R

statistical software, one software to Rule them all

Cookbook (niveau M2)

La Bible R en images (et en couleurs…)

Vincent ISOZ, Daname KOLANI

V4.0 Révision 275 (12024-01-26), {oUUID 1.851}

R *you ready?*

Table des matières

1. Introduction

Contenu

R est un projet GNU libre de traitement des données et d'analyse statistiques (sans AUCUNE GARANTIE COMMERCIALE quant à la fiabilité des algorithmes) mettant en œuvre le langage de programmation S créé à l'époque par John Chambers et ses collègues lorsqu'il était aux laboratoires Bell en 1976. Depuis plusieurs années, deux nouvelles versions apparaissent au printemps et à l'automne. Il dispose de nombreuses fonctions graphiques.

Le logiciel R est considéré par ses créateurs comme étant une exécution de S, avec la sémantique dérivée du langage Scheme. C'est un logiciel libre distribué selon les termes de la licence GNU GPL et est disponible sous GNU/Linux, FreeBSD, NetBSD, OpenBSD, Mac OS X et Windows (ce qui est un énorme avantage dans l'enseignement et le pratique des universités et instituts de recherche). Il représente aujourd'hui l'un des objectifs techniques majeurs de la communauté GNU.

Il convient de signaler que le 6 Avril 2015 Microsoft a acheté la version entreprise de R (Revolution Analytics) dont la rapidité d'exécution serait de 20 fois supérieure à la version standard de R et 42 fois plus rapide que SAS[1]. Ce qui présage que R avec ses 2 millions d'utilisateurs va encore plus dominer le domaine de l'analyse statistique de données.

[1] http://blogs.technet.com/b/machinelearning/archive/2015/04/06/microsoft-closes-acquisition-of-revolution-analytics.aspx

Le choix de R dans une organisation peut se justifier principalement selon moi (Vincent ISOZ) par le fait que:

- R est gratuit et la documentation gratuite de qualité ET homogène sur l'Internet est considérable de plus il marche sur PC, Mac, Unix et Linux ainsi que sur les pages web.

- Permet de faire des graphiques statiques ou dynamiques assez intéressant et épurés de tout gadget esthétique inutile (les possibilités techniques sont bien supérieures à Excel, SPSS, SAS[2] ou Minitab)

- Les étudiants provenant d'universités ont la plupart fait leurs premières armes avec R et sont donc déjà habitués à cet environnement.

- La communauté d'utilisateurs (et donc d'intervenants) semble plus vaste que toutes les autres solutions alternatives d'où une plus grande réactivité sur les forums (même si le nom du logiciel pose parfois problème avec les moteurs de recherche puisqu'il ne fait qu'une seule lettre...)

- Il contient des packages de haut niveau et souvent dès que de nouveaux modèles théoriques de statistiques (analyses de données en général) sont publiés, ils sont en premier disponible sur R pour la communauté académique.

Enfin rappelons que ce logiciel se destine à des ingénieurs et universitaires donc son rôle n'est pas de faire de l'esthétique mais de l'efficace!

Le logiciel R est devenu depuis sa création un outil incontournable de statistique et de visualisation de données tant dans le monde universitaire que dans celui de l'entreprise. Ce développement exceptionnel s'explique par ses trois principales qualités: il est gratuit, très complet et en essor permanent.

Dans tout ce livre nous avons fait le choix de travailler principalement avec des fichiers *.csv. Pourquoi ce choix?:

1. Parce que les fichiers de tableurs comme Microsoft Excel ou autres ne sont pas pérennes sur les très longues durées (au-delà du demi-siècle)

2. Parce que les fichiers XML sont difficiles à produire par les pratiquants dans les laboratoires qui n'ont pas de formation de programmeur et ne connaissant pas les normes internationales d'échanges de données.

Le lecteur remarquera que le début du présent document rassemble uniquement des points qui sont aussi faisables avec Microsoft Excel avec plus ou moins de facilité (parfois plus rapidement, parfois pas…). Le but est d'abord de montrer les différences entre les deux logiciels comme le font de nombreux ouvrages avant de passer à des cas très spécifiques à R.

Enfin signalons que l'idée de ce support de cours sera au long terme de traiter des sujets suivants (et dans l'ordre indiqué) dont les démonstrations mathématiques ont été faites pendant les cours théoriques:

[2] Si vous étouffez avec SAS... prenez une bouffée d'R !

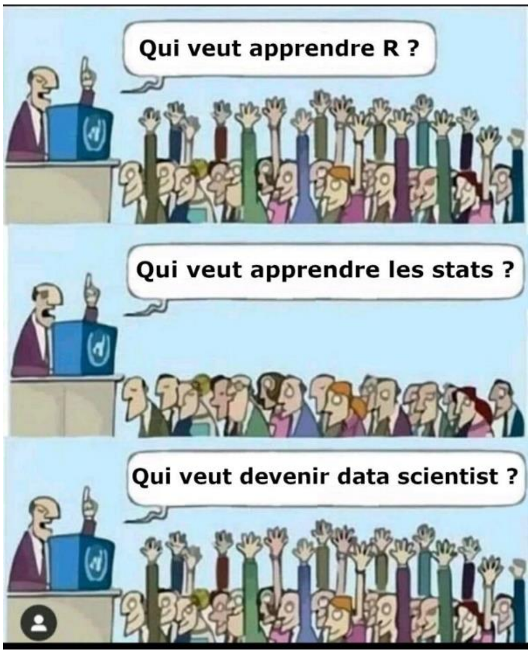
- Fondamentaux (manipulations scolaires)
- Manipulation et sources de données avancées
- Statistiques descriptives
- Statistiques inférentielles paramétriques
- Statistiques inférentielles non paramétriques
- Représentations graphiques
- Statistiques bayésiennes
- Méthodes de Monte-Carlo
- Fouille de données/Data Mining (+statistique multivariée)
- Analyses de texte/Text Mining
- Analyses de séries temporelles
- Actuariat & Économie (finance quantitative)
- Ingénierie de la fiabilité (modèles de survie/garanties)
- Plans d'expérience
- Recherche opérationnelle/Optimisation
- Structural Equation Modeling (SEM)
- Logique Floue
- Scripting (programmation)
- Traitement des images

Un ordinateur est un outil incomparable entre les mains de celui qui sait. Sous les doigts du Crétin, c'est un revolver manipulé par un aveugle au milieu de la foule Chester Himes

Remarques:

R1. Les premiers exercices sont donc quasiment les mêmes que ceux effectués dans les cours de statistique théorique, maîtrise statistique des procédés, d'analyse décisionnelle avec Microsoft Excel, Minitab, MATLAB et SPSS ou lors du cursus d'ingénierie de gestion projets (dans lesquels les calculs sont faits à la main).

R2. Nous présentons dans ce livre que des outils utilisant des concepts mathématiques dont nous exposons la démonstration mathématique détaillée (démonstrations disponibles dans mon livre sur les Mathématiques Appliquées) dans nos cours et qui sont utilisés dans le cadre de nos activités de consultants. **Si cela vient à ne pas être le cas, le titre du sujet est suivi de l'abréviation *WP* qui est l'abréviation de: Without Proof**.



R3. J'ai rédigé ce document uniquement pour le fun afin de valider les résultats obtenus à la main et surtout pour jouer avec les théorèmes mathématiques que j'ai étudiés en détail dans le cadre de mes études (démonstrations mathématiques disponibles dans mon ouvrage sur les Mathématiques Appliquées).

R4. Suite à des questions de la part de lecteurs: Non, nous ne sommes pas rétribués par R pour leur faire de la pub… Des logiciels comme:

XLStat, SPSS, JASP, Minitab (voir mon PDF sur ce logiciel), SAS, PSPP, Gauss, MATLAB, Statistica, Stata, Medcalc, StatsDirect, SigmaXL, NumXL, JMP, Weibull++, Design-Expert, PlanExpert, UNISTAT, QIMacros (Quality Improvement Macros), EasyFitXL, NemrodW, qs-STAT, NeoLiCy, etc

font à peu près pareil relativement aux sujets couverts dans ce document. Il fallait juste que nous fassions un choix... (nous ne pouvons pas passer notre temps à écrire des supports sur tous les types de logiciels!) et celui-ci s'est porté sur le logiciel utilisé par des écoles dans lesquelles nous intervenons parfois en tant que consultants. Cependant si des passionnés veulent reproduire le contenu du présent livre avec leur logiciel de statistique favori qu'ils n'hésitent pas! Ce serait même fort intéressant de comparer les résultats!! Si nous avons le temps nous écrirons le même contenu mais avec SPSS (cela étant déjà fait avec Minitab et partiellement avec Microsoft Excel).

Pour terminer, nous tenons à remercier ici les quelques collègues et clients qui ont bien voulu nous faire part de leurs remarques pour améliorer le contenu de ce livre électronique. Il est cependant certain qu'il est encore perfectible sur de nombreux points et qu'il va encore évoluer puisqu'il y a encore une petite dizaine de sujets dont les démonstrations mathématiques sont en cours de rédaction pour la prochaine édition du livre de Vincent ISOZ sur les Mathématiques Appliquées et nous montrerons comment les utiliser aussi avec R.

Si vous souhaitez être informé des nouvelles versions **majeures** de ce document n'hésitez pas à écrire un mail dans ce sens à Vincent Isoz: isoz@sciences.ch.

PS: Les fichiers utilisés pour les exemples qui vont suivre ne sont fournis qu'aux entreprises, administrations et écoles faisant appel à nos services de consultants/professeurs.

À propos des rédacteurs

ISOZ Vincent



Domicilié à ce jour à Lausanne (Suisse)

Formation: Ingénieur Physicien HES
Équivalence Internationale: Bachelor of Science
Fonction: Consultant scientifique senior

Année de naissance: 1978

Courriel: isoz@sciences.ch

Actuellement, je travaille en tant qu'employé pour l'entreprise Scientific Evolution Sàrl spécialiste dans le conseil et la formation et techniques quantitatives pour les hauts-potentiels. Mes tâches consistent à faire du conseil (français, anglais et allemand) en mathématiques appliquées dans le tutorat d'analystes quantitatifs (niveau Bac+5 à Bac+7). Accessoirement, je suis le rédacteur de plusieurs livres électroniques dans les domaines suivants:

- maîtrise statistique des processus/procédés (méthodes paramétriques et non paramétriques)
- modélisation prévisionnelle/décisionnelle avancée (arbres de décisions, chaînes de Markov)
- recherche opérationnelle (simplexe, algorithmes génétiques, algorithme GRG)
- data mining (réseaux de neurones, ACP, AFC, régressions, scoring, clustering, etc.)
- modélisation du risque en gestion de projets et finance d'entreprise (monte-carlo, etc.)
- gestion de projets (modèles et best practices théoriques EFQM+Six Sigma, MS Project)
- ISO 9001:2008, 5807:1985, 10015:1999, 31000+31010:2009, 8258:1991, 10017:2003, etc.
- Adobe Photoshop et Illustrator
- 12 applications de la suite Microsoft Office System (Project, Visio, SharePoint, Access, etc.)

À ce jour interventions dans plus de ~200 entreprises dont 10 du *Fortune 500* selon listing 2009 et 3 universités et écoles d'ingénieurs suisses dans des cours de modélisation de bases de données et simulations stochastiques du risque. Formation de plusieurs dirigeants de multinationales en one to one.

Accessoirement j'interviens pour des formations sur des logiciels comme MS Project, MS Visio, MS Access et une vingtaine d'autres dont je délègue l'organisation à des entreprises spécialisées dans la formation continue en bureautique (niveau licence universitaire et en-dessous).

Enfin, je conseille aussi vivement à toute personne souhaitant vraiment maîtriser le sujet de lire mon e-book sur les *Mathématiques Appliquées* (~5'900 pages) ou de préférence la version anglophone (>7'000 pages).

KOLANI Daname



Domicilié à ce jour à Casablanca (Maroc)

Formation: École Nationale de Commerce et de Gestion - Casablanca (BAC + 5)
Fonction: Consultant scientifique junior

Année de naissance: 1990

Courriel: daname.kolani@hotmail.com

Actuellement, diplômé d'école de commerce (option finance), je suis un passionné de la finance, de l'économie, de l'économétrie, des statistiques & probabilités. Outre, les théories qui y sont associées, j'ai un grand intérêt pour les outils qui fondent l'ensemble de ces domaines. Je suis bilingue (Français & Anglais) et mes compétences spécifiques concernent:

- La finance d'entreprise (Evaluation d'investissement & de financement, Analyse et diagnostic financier, Contrôle de gestion et Evaluation de société …)

- La finance de marché (gestion de portefeuille financier, l'évaluation d'actif financier, outils de gestion de risque…)

J'ai aussi un intérêt pour l'outil informatique notamment les logiciels comme R Project, VBA – Microsoft Excel, ,MATLAB, EViews et IBM SPSS…

J'ambitionne me lancer dans le consulting et par là, contribuer à l'optimisation des décisions financières et à la mise en place de stratégie financière efficiente cadrant avec le modèle d'affaire des entreprises.

Avertissements

Le contenu du présent support est élaboré par un processus de développement par lequel des experts en statistiques parviennent à un consensus. Ce processus qui rassemble des participants bénévoles recherche également les points de vue de personnes intéressées par le sujet de cet ouvrage. En tant que responsable du présent support, j'assure l'administration du processus et je fixe les règles qui permettent de promouvoir l'équité dans l'approche d'un consensus. Je me charge également de rédiger les textes, parfois de les tester/évaluer ou de vérifier indépendamment l'exactitude/solidité ou l'exhaustivité des informations présentées.

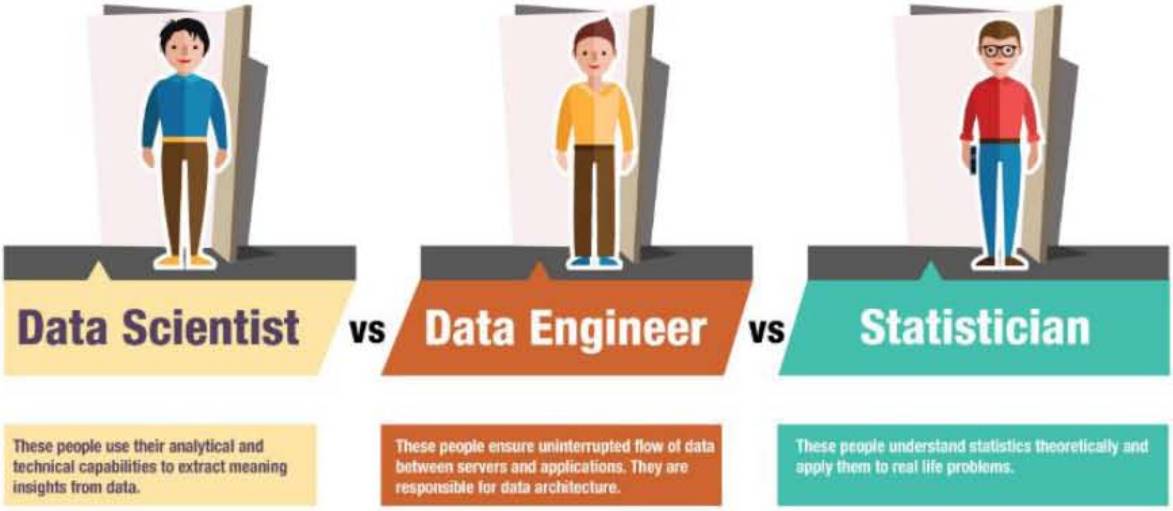
Je décline toute responsabilité en cas de dommages corporels, matériels ou autres de quelque nature que ce soit, particuliers, indirects, accessoires ou compensatoires, résultant de la publication, de l'application ou de la confiance accordée au contenu du présent support. Je n'émets aucune garantie expresse ou implicite quant à l'exactitude ou à l'exhaustivité de toute information publiée dans le présent support, et ne garantit aucunement que les informations contenues dans cet ouvrage satisfassent un quelconque objectif ou besoin spécifique du lecteur. Je ne garantis pas non plus les performances de produits ou de services d'un fabricant ou d'un vendeur par la seule vertu du contenu du présent support.

En publiant des textes, il n'est pas dans l'intention principale du présent support de fournir des services de spécialistes ou autres au nom de toute personne physique ou morale ni pour mon compte, ni d'effectuer toute tâche devant être accomplie par toute personne physique ou morale au bénéfice d'un tiers. Toute personne utilisant le présent support devrait s'appuyer sur son propre jugement indépendant ou, lorsque cela s'avère approprié, faire appel aux conseils d'un spécialiste compétent afin de déterminer comment exercer une prudence raisonnable en toute circonstance. Les informations et les normes concernant le sujet couvert par le présent support peuvent être disponibles auprès d'autres sources que le lecteur pourra souhaiter consulter en quête de points de vue ou d'informations supplémentaires qui ne seraient pas couverts par le contenu du présent site Internet.

Je ne dispose (malheureusement...) d'aucun pouvoir dans le but de faire respecter la conformité au contenu du présent ouvrage, et je ne m'engage nullement à surveiller ni à faire respecter une telle conformité. Je n'exerce (à ce jour...) aucune activité de certification, de test ni d'inspection de produits, de conceptions ou d'installations à fins de santé ou de sécurité des personnes et des biens. Toute certification ou autre déclaration de conformité en matière d'informations ayant trait à la santé ou à la sécurité des personnes et des biens, mentionnée dans le présent support, ne peut aucunement être attribuée au contenu du présent support et demeure sous l'unique responsabilité de l'organisme de certification ou du déclarant concerné.

Et pour rappel:

- Tous les modèles sont faux
- Certains modèles sont meilleurs que d'autres
- Le modèle le meilleur ne peut jamais être connu avec certitude!
- Plus simple est le modèle, mieux c'est!


Data Scientist
vs
Data Engineer
vs
Statistician
These people use their analytical and technical capabilities to extract meaning insights from data.
These people ensure uninterrupted flow of data between servers and applications. They are responsible for data architecture.
These people understand statistics theoretically and apply them to real life problems.

Comment utiliser ce livre

Dans le cadre des études supérieures, ce livre peut être utilisé pour des séminaires de niveau licence, master ou doctorat dans de nombreux domaines des sciences pures et exactes. Les séminaires dans lesquels nous utilisons ce matériel est une partie intégrale des programmes de Scientific Evolution Sàrl, où les apprenants ont déjà derrière eux un cursus universitaire ni licence ou maîtrise dans leur spécialisation.

Parce que les méthodes des mathématiques appliquées s'apprennent par la pratique et l'expérience, nous concevons un séminaire sur R comme un processus d'apprentissage par la pratique. Nous structurons nous séminaires de modélisation mathématiques autour d'un ensemble de problèmes qui requière aux apprenants de construire des modèles qui aident à s'organiser et à prendre des décisions. L'important est que les modèles doivent être consistants avec la théorie et back-testés. Pour répondre à cet impératif, il est nécessaire pour l'apprenant de combiner les mathématiques théoriques avec la modélisation. Le résultat est que l'apprenant maîtrise la théorie et, plus important, qu'il acquière les méthodes permettant d'appliquer la théorie à des cas pratiques réels. La capacité de critiquer et identifier les limitations d'outils mathématiques sensibles est la valeur ajoutée la plus importante de nos séminaires.

Les problèmes avec solutions dans le texte fournissent une opportunité d'appliquer les notions théoriques à de petits cas pratiques tirés ou inspirés pour la majorité de cas réels. À la fin des séminaires les apprenants ont amélioré leurs compétences et connaissances des outils théoriques et computationnels majeurs.

Il est vraiment difficile de traiter toute la matière de ce livre en un semestre. Il faut beaucoup de temps pour expliquer la finesse des concepts aux apprenants. Le lecteur est alors encouragé de choisir les sujets qui lui sont nécessaires pendant son temps d'étude. Ainsi, il n'est pas obligatoire de parcourir les chapitres dans l'ordre dans lesquels ils sont présentés mais cela peut toutefois aider de façon significative…

En un mot, ce livre offre une large variété de sujets qui se prêtent à la modélisation, il est le complément pratique du livre théorique *Opera Magistris* (8'000 pages) et la version écrite des formations vidéos suivantes chez ALPHORM:



Pour ceux qui souhaitent un échantillon de ce support de cours dans un format plus académique, vous pouvez commander notre ouvrage papier sur Amazon.fr:

https://www.amazon.fr/Langage-Daname-KOLANI-Vincent-ISOZ/dp/2409014208/ref=sr_1_1?ie=UTF8&qid=1525176182&sr=8-1&keywords=vincent+isoz



Fichiers de données

Les fichiers de données utilisées dans ce livre, c'est-à-dire:

- 579 fichiers (*.r, *.rprofile, *.csv, *.xml, *.jpg, *.mdb, *.xls, etc)
- Classés dans 32 dossiers
- Pour un total 579 MB de de données

sont mis à disposition, avec mises à jour gratuites perpétuelles, uniquement aux apprenants ayant suivi nos formations correspondantes à ce livre ou ayant payé (ou fait un don) d'une somme équivalente de **499$**.

Les universités, écoles d'ingénieurs et les sociétés privées doivent acheter une seule fois le package de fichiers pour une utilisation illimitée de leurs étudiants ou employés, mais limité à une seule localisation géographique.

Pour obtenir plus d'informations sur le processus d'achat, vous pouvez contacter envoyer un courriel à l'adresse suivante: isoz@sciences.ch

Votre avis nous intéresse!

En tant que lecteur de ce document, vous êtes le critique et le commentateur le plus important. Votre opinion compte et il est très intéressant de savoir ce qui est bien, ce qui peut être mieux et les sujets que vous souhaiteriez voir être traités.

Vous pouvez m'envoyer un e-mail pour partager ce que vous avez aimé ou détesté dans le présent document afin d'en assurer une amélioration continue.

Si vous souhaitez compléter le présent support avec un sujet qui vous tient à coeur et pour lequel vous avez la démonstration mathématique n'hésitez pas à me contacter. J'intégrerai le sujet en précisant votre nom et prénom.

Notez que malheureusement, je ne peux pas répondre gratuitement à des questions techniques d'ingénierie ou de problématique d'entreprise par e-mail pour des raisons professionnelles évidentes.

E-mail: isoz@sciences.ch

Remerciements

Cet e-book n'aurait probablement jamais pu voir le jour sans l'apport désintéressé de nombreux contributeurs à ce bijou qu'est R. Nous souhaiterions donc exprimer notre plus grande gratitude et nos plus sincères remerciements à l'équipe qui maintient R et tous les contributeurs des différentes packages utilisés et cités dans le présent ouvrage à ce jour :

Tony Plate, Richard Heiberger, Andreas Kiermeier, Peter Bloomfield, Phil Spector, Jerome Friedman, Robert Tibshirani, Thomas Lumley, Shawn Garbett, Jonathan Baron, Ezra Haber Glenn, Vincent Goulet, Sébastien Auclair, Christophe Dutang, Xavier Milhaud, Tommy Ouellet, Alexandre Parent, Mathieu Pigeon, Louis-Philippe Pouliot, Stéphane Dray, Thibaut Jombart, Sandrine Pavoine, Jean R. Lobry, Sébastien Ollier, Daniel Borcard, Pierre Legendre, Aurélie Siberchicot, Daniel Chessel, Clement Calenge, Mathieu Basille, Stephane Dray, Scott Fortmann-Roe, Manuela Royer, Carlos J. Gil Bellosta, Christian Kleiber, Achim Zeileis, Felipe de Mendiburu, Hiroshi Akima, Albrecht Gebhardt, Thomas Petzold, Martin Maechler, Association for Computing Machinery Inc, Bob Wheeler, Michal Bojanowski, Robin Edwards, Manuel Castejon Limas, Joaquin B. Ordieres Mere, Ana Gonzalez Marcos, Francisco Javier Martinez de Pison Ascacibar, Alpha V. Pernia Espinoza, Fernando Alba Elias, Jose Maria Perez Ramos, P Lafaye de Micheaux, Yihui Xie, Christian Mueller, Lijia Yu, Weicheng Zhu, Dirk Eddelbuettel, Emmanuel Paradis, Simon Blomberg, Ben Bolker, Joseph Brown, Julien Claude, Hoa Sien Cuong, Richard Desper, Gilles Didier, Benoit Durand, Julien Dutheil, RJ Ewing, Olivier Gascuel, Christoph Heibl, Anthony Ives, Bradley Jones, Franz Krah, Daniel Lawson, Vincent Lefort, Jim Lemon, Rosemary McCloskey, Johan Nylander, Rainer Opgen-Rhein, Andrei-Alin Popescu, Manuela Royer-Carenzi, Klaus Schliep, Korbinian Strimmer, Damien de Vienne, Hans Peter Wolf, Uni Bielefeld, Jenise Swall, Kristen Foley, Dylan Beaudette, Pierre Roudier, Andrew Gelman, Yu-Sung Su, Masanao Yajima, Jennifer Hill, Maria Grazia Pittau, Jouni Kerman, Tian Zheng, Vincent Dorie, Michael Hahsler, Christian Buchta, Bettina Gruen, Kurt Hornik, Ian Johnson, Christian Borgelt, Sudheer Chelluboina, Hadley Wickham, Cqls Team, Yvonnick Noel, Simon Urbanek, Michel Lang, Jeroen Ooms, Rasmus Bååth, Richard D. Morey, Jeffrey N. Rouder, Tahira Jamil, David Ardia, Jonah Gabry, Tristan Mahr, Paul-Christian Buerkner, Ravi Varadhan, Paul Gilbert, Marcos Raydan, JM Martinez, EG Birgin, W LaCruz, R Development Core Team, John W. Emerson, Michael J. Kane, Agner Fog, Kirill Müller, RStudio, Frederic Bertrand, Bart Smeets, Iñaki Ucar, Jens Oehlschlägel, Steve Dutky, Jo Hardin, Bernhard Lehnert, Angelo Canty, Brian Ripley, Rob Tibshirani, Friedrich Leisch, Heather Turner, David Firth, Jeffrey Horner, Ioannis Kosmidis, Quentin F. Gronau, Henrik Singmann, Jonathan J. Forster, Eric-Jan Wagenmakers, The JASP Team, Jiqiang Guo, Ben Goodrich, Paul-Christian Bürkner, Robin K. S. Hankin, David Robinson, Matthieu Gomez, Boris Demeshev, Dieter Menne, Benjamin Nutter, Luke Johnston, Francois Briatte, Jeffrey Arnold, Luciano Selzer, Gavin Simpson, Jens Preussner, Jay Hesselberth, Matthew Lincoln, Alan T. Arnholt, Ben Evans, Ernesto Barrios, Daniel Meyer, Gert Janssenswillen, Max Kuhn, Weston Steve, Culp Mark, Coulter Nathan, Ross Quinlan, RuleQuest Research, Rulequest Research Pty Ltd, Michael Lawrence, Gábor Csárdi, Winston Chang, John Fox, Sanford Weisberg, Daniel Adler, Douglas Bates, Gabriel Baud-Bovy, Steve Ellison, Michael Friendly, Gregor Gorjanc, Spencer Graves, Rafael Laboissiere, Georges Monette, Duncan Murdoch, Henric Nilsson, Derek Ogle, William Venables, David Winsemius, R-Core, Brad Price, Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, Tyler Hunt, Sebastian Jeworutzki, Timothee Giraud, Nicolas Lambert, Roger Bivand, Edzer Pebesma, Jarek Tuszynski, Ignacio González, Sébastien Déjean, Jennifer Bryan, Markus Gesmann, Daniel Murphy, Yanwei Zhang, Alessandro Carrato, Giuseppe Crupi, Arnaud Lacoume, Arthur Charpentier, Mario Wuthrich, Fabio Concina, Bernd Bischl, Microsoft Corporation, Ari Lamstein, David James, Gabor Grothendieck, R Core Team, Zuguang Gu, Ulric Lund, Federico Rotolo, Hisaji Ono, Richard Dunlap, Matthieu Stigler, Louis Maddox, Steve Simpson, Walter Böhm, Peter Rousseeuw, Anja Struyf, Mia Hubert, Matthias Studer, Juan Gonzalez, Bob Gray, Martyn Plummer, Nicky Best, Kate Cowles, Karen Vines, Deepayan Sarkar, Russell Almond, Arni Magnusson, Luke Tierney, Torsten Hothorn, Mark A. van de Wiel, Henric Winell, Tim Keitt, Ross Ihaka, Paul Murrell, Jason C. Fisher, Reto Stauffer, Dean Attali, David Griswold, Scott Chasalow, John MacFarlane, Bill Venables, JJ Allaire, RStudio Inc, P. Lafaye de Micheaux, Marius Hofert, Juliane Schafer, Verena Zuber, Miika Ahdesmaki, A. Pedro Duarte Silva, Taiyun Wei, Viliam Simko, Michael Levy, Yan Jin, Jeff Zemla, Claus O. Wilke, Carsten Franz, Brodie Gaslam, Martin Oliver Sailer, Joe Cheng, jQuery Foundation, jQuery contributors, Mark Otto, Jacob Thornton, Bootstrap contributors, Twitter Inc, Brian Reavis, Kristopher Michael Kowal, es5-shim contributors, Denis Ineshin, Sami Samhuri, Balasubramanian Narasimhan, Manuel Koller, Steven G. Johnson, Keefer Chris, Tammo Krueger, Mikio Braun, Mike Bostock, Kent Russell, Gregor Aisch, Adam Pearce, Ben Ortiz, Matt Dowle, Arun Srinivasan, Jan Gorecki, Tom Short, Steve Lianoglou, Eduard Antonyan, Facundo Munoz, Markus Wamser, Pierre Formont, Kent Russel, Noam Ross, Duncan Garmonsway, Christoph Glur, Christopher Gandrud, Boxuan Cui, R Special Interest Group on Databases, Edgar Ruiz, Oleksii Pokotylo, Pavlo Mozharovskyi, Rainer Dyckerhoff, Stanislav Nagy, Ian Fellows, Rolf Turner, Tal Galili, Gregory Jefferis, Marco Gallotta, Johan Renaudie, Uwe Ligges, Andrej-Nikolai Spiess, Steve Horvath, Peter Langfelder, skullkey, Mark Van Der Loo, Andrie de Vries, Cath, John Ma, Krzysiek G, Yoav Benjamini, Koji Makiyama, Eduardo L. T. Conceicao, Andrew Clausen, Serguei Sokol, Aravind Hebbali, Andri Signorell, Ken Aho, Andreas Alfons, Nanina Anderegg, Tomas Aragon, Antti Arppe, Adrian Baddeley, Kamil Barton, Frederico Caeiro, Stephane Champely, Leanne Chhay, Clint Cummins, Michael Dewey, Harold C. Doran, Charles Dupont, Jeff Enos, Claus Ekstrom, Martin Elff, Kamil Erguler, Richard W. Farebrother, Romain Francois, Matthias Gamer, Joseph L. Gastwirth, Yulia R. Gel, Juergen Gross, Frank E. Harrell Jr, Michael Hoehle, Christian W. Hoffmann, Markus Huerzeler, Wallace W. Hui, Pete Hurd, Rob J. Hyndman, Pablo J. Villacorta Iglesias, Matthias Kohl, Mikko Korpela, Detlew Labes, Friederich Leisch, Dong Li, Daniel Malter, George Marsaglia, John Marsaglia, Alina Matei, David Meyer, Weiwen Miao, Giovanni Millo, Yongyi Min, David Mitchell, Markus Naepflin, Daniel Navarro, Klaus Nordhausen, Hong Ooi, Nick Parsons, Tony Plate, Roland Rapold, William Revelle, Tyler Rinker, Brian D. Ripley, Caroline Rodriguez, Nathan Russell, Nick Sabbe, Venkatraman E. Seshan, Greg Snow, Michael Smithson, Werner A. Stahel, Mark Stevenson, Matthias Templ, Terry Therneau, Yves Tille, Adrian Trapletti, Kevin Ushey, Jeremy VanDerWal, John Verzani, Gregory R. Warnes, Stefan Wellek, Rand R. Wilcox, Peter Wolf, Daniel Wollschlaeger, Thomas Yee, Karline Soetaert, Thomas Petzoldt, R. Woodrow Setzer, odepack authors, Jim Hester, Johns Hopkins University, Hans W. Borchers, ABB Corporate Research, Richard Barnes, Kevin Sahr, Gerald Evenden, Angus Johnson, Frank Warmerdam, Taylor B. Arnold, Richard Iannone, Jessica Franco, Delphine Dupuy, Olivier Roustant, Guillaume Damblin, Bertrand Iooss, Ken Knoblauch, Scott Waichler, Antoine Lucas, Henrik Bengtsson, Mario Frasca, Bryan Lewis, Murray Stokely, Hannes Muehleisen, Wush Wu, Qiang Kou, Thierry Onkelinx, Guido Kraemer, Gaston Sanchez, Charles Determan, Robert J. Hijmans, Steven Phillips, John Leathwick, Jane Elith, Florian Camphausen, Peter Ruckdeschel, Thomas Stabla, Matt Taddy, Ulrike Groemping, Boyko Amarov, Hongquan Xu, Lenth Russ, Rich Calaway, Dan Tenenbaum, Kaspar Moesinger, Florian Gerber, Reinhard Furrer, Andy Bunn, Franco Biondi, Filipe Campelo, Pierre Mérian, Fares Qeadan, Christian Zang, Allan Buras, Jacob Cecile, Manfred Mudelsee, Michael Schulz, Darwin Pucha-Cofrep, Jakob Wernicke, Lionel Henry, Xianying Tan, Maximilian Girlich, SpryMedia Limited, Leon Gersen, Bartek Szopka, Alexis Dinno, Dan Vanderkam, Jonathan Owen, Daniel Gromer, Petr Shevtsov, Benoit Thieurmel, Evgenia Dimitriadou, Andreas Weingessel, Chih-Chung Chang, Chih-Chen Lin, Stephen Milborrow, Trevor Hastie, Alan Miller, Thomas Lumley, Emmanuel Arnhold, Stephen H-T. Lihn, Yves

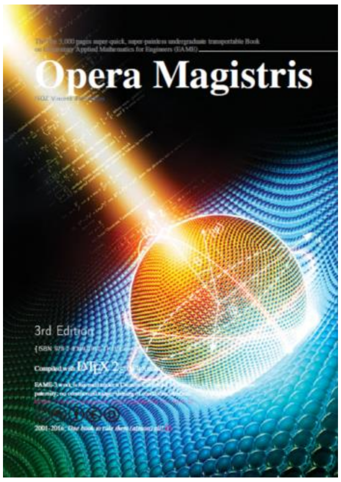
Croissant, Sarah Goslee, Dean Urban, Marijke Swennen, Jangman Hong, Robert Andersen, Steve Taylor, E. D. Chow, Maria L. Rizzo, Gabor J. Szekely, Bendix Carstensen, Esa Laara, Michael Hills, Telmo Nunes, Cord Heuer, Jonathon Marshall, Javier Sanchez, Ron Thornton, Jeno Reiczigel, Jim Robison-Cox, Paola Sebastiani, Peter Solymos, Kazuki Yoshida, Geoff Jones, Sarah Pirikahu, Simon Firestone, Ryan Kyle, Johann Popp, Mathew Jay, Tomas J. Aragon, Michael P. Fay, Adam Omidpanah, Russell Lenth, Paul Buerkner, Arthur Allignol, Thomas Kluyver, Barret Schloerke, Adam Ryczkowski, Alec Stephenson, Chris Ferro, Yilei Yu, Lu Tian, Gerard Jungman, Brian Gough, Jeffrey A. Ryan, Robert Gentleman, R Foundation, Marina Shapira, Michael Stadelmann, Tymoteusz Wolodzko, Alboukadel Kassambara, Fabian Mundt, Francois Husson, Julie Josse, Sebastien Le, Jeremy Mazet, Diethelm Wuertz, Tobias Setz, Yohan Chalabi, J L Marchini, Yohan Chalabi Martin Maechler, J. O. Ramsay, Bernd Klaus, Andrew Collier, Edwin de Jonge, Jan Wijffels, Jan van der Laan, Chris Boudt, Pierre Chausse, Michal Miklovac, Douglas Nychka, John Paige, Stephan Sain, Roger D. Peng, Lukasz Komsta, Marie-Laure Delignette-Muller, Regis Pouillot, Jean-Baptiste Denis, Fionn Murtagh, Frederic Mortier, Nicolas Picard, David Gohel, Maxim Nazarov, Alina Beygelzimer, Sham Kakadet, John Langford, Sunil Arya, David Mount, Shengqiao Li, Microsoft, Rob Hyndman, Mitchell O'Hara-Wild, Christoph Bergmeir, Slava Razbash, Earo Wang, Vincent J. Carey, Saikat DebRoy, Stephen Eglen, Rajarshi Guha, Swetlana Herbrandt, Nicholas Lewin-Koh, Mark Myatt, Ben Pfaff, Brian Quistorff, Stephen Weigand, Free Software Foundation Inc, Eugene Ha, Kohske Takahashi, Ed Lee, Kun Ren, Kenton Russell, Christian Hennig, William Chen, Chris Fraley, U.Washington, Seattle, Fritz Leisch at TU Wien, Valderio Reisen, Artur Lemonte, Lala Septem Riza, Francisco Herrera, Jose Manuel Benitez, Brian Lee Yung Rowe, Jon Garibaldi, Chao Chen, Tajul Razak, Craig Knott, Luke Hovell, Nathan Karimian with supervision from Dr. Jon Garibaldi, Catherine Hurley, Jonathan Asher Greenberg, Matteo Mattiuzzi, Ales Korosec, Don MacQueen, Jim Rogers, Jacob van Etten, Yixuan Qiu, Kamil Wais, Nathan VanHoudnos, John Ramey, Konstantinos Fokianos, Scott Chamberlain, Andy Teucher, Tom MacWright, Mads Kristensen, Jonathan M. Lees, Paulo J. Ribeiro Jr, Ed Williams, Chris Vennes, Christopher Baker, Speros Kokenes, David Kahle, Kamil Slowikowski, Jean-Olivier Irisson, Saulius Lukauskas, Alicia Schep, Zhian N Kamvar, Sean Hughes, Pierre Gramme, Nan Xiao, Miaozhu Li, Constantin Ahlmann-Eltze, Jeffrey B. Arnold, Gergely Daroczi, Bo Werth, Brian Weitzner, Joshua Kunst, Baptise Auguie, Bob Rudis, Justin Talbot, Joshua London, Ian Marschner, Noah Simon, Junyang Qian, Vincent Calcagno, Randall C Johnson, Randall C. Johnson, SAIC-Frederick Inc. Funded, National Cancer Institute, Center for Cancer Research under NCI Contract NO1-CO-12400, Immanuel Scholz, Rainer Boehme, Douglas M. Bates, Julian Faraway, Diego de Castillo, Coen Bernaards, Robert Jennrich, Marlene Mueller, Lodewijk Bonebakker, Wolfgang Huber Andy Liaw, Steffen Moeller, Marc Schwartz, Stephen Gardiner, Baptiste Auguie, Anton Antonov, Chong Gu, Benedikt Graeler, G. Grothendieck, Philippe Massicotte, S Subramanian, Philippe Grosjean and Michael Lawrence, H2O.ai team, Evan Miller, Mike Meredith, John Kruschke, Dan Carr, Richard M. Heiberger, Qiang Li, The R Consortium, Brian S. Everitt, Max Gordon, Stephen Gragg, Peter Konings, Ramnath Vaidyanathan, Hector Corrada Bravo, Matthew Wolak, Wencheng Lau-Medrano, Jacobs Simon, Khanna Aditya, Madduri Kamesh, Bader David, Oleg Sklyar, Mike Smith, Barry Rowlingson, Boris Hejblum, Dason, Felix Schonbrodt, Heuristic Andrew, James, Thomas Leeper, VitoshKa, Kornelius Rohmeyer, Takekatsu Hiramura, Berry Boessenkool, Jonathan Godfrey, Tom Allard, ChingChuan Chen, Jonathan Hill, Pablo Barbera, Tiago Dantas, Jonne Guyt, Richard Bourgon, Andrea Peters, Beth Atkinson, Jim Baglama, Lothar Reichel, B. W. Lewis, Revolution Analytics, Markus Helbig, Rich FitzJohn, Stefan Milton Bache, Duncan Temple Lang, Lloyd Hilaiel, Mathias Buus, Arsalane Chouaib Guidoum, A.I. McLeod, Alexandros Karatzoglou, Alex Smola, Matt Wand, Christian Roever, Nils Raabe, Karsten Luebke, Gero Szepannek, Marc Zentgraf, Adam Vogt, Alastair Andrew, Alex Zvoleff, Andre Simon, Aron Atkins, Aaron Wolen, Ashley Manton, Ben Baumer, Brian Diggs, Cassio Pereira, Christophe Dervieux, David Hugh-Jones, Donald Arseneau, Doug Hemken, Elio Campitelli, Fabian Hirschmann, Fitch Simeon, Forest Fang, Garrick Aden-Buie, Gregoire Detrez, Heewon Jeon, Hiroaki Yutani, Ian Lyttle, Hodges Daniel, Jake Burkhead, James Manton, Jared Lander, Jason Punyon, Javier Luraschi, Jeff Arnold, Jenny Bryan, Jeremy Ashkenas, Jeremy Stephens, Johannes Ranke, John Honaker, John Muschelli, Jonathan Keane, Johan Toloe, Jonathan Sidi, Joseph Larmarange, Julien Barnier, Kaiyin Zhong, Karl Forner, Kevin K. Smith, Kirill Mueller, Martin Modrák, Michael Chirico, Michel Kuhlmann, Nacho Caballero, Nick Salkowski, Obada Mahdi, Richard Cotton, Robert Krzyzanowski, Ruaridh Williamson, Scott Kostyshak, Sebastian Meyer, Sietse Brouwer, Simon de Bernard, Sylvain Rousseau, Thibaut Assus, Thibaut Lamadon, Tom Torsney-Weir, Trevor Davis, Viktoras Veitas, Zachary Foster, Daniel Ludecke, Bojanowski Michal, Josef Holzer, Klaus K. Holst, Brice Ozenne, Thomas Gerds, Yves Rosseel, Daniel Oberski, Jarrett Byrnes, Leonard Vanbrabant, Victoria Savalei, Ed Merkle, Michael Hallquist, Mijke Rhemtulla, Myrsini Katsikatsou, Mariska Barendse, Michael Chow, Terrence Jorgensen, W. L. Wallace Hui, Vyacheslav Lyubchich, Kimihiro Noguchi, Bhaskar Karambelkar, Kent Johnson, Vladimir Agafonkin, CloudMade, Leaflet contributors, Leaflet Providers contributors, Brandon Copeland, Jacob Toye, Joerg Dietrich, Benjamin Becquet, Norkart AS, L. Voogdt, Daniel Montague, Kartena AB, Robert Kajic, Mapbox, Michael Bostock, Alan Miller, Jim Albert, Rob Carnell, Giorgio Alfredo Spedicato, Reinhold Kainhofer, Kevin J. Owens, Ernesto Schirmacher, Gian Paolo Clemente, Jason Bryer, Steven Walker, Rune Haubo Bojesen Christensen, Bin Dai, Fabian Scheipl, Peter Green, Aki Vehtari, Juho Piironen, Michel Berkelaar, Jonathon Love, Dimitris Rizopoulos, Nikola Kasprikova, Vitalie Spinu, Garrett Grolemund, Imanuel Constigan, Jason Law, Doug Mitarotonda, Jonathan Boiser, Chel Hee Lee, Rahul Premraj, Richard A. Becker, Allan R. Wilks. Ray Brownrigg, Doug McIlroy, Thomas P Minka, Allan R. Wilks, Ray Brownrigg, Alex Deckmyn, Eric Archer, Nick Bearman, Hans-Jörg Bibiko, Steven Brey, Jonathan Callahan, German Carrillo, David Forrest, Patrick Giraudoux, Duncan Golicher, Virgilio Gómez Rubio, Patrick Hausmann, Karl Ove Hufthammer, Thomas Jagger, Sebastian Luque, Andrew Niccolai, Oscar Perpiñán Lamigueiro, Ben Stabler, Tim Appelhans, Florian Detsch, Christoph Reudenbach, Stefan Woellauer, Spaska Forteva, Thomas Nauss, Michael Sumner, Environmental Informatics Marburg, Adam November, Andrzej Oles, Vicent Marti, Natacha Porte, Tae Seung Kang, Sai Bhargav Yalamanchi, Mildenberger Thoralf, Deepak Yadav, Nacho Cordón Castillo, Vandit Jain, Eric Pante, Benoit Simon-Bouhet, P. Roebuck, Timothy A. Davis, Jason Riedy, Regents of the University of California, Frederick Novomestky, Ola Hossjer, Harris Jaffee, Dongcan Jiang, Peter Hickey, Timothy P. Jurka, Yoshimasa Tsuruoka, Andrew Finley, Sudipto Banerjee, Øyvind Hjelle, Adrian E. Raftery, Thomas Brendan Murphy, Michael Fop, Daniel Cook, Lyle Gurrin, Michal Figurski, Simon Wood, Maria Pittau, Yajuan Si, Jon Kropko, Stef van Buuren, Karin Groothuis-Oudshoorn, Alexander Robitzsch, Gerko Vink, Lisa Doove, Shahab Jolani, Rianne Schouten, Philipp Gaffert, Florian Meinfelder, Bernie Gray, Olaf Mersmann, Claudia Beleites, Rainer Hurling, Ari Friedman, Joshua M. Ulrich, Guy J. Abel, Timur V. Elzhov, Katharine M. Mullen, John C. Nash, Dai Feng, Patrick Applegate, Kim-Anh Le Cao, Florian Rohart, Ignacio Gonzalez, Sebastien Dejean, Benoit Gautier, Francois Bartolo, Pierre Monget, Jeff Coquery, FangZou Yao, Benoit Liquet, Derek Young, Tatiana Benaglia, Didier Chauveau, David Hunter, Ryan Elmore, Thomas Hettmansperger, Hoben Thomas, Fengjuan Xuan, Alan Genz, Adelchi Azzalini, Gemechis Dilba Djira, Mario Hasler, Daniel Gerhard, Frank Schaarschmidt, Fernando Miguez, YouGov, Christopher Jackson, Frank Bretz, Peter Westfall, Andre Schuetzenmeister, Susan Scheibe, Hans-Peter Piepho, Luciano Selzer with help from Sundar Dorai-Raj, Paul Bliese, Pablo J. Villacorta, Charlotte Wickham, Slawomir Jarek, Tetsuhisa Miwa, Xuefei Mi, Bjoern Bornkamp, David Pierce, J.J. Allaire, Peter Ellis, Kevin Kuo, B.W. Lewis, Jennifer Rogers, Charles Sese, CJ Yetman, Stefan Fritsch, Frauke Guenther, Marc Suling, Sebastian M. Mueller, Berend Hasselman, José Pinheiro, EISPACK authors, Siem Heisterkamp, Bert Van Willigen, Jelmer Ypma, Renaud Gaujoux, Cathal Seoighe, Hans Werner Borchers, Frank Hangler, Liz Sander, Jon Calder, Petr Simecek, Lampros Mouselimis, Sight Machine, Johannes Buchner, Oliver Keyes, Jan Peter Stotz, John C Nash, Thomas Schlesinger, Manuel J. A. Eugster, Yang Lu, David Kane, Jonathan McPherson, Dason Kurkiewicz, Keith Hughitt, Albert Wang, Carolin Strobl, Heidi Seibold, Philippe Grosjean, Frederic Ibanez, Michele Etienne, Zygmunt Zawadzki, Ulrich Halekoh, Peter Filzmoser, Heinrich Fritz, Klaudius Kalcher, Brian G. Peterson, Peter Carl, Kris Boudt, Ross Bennett, Joshua Ulrich, Eric Zivot, Dries Cornilly, Eric Hung, Matthieu Lestel, Hannes Riebl, Kyle Balkissoon, Gavin L. Simpson, Jari Oksanen, Raivo Kolde, Sergey Podobry, Tomislav Hengl, Michael Blaschek, Carson Sievert, Chris Parmer, Toby Hocking, Karthik Ram, Marianne Corvellec, Pedro Despouy, Plotly Technologies Inc, Javier Celigueta Muoz, Sander Oom, Eduardo Klein, Anupam Tyagi, Olivier

Eterradossi, Michael Toews, John Kane, Carl Witthoft, Julian Stander, Remko Duursma, Elisa Biancotto, Ofir Levy, Robby Engelmann, Michael Hecker, Felix Steinbeck, Hans Borchers, Ted Toal, Darshan Baral, Bjørn-Helge Mevik, Ron Wehrens, Kristian Hovde Liland, Nicole Kraemer, Mikio L. Braun, Anne-Laure Boulesteix, Paul H. Lee, Philip L. H. Yu, Seongho Kim, Gabor Csardi, Sindre Sorhus, Brecht Devleesschauwer, Paul Torgerson, Johannes Charlier, Bruno Levecke, Nicolas Praet, Sophie Roelandt, Suzanne Smit, Pierre Dorny, Dirk Berkvens, Niko Speybroeck, Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, Markus Müller, Stefan Siegert, Benoît Depaire, Felix Mannhardt, Thijs Beuving, Thomas A. Gerds, Louis Kates, James E. Helmreich . We are grateful to KuangNan Xiong, Simon Jackman, Alex Tahk, Christina Maimone, Jim Fearon, Zoe Meers, Petr Savicky, Jim Ramsey, Bruno Falissard, Peter Dalgaard, Jeffrey Gill, Stephan Weibelzahl, Aditya Anandkumar, Clay Ford, Robert Volcic, Helios De Rosario, Franz Rendl, Berwin A. Turlach, Thomas Roth, Raymond McTaggart, Clement Leung, Quandl Inc, Wouter Thielen, Paul Teetor, Roger Koenker, Stephen Portnoy, Pin Tian Ng, Philip Grosjean, Brian D Ripley, François Briatte, Pedro Canadilla, Andy Jacobson, Martin Schlather, Alexander Malinowski, Marco Oesting, Daphne Boecker, Kirstin Strokorb, Sebastian Engelke, Johannes Martini, Felix Ballani, Olga Moreva, Jonas Auel, Peter Menck, Sebastian Gross, Ulrike Ober, Christoph Berreth, Katharina Burmeister, Juliane Manitz, Paulo Ribeiro, Richard Singleton, Martin Kroll, Leo Breiman, Adele Cutler, Andy Liaw, Matthew Wiener, D. Labes, Diethelm, M. Matsumoto, M. Saito, the Knuth-TAOCP RNG from D. Knuth, Samuel E. Kemp, R Core, Sridhar Ratnakumar, Trent Mick, ActiveState, Eddy Petrisor, Jiali Mei, Jonathan A. Greenberg, Oscar Perpinan Lamigueiro, Andrew Bevan, Etienne B. Racine, Ashton Shortridge, Aniruddha Ghosh, Graham Williams, Mark Vere Culp, Ed Cox, Anthony Nolan, Denis White, Daniele Medri, Akbar Waljee, Jose Magana, Surendra Tipparaju, Durga Prasad Chappidi, Dinesh Manyam Venkata, Mrinal Chakraborty, Fang Zhou, Wacek Kusnierczyk, Milan Bouchet-Valat, Liviu Andronic, Michael Ash, Theophilius Boye, Stefano Calza, Andy Chang, Kosar Karimi Pour, G. Jay Kerns, Renaud Lancelot, Matthieu Lesnoff, Samir Messad, Robert Muenchen, Erich Neuwirth, Dan Putler, Miroslav Ristic, Kevin Wright, Fox John, John Chambers, Doug Bates and Binxiang Ni, and Yixuan Qiu, Gregory Vandenbrouck, Marcus Geelnard, Intel, CRAN team, Gunnlaugur Thor Briem, Jukka Jylänki, Mikkel Jørgensen, Jan Marvin Garbuszus, Magnus Thor Torfason, Luke M. Olson, Giovanni Righi, Marcin Kalicinski, Komarov Valery, Christophe Leitienne, Bob Colbert, David Hoerl, Andrew Geisler, Wouter van Atteveldt, Arthur Tenenhaus, Vincent Guillemot, Robert Hijmans, Even Rouault, Colin Rundel, Rainer Stuetz, George Vega Yon, Stefan Theussl, Florian Schwendinger, Heinrich Schuchardt, Markus Loecher, Junji NAKANO, Alexey Stukalov, Matt Denwood, Alex Couture-Beil, Matthew Bloch, Jeff Allen, Roy Storey, Ruben Arslan, jQuery UI contributors, Alexander Farkas, Scott Jehl, Ivan Sagalaev, Greg Franko, Google Inc, Dave Raggett, W3C, Dave Gandy, Ben Sperry, Drifty, Aidan Lister, Bernhard Pfaff, F. Panneton, P. L'Ecuyer, Daniel C. Bowman, Christophe Croux, Valentin Todorov, Andreas Ruckstuhl, Matias Salibian-Barrera, Tobias Verbeke, Maria Anna di Palma, Tobias Sing, Oliver Sander, Niko Beerenwinkel, Thomas Lengauer, Michael Lapsley, Andrzej Janusz, Dominik Slezak, Chris Cornelis, Sebastian Stawicki, Peter Danenberg, Manuel Eugster, Bowman, Gibson, Crawford, Jake Anderson, Yasuyuki Okumura, Khanh Nguyen, Terry Leitch, Alexander Brenning, Donovan Bangs, Marc Becker, José M. Benítez, Andreas Zell, Niels Mache, Günter Mamier, Michael Vogt, Sven Döring, Ralf Hübner, Kai-Uwe Herrmann, Tobias Soyez, Michael Schmalzl, Tilman Sommer, Artemis Hatzigeorgiou, Dietmar Posselt, Tobias Schreiner, Bernward Kett, Martin Reczko, Martin Riedmiller, Mark Seemann, Marcus Ritt, Jamie DeCoster, Jochen Biedermann, Joachim Danz, Christian Wehrfritz, Patrick Kursawe, Andre El-Ama, Alexios Ghalanos, Gael Guennebaud, Jitse Niesen, David A. James, Seth Falcon, SQLite Authors, Liam Healy, R Consortium, Daniel Lee, Krzysztof Sakrejda, Trustees of Columbia University, Jens Oehlschlaegel-Akiyoshi, Joel de Guzman, John Fletcher, Thomas Heller, Eric Niebler, Gary Ritchie, Reinhard Harter, Cyrille Szymanski, G Grothendieck, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, Andrew Weeks, Frank M. Siegert, Mark Heath, Thomas Henlick, Sergey Babkin, Turgut Uyar, Rihardas Hepas, Szalay Tamas, Johan Vromans, Petr Titera, Lei Wang, Chen Xiangyang, Zvezdan Petkovic, Rigel, I. Lee Hetherington, Michael W. Kearney, Maxime Hervé, Francois Brunetti, University of Waikato, Dominik Wabersich, Andy South, Yves Tillé, Susanne Berger, Nathaniel Graham, Hanna Meyer, Sarah Schnackenberg, Marlene Marchena, Ryan M. Acton, Vito M. R. Muggeo, Simon Potter, Simon Sapin, Ian Bicking, Zhenghua Nie, Michael Culbertson, Benjamin Goodrich, Richard H. Jones, Adam Kramer, Quinn Weber, Gilles Pujol, Alexandre Janon, Khalid Boumhaout, Sebastien Da Veiga, Thibault Delage, Jana Fruth, Laurent Gilquin, Joseph Guillaume, Loic Le Gratiet, Paul Lemaitre, Barry L. Nelson, Filippo Monari, Roelof Oomen, Bernardo Ramos, Eunhye Song, Jeremy Staum, Taieb Touati, Frank Weber, Delphine Charif, Olivier Clerc, Carolin Frank, Anamaria Necsulea, Leonor Palmeira, Simon Penel, Guy Perrière, Michael Brusco, Stephanie Stahl, Hans-Friedrich Koehn, Ian Cook, Robin Lovelace, Etienne Racine, Werner Stahel, Christian Keller, Kjetil Halvorsen, Alain Hauser, Christoph Buser, Lorenz Gygax, Isabelle Flückiger, Marcel Wolbers, Markus Keller, Sandrine Dudoit, Jane Fridlyand, Henrik Aa. Nielsen, Vincent Carey, Frédéric Ibanez, Caterina Savi, Charles Geyer, Ian L. Dryden, Stefan Petre, Andrew Rowls, John Fraser, John Gruber, Barbara Borges Ribeiro, Almasaeed Studio, Adobe Systems Incorporated, Stan Development Team, Michael Andreae, Michael Betancourt, Bob Carpenter, Yuanjun Gao, Dongying Song, Rob Trangucci, Thomas Park, Lukasz Dziedzic, Nathan Willis, Google Corporation, Matt McInerney, Canonical Ltd, Gerhard Burger, Paul Kienzle, David Billinghurst, Hans-Werner Borchers, Andre Carezia, Pascal Dupuis, John W. Eaton, E. Farhi, Kai Habel, Sebastian Krey, Bill Lash, Paulo Neis, Jaakko Ruohio, Julius O. Smith III, Doug Stewart, Emilio L. Cano, Javier M. Moguerza, Mariano Prieto, Andrés Redchuk, Adrian Bowman, B. D. Ripley, Sundar Dorai-Raj, Virgilio Gomez-Rubio, Josh O'Brien, Joseph O'Rourke, Tom Gottfried, Daniel Gerber, Youcef Saad, Esmond G. Ng, Barry W. Peyton, Joseph W.H. Liu, Alan D. George, Alexander Kowarik, Bernhard Meindl, Kasper Klitgaard Berthelsen, Ottmar Cronie, Yongtao Guan, Ute Hahn, Abdollah Jalilian, Marie-Colette van Lieshout, Greg McSwiggan, Tuomas Rajala, Suman Rakshit, Dominic Schuhmacher, Rasmus Waagepetersen, Hangsheng Wang, M. Adepeju, C. Anderson, Q.W. Ang, M. Austenfeld, S. Azaele, M. Baddeley, C. Beale, M. Bell, R. Bernhardt, T. Bendtsen, A. Bevan, B. Biggerstaff, A. Bilgrau, L. Bischof, C. Biscio, R. Bivand, J.M. Blanco Moreno, F. Bonneu, J. Burgos, S. Byers, Y.M. Chang, J.B. Chen, I. Chernayavsky, Y.C. Chin, B. Christensen, J.-F. Coeurjolly, K. Colyvas, R. Constantine, R. Corria Ainslie, R. Cotton, M. de la Cruz, P. Dalgaard, M. D'Antuono, S. Das, T. Davies, P.J. Diggle, P. Donnelly, I. Dryden, S. Eglen, A. El-Gabbas, B. Fandohan, O. Flores, E.D. Ford, P. Forbes, S. Frank, J. Franklin, N. Funwi-Gabga, O. Garcia, A. Gault, J. Geldmann, M. Genton, S. Ghalandarayeshi, J. Gilbey, J. Goldstick, P. Grabarnik, C. Graf, U. Hahn, A. Hardegen, M.B. Hansen, M. Hazelton, J. Heikkinen, M. Hering, M. Herrmann, P. Hewson, K. Hingee, K. Hornik, P. Hunziker, J. Hywood, R. Ihaka, C. Icos, A. Jammalamadaka, R. John-Chandran, D. Johnson, M. Khanmohammadi, R. Klaver, P. Kovesi, L. Kozmian-Ledward, M. Kuhn, J. Laake, F. Lavancier, T. Lawrence, R.A. Lamb, J. Lee, G.P. Leser, H.T. Li, G. Limitsios, A. Lister, B. Madin, M. Maechler, J. Marcus, K. Marchikanti, R. Mark, J. Mateu, P. McCullagh, U. Mehlig, F. Mestre, S. Meyer, X.C. Mi, L. De Middeleer, R.K. Milne, E. Miranda, J. Moller, M. Moradi, V. Morera Pujol, E. Mudrak, G.M. Nair, N. Najari, N. Nava, L.S. Nielsen, F. Nunes, J.R. Nyengaard, J. Oehlschlaegel, T. Onkelinx, S. O'Riordan, E. Parilov, J. Picka, N. Picard, M. Porter, S. Protsiv, A. Raftery, S. Rakshit, B. Ramage, P. Ramon, X. Raynaud, N. Read, M. Reiter, I. Renner, T.O. Richardson, B.D. Ripley, E. Rosenbaum, B. Rowlingson, J. Rudokas, J. Rudge, C. Ryan, F. Safavimanesh, A. Sarkka, C. Schank, K. Schladitz, S. Schutte, B.T. Scott, O. Semboli, F. Semecurbe, V. Shcherbakov, G.C. Shen, P. Shi, H.-J. Ship, T.L. Silva, I.-M. Sintorn, Y. Song, M. Spiess, M. Stevenson, K. Stucki, M. Sumner, P. Surovy, B. Taylor, T. Thorarinsdottir, L. Torres, B. Turlach, T. Tvedebrink, K. Ummer, M. Uppala, A. van Burgel, T. Verbeke, M. Vihtakari, A. Villers, F. Vinatier, S. Voss, S. Wagner, H. Wang, H. Wendrock, J. Wild, C. Witthoft, S. Wong, M. Woringer, M.E. Zamboni, A. Zeileis, Ege Rubak, Jakub Nowosad, Micah Altman, Luc Anselin, Renato Assunção, Olaf Berke, Andrew Bernat, Guillaume Blanchet, Eric Blankmeyer, Marilia Carvalho, Bjarke Christensen, Yongwan Chun, Carsten Dormann, Virgilio Gómez-Rubio, Martin Gubri, Rein Halbersma, Elias Krainski, Hongfei Li, Jielai Ma, Abhirup Mallik, Werner Mueller, Pedro Peres-Neto, Gianfranco Piras, Markus Reder, Michael Tiefelsdorf, Danlin Yu, Peter Diggle, Giovanni Petris, Rmetrics core team members, Joshua Pritikin, Matt Hoffman, Marcus Brubaker, Peter Li, Allen Riddell, Marco Inacio, Mitzi Morris, Rob Goedman, Brian Lau, Alp Kucukelbir, Robert Grant, Dustin Tran, Michael Malecki, Lawrence Livermore National Security, The Regents of the University of California, Southern Methodist University, Marcello D'Orazio, Gordon Smyth, Yifang Hu, Peter

Dunn, Belinda Phipson, Yunshun Chen, Marek Gagolewski, Bartek Tartanus, IBM, Bruce Hansen, Edgar C. Merkle, Kerry Rodden, Florian Breitwieser, Terry M Therneau, T Jake Luciani, Matthieu Decorde, Vaudor Lise, Håkon Malmedal, Arne Henningsen, Jeff D. Hamann, Ingo Feinerer, Jonathan Rougier, Nick Golding, Johannes Rauch, Masayuki Tanaka, Shaun Bowe, Materialize, Yuxi You, Kevin Decker, Rodrigo Fernandes, Yauheni Pakala, Three.js authors, Andreas Briese, B. Thieurmel, Kyle Walker, Joe W. Byers, Almende B.V, Jeff Hallman, Artifex Software Inc, Martijn Tennekes, Joel Gombin, Richard Zijdeman, John Clouse, Stefan Wilhelm, Thorsten Pohlert, Heike Trautmann, Detlef Steuer, Björn Bornkamp, Blake LeBaron, Peter K. Dunn, Jeff Gentry, Hans Bruun Nielsen, Stig Bousgaard Mortensen, James Hiebert, Thomas Mailund, Tomasz Kalinowski, Patrick O. Perry, Unicode Inc, F. Guillaume Blanchet, Roeland Kindt, Dan McGlinn, Peter R. Minchin, R. B. O'Hara, M. Henry H. Stevens, Eduard Szoecs, Helene Wagner, Adrian Dusa, Hanbo Chen, Lee Wilkinson, Thomas W. Yee, Bernd Prantner, Simon Garnier, Marco Sciaini, Cédric Scherer, Titouan Robert, James P. Howard, II, Vincent Arel-Bundock, Francois Guillem, Nicolas Perriault, John McNamara, Mirai Solutions GmbH, Martin Studer, The Apache Software Foundation, Graph Builder, Stephen Colebourne, Adrian A. Dragulescu, The Apache Software Foundation Apache Commons Codec, MetaStuff Ltd, James Hester, David B. Dahl, Nathaniel Phillips, Nathan Teetor, Kuba Podgórski, Rich Geldreich, Felix Andrews, Jens Oehlschlégel, JoséPinheiro, Victor Perrier, Shinichiro Tomizono, Przemyslaw Biecek, Frank E Harrell, Duncan Garmonsway, Yachen Yan, Timothée Giraud, Nicolas Lambert, Ian Fellows, Robert Garrett, Thomas Fisher, Hans Peter Wolf, Robert Hickman, Kameron Penn, Jack Schmidt, Karolis Koncevičius, Enrico Schumann, Jayanth Varma, Liam Revell, Indrajeet Patil, Subhash R. Lele, Jonah L. Keim, Peter Solymos, Sean Kross, Tanguy Lucci, Shivam P., George Pipis…

Dons

Ce livre fait partie des livres compagnons du compendium de mathématiques théorique *Opera Magistris* d'environ 7'000 pages qui est lui aussi gratuit.



et ainsi des livres gratuits équivalents sur Minitab et MATLAB.

Afin de poursuivre la rédaction de ce compendium et continuer à faire évoluer ce livre gratuit sur R vos dons sont la bienvenue et nous vous en remercions d'avance.

Vous pouvez faire un don via Tipeee:

https://www.tipeee.com/elements-of-applied-mathematics

Merci d'avance!

Histoire de R

Voici quelques dates majeures concernant le logiciel R:

- 1992: Le développement de R comment en tant que projet de recherché à Auckland, (Nouvelle-Zélande) par Robert Gentleman et Ross Ihaka

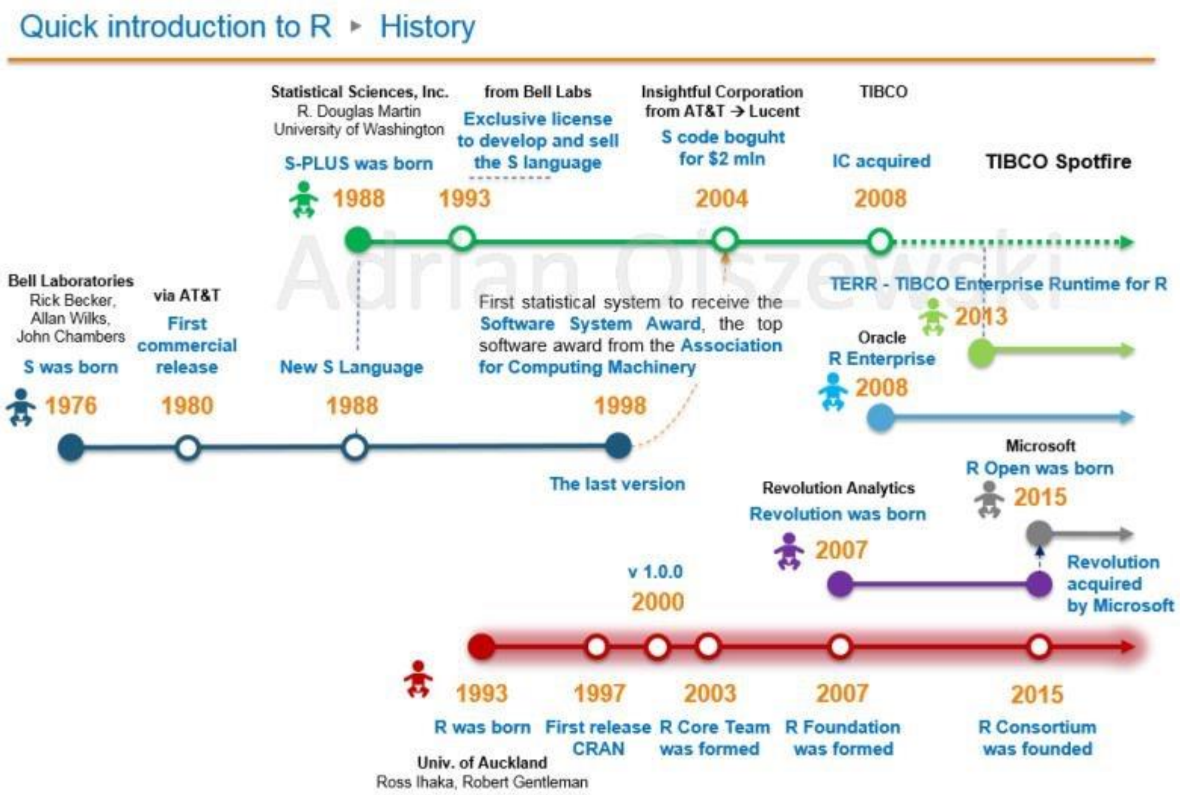


- 1993: Publication des premiers fichiers binaires à Statlib
- 1995: R est distribué sous la licence logiciel open-source, GPL2

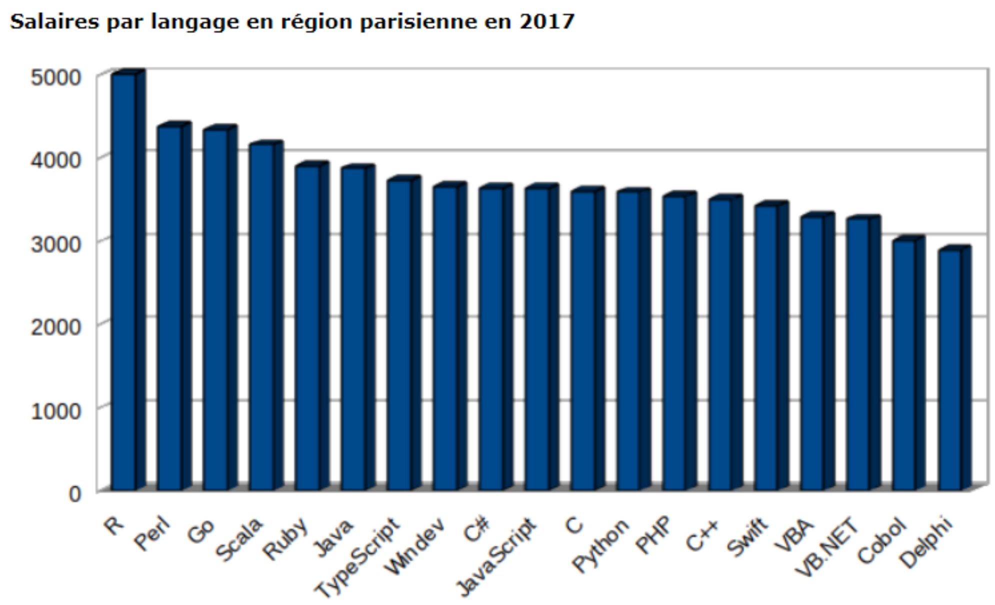
- 1997: Composition de l'équipe "R core"
- 1997: L'association CRAN est fondée (par Kurt Hornik et Fritz Leisch)
- 1999: Le site internet R *r-project.org* est créé
- 2000: Publication de R 1.0.0 (29 février)
- 2001: Création de R News (qui deviendra plus tard le Journal R)
- 2003: La fondation R est créée
- 2004: Première conférence UseR! (à Vienne)
- 2004: Publication de R 2.0.0
- 2009: Première édition du R Journal
- 2011: 14 Novembre, publication de la version 0.1 de ce livre
- 2013: Publication de R 3.0.0

- 2015: Fondation du Consortium R avec la participation de la fondation R. Dépassement des 10'000 packages.

- 2016: Nouveau logo R

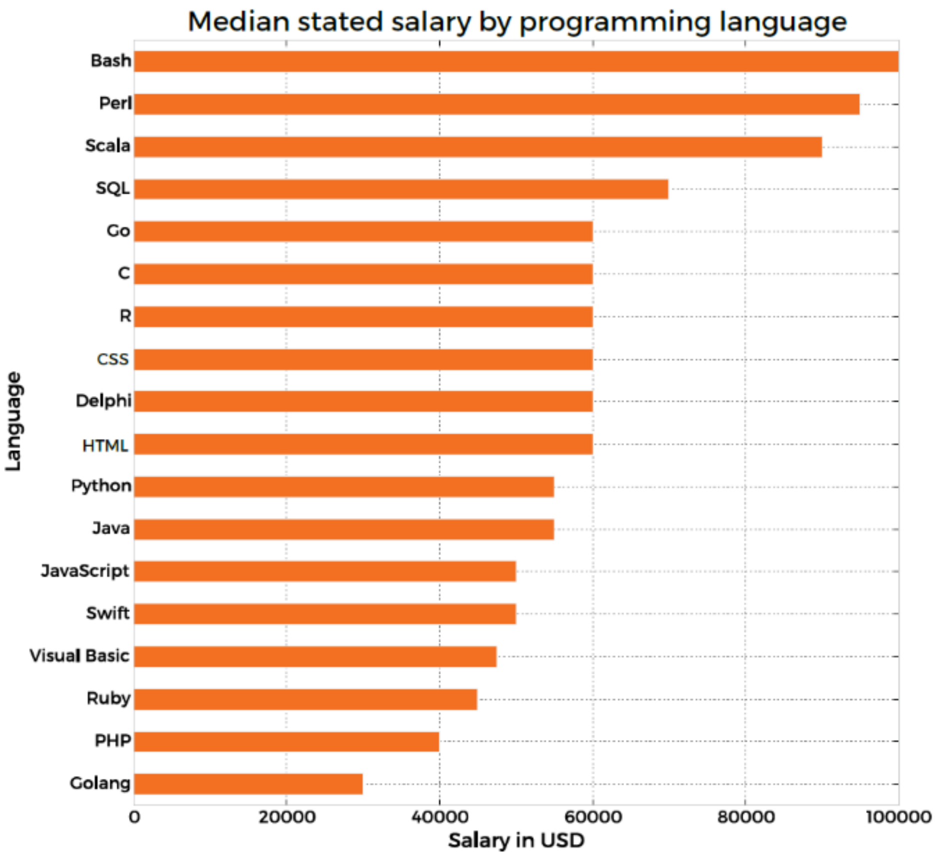


Enfin signalons quelques informations concernant R. D'abord son ranking dans les salaires à Paris (France):



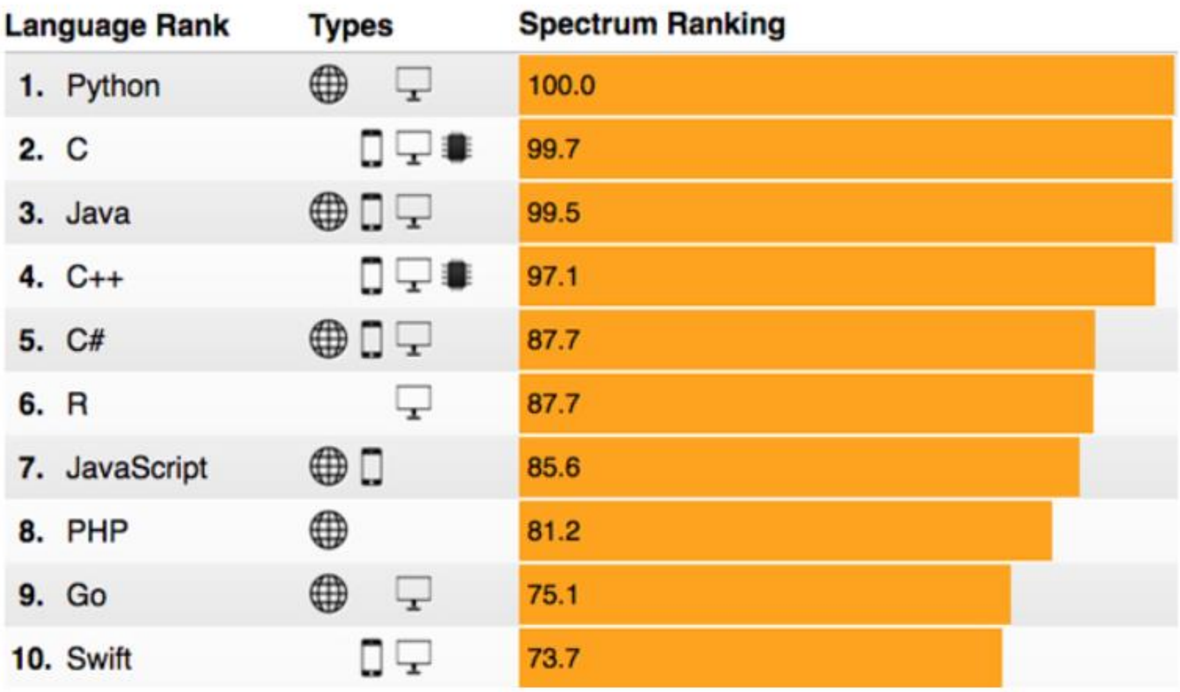
source: https://emploi.developpez.com/actu/199863/Emploi-developpeur-2017-les-langages-les-plus-demandes-et-les-mieux-payes-Java-JavaScript-et-PHP-plus-demandes-mais-Perl-Go-et-Scala-mieux-payes/

Ce qui équivaut en Suisse à un salaire au minimum de 15'000 CHF/mois! Voici l'équivalent aux USA:



source: https://adtmag.com/blogs/dev-watch/2016/07/bash-salaries.aspx

Et enfin le fameux ranking Spectrum:



source: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Liens Internet

- http://www.sciences.ch (le site Internet compagnon avec les démonstrations mathématiques)
- http://www.r-project.org/
- http://www.rstudio.com/
- https://www.visualstudio.com
- http://cran.univ-lyon1.fr/index.html
- http://cran.r-project.org/web/packages/
- http://www.itl.nist.gov/div898/handbook/
- http://www.cookbook-r.com
- http://www.rcommander.com
- http://rgraphgallery.blogspot.ch/
- http://statcheck.io/
- http://shiny.rstudio.com/
- https://radiant-rstats.github.io/docs/
- https://plot.ly/

Forums

- http://www.talkstats.com
- http://statistiques.forumpro.fr
- http://www.les-mathematiques.net
- http://www.developpez.net/forums/f1179/autres-langages/autres-langages/r/
- http://forums.cirad.fr/logiciel-R/
- https://pubpeer.com/

Graphiques

- https://www.r-graph-gallery.com/all-graphs/
- https://plot.ly/r/
- https://exts.ggplot2.tidyverse.org/gallery/
- https://shiny.gmw.rug.nl/ggplotgui/

YouTube

- https://www.youtube.com/@StatisticsGlobe/videos

Solutions de Calcul Distribué

- http://www.techilatechnologies.com

- https://www.microsoft.com/en-us/cloud-platform/r-server

Blogs

- http://freakonometrics.hypotheses.org

Sur l'aberration de gérer du business de haut niveau avec des tableurs

- http://staffweb.cms.gre.ac.uk/~cd02/EUSPRIG/

Data

- https://opendata.swiss
- https://github.com/rnckp/awesome-ogd-switzerland
- http://ec.europa.eu/eurostat/data/database
- http://www.quandl.com
- https://data.gov.uk
- http://www.data.gov
- http://www.census.gov/data.html
- http://www.opendatanetwork.com
- http://open-data.europa.eu/en/data/
- http://wiki.dbpedia.org/
- http://dataportals.org/
- http://www.healthdata.gov/
- https://azure.microsoft.com/en-us/marketplace/
- http://www.gapminder.org/data/
- http://finance.yahoo.com/
- http://archive.ics.uci.edu/ml/index.html
- http://www.ncdc.noaa.gov/data-access/quick-links#loc-clim

Quiz/Exemples

- http://www.scientific-evolution.com/qcm/fr/start_session/2a6fca7473/
- http://www.r-exercises.com

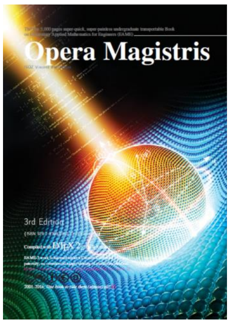
R Shiny et TensorFlow

- http://deanattali.com/blog/advanced-shiny-tips/
- https://tensorflow.rstudio.com/learn/examples.html
- https://www.shinyproxy.io/
- https://github.com/merlinoa/shinyFeedback

Médiagraphie

Voici une liste de livres ou vidéos d'une qualité pédagogique et de rigueur remarquable que nous avons eue la chance d'avoir entre les mains et nous recommandons l'acquisition. Nous en avons lu beaucoup d'autres mais qui sont tellement mauvais qu'ils ne valent pas la peine d'être mentionnés.

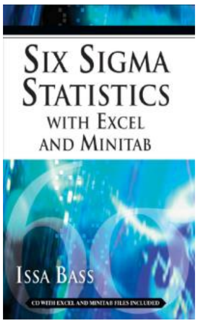
Le lecteur aura donc compris que je recommande très fortement de compléter la lecture du présent e-book (non exhaustif concernant R!) par la liste de lecture ci-dessous:



Opera Magistris / 7'950 pages / Editions Sciences.ch / Vincent ISOZ / 3rd édition
ISBN: 978283999327

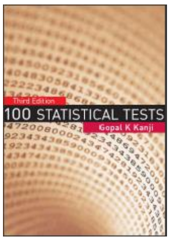
Commentaire: Équivalent du livre en français ci-dessus mais en anglais, bien plus complet et de qualité graphique considérablement meilleure!

Attention!!! Pendant les séminaires et conférences de moins ou égal à 3 jours je prends oralement et délibérément des raccourcis théoriques dangereux voire parfois partiellement faux délibérément à cause du manque de temps pour aller dans les détails.



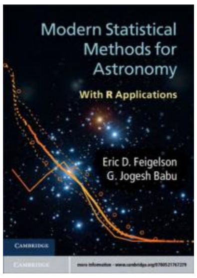
Six Sigma Statistics With Excel and Minitab / 386 pages / Éditions McGrawHill / Issa BASS
ISBN: 9780071542685

Commentaire: C'est avec ce livre que j'ai été initié à Minitab lors de ma formation Green Belt chez mon deuxième employeur. Il n'y a aucune démonstration mathématique et très peu d'équations mais les exemples donnés ont une orientation industrielle très utile et cela avec un clarté pédagogique remarquable



100 Statistical tests / 257 pages / Éditions SAGE / Gopal K. KANJI
ISBN: 101412923751

Commentaire: Un "must have" listant 100 tests statistiques avec les hypothèses d'utilisation, les limites, les formules (sans démonstration) et un exemple à chaque fois.



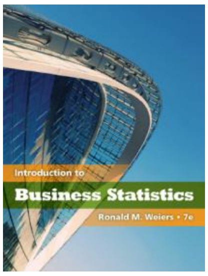
Modern Statistical Methods for Astronomy – with R applications / 504 pages / Éditions Cambridge / Eric D. Feigelson, G. Jogesh Babu
ISBN: 101412923751

Commentaire: Un petit bijou pour ceux qui sont intéressés à l'application des techniques de Data Science/Data Mining à l'astronomie sur des jeux de données et des cas réels dans le domaine!



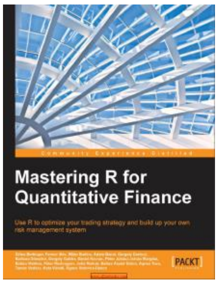
Applied Multivariate Statistics for the social sciences / 663 pages / Éditions Routledge / James P. STEVENS
ISBN: 9780805859010

Commentaire: Livre un peu vieillissant mais contient des exemples très détaillés avec étapes de calcul de test multivariés complexes, ce qui est difficile à trouver dans la littérature. Par contre, il n'y a aucune démonstration détaillée.



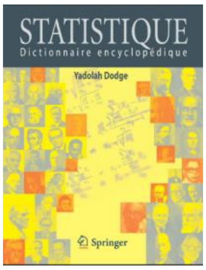
Business Statistics / 892 pages / Éditions Routledge / Ronald M. WEIERS / 7ème édition
ISBN: 053845217X

Commentaire: Le livre !!! à mettre entre les mains à ceux qui commencent à étudier les statistiques et qui cherchent un ouvrage sans démonstrations mathématiques mais présentant de façon très claire et pédagogique des tests jusqu'au niveau du Master et accompagnés à chaque fois de façon détaillée de Microsoft Excel et Minitab.



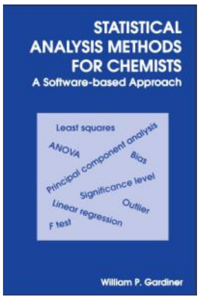
Mastering R for Quantitative Finance / 362 pages / Éditions Packt / Edina Berlinger, Ferenc Illés, Milán Badics, Ádám Banai, Gergely Daróczi, Barbara Dömötör, Gergely Gabler, Dániel Havran, Péter Juhász, István Margitai, Balázs Márkus, Péter Medvegyev, Julia Molnár, Balázs Árpád Szűcs, Ágnes Tuza, Tamás Vadász, Kata Váradi, Ágnes Vidovics-Dancs
ISBN: 9781783552078

Commentaire: Bon que dire… On ne peut que conclure que faire intervenir presque 20 auteurs sur un livre ne permet pas toujours un résultat cohérent et de qualité. On remarque parfois qu'il y a beaucoup de remplissage avec du bla bla bla… On note aussi suite à la lecture, la différence de style entre les différentes auteurs… entre ceux qui utilisent zéro packages, et ceux qui font que d'en utiliser… Bref pas brillant mais quelques pages sont toutefois intéressantes (peut-être une vingtaine sur les 362…).



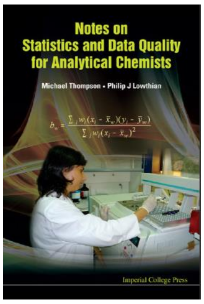
STATISTIQUES: Dictionnaire encyclopédique / 614 pages / Éditions Springer / Yadolah DODGE
ISBN: 9782287720932

Commentaire: Un petit dictionnaire très utile contenant un peu tout de façon très condensée sur tout le domaine des statistiques. Très pratique pour retrouver des définitions qui génèrent des débats en entreprise.



Statistical analysis methods for chemists / 390 pages / Éditions The Royal Society of Chemistry / William P. GARDNIER
ISBN: 085404549X

Commentaire: Un livre vieillissant sans démonstrations mathématiques mais qui intéressera les chimistes utilisant Minitab et qui veulent apprendre une petite partie du logiciel avec des exemples familiers à leur domaine d'activité.



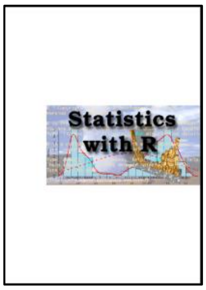
Statistical analysis methods for chemists / 260 pages / Éditions Imperial College Presse / Michael THOMPSON, Philip J. LOWTHIAN
ISBN: 1848166168

Commentaire: Uniquement théorique, mais sans démonstrations mathématiques ni exemples ou exercices il constitue un bon complément à l'ouvre proposé juste précédemment qui était aussi destiné aux chimistes.



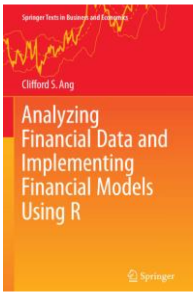
GGPlot2 / 118 pages / PDF gratuit sur Internet / Daname Kolani
ISBN: -

Commentaire: À ce jour le meilleur livre sur GGPlot2 toutes langues confondues et l'unique en Français sur le sujet. Contrairement au livre du créateur originel du package, ce PDF est compréhensible et progresse pas à pas de manière structurée. Lecture recommandée à tous ceux qui veulent faire du R Shiny!



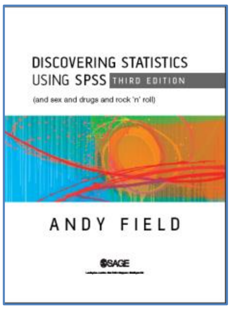
Statistics with R / 1266 pages / PDF gratuit sur Internet / Vincent Zoonekynd
ISBN: -

Commentaire: Un magnifique e-book extrêmement complet sur de nombreux domaines d'applications de R. C'est selon mon opinion la bible R en anglais à ce jour (même plus complet que les ouvrages payants qui se disent être des bibles). Je lui suis redevable de nombreux emprunts et cela restera une référence!



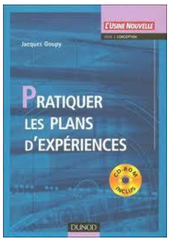
Analyzing Financial Data and Implementing Financial Models Using R/ 360 pages / Clifford S. Ang
ISBN: 9783319140742

Commentaire: Bon… je n'ai pas aimé du tout. Cela manque cruellement d'équations mathématiques pour expliquer les concepts. Les exemples ne sont même pas vraiment techniquement avancés. Bref je l'ai mis ici parce que parfois on me demande ce que j'en pense…



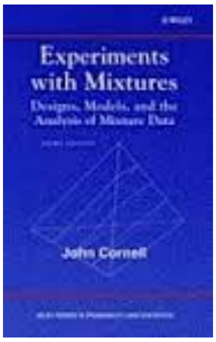
Discovering Statistics using SPSS / 854 pages / Éditions Sage / Andy Field / 3ème édition
ISBN: 9781847879066

Commentaire: Comme je le dis toujours au début des cours: dans un contexte scientifique avoir seulement un logiciel d'analyse est insuffisant ne serait-ce que pour se couvrir contre les bugs et les variantes d'algorithmes. Dès lors, un deuxième outil complémentaire et bien supérieur à Minitab pour la partie statistique est SPSS! L'auteur de ce livre est remarquable! Outre le fait qu'il ait aussi écrit un ouvrage de la même qualité sur *R* il a beaucoup d'humour, un sens aigu de la pédagogie et sait particulièrement bien présenter des outils informatiques avec des exemples provenant de données réelles! Un must have absolu!



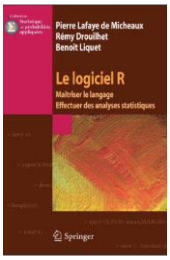
Pratiquer les plans d'expériences / 551 pages / Éditions Dunod / Jacque Goupy
ISBN: 2100042173

Commentaire: Comme je le dis toujours au début des cours: dans un contexte scientifique avoir seulement un logiciel d'analyse est insuffisant ne serait-ce que pour se couvrir contre les bugs et les variantes d'algorithmes. Dès lors, un deuxième outil complémentaire à Minitab pour les plans d'expérience est JMP! Et ce livre est d'un niveau de qualité remarquable tant par le niveau de détail des exemples industriels qui y sont donnés étape par étape que par la clarté des explications et de la présence parfois de quelques démonstrations mathématiques accompagnées de nombreuses figures. De plus, l'auteur est connu dans tout le monde francophone est un consultant dont la renommée n'est plus à faire.



A primer on experiments with mixture / 368 pages / Éditions Wiley / John CORNELL
ISBN: 0470643382

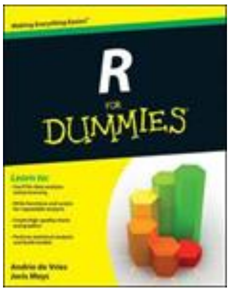
Commentaire: Il est très difficile de trouver de la littérature détaillée et rigoureuse sur le sujet des plans de mélange ("mixture design" en anglais) mais John Cornell réussit ici un coup de maître. Son ouvrage est complet, rigoureux, détaillé, fourni d'exemple pertinents! On regrettera peut-être juste le fait qu'il ne fasse pas le parallèle avec un logiciel informatique de son choix mais bon... Achat recommandé!



Le logiciel R / 527 pages / Éditions Springer / Pierre LAFAYE DE MICHEAUX + Remy DROUILHET + Benoit LIQUET
ISBN: 9782817801148

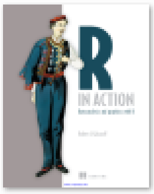
Commentaire: Un ouvrage sur R d'une qualité pédagogique remarquable. Il pourrait intéresser les élèves de Terminale et de 1er

cycle. La structure est excellente et des trucs et astuces ainsi que des avertissements sont donnés tout au long de l'ouvrage.



R for dummies/ 406 pages / Éditions For Dummies / Joris MEYS + Andrie DE VRIES
ISBN: 9781119962847

Commentaire: Non ce n'est pas une blague. Curieusement ce livre traite de sujets de base qui manquent dans de nombreux livres considérés comme des références et qui sont pourtant souvent demandés par les employés en entreprise. Ce n'est rien que pour cette raison que je le conseille!



R in Action/ 474 pages / Éditions MANNING/ Robert I. KABACOFF
ISBN: 9781935182399

Commentaire: Un petit livre bien sympathique pour début sur R. À lire peut-être après avoir parcouru R for Dummies. Mais l'achat en vaut la peine, il y a quelques bonnes infos et le site web associé est très complet.



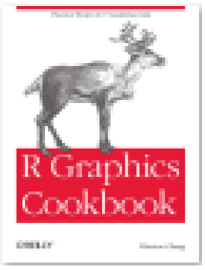
The R Book / 951 pages / Éditions Wiley/ Michael J. KRAWLEY
ISBN: 9780470510247

Commentaire: L'ouvrage se fait un peu vieux (manque de couleurs et de captures d'écrans) mais il est relativement complet en ce qui concerne les tests statistiques traditionnels surtout dans les domaines de l'analyse de la variance (par contre il ne traite pas de finance ou de data mining).



Comprendre et réaliser des tests statistiques à l'aide de R / 695 pages / Éditions de Boeck / Gaël Millot
ISBN: 9782804101794

Commentaire: À ma connaissance c'est l'ouvrage en français le plus complet sur R relativement aux tests statistiques. Plus complet que *Le logiciel R* présenté plus haut mais mois complet que le présent support électronique (qui est en français pour rappel...).



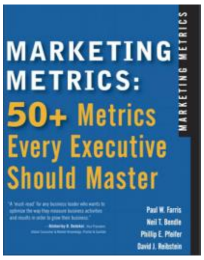
R Graphics Cookbook / 413 pages / Éditions O'Reilly/ Winston Chang
ISBN: 9781449316952

Commentaire: Un bon livre pour se familiariser en douceur avec la package ggplot2. Les forums et le web en général sont cependant devenus bien plus complets que cet ouvrage qui perd alors un peu de son intérêt (une deuxième édition avec deux ou trois fois plus de pages en relancerait l'intérêt).



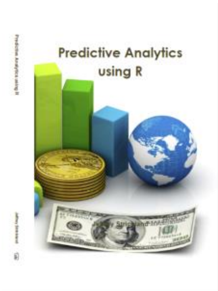
Biostatistical Design and Analysis Using R / 577 pages / Éditions Wiley-Blackwell/ Murray Logan
ISBN: 97814433524

Commentaire: Quelques bons éléments à tirer particulièrement au niveau des analyses de la variance. Bon si on retire les notions de base de R le livre se réduit en réalité à 200 pages...



Marketing Metrics / 432 pages / Éditions Pearson FT Press / Paul W. Farris, Neil T. Bendle, Phillip E. Pfeifer, David J. Reibstein
ISBN: 0137058292

Commentaire: Suite à la demande d'une grosse entreprise suisse dont un responsable Marketing ne savait pas ce dont il pouvait mesurer en Marketing… Évidemment, il ne faut pas chercher en français sur Google pour trouver ce genre de livre.



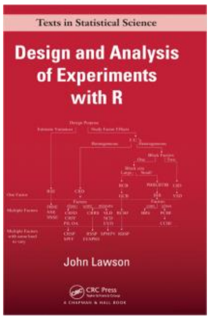
Predictive Analytics using R / 553 pages / LuLu Press / Jeffrey Strickland
ISBN: 9781312841017

Commentaire: Suite à la demande d'une grosse entreprise suisse dont un responsable Marketing ne savait pas ce dont il pouvait mesurer en Marketing… Évidemment, il ne faut pas chercher en français sur Google pour trouver ce genre de livre (depuis que ces lignes ont été écrite, le livre francophone ci-dessous a été écrit!!!).



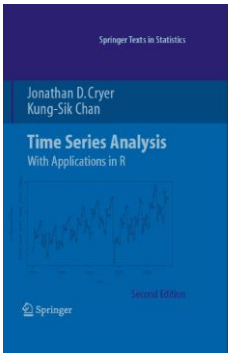
Data Marketing - Statistiques appliquées au marketing avec Excel et R / 262 pages / Éditions ENI / Magali Trelohan
ISBN: 9782409014857

Commentaire: Cet ouvrage s'adresse aux managers marketing et commerciaux soucieux de développer une stratégie de data marketing au sein de leur organisation (entreprise, service public ou association). Il aborde de manière simple et pragmatique l'intérêt de la data marketing et explique comment la mettre en oeuvre au quotidien à l'aide d'Excel et de R.



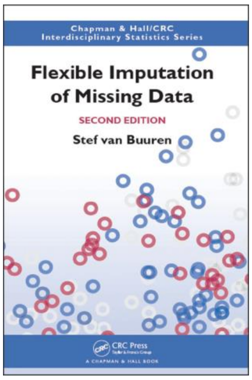
Design and Analysis of Experiments with R / 618 pages / Éditions CRC Press / John Lawson
ISBN: 9781498728485

Commentaire: Un livre, ou plutôt devrait-t-on parler de "bible", d'une qualité exceptionnelle tant il parcourt de manière exhaustive de nombreux plans d'expérience. On regrettera l'absence de démonstration mathématiques, mais l'aspect pratique avec R est très détaillé et permet de rentabiliser rapidement le l'achat du livre pour tout praticien travaillant dans un laboratoire.



Time Series Analysis with application in R / 500 pages / Éditions Springer / Johnathan D. Cryer et Kung-Sik Chan
ISBN: 9780387759586

Commentaire: Sans aucun doute, le meilleur livre sur les séries temporelles financières que j'ai pu avoir à ce jour entre les mains. Un bon équilibre entre pratique et formalisme mathématique. Dommage qu'il manque juste les démonstrations mathématiques détaillées mais bon… on pardonnera cette faut sinon le livre ferait probablement 2'000 pages.

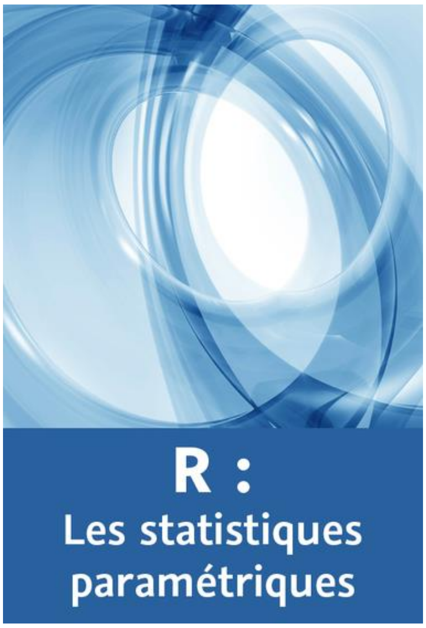


Flexible imputation of Missing Data (2nd Edition) / 444 pages / Éditions CRC Press / Stef van Buuren
ISBN: 9781138588318

Commentaire: Livre unique avec contenu très rare qui présente le sujet de manière très satisfaisantes et superbement illustré avec les détails des scripts R correspondants!

En ce qui concerne les vidéos pour l'instant nous ne pouvons que recommander en ce qui concerne la langue française (même si ce n'est qu'une découverte) les vidéos structurées et retravaillées avec un niveau de qualité professionnel de Video2Brain et effectuées par Vincent (cliquez sur les images pour ouvrir la page web correspondante):





et R pour Data Scientists:


00110
R

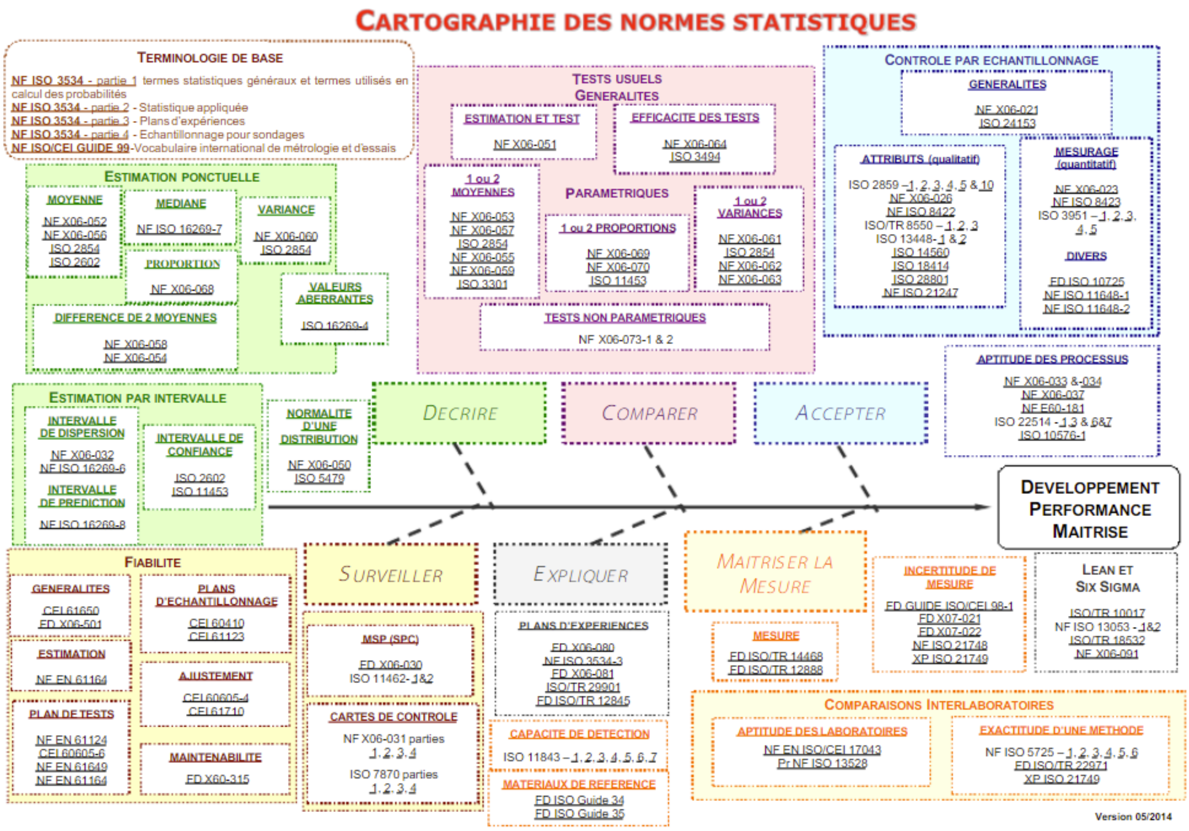
2. Normes et validation

Rappelons conformément à ce que nous avons vu dans le cours théorique qu'il est indispensable pour le chercheur/statisticien/ingénieur professionnel de se baser sur les normes suivantes (dans l'ordre des plus utilisées) pour son travail et tous les outils dont le présent support fait l'objet:

- **ISO 31:2006**
  Système international d'unités
- **ISO 8000**
  Qualité des données
- **ISO 3534-1:1999**
  Vocabulaire et symbole des statistiques
- **ISO 2602:1980**
  Interprétation statistique de résultats d'essais - Estimation de la moyenne - Intervalle de confiance
- **ISO 3301:1975**
  Interprétation statistique des données - Comparaison de deux moyennes dans le cas d'observations appariées
- **ISO 5479:1997**
  Interprétation statistique des données - Tests pour les écarts à la distribution normale
- **ISO 3494:1976**
  Interprétation statistique des données -- Efficacité des tests portant sur des moyennes et des variances
- **ISO 11453:1996**
  Interprétation statistique des données - Tests et intervalles de confiance portant sur les proportions
- **ISO 16269-4:2010**
  Interprétation statistique des données Détection et traitement des valeurs aberrantes
- **ISO 16269-6:2005**
  Interprétation statistique des données - Détermination des intervalles statistiques de tolérance
- **ISO 16269-8:2004**
  Interprétation statistique des données - Détermination des intervalles de prédiction
- **ISO/TR 18532:2009**
  Lignes directrices pour l'application des méthodes statistiques à la qualité et à la normalisation industrielle
- **ISO 3534-3:1999**
  Plans d'expérience (ou AFNOR NF X 06-080 + NF X 06-081)
- **ISO 8285:1991**
  Cartes de contrôle de Shewhart
- **ISO 17025:2005**
  Exigences générales concernant la compétence des laboratoires d'étalonnages et d'essais
- **ISO 10017:2003**
  Lignes directrices pour les techniques statistiques relatives à l'ISO 9001:2000
- **ISO 13300:2006**
  Guide général à l'attention du personnel des laboratoires d'analyse sensorielle

- **ISO 31010:2009**
  Techniques d'évaluations des risques
- **ISO 3951:2006**
  Règles d'échantillonnage pour les contrôles par mesures
- **ISO 11095:1996**
  Étalonnage linéaire utilisant des matériaux de référence
- **ISO 22514-2:2013**
  Indices de capabilité
- **ISO 5725**
  Précision et fiabilité des mesures en laboratoire (on y retrouve le test C de Cochran et aussi celui de Dixon)

Bref en gros en 2014 les normes relatives à l'analyse statistique peuvent se résumer avec la cartographie suivante:



À propos de la validation de l'utilisation du logiciel R par la FDA (Food and Drug administration) il est peut-être utile d'informer le lecteur que ceci n'est par le rôle de la FDA mais de l'acheteur de s'assurer du bon fonctionnement de la solution informatique choisie (conformément à la norme **ISO 9241-11** vue dans le cours de gestion de projets pour tout logiciel ou machine). Par ailleurs, signalons l'article du code des régulations Titre 21 partie 820.70(i):

> *"When computers or automated data processing systems are used as part of production or the quality system, the manufacturer shall validate computer software for its intended use according to an established protocol. All software*

> *changes shall be validated before approval and issuance. These validation activities and results shall be documented."*

et aussi (Final Guidance for Industry and FDA Staff, January 11, 2002):

> *"The device manufacturer is responsible for ensuring that the product development methodologies used by the off-the-shelf (OTS) software developer are appropriate and sufficient for the device manufacturer's intended use of that OTS software. For OTS software and equipment, the device manufacturer may or may not have access to the vendor's software validation documentation. If the vendor can provide information about their system requirements, software requirements, validation process, and the results of their validation, the medical device manufacturer can use that information as a beginning point for their required validation documentation."*

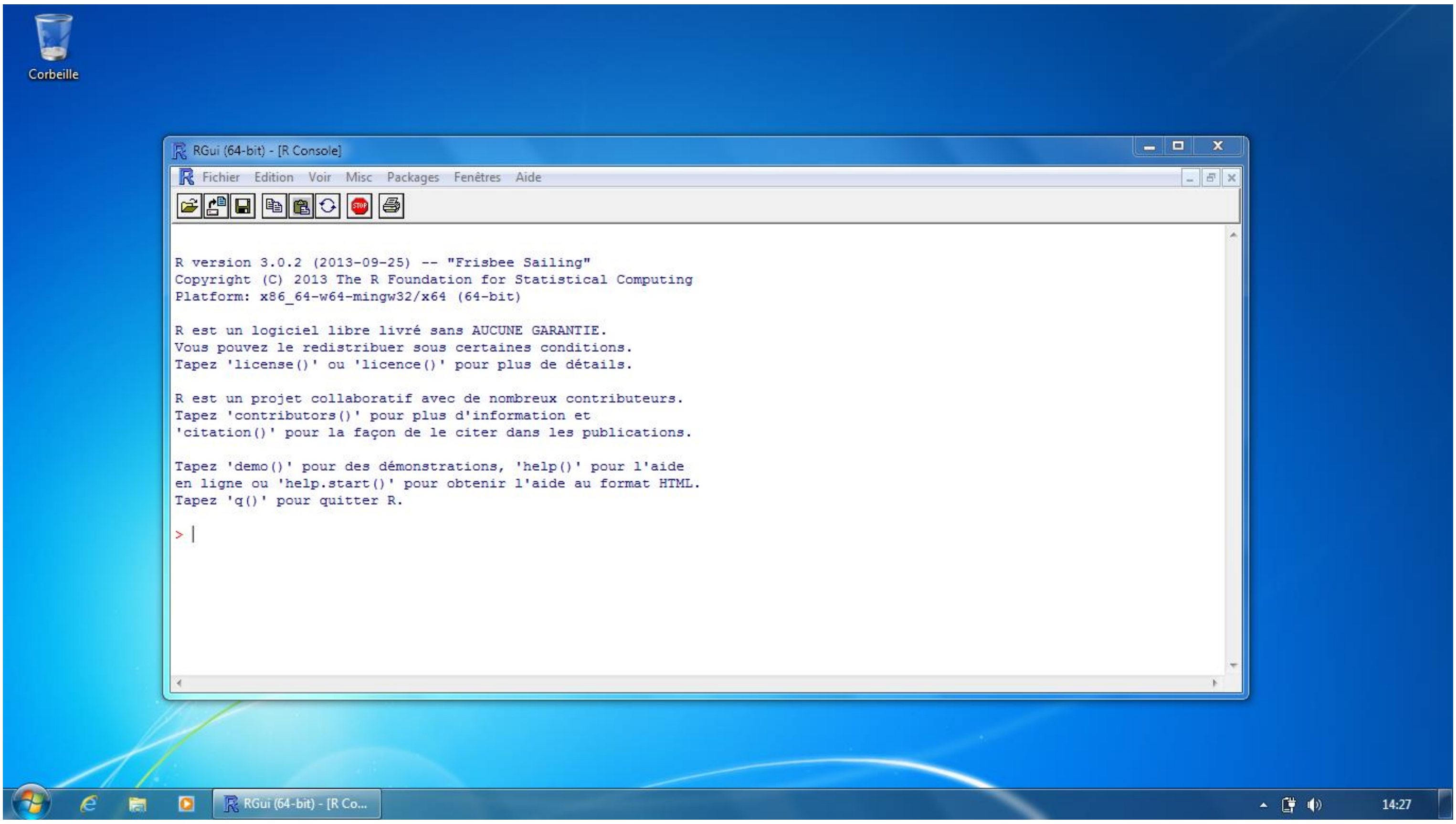
3. Installer R

En première année d'Université je considère au 21ème siècle que tout étudiant doit déjà savoir installer un logiciel seul sur Microsoft Windows ou Mac OS (sinon quoi il peut se faire des soucis pour sa carrière professionnelle). Donc ici je ne vais pas montrer comment installer R mais juste présenter le résultat de l'installation excepté pour Linux.

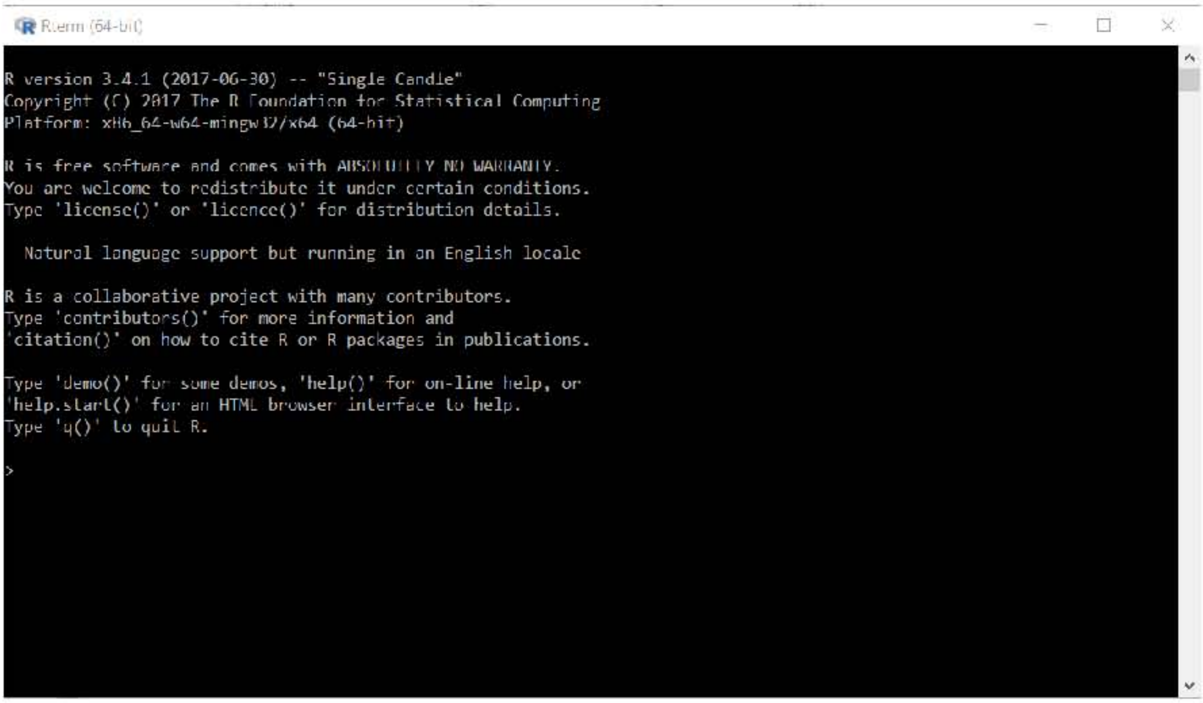
Contenu

Microsoft Windows

Sous **Microsoft Windows 7** (cas le plus courant mais scientifiquement le plus mauvais choix):

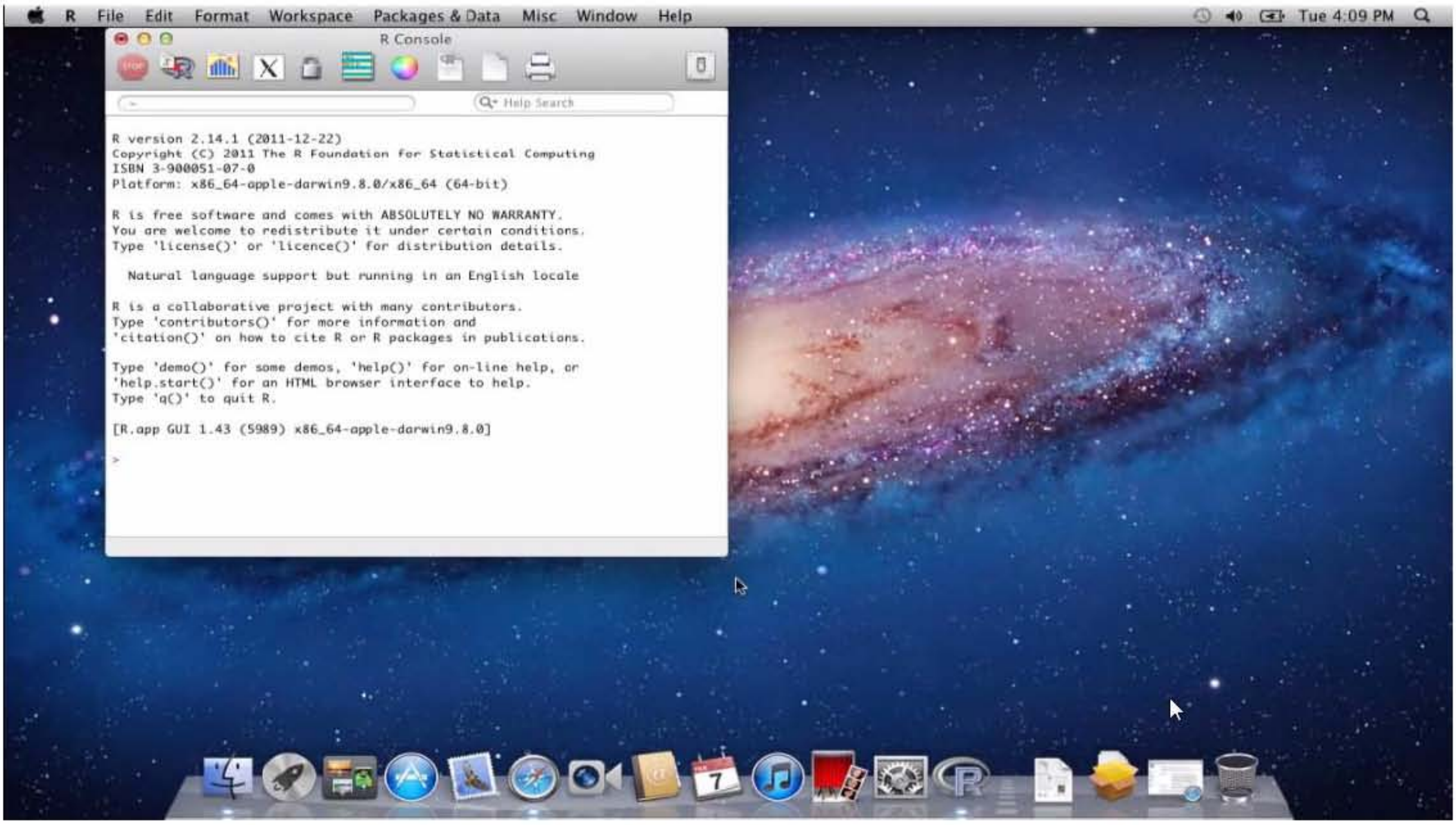


On peut bien évidemment travailler avec R dans Microsoft Windows en utilisant le Shell (suffit de double cliquer sur le fichier R.exe qui est dans le dossier d'installation de R):


Rterm (64-bit)
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

Mac OS

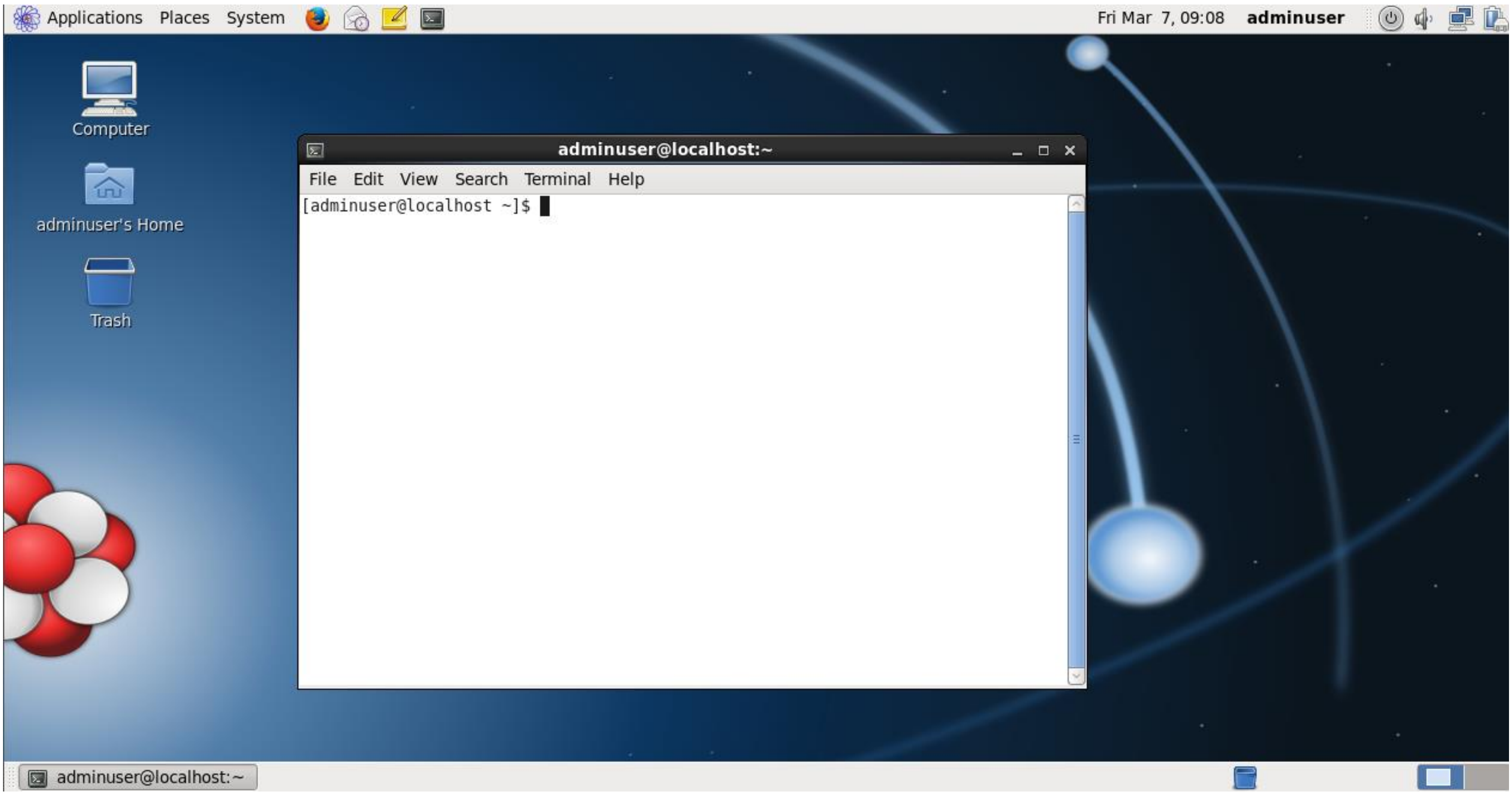
Sous Mac OS l'installation est aussi triviale. Le résultat donnant:



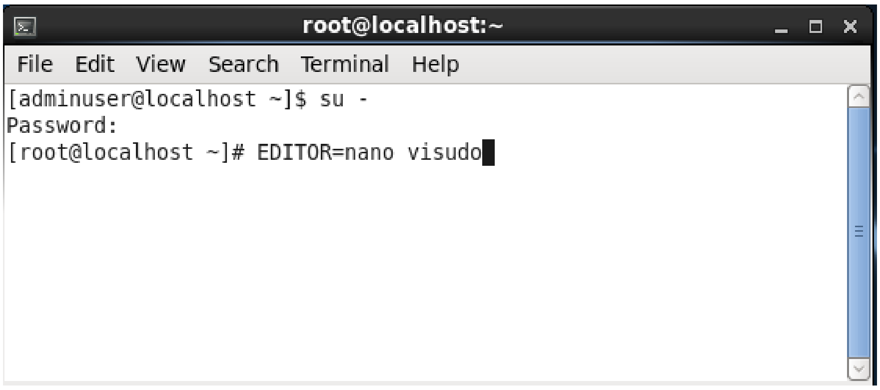
Scientific Linux 6.5

Sous **Scientific Linux (RedHat) 6.5** (cas le moins courant mais le meilleur choix car optimisé par le FermiLab et le CERN pour les applications scientifiques) voici comment procéder pour installer R (pour plus d'informations sur Scientifc Linux consulter mon e-book en anglais ici http://www.sciences.ch/dwnldbl/divers/ScientificLinux.pdf).

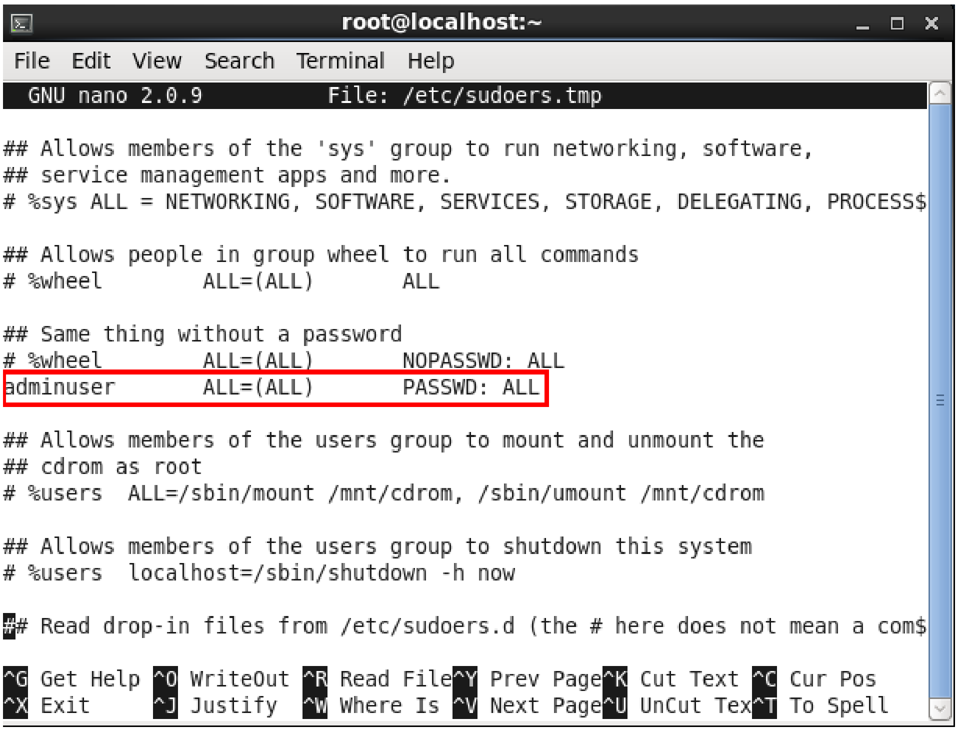
D'abord vous ouvrez une fenêtre de terminal:



et nous allons d'abord nous assurer d'avoir les droits suffisants pour installer R en faisant quelques manipulations non triviales. D'abord nous ouvrons le fichier de gestion des droits généraux avec l'éditeur nano:

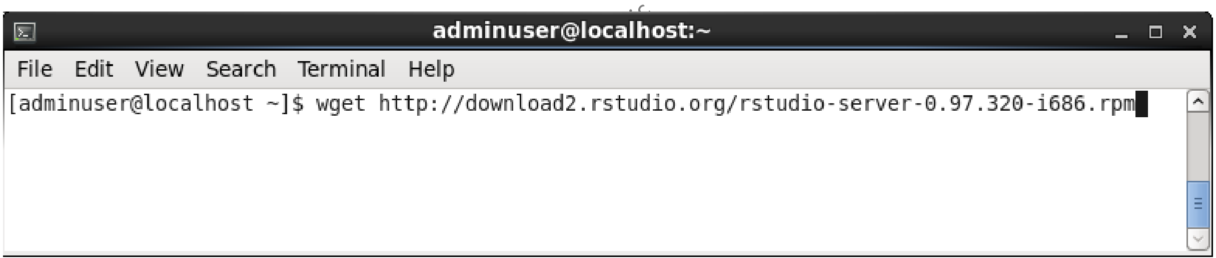


Et dans l'éditeur il faut rajouter la ligne mise en évidence ci-dessous en rouge (il faut descendre un peu dans le fichier texte pour trouver l'endroit où mettre cette ligne):



puis on fait un Ctrl+O, Entrée, Ctrl+X.

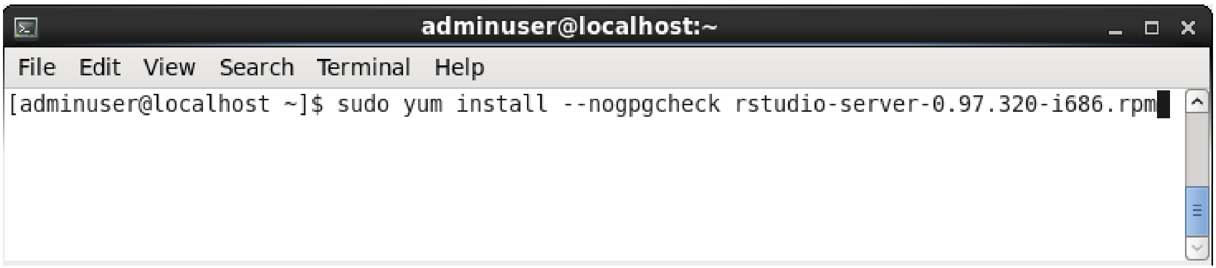
Ensuite, nous écrivons (il s'agit de RStudio Server mais il contient le coeur – core – de la dernière version de R):



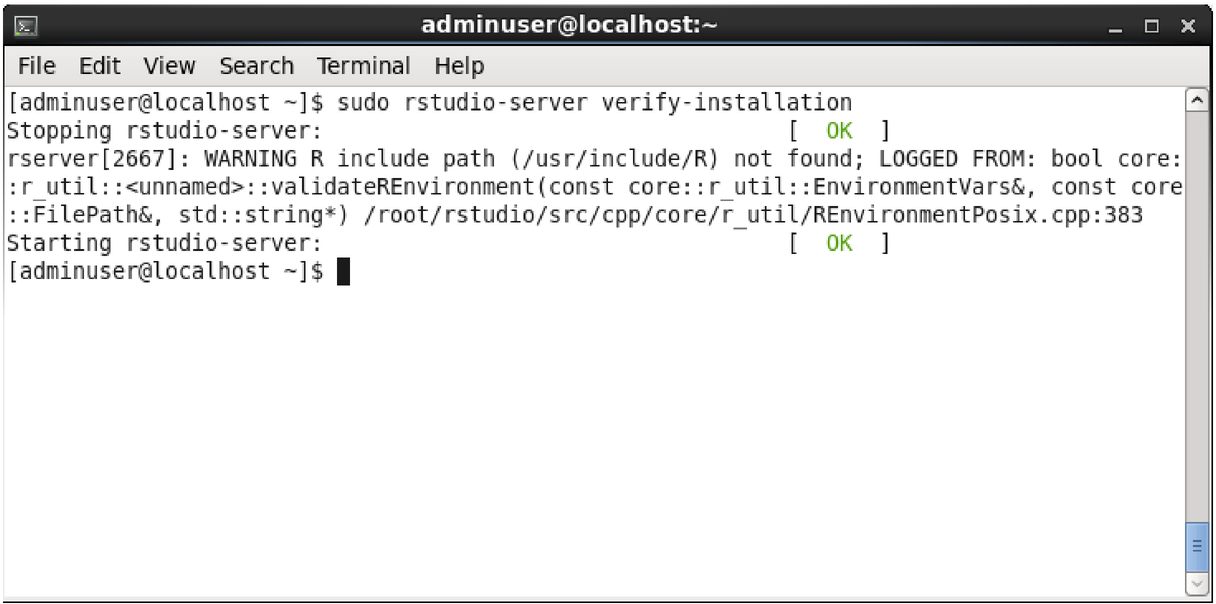
Cela va le télécharger comme le montre la capture suivante:



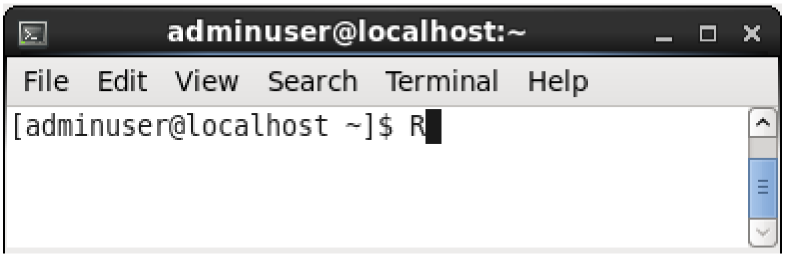
Ensuite, nous lançons l'installation:



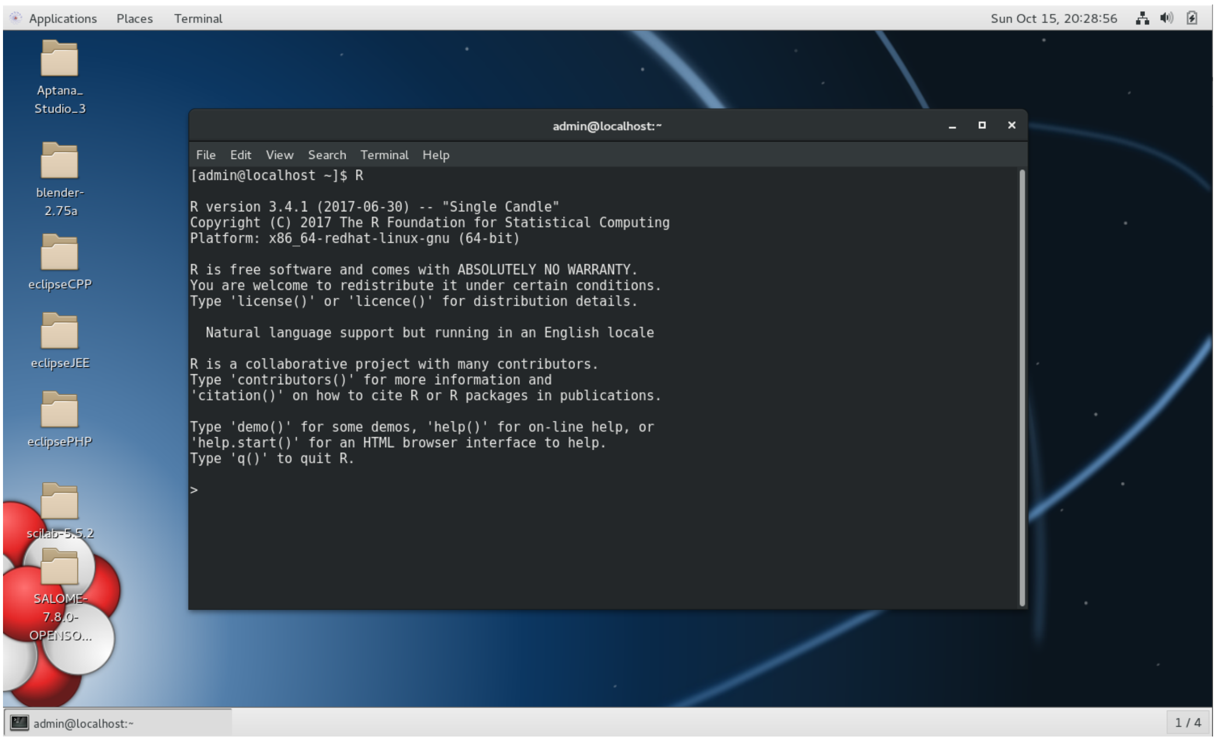
Après 10 à 15 minutes (le temps de télécharger environ 60 modules et de les installer) vous pouvez vérifier que l'installation soit OK.



Ensuite, pour lancer R, il suffit d'écrire:



Ce qui donne:



Voilà cela donne le squelette de base de R pour Scientific Linux. On doit probablement pouvoir faire plus simple mais je n'ai pas trouvé (et surtout je n'ai pas le temps de chercher).

Remarque: Depuis Scientific Linux 7 il suffit d'écrire:

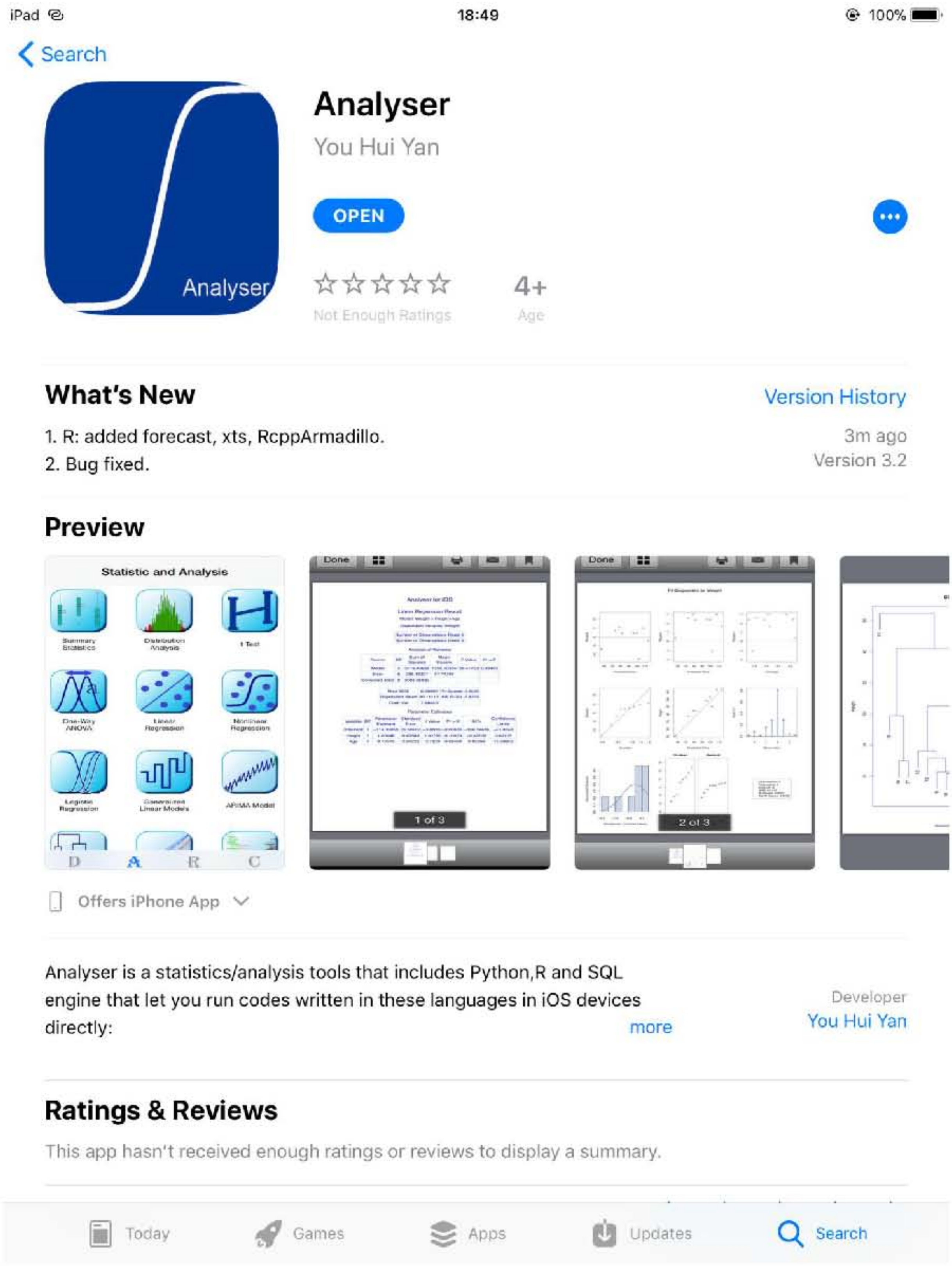
yum install R

Attention!!! La totalité des fichiers *.csv ont été créés sous Microsoft Windows et donc sont incompatibles au niveau encodage dans Linux. Il faut les soit copier/coller le contenu dans un fichier texte nouvellement créé, soit refaire une enregistré sous avec un encodage Linux…

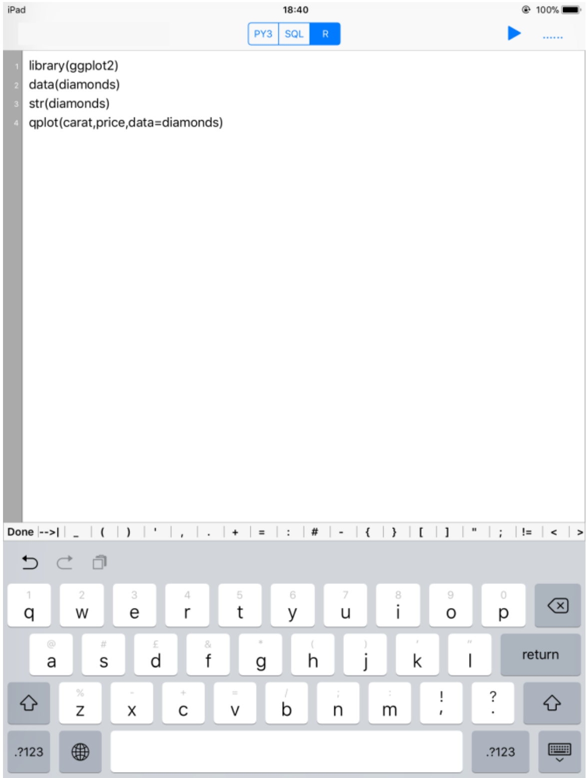
Pour la suite nous nous restreindrons au cas le plus fréquent chez les praticiens de l'industrie: MS Windows (sic!).

iPad

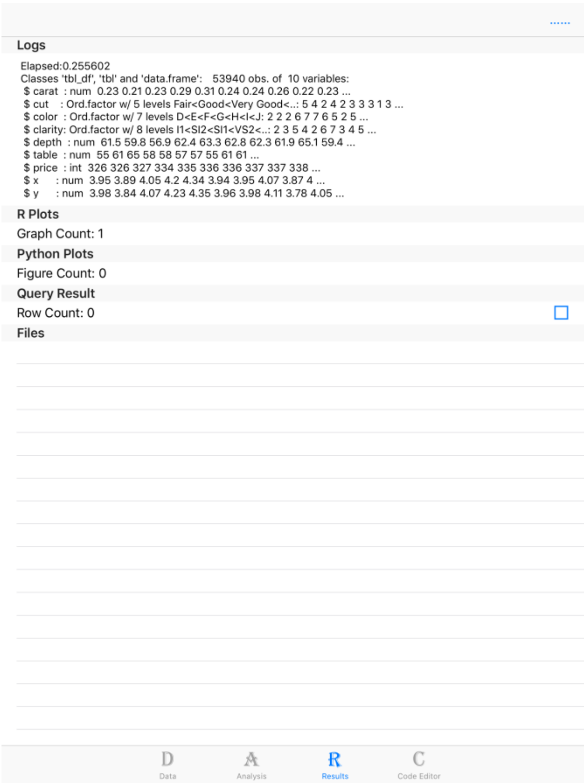
Pour ceux qui aiment faire du R en voyages, dans l'avion, dans le train, sur la plage, voici une version de R pour iPad que j'ai testé est qui est vraiment satisfaisant:



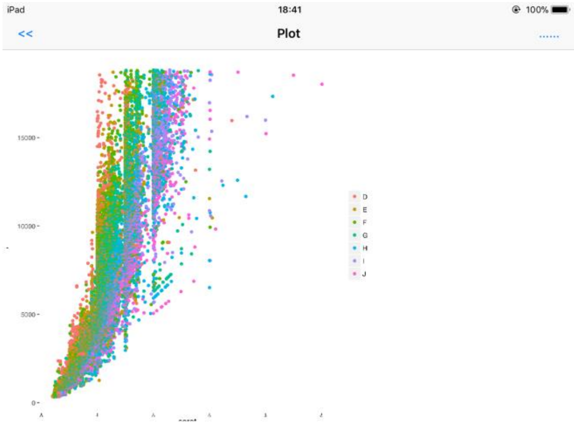
Une fois installé et ouvert on peut écrire du R presque sans limite vue le nombre de packages installés par défaut:



Ce qui exécuté donne:



ou même pour les grpahiques (ggplot2 dans la cas présent):



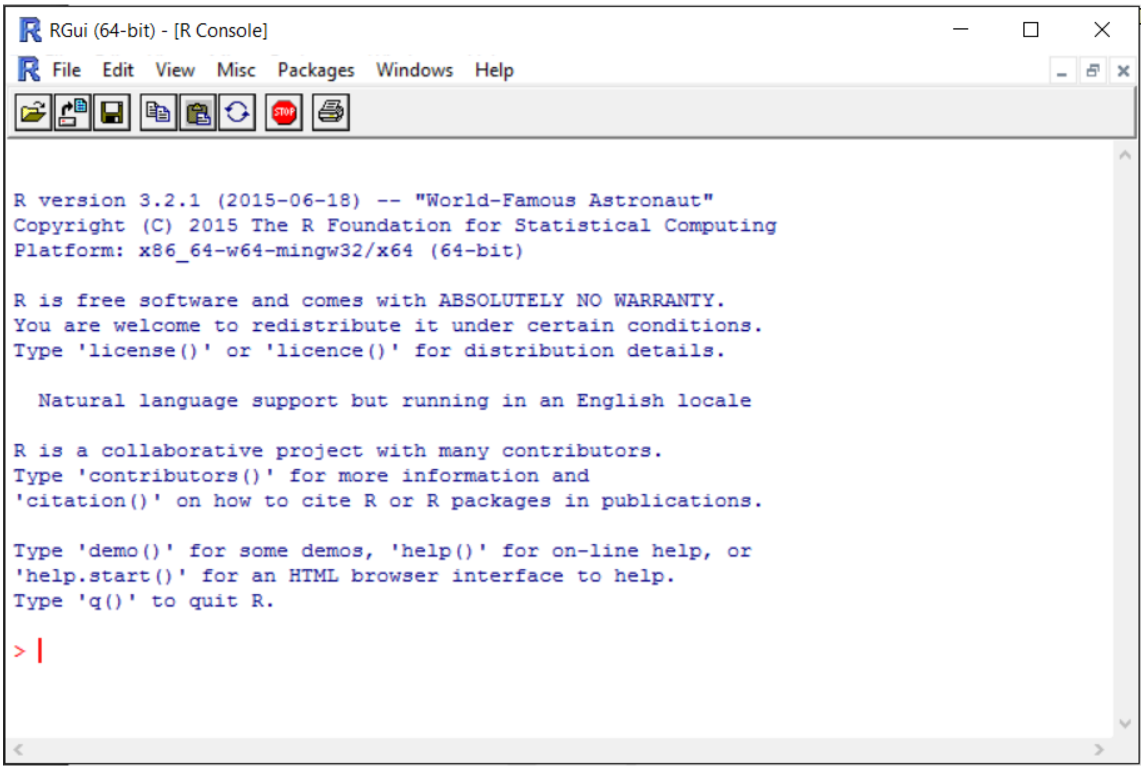
•••

Donc excellente application de You Hui Yan.

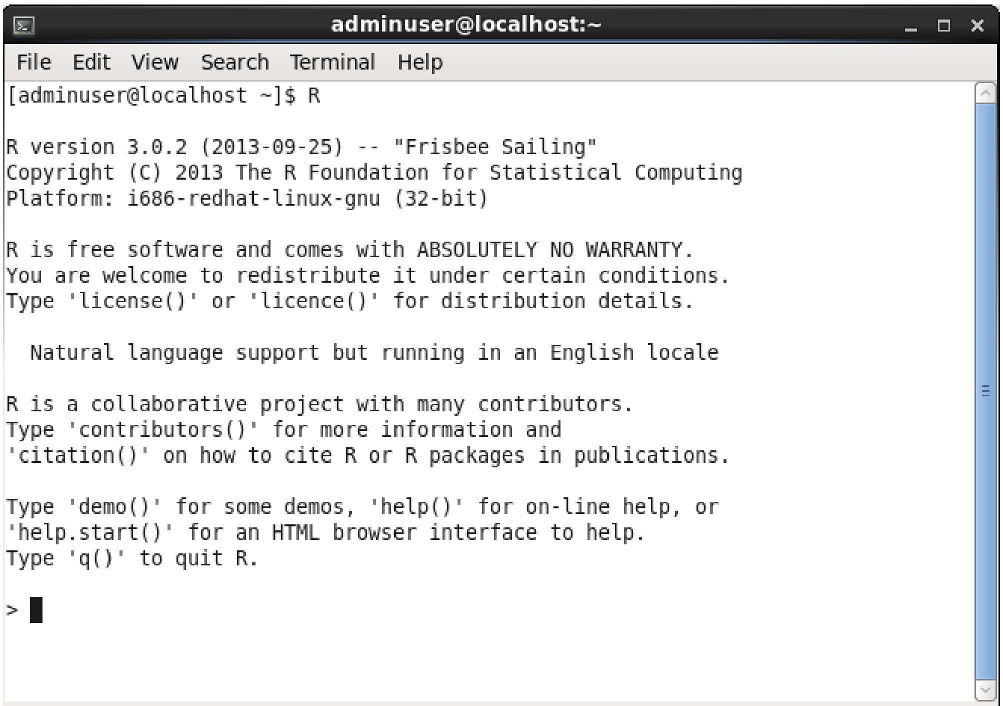
RAD IDE Divers

Pour la culture générale du lecteur nous allons montrer ici quelques interfaces classiques de R.

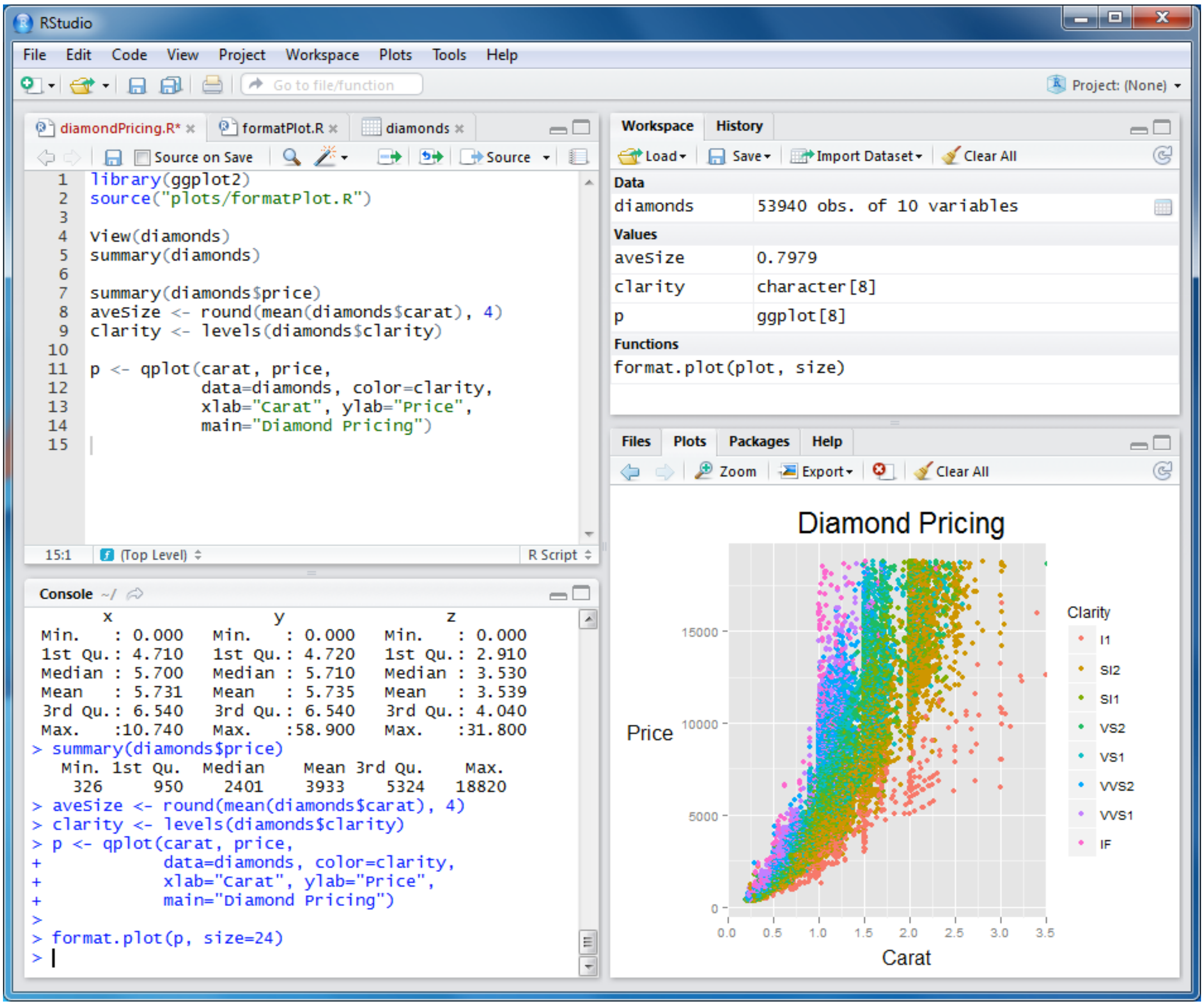
D'abord celle par défaut pour le monde Microsoft:



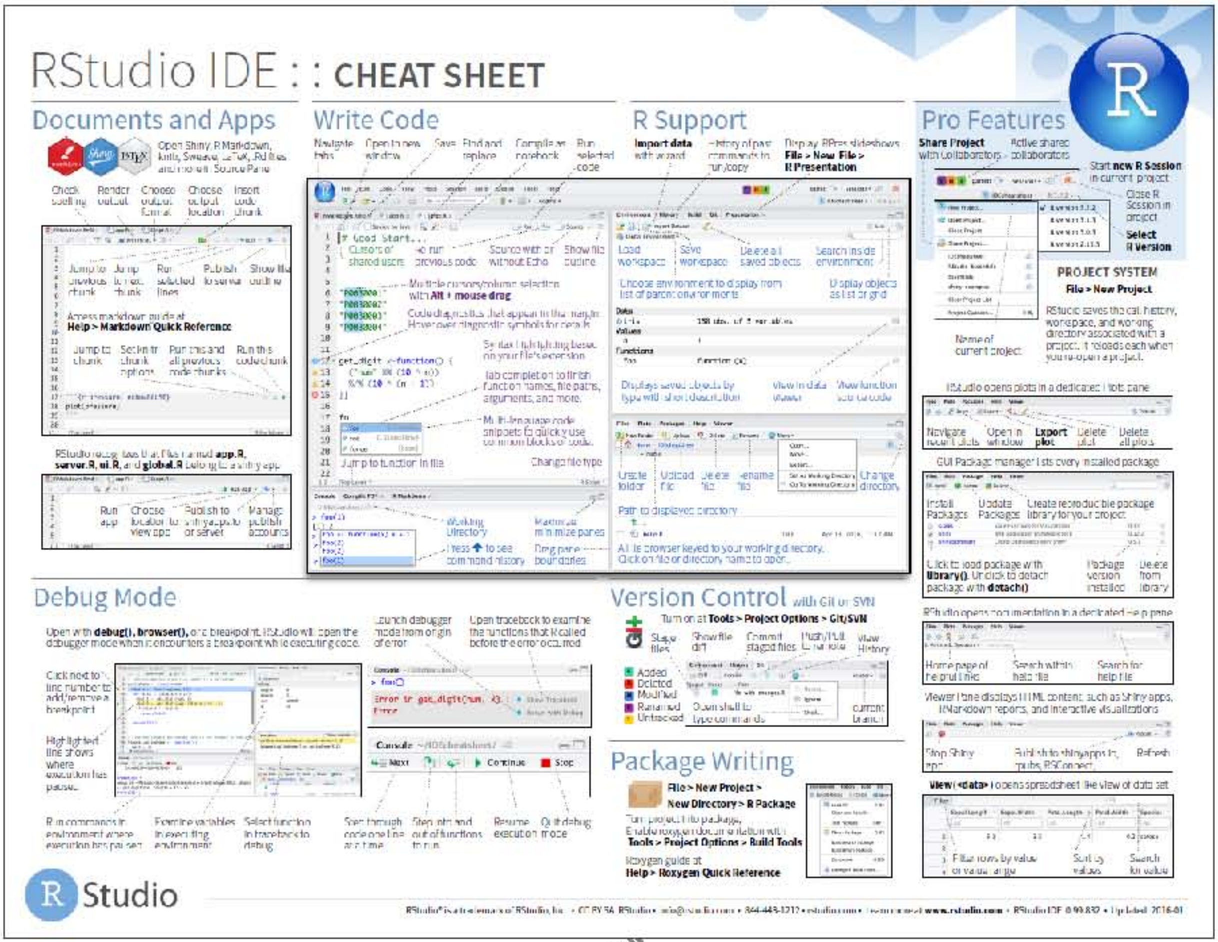
Ensuite celle par défaut dans le monde Linux (Ubuntu ou Scientific Linux ou CentOS avec une interface du type GNOME) :

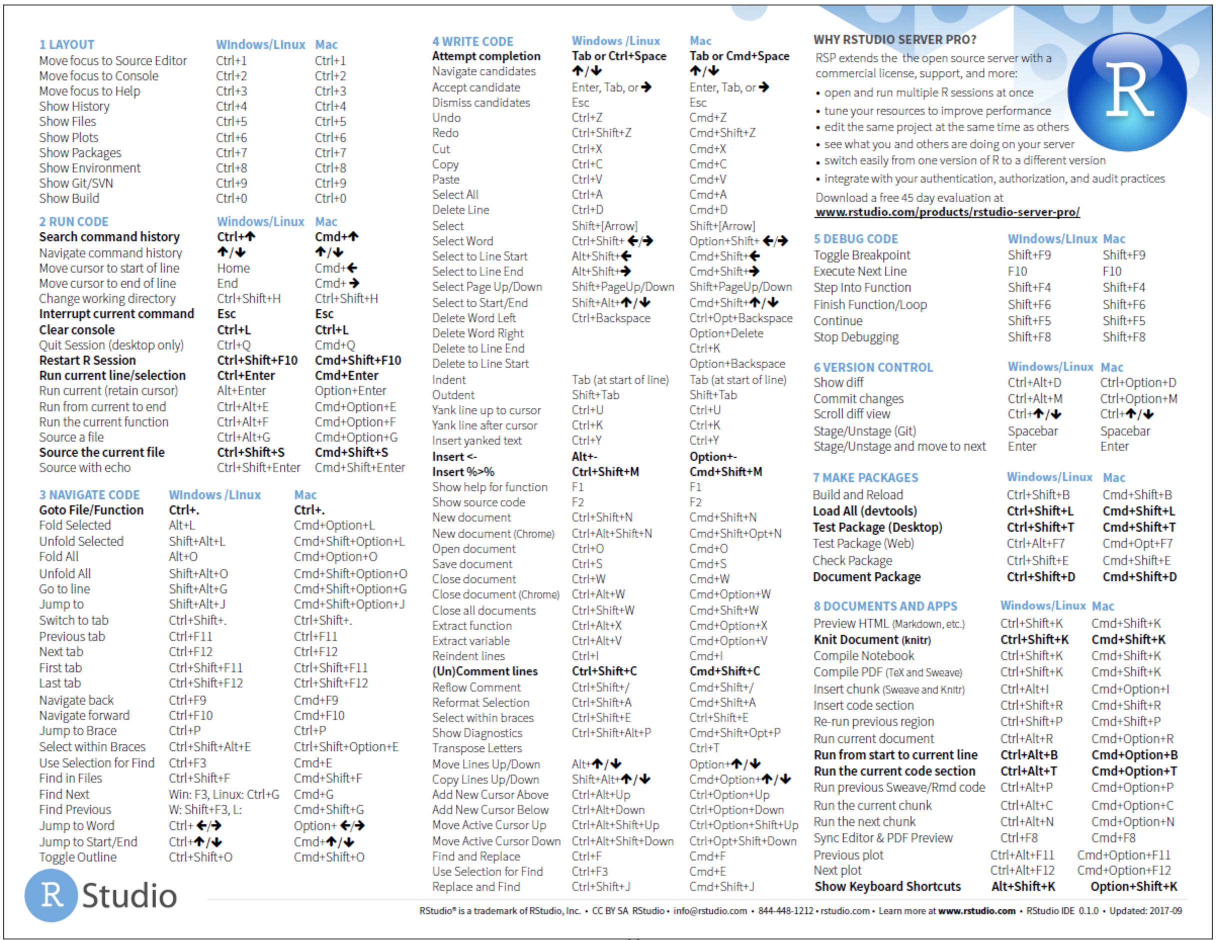


R Studio (dont nous avons déjà donné le lien Internet au début de ce livre) :



Pour ceux qui aiment R Studio, voici sa cheatsheet:





1 LAYOUT	Windows/Linux	Mac
Move focus to Source Editor	Ctrl+1	Ctrl+1
Move focus to Console	Ctrl+2	Ctrl+2
Move focus to Help	Ctrl+3	Ctrl+3
Show History	Ctrl+4	Ctrl+4
Show Files	Ctrl+5	Ctrl+5
Show Plots	Ctrl+6	Ctrl+6
Show Packages	Ctrl+7	Ctrl+7
Show Environment	Ctrl+8	Ctrl+8
Show Git/SVN	Ctrl+9	Ctrl+9
Show Build	Ctrl+0	Ctrl+0

2 RUN CODE	Windows/Linux	Mac
Search command history	**Ctrl+↑**	**Cmd+↑**
Navigate command history	↑/↓	↑/↓
Move cursor to start of line	Home	Cmd+←
Move cursor to end of line	End	Cmd+→
Change working directory	Ctrl+Shift+H	Ctrl+Shift+H
Interrupt current command	**Esc**	**Esc**
Clear console	**Ctrl+L**	**Ctrl+L**
Quit Session (desktop only)	Ctrl+Q	Cmd+Q
Restart R Session	**Ctrl+Shift+F10**	**Cmd+Shift+F10**
Run current line/selection	**Ctrl+Enter**	**Cmd+Enter**
Run current (retain cursor)	Alt+Enter	Option+Enter
Run from current to end	Ctrl+Alt+E	Cmd+Option+E
Run the current function	Ctrl+Alt+F	Cmd+Option+F
Source a file	Ctrl+Alt+G	Cmd+Option+G
Source the current file	**Ctrl+Shift+S**	**Cmd+Shift+S**
Source with echo	Ctrl+Shift+Enter	Cmd+Shift+Enter

3 NAVIGATE CODE	Windows /Linux	Mac
Goto File/Function	**Ctrl+.**	**Ctrl+.**
Fold Selected	Alt+L	Cmd+Option+L
Unfold Selected	Shift+Alt+L	Cmd+Shift+Option+L
Fold All	Alt+O	Cmd+Option+O
Unfold All	Shift+Alt+O	Cmd+Shift+Option+O
Go to line	Shift+Alt+G	Cmd+Shift+Option+G
Jump to	Shift+Alt+J	Cmd+Shift+Option+J
Switch to tab	Ctrl+Shift+.	Ctrl+Shift+.
Previous tab	Ctrl+F11	Ctrl+F11
Next tab	Ctrl+F12	Ctrl+F12
First tab	Ctrl+Shift+F11	Ctrl+Shift+F11
Last tab	Ctrl+Shift+F12	Ctrl+Shift+F12
Navigate back	Ctrl+F9	Cmd+F9
Navigate forward	Ctrl+F10	Cmd+F10
Jump to Brace	Ctrl+P	Ctrl+P
Select within Braces	Ctrl+Shift+Alt+E	Ctrl+Shift+Option+E
Use Selection for Find	Ctrl+F3	Cmd+E
Find in Files	Ctrl+Shift+F	Cmd+Shift+F
Find Next	Win: F3, Linux: Ctrl+G	Cmd+G
Find Previous	W: Shift+F3, L:	Cmd+Shift+G
Jump to Word	Ctrl+ ←/→	Option+ ←/→
Jump to Start/End	Ctrl+↑/↓	Cmd+↑/↓
Toggle Outline	Ctrl+Shift+O	Cmd+Shift+O

RStudio

4 WRITE CODE	Windows /Linux	Mac
Attempt completion	**Tab or Ctrl+Space**	**Tab or Cmd+Space**
Navigate candidates	↑/↓	↑/↓
Accept candidate	Enter, Tab, or →	Enter, Tab, or →
Dismiss candidates	Esc	Esc
Undo	Ctrl+Z	Cmd+Z
Redo	Ctrl+Shift+Z	Cmd+Shift+Z
Cut	Ctrl+X	Cmd+X
Copy	Ctrl+C	Cmd+C
Paste	Ctrl+V	Cmd+V
Select All	Ctrl+A	Cmd+A
Delete Line	Ctrl+D	Cmd+D
Select	Shift+[Arrow]	Shift+[Arrow]
Select Word	Ctrl+Shift+ ←/→	Option+Shift+ ←/→
Select to Line Start	Alt+Shift+←	Cmd+Shift+←
Select to Line End	Alt+Shift+→	Cmd+Shift+→
Select Page Up/Down	Shift+PageUp/Down	Shift+PageUp/Down
Select to Start/End	Shift+Alt+↑/↓	Cmd+Shift+↑/↓
Delete Word Left	Ctrl+Backspace	Ctrl+Opt+Backspace
Delete Word Right		Option+Delete
Delete to Line End		Ctrl+K
Delete to Line Start		Option+Backspace
Indent	Tab (at start of line)	Tab (at start of line)
Outdent	Shift+Tab	Shift+Tab
Yank line up to cursor	Ctrl+U	Ctrl+U
Yank line after cursor	Ctrl+K	Ctrl+K
Insert yanked text	Ctrl+Y	Ctrl+Y
Insert <-	**Alt+-**	**Option+-**
Insert %>%	**Ctrl+Shift+M**	**Cmd+Shift+M**
Show help for function	F1	F1
Show source code	F2	F2
New document	Ctrl+Shift+N	Cmd+Shift+N
New document (Chrome)	Ctrl+Alt+Shift+N	Cmd+Shift+Opt+N
Open document	Ctrl+O	Cmd+O
Save document	Ctrl+S	Cmd+S
Close document	Ctrl+W	Cmd+W
Close document (Chrome)	Ctrl+Alt+W	Cmd+Option+W
Close all documents	Ctrl+Shift+W	Cmd+Shift+W
Extract function	Ctrl+Alt+X	Cmd+Option+X
Extract variable	Ctrl+Alt+V	Cmd+Option+V
Reindent lines	Ctrl+I	Cmd+I
(Un)Comment lines	**Ctrl+Shift+C**	**Cmd+Shift+C**
Reflow Comment	Ctrl+Shift+/	Cmd+Shift+/
Reformat Selection	Ctrl+Shift+A	Cmd+Shift+A
Select within braces	Ctrl+Shift+E	Ctrl+Shift+E
Show Diagnostics	Ctrl+Shift+Alt+P	Cmd+Shift+Opt+P
Transpose Letters		Ctrl+T
Move Lines Up/Down	Alt+↑/↓	Option+↑/↓
Copy Lines Up/Down	Shift+Alt+↑/↓	Cmd+Option+↑/↓
Add New Cursor Above	Ctrl+Alt+Up	Ctrl+Option+Up
Add New Cursor Below	Ctrl+Alt+Down	Ctrl+Option+Down
Move Active Cursor Up	Ctrl+Alt+Shift+Up	Ctrl+Option+Shift+Up
Move Active Cursor Down	Ctrl+Alt+Shift+Down	Ctrl+Opt+Shift+Down
Find and Replace	Ctrl+F	Cmd+F
Use Selection for Find	Ctrl+F3	Cmd+E
Replace and Find	Ctrl+Shift+J	Cmd+Shift+J

WHY RSTUDIO SERVER PRO?

RSP extends the the open source server with a commercial license, support, and more:

- open and run multiple R sessions at once
- tune your resources to improve performance
- edit the same project at the same time as others
- see what you and others are doing on your server
- switch easily from one version of R to a different version
- integrate with your authentication, authorization, and audit practices

Download a free 45 day evaluation at
www.rstudio.com/products/rstudio-server-pro/

5 DEBUG CODE	Windows/Linux	Mac
Toggle Breakpoint	Shift+F9	Shift+F9
Execute Next Line	F10	F10
Step Into Function	Shift+F4	Shift+F4
Finish Function/Loop	Shift+F6	Shift+F6
Continue	Shift+F5	Shift+F5
Stop Debugging	Shift+F8	Shift+F8

6 VERSION CONTROL	Windows/Linux	Mac
Show diff	Ctrl+Alt+D	Ctrl+Option+D
Commit changes	Ctrl+Alt+M	Ctrl+Option+M
Scroll diff view	Ctrl+↑/↓	Ctrl+↑/↓
Stage/Unstage (Git)	Spacebar	Spacebar
Stage/Unstage and move to next	Enter	Enter

7 MAKE PACKAGES	Windows/Linux	Mac
Build and Reload	Ctrl+Shift+B	Cmd+Shift+B
Load All (devtools)	**Ctrl+Shift+L**	**Cmd+Shift+L**
Test Package (Desktop)	**Ctrl+Shift+T**	**Cmd+Shift+T**
Test Package (Web)	Ctrl+Alt+F7	Cmd+Opt+F7
Check Package	Ctrl+Shift+E	Cmd+Shift+E
Document Package	**Ctrl+Shift+D**	**Cmd+Shift+D**

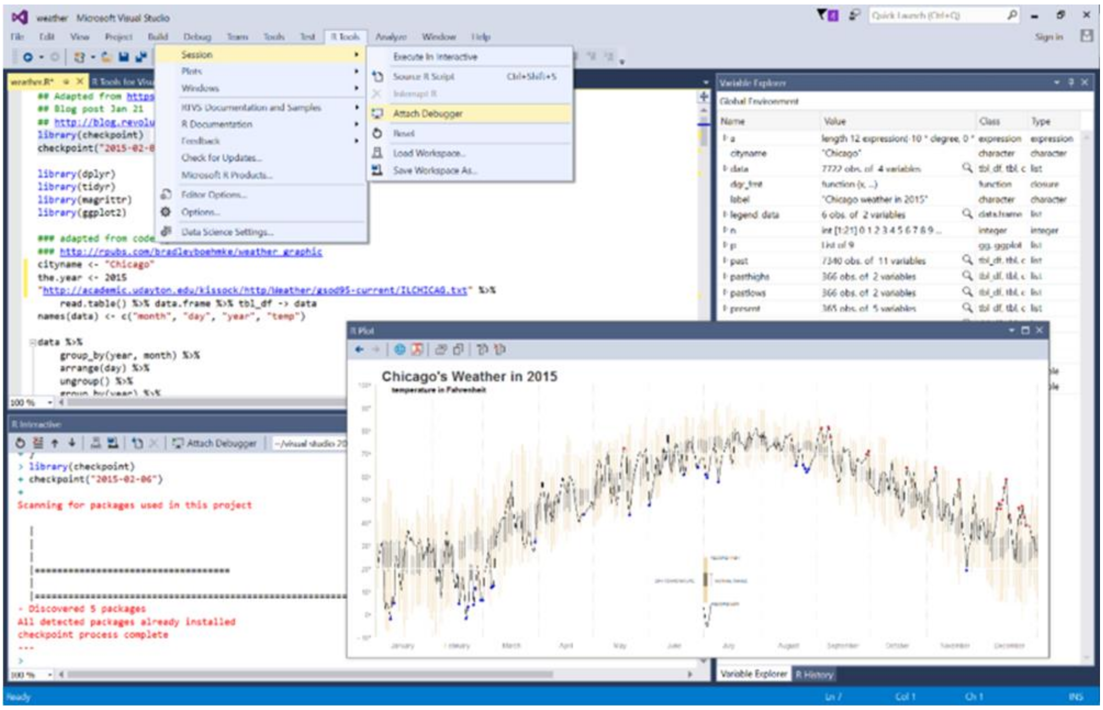
8 DOCUMENTS AND APPS	Windows/Linux	Mac
Preview HTML (Markdown, etc.)	Ctrl+Shift+K	Cmd+Shift+K
Knit Document (knitr)	**Ctrl+Shift+K**	**Cmd+Shift+K**
Compile Notebook	Ctrl+Shift+K	Cmd+Shift+K
Compile PDF (TeX and Sweave)	Ctrl+Shift+K	Cmd+Shift+K
Insert chunk (Sweave and Knitr)	Ctrl+Alt+I	Cmd+Option+I
Insert code section	Ctrl+Shift+R	Cmd+Shift+R
Re-run previous region	Ctrl+Shift+P	Cmd+Shift+P
Run current document	Ctrl+Alt+R	Cmd+Option+R
Run from start to current line	**Ctrl+Alt+B**	**Cmd+Option+B**
Run the current code section	**Ctrl+Alt+T**	**Cmd+Option+T**
Run previous Sweave/Rmd code	Ctrl+Alt+P	Cmd+Option+P
Run the current chunk	Ctrl+Alt+C	Cmd+Option+C
Run the next chunk	Ctrl+Alt+N	Cmd+Option+N
Sync Editor & PDF Preview	Ctrl+F8	Cmd+F8
Previous plot	Ctrl+Alt+F11	Cmd+Option+F11
Next plot	Ctrl+Alt+F12	Cmd+Option+F12
Show Keyboard Shortcuts	**Alt+Shift+K**	**Option+Shift+K**

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **www.rstudio.com** • RStudio IDE 0.1.0 • Updated: 2017-09

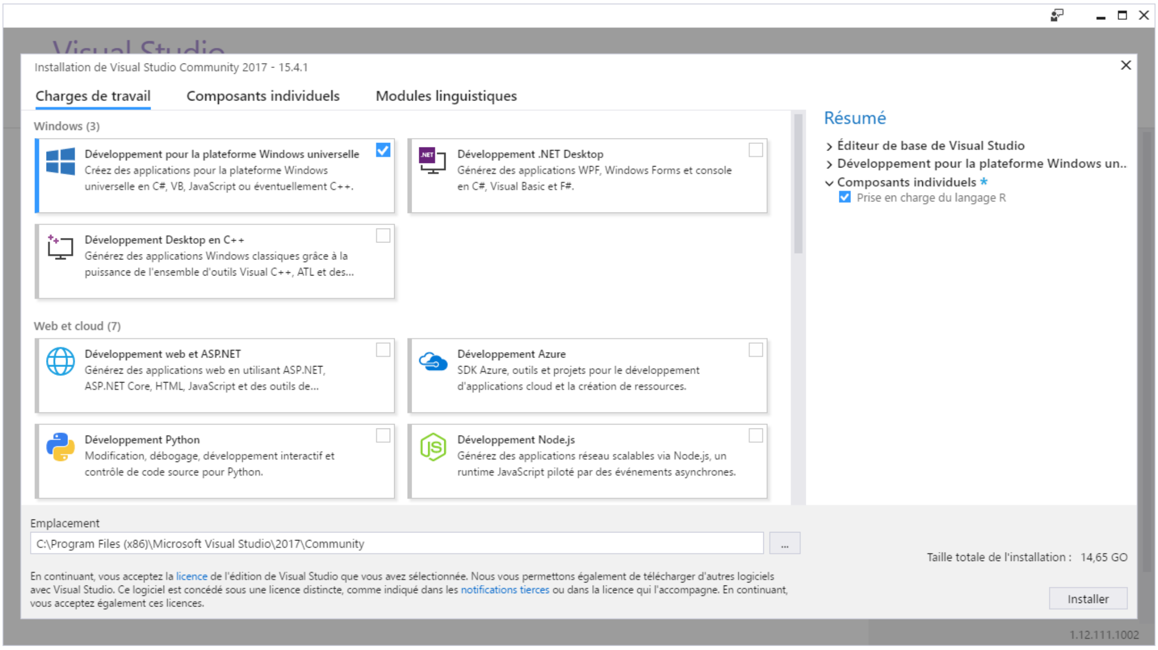
Figure 1 Cheatsheet R Studio

Visual Studio.Net depuis le 1er trimestre 2016 en installant le complément RVTS disponible sur (attention à bien suivre la procédure et particulièrement que l'option Web Developer Tools soit cochée à l'installation de VS.Net!) :

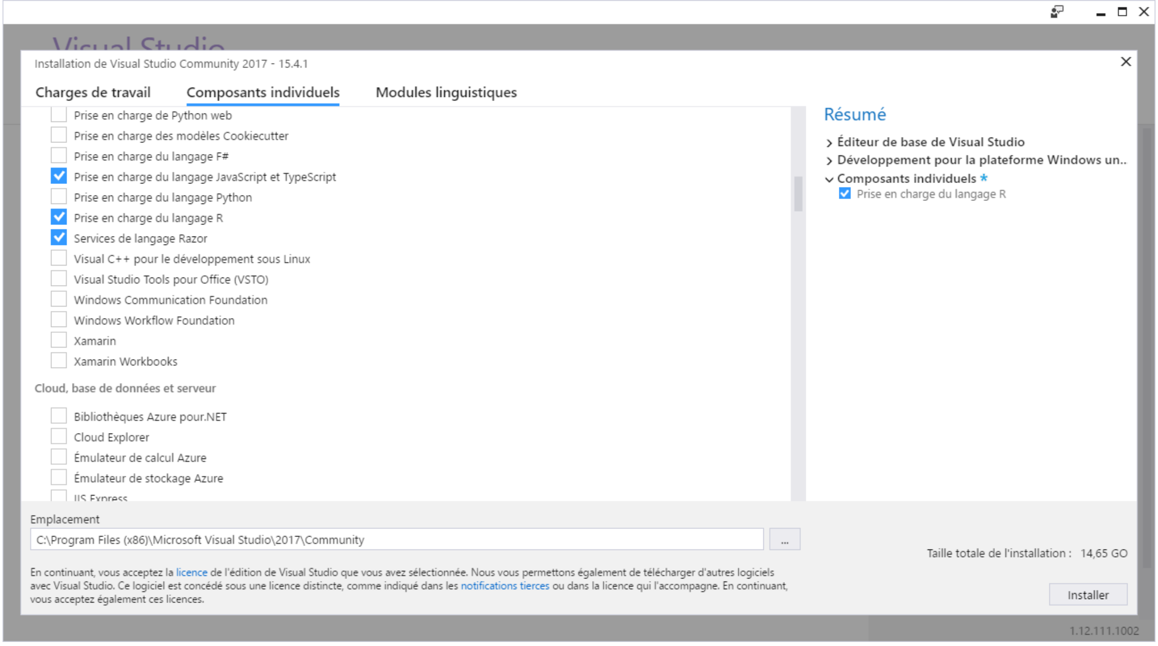
http://microsoft.github.io/RTVS-docs/installation.html



Ce dernier est utile si vous travaillez chez un employeur qui ne fait pas confiance en ce qui ne provient pas de chez Microsoft (…no comment…). Attention, pendant son installation (!), pas besoin de cocher l'option *Développement pour la plateforme Windows universelle*:



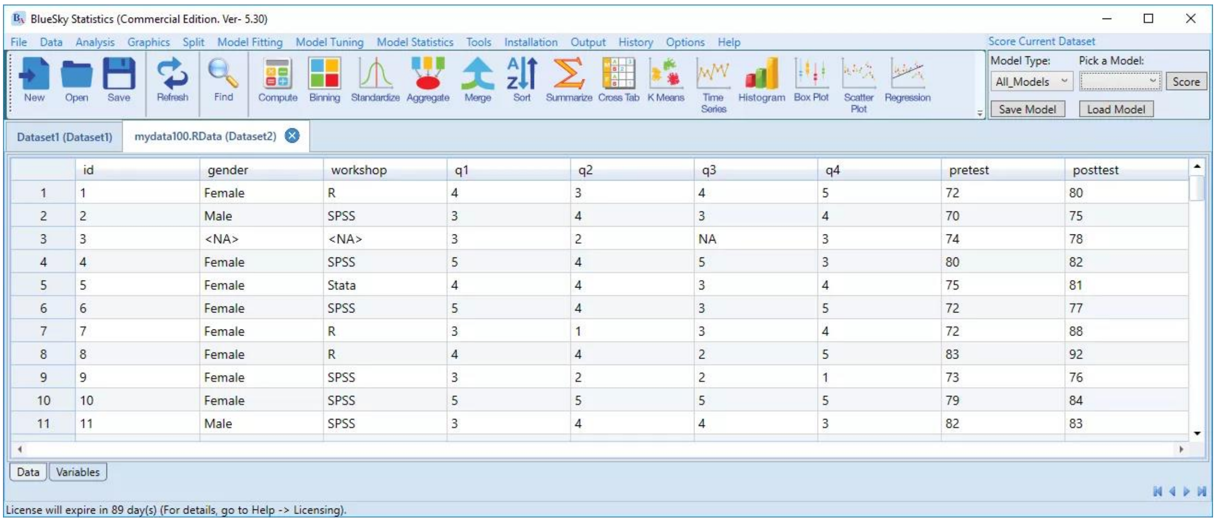
Mais il faut aller dans l'onglet *Composant individuels* et cocher *Prise en charge du langage R* (vous pouvez décocher le reste!):



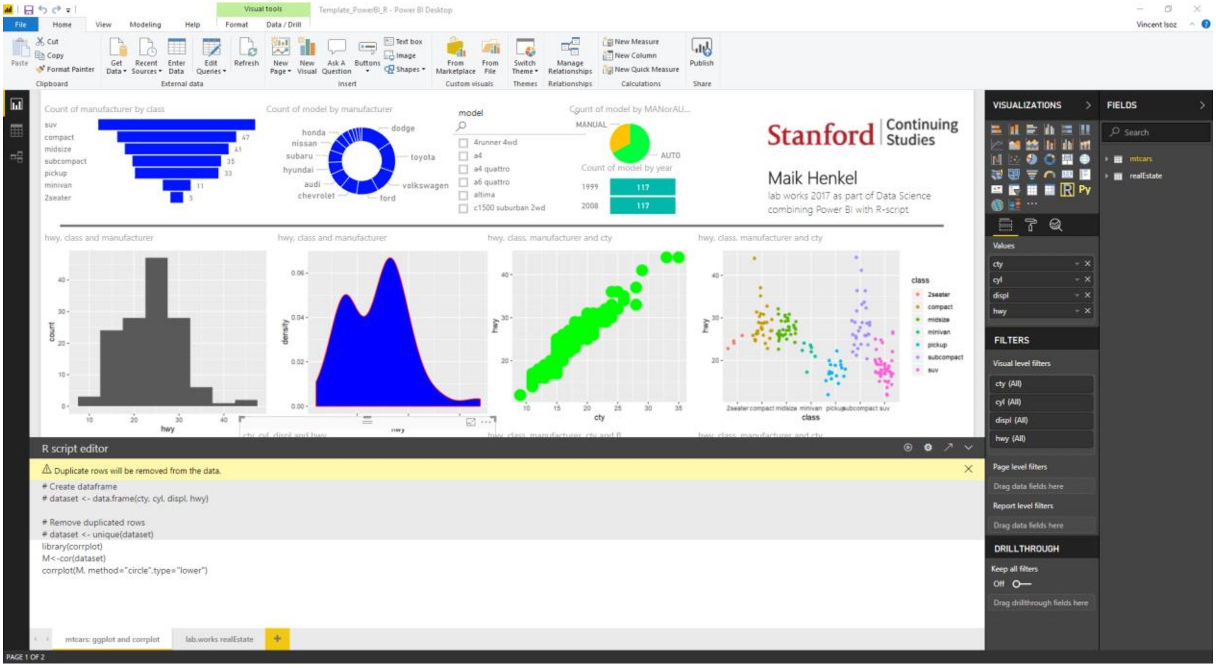
Ou encore pour les fans de SPSS, il y a BlueSky Statistics disponible à l'adresse suivante gratuitement:

https://www.blueskystatistics.com

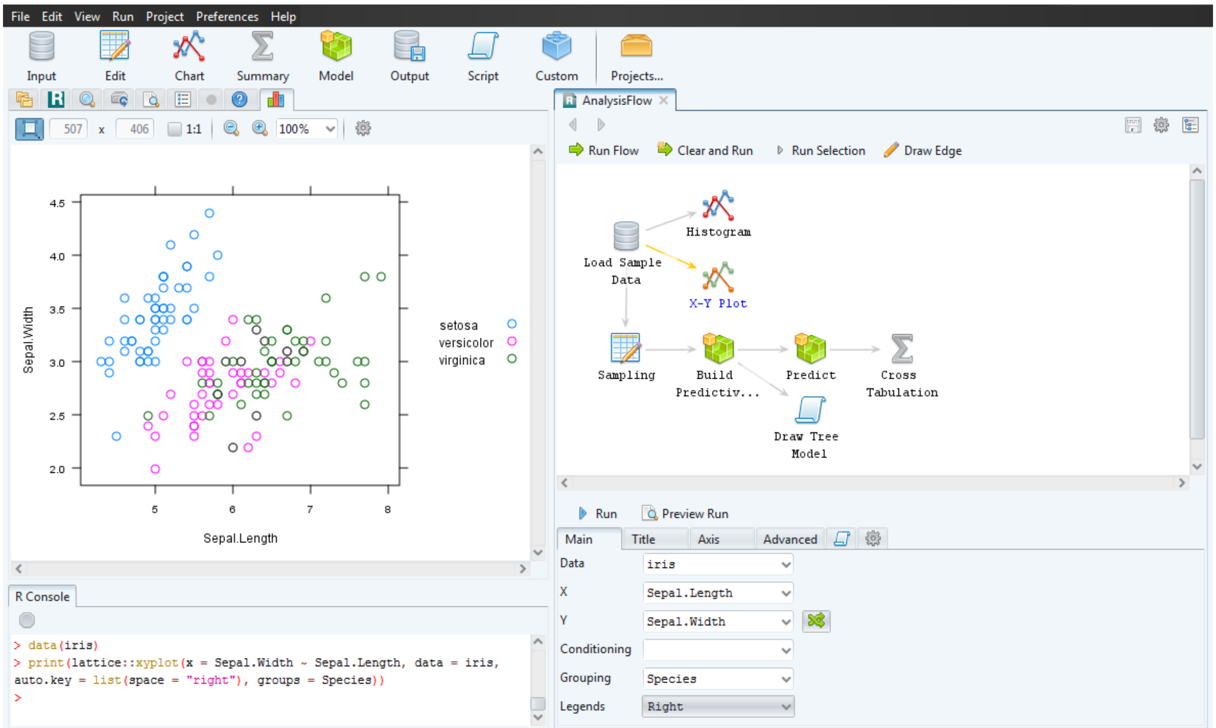
Qui une fois installé, ouvert et utilisé ressemble à:



Il y a aussi Microsoft Power BI qui est de mon point de vue le plus prometteur:



Pour ceux qui aiment bien le style RapidMiner, il y a aussi R AnalyticFlow:

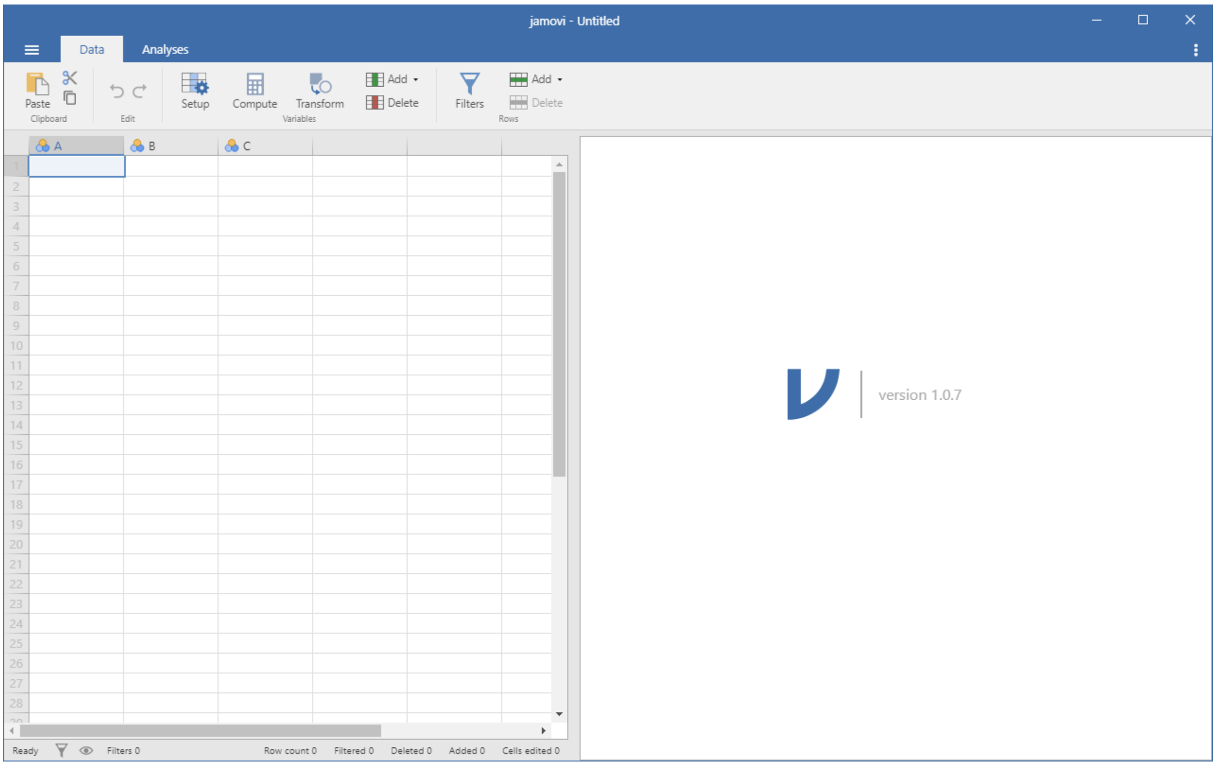


Disponible ici:

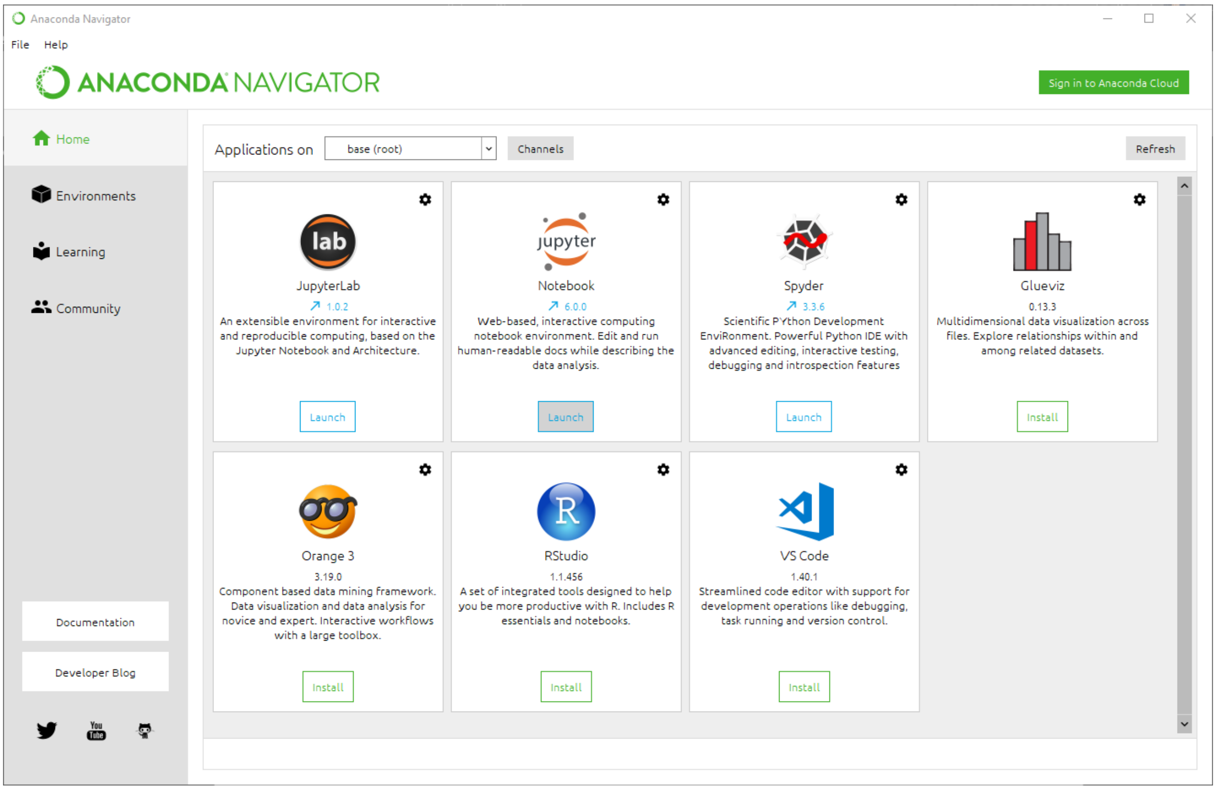
http://r.analyticflow.com/en/

Il y a aussi jamovi qui ressemble beaucoup à SPSS (il paraît que c'est son objectif… par ailleurs):

https://www.jamovi.org



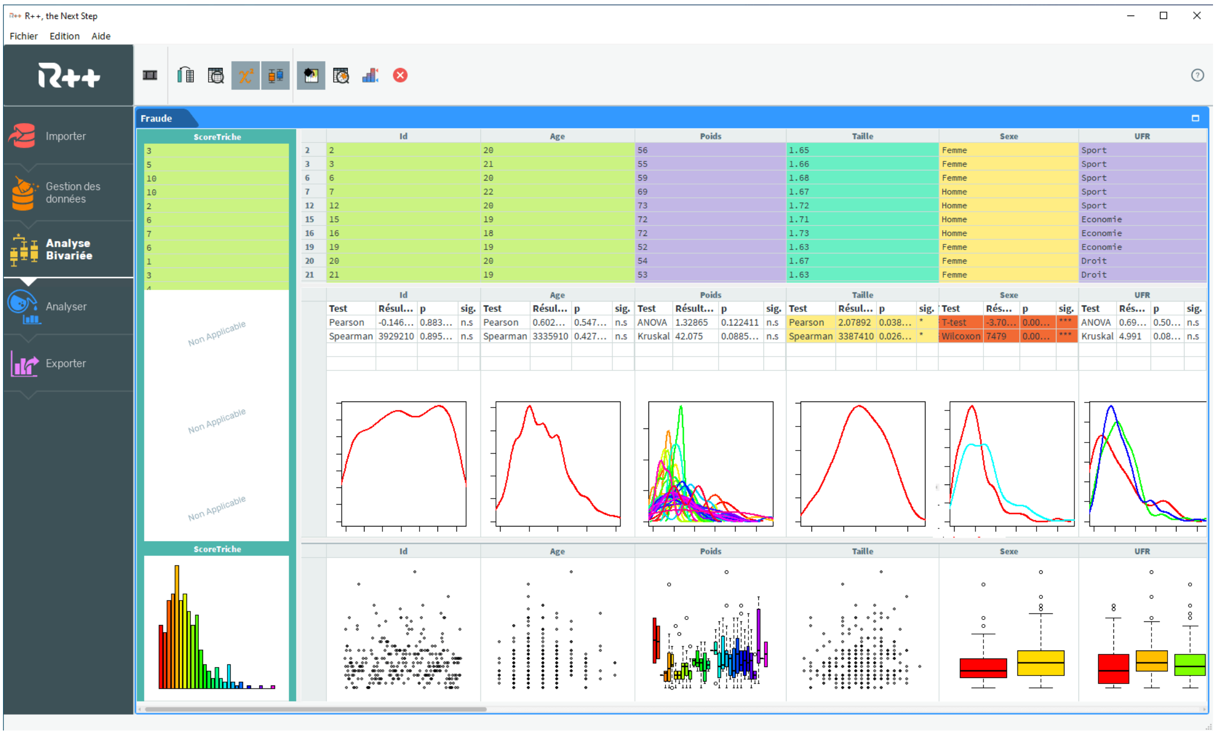
Et pour ceux qui aiment Python, on peut aussi faire du R dans Anaconda pour information (on peut choisir R lors de la création d'un nouvel environnement dans Anaconda):



Il y aussi R++:

https://rplusplus.com/

Qui ressemble à:



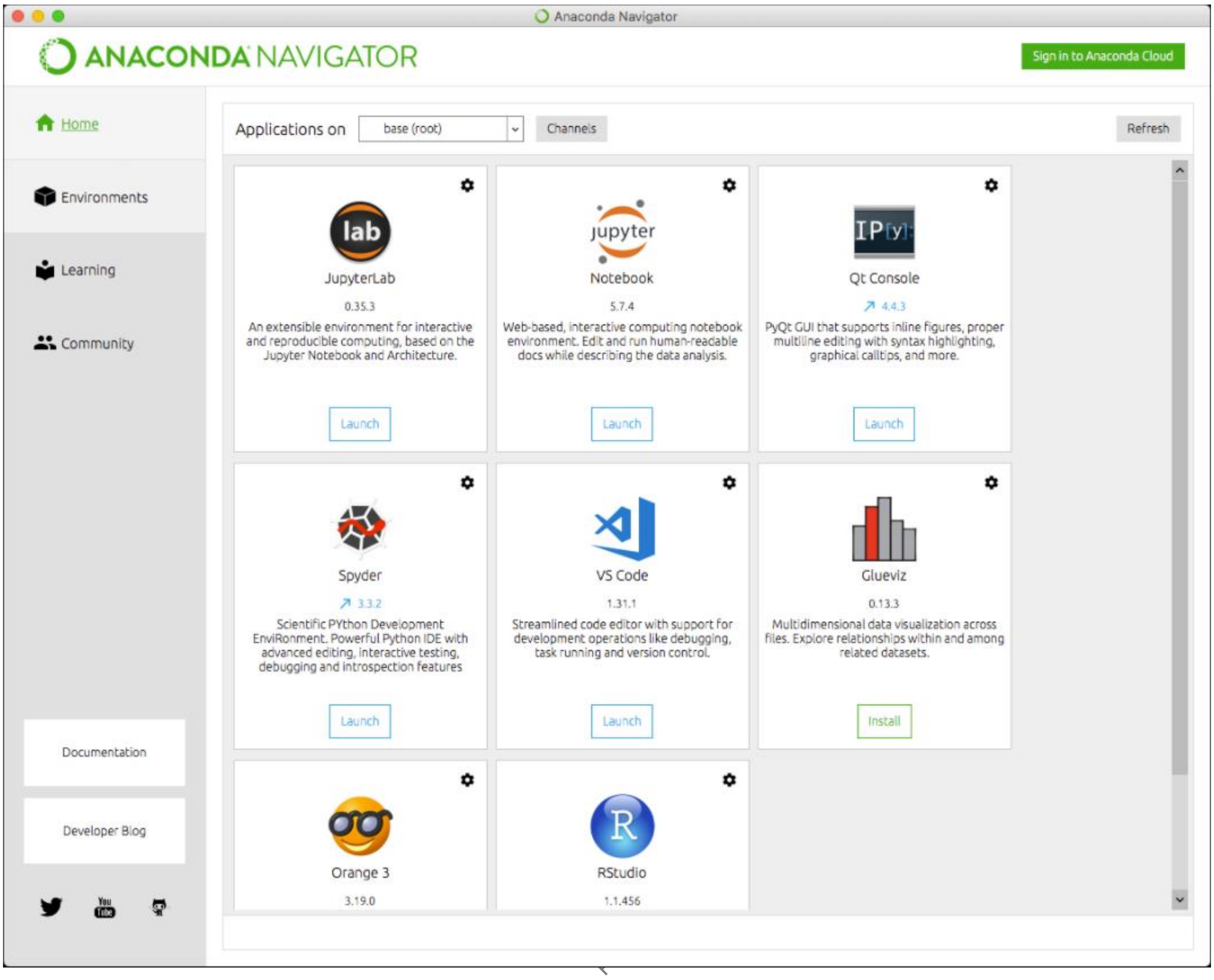
Sinon en ce qui concerne les IDE en ligne:

- http://rextester.com
- http://www.r-fiddle.org/#/
- https://www.jdoodle.com/execute-r-online

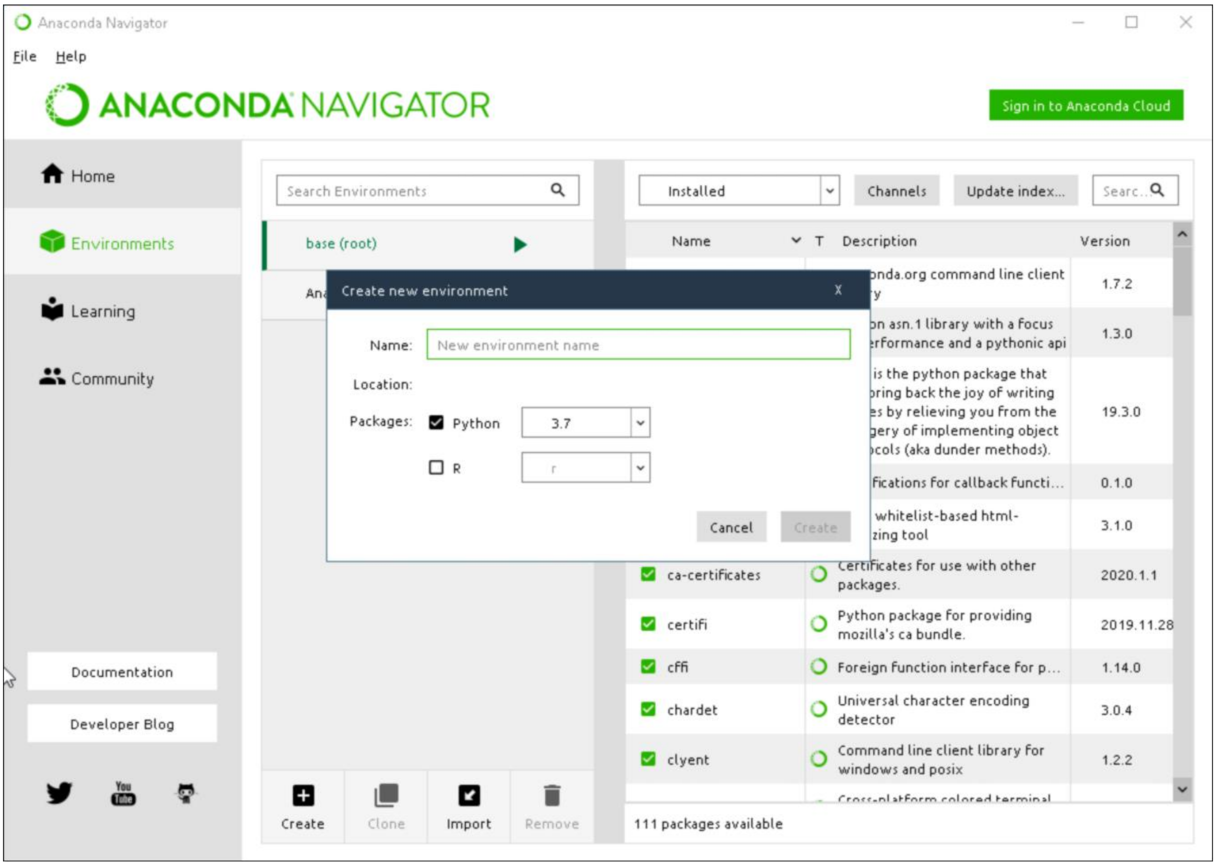
JupyterLab

Suite à la demande d'un lecteur détaillons un peu l'utilisation de JupyterLab pour faire du R.

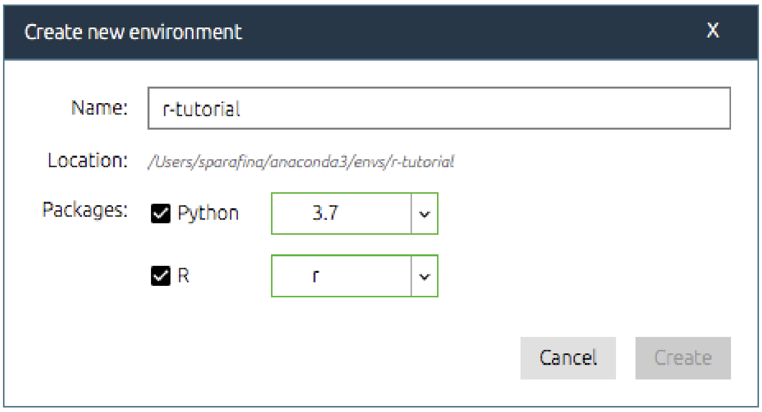
Pour installer et exécuter R dans un bloc-notes Jupyter on lancer Anaconda:



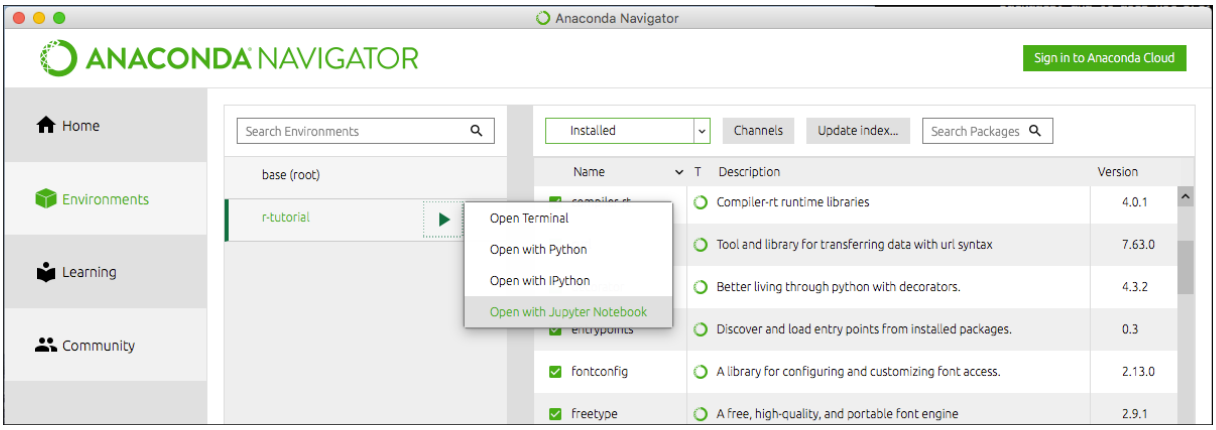
Pour installer le langage R et les packages r-essentials, sélectionnez **Environnements** pour créer un nouvel environnement. Cliquez sur **Create**:



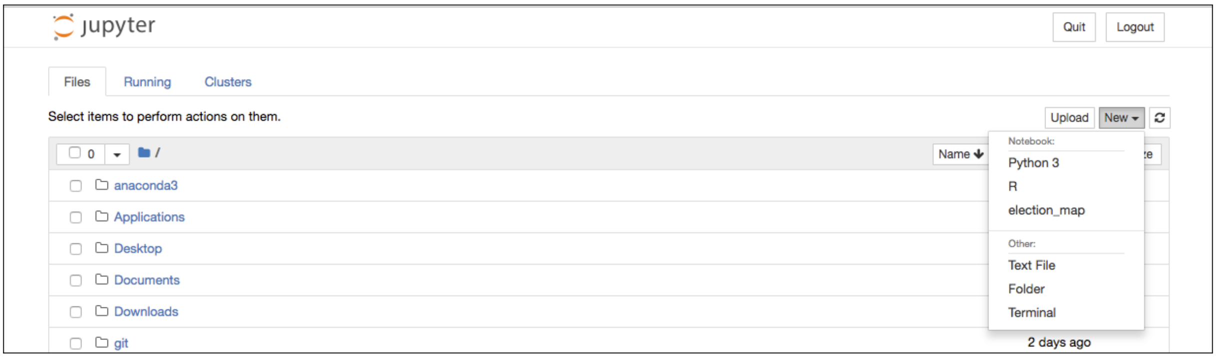
Nommez l'environnement *r-tutorial*. À côté de **Packages**, sélectionnez Python 3.7 et R. Sélectionnez r dans le menu déroulant. Cliquez sur **Create**:



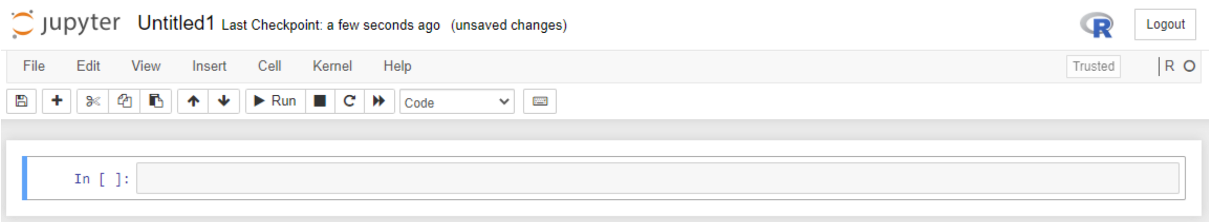
Ouvrez l'environnement avec le package R à l'aide de l'option **Open with Jupyter Notebook**:



Pour créer un nouveau bloc-notes pour le langage R, dans le menu **Jupyter Notebook**, sélectionnez **New**, puis sélectionnez **R**:

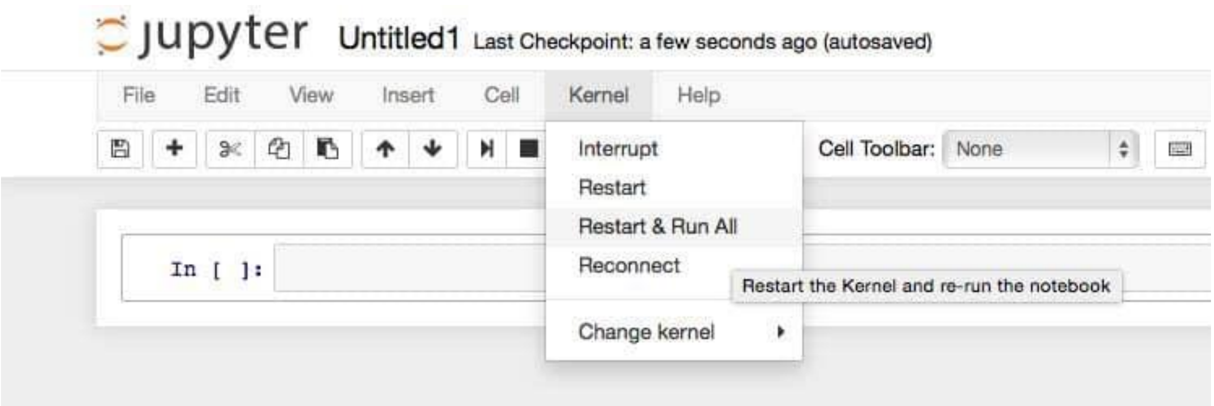


Vous aurez alors:



Il n'a plus qu'à écrire vos codes dans les cellules.

Notez que si vous voulez tout exécuter, il vous faudra aller dans:



4. Divers R (configuration, packages,…)

Contenu

Changer la langue de l'interface

Par défaut, R s'ouvrira dans la langue que vous avez choisie lors de l'installation. Avec les versions 3.0.2 et antérieures, il n'existe pas à ce jour de menu permettant de changer la langue après coup.

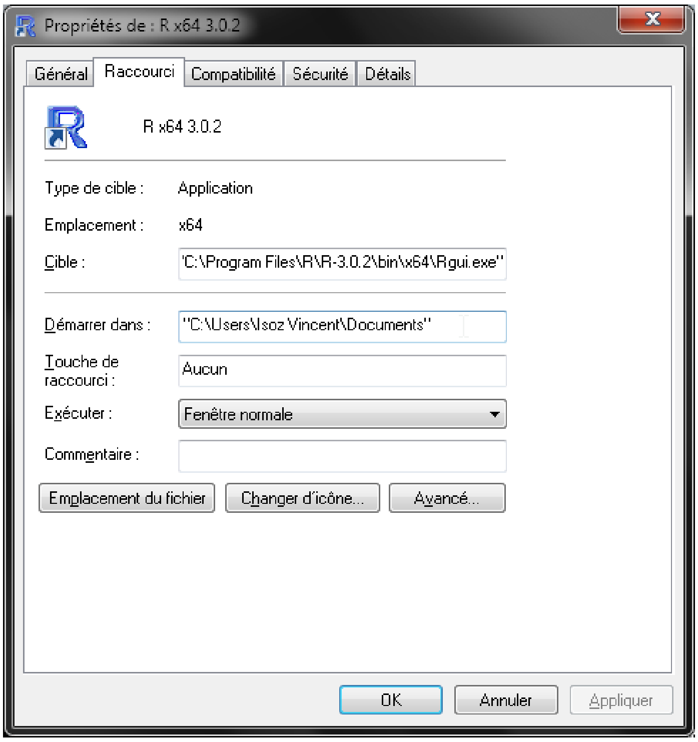
Dans les multinationales en Suisse, où l'on travaille souvent en 3 ou 4 langues et il est donc utile de savoir comment changer la langue de l'interface lorsque l'on rédige une documentation.

Voici la procédure:

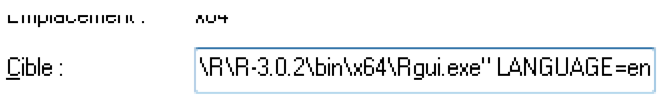
Créez une copie du raccourci de R:



Et dans les propriétés du raccourci:



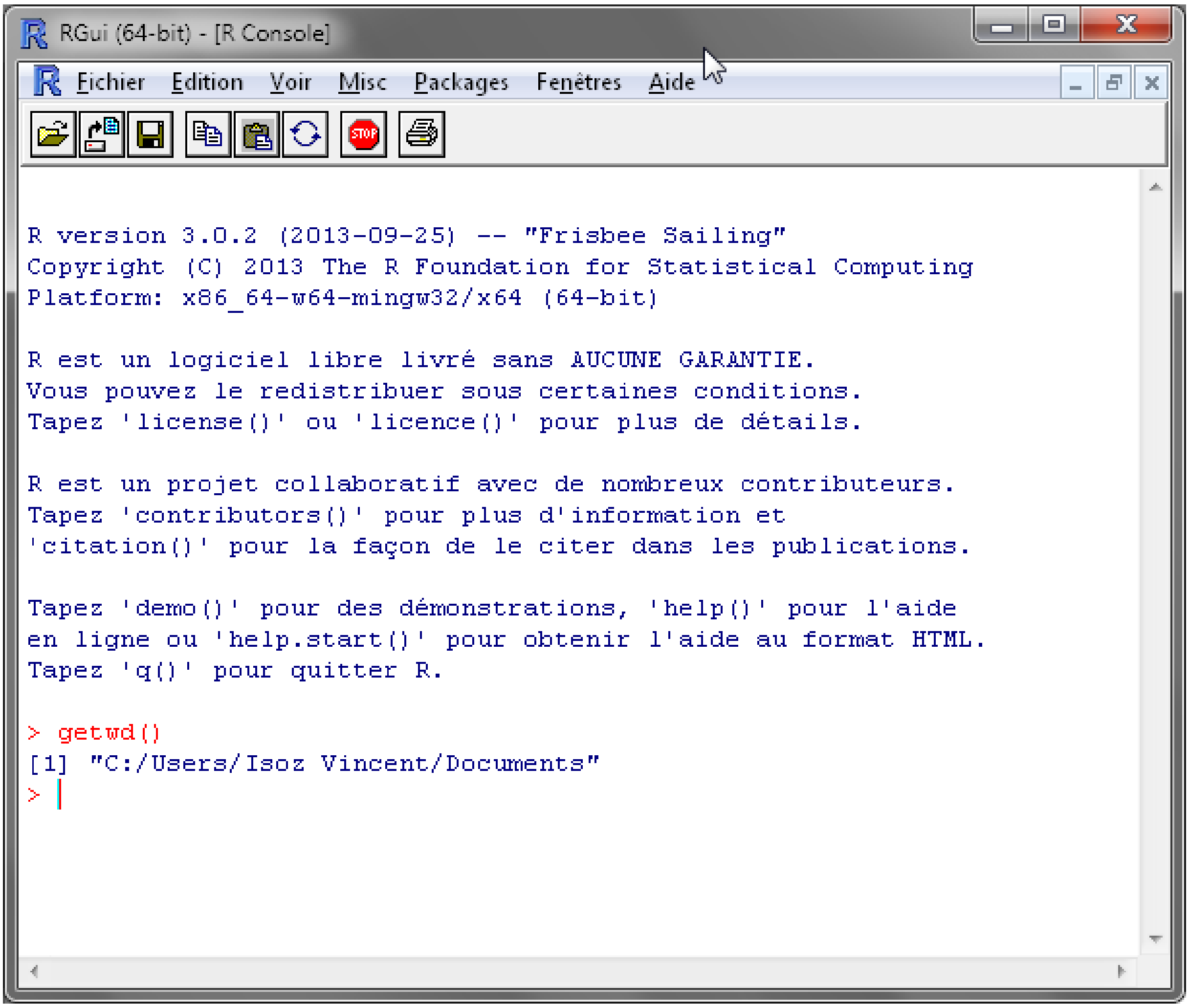
On change le paramètre cible en rajoutant un argument:



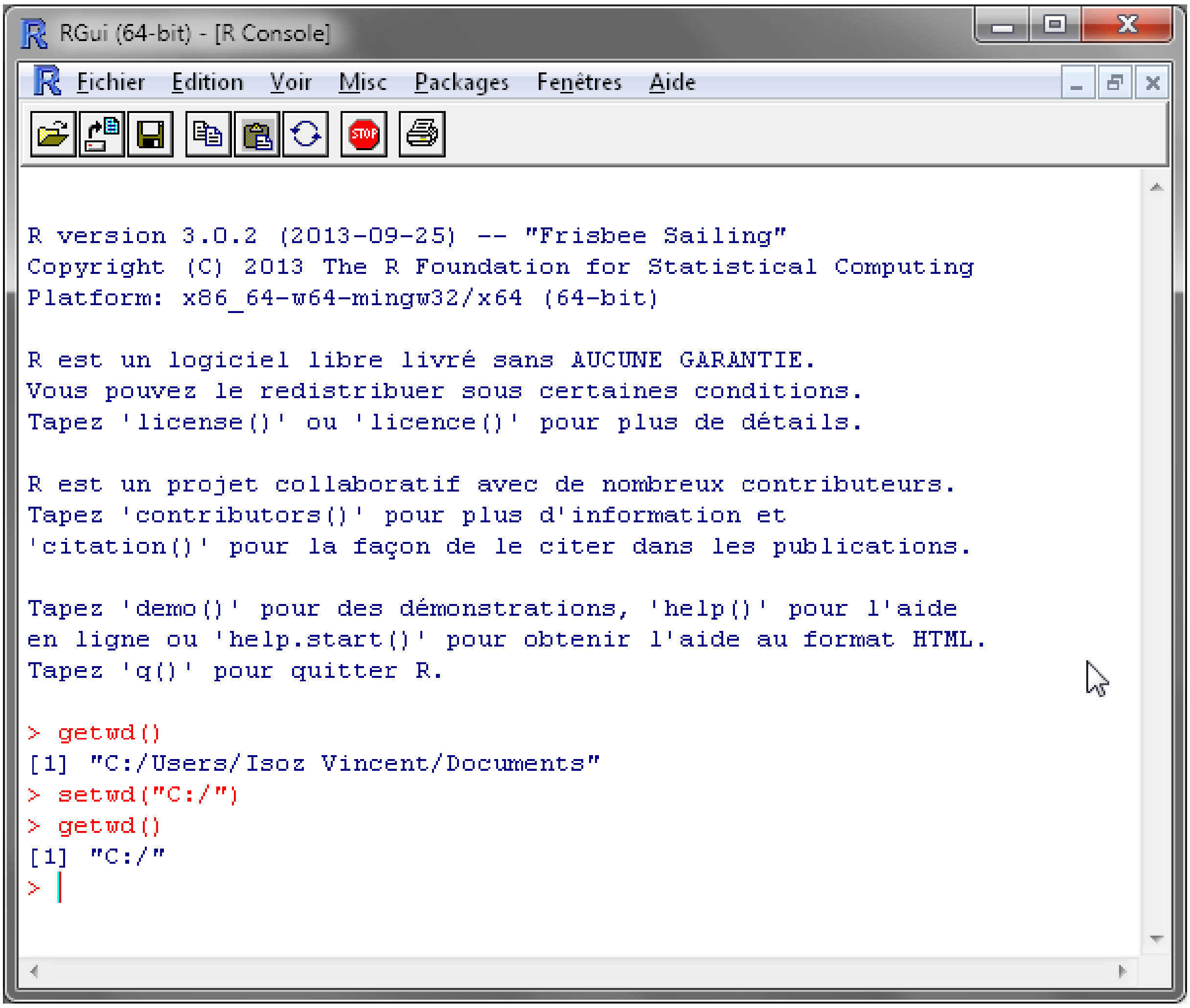
et voilà... c'est tout!

Changer le dossier de travail par défaut à chaque session

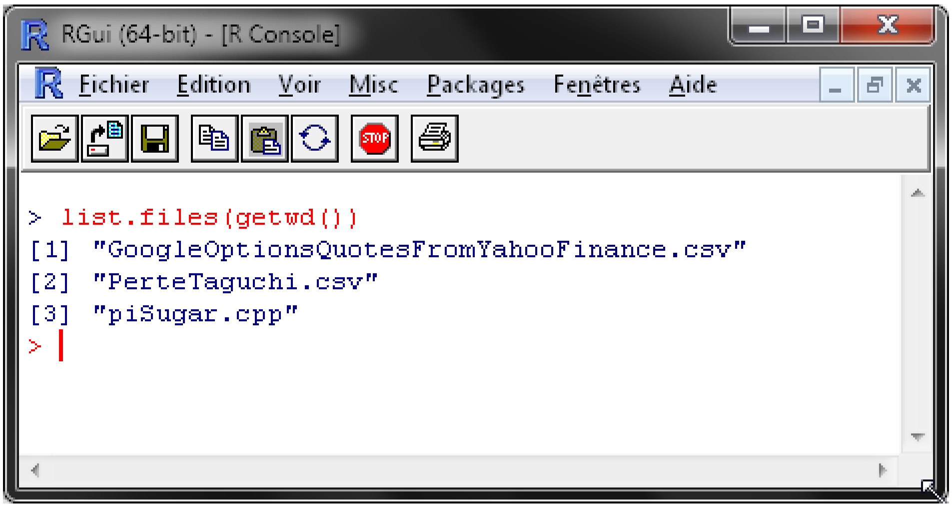
Le paramétrage le plus demandé en cours pour R est celui le changement du dossier de travail par défaut. Pour faire simple imaginons que nous allons mettre tous nos fichiers d'exercices à la racine du disque C:\. Dès lors, vous pouvez utiliser la commande **getwd( )** pour voir déjà quel est le dossier par défaut par curiosité:



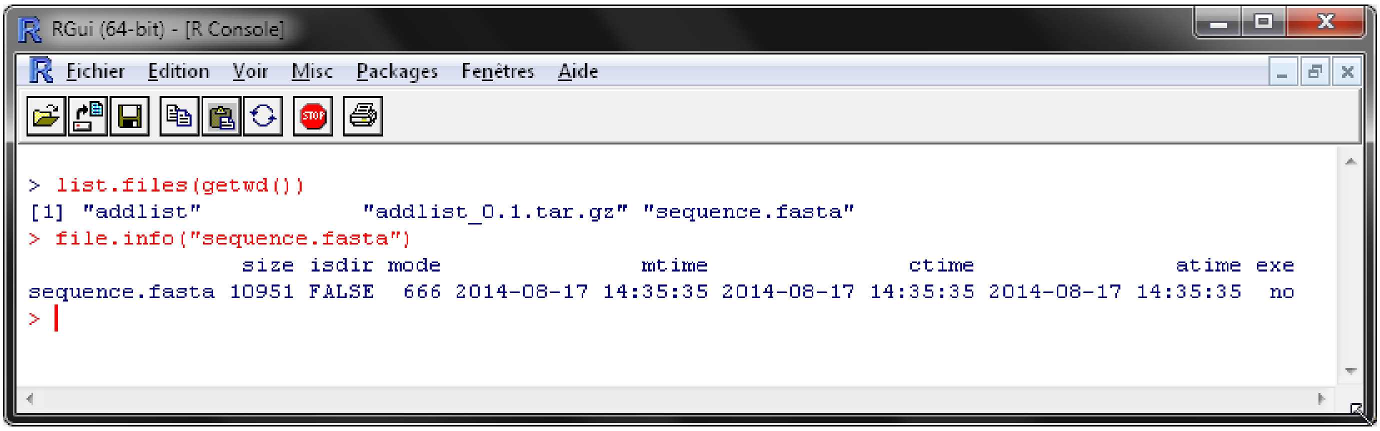
et ensuite vous pourrez le changer avec la commande **setwd( )** et vérifier que le nouveau dossier de travail par défaut a bien été changé:



Nous pouvons aussi voir les fichiers qu'il y dans le dossier de notre choix avec la commande **list.files( )**. Par exemple:

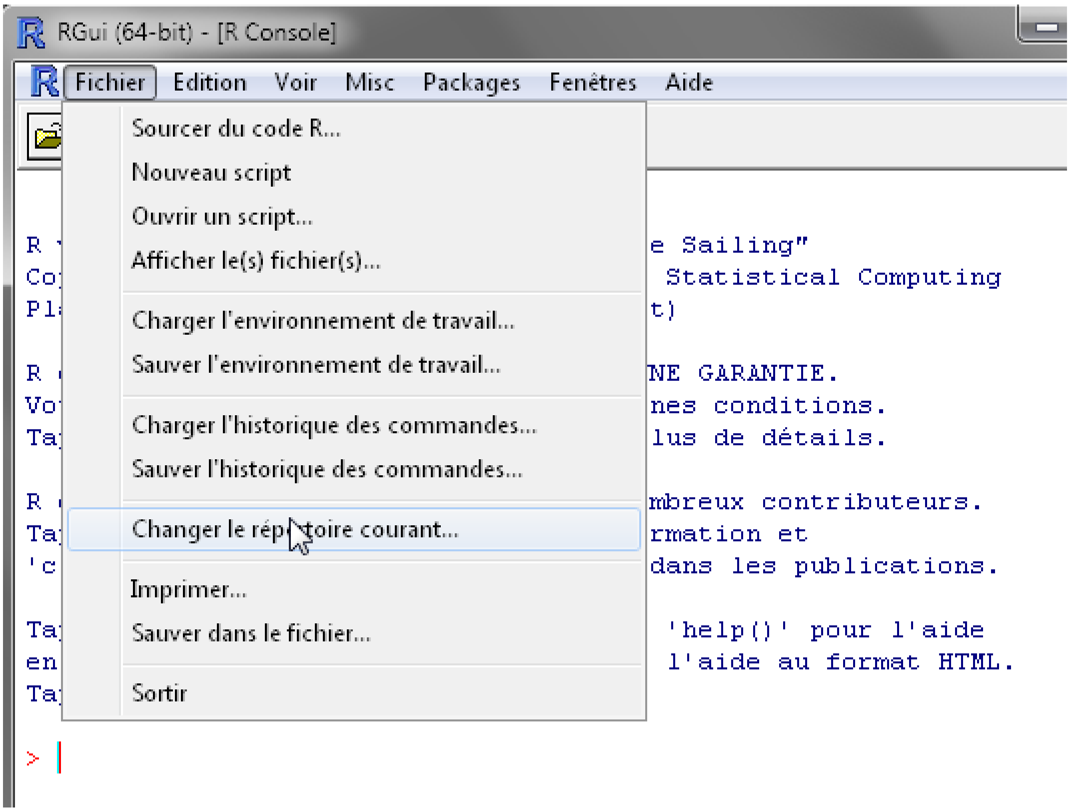


avec la commande **file.info( )** nous pouvons obtenir certaines informations utiles sur un fichier:

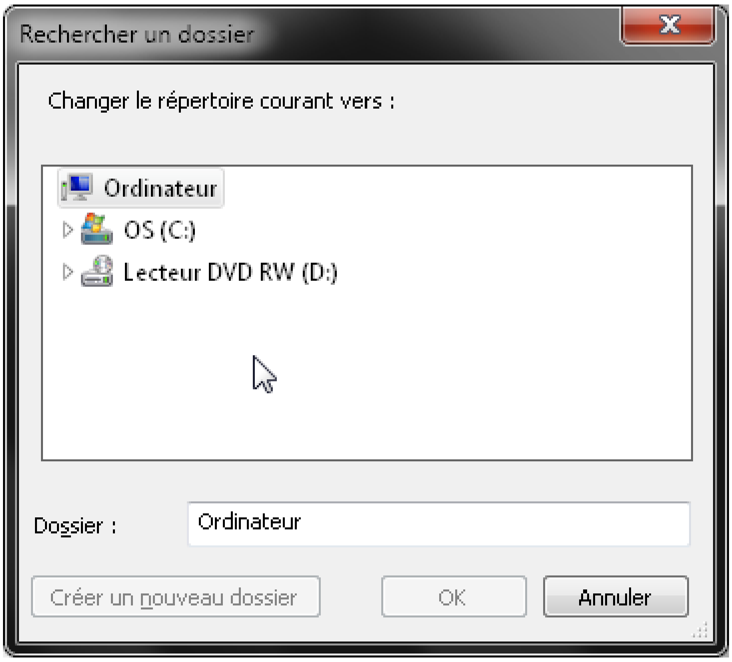


Changer définitivement le dossier de travail par défaut

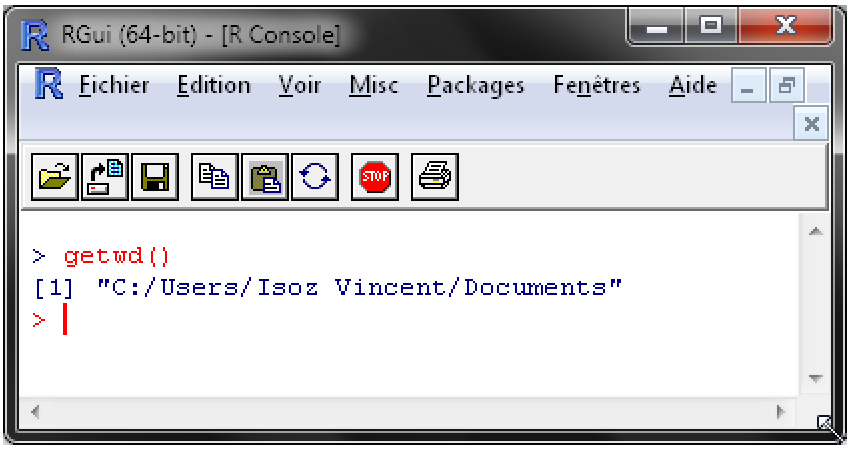
Pour changer définitivement le dossier de travail par défaut de R, vous pouvez vous rendre dans le menu **Fichier/Changer le répertoire courant**:



et vous faites votre choix:



Mais lorsque vous fermerez et ouvrirez à nouveau R, il aura oublié ce changement. Vous pouvez vérifier cela en tapant la commande **getwd( )**:



Dès lors, la méthode consiste à créer un fichier .Rprofile dans le dossier de démarrage par défaut avec la commande suivante au début:



Nous verrons tout au long de ce livre l'intérêt d'avoir au fait un fichier .Rprofile ayant plutôt le contenu suivant:

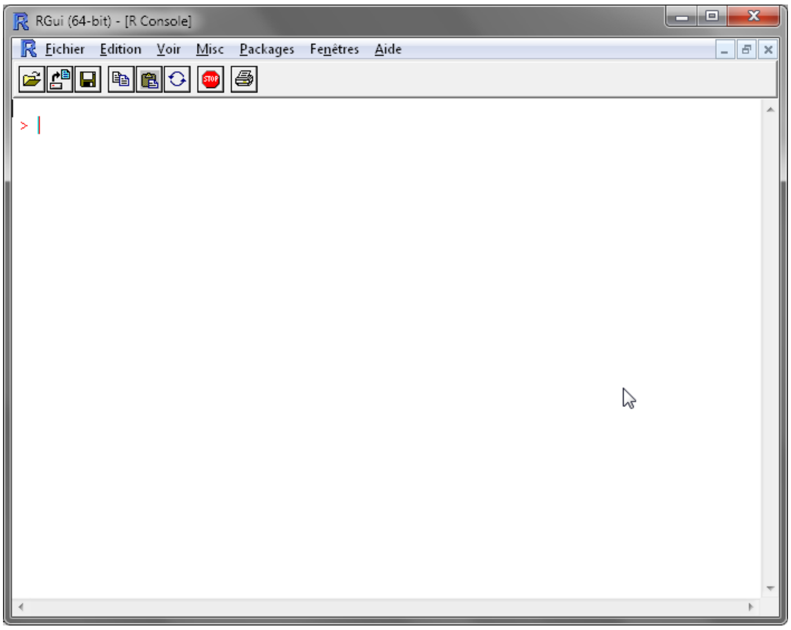
.Rprofile - Notepad

File Edit Format View Help

```
setwd("C:/tmp")
options(width=200)
#options(scipen=-100,digits=4)
options(show.signif.stars=FALSE)
#options(prompt="R> ")
qq<-function(){q(save="no")} #pour éviter à devoir fermer avec le message de sauvegarde de l'espace de travail
print(date())
#.libPaths("D:/Systems/R_packages")
set.seed(1234) #pour rendre les expérience aléatoire reproductibles
#definition of the default CRAN Mirror
r <- getOption("repos")
r["CRAN"] <- "http://cran.cnr.berkeley.edu/"
options(repos = r)
#we open the wd in the explorer
suppressWarnings(shell(paste("explorer",gsub('/','\\\\',getwd()))))
```

Vider la fenêtre de session

Pour vider la fenêtre de commande, sur PC il faut faire **Ctrl+L**. Vous obtiendrez alors une fenêtre toute vide (de toute façon ce raccourci est indiqué dans le menu **Edition**):



Pour "vider" (vider est un bien grand mot… on laissera le lecteur vérifier sur son écran!) la fenêtre en mode Shell Microsoft DOS, cette dernière commande ne fonctionne pas! Le mieux est d'utiliser la fonction **mise( )** du package **mise** (qui au passage, pour information, efface aussi par défaut toutes les variables de la mémoire!):



Version de R et autres informations

Pour connaître sa version de R et avoir quelques informations pertinentes sur son installation, on peut toujours exécuter la commande suivante, que l'on retrouve fréquemment dans les documentations de scripts R pour des raisons de reproductibilité, **sessionInfo( )** :

```
> library("data.table")
> library("dplyr")
> sessionInfo()
R version 3.6.3 (2020-02-29)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18363)

Matrix products: default

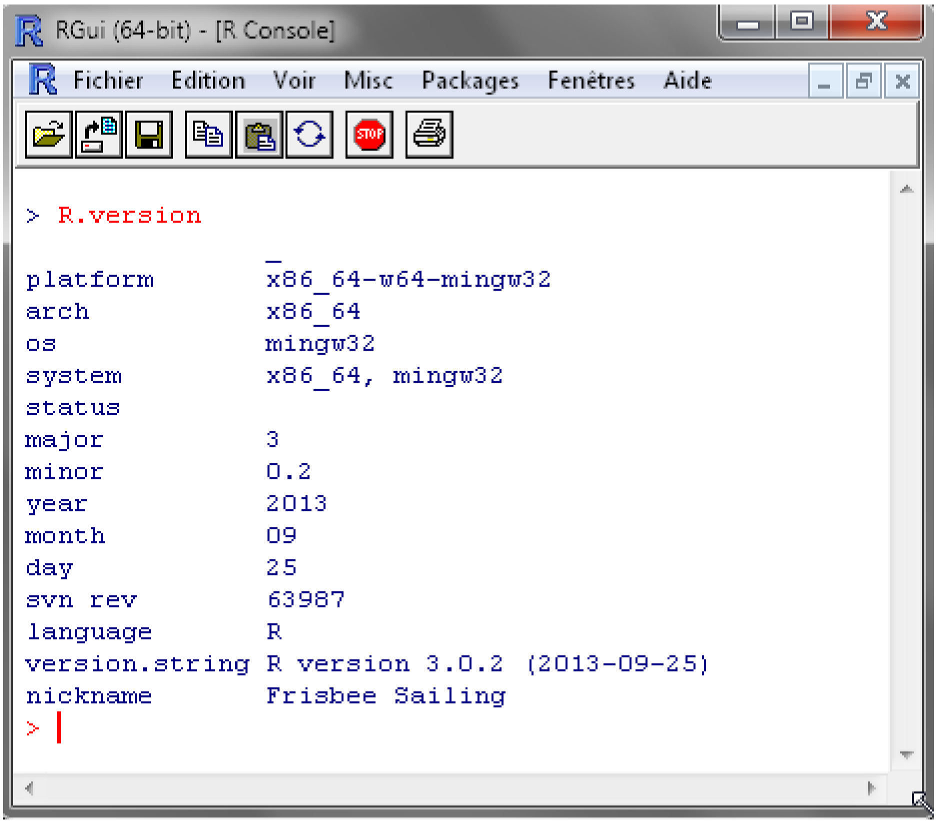
locale:
[1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

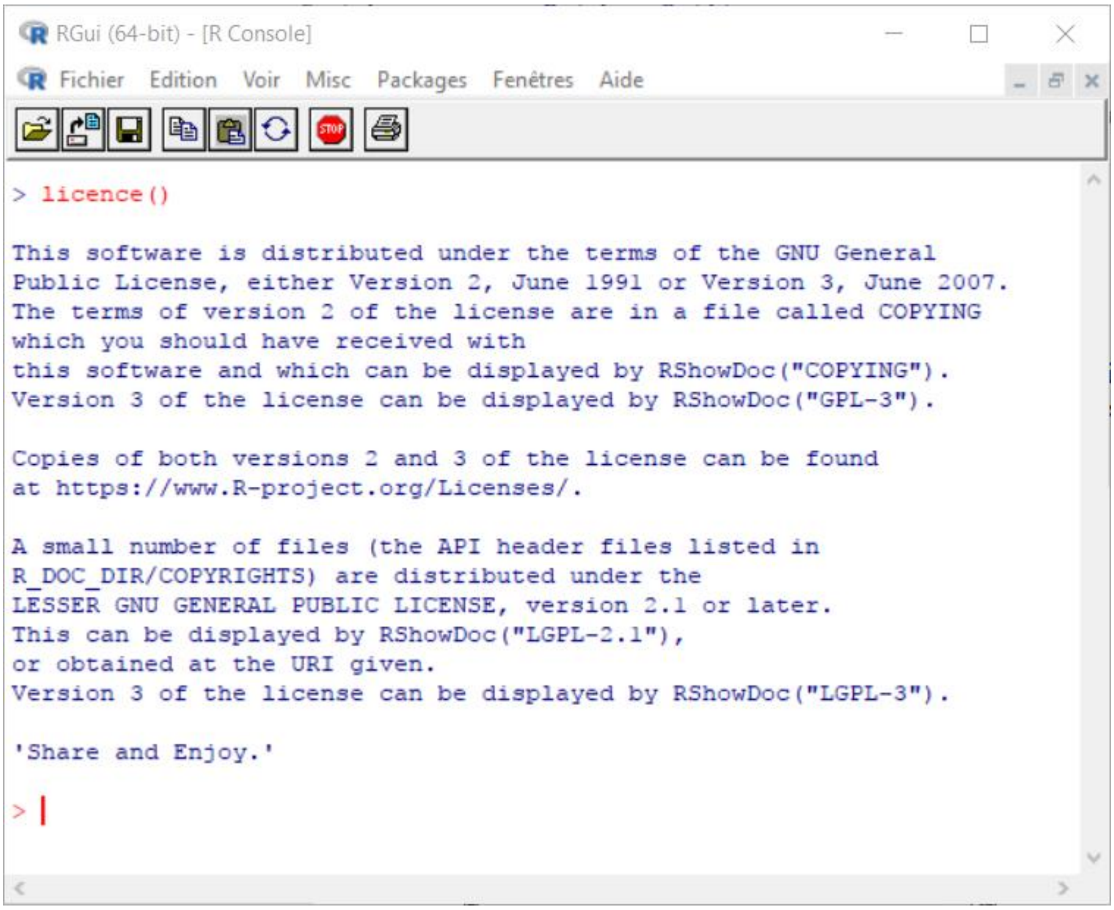
other attached packages:
[1] dplyr_0.8.5       data.table_1.12.8

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.4.6     crayon_1.3.4     assertthat_0.2.1 R6_2.4.1
 [5] lifecycle_0.2.0  magrittr_1.5     pillar_1.4.3     rlang_0.4.5
 [9] vctrs_0.2.4      ellipsis_0.3.0   glue_1.4.0       purrr_0.3.4
[13] compiler_3.6.3   pkgconfig_2.0.3  tidyselect_1.0.0 tibble_3.0.1
>
```

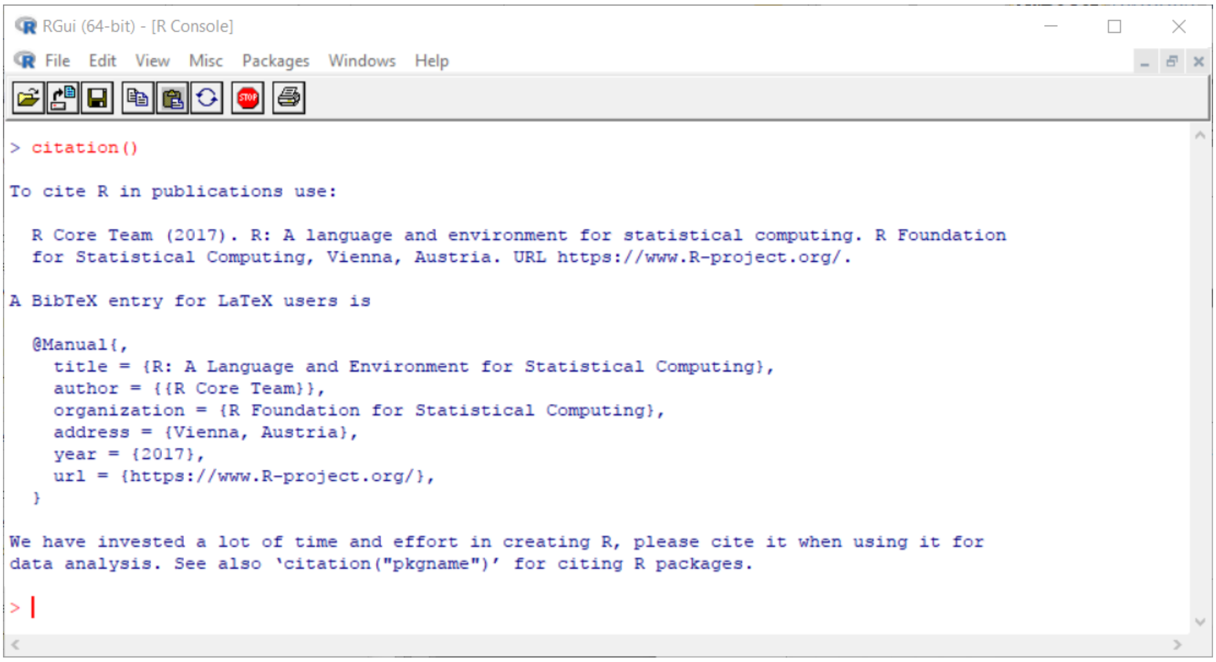
ou pour avoir plus d'informations en utilisant la commande **R.version**:



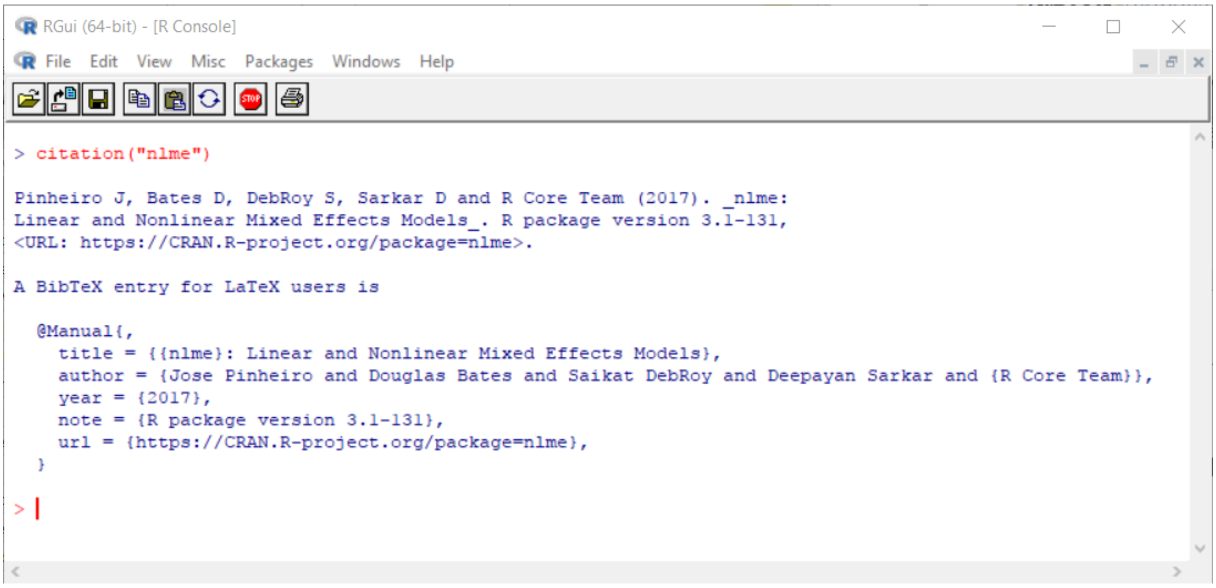
Et pour obtenir les informations concernant la licence, nous utilisons la fonction **licence( )**:



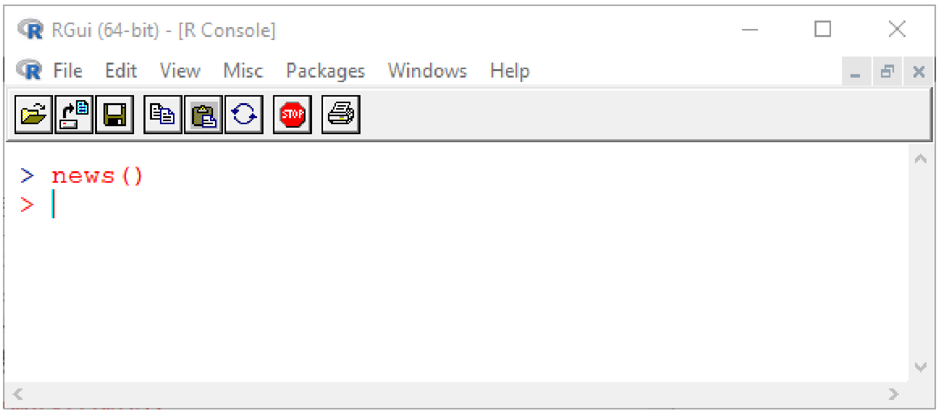
Ou pour citer le logiciel dans vos publications scientifiques, travail de mémoire ou de thèse avec la fonction **citation( )**:



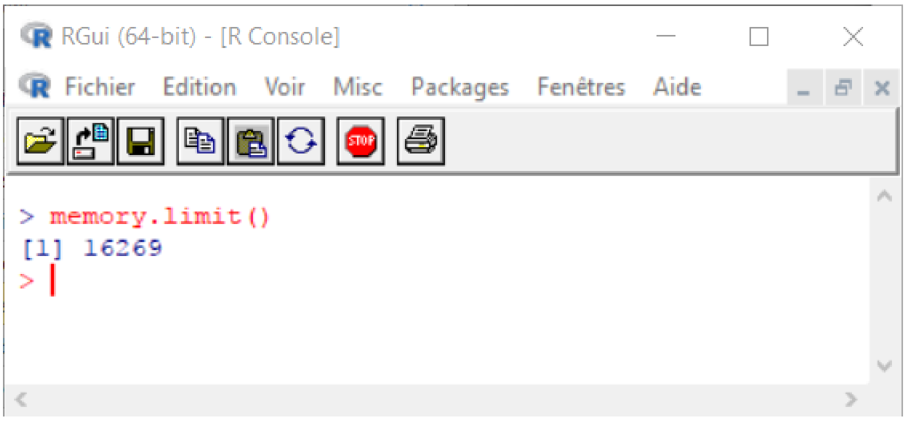
ou encore, citer un package:



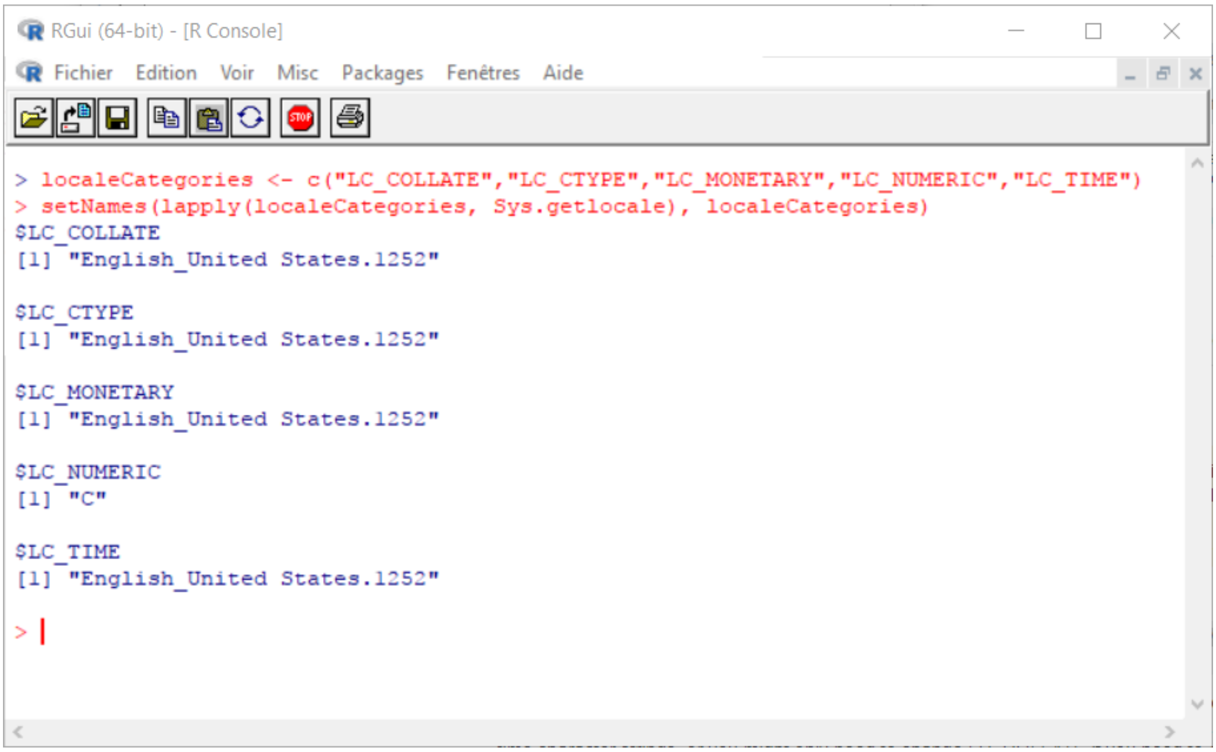
On peut aussi consulter les dernières nouveautés avec la commande **news( )**:



Ou encore le total de mémoire vive installé avec **memory.limit( )**:



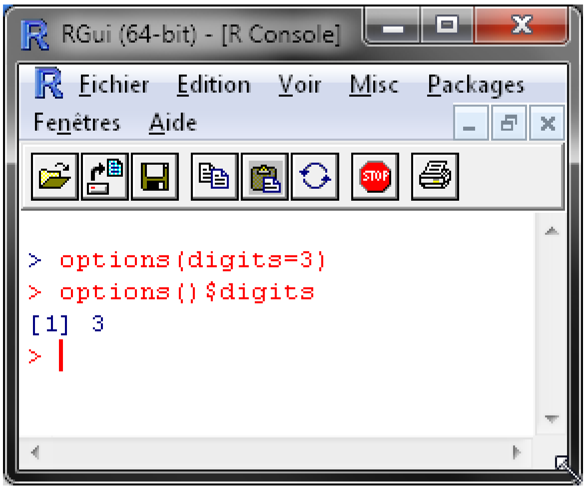
ou encore certains paramètres systèmes en utilisation une transformation de liste (**lapply( )**) sur la base de la commande système **Sys.getlocale( )**:



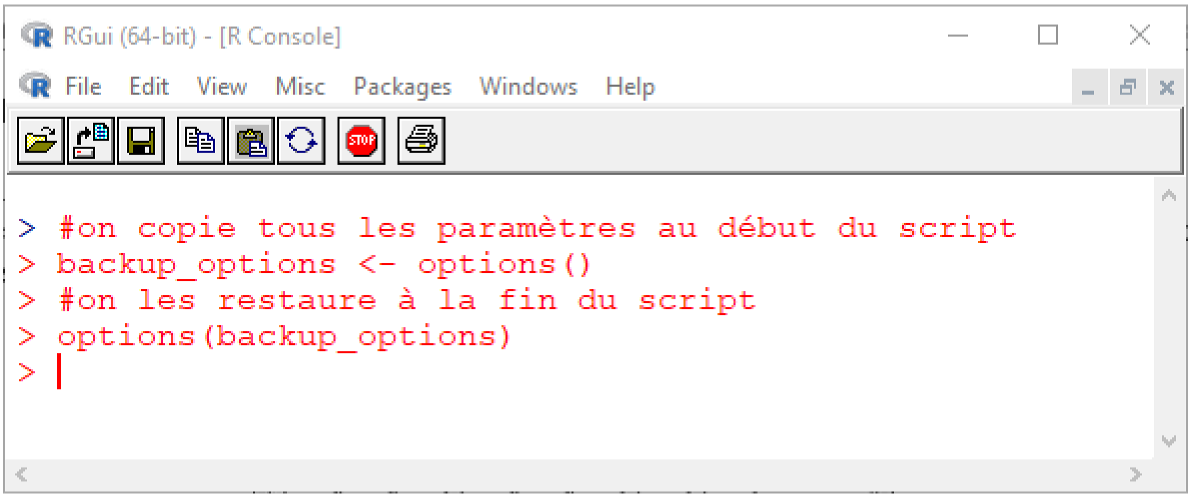

DID YOU UPDATE R?
YES
WHAT DID IT COST?
Warning in install.packages :
package 'memer' is not available
(for R version 3.6.1)
> library("memer")
Error in library("memer") : there
is no package called 'memer'
>
EVERYTHING
imgflip.com

Définir le nombre de décimales

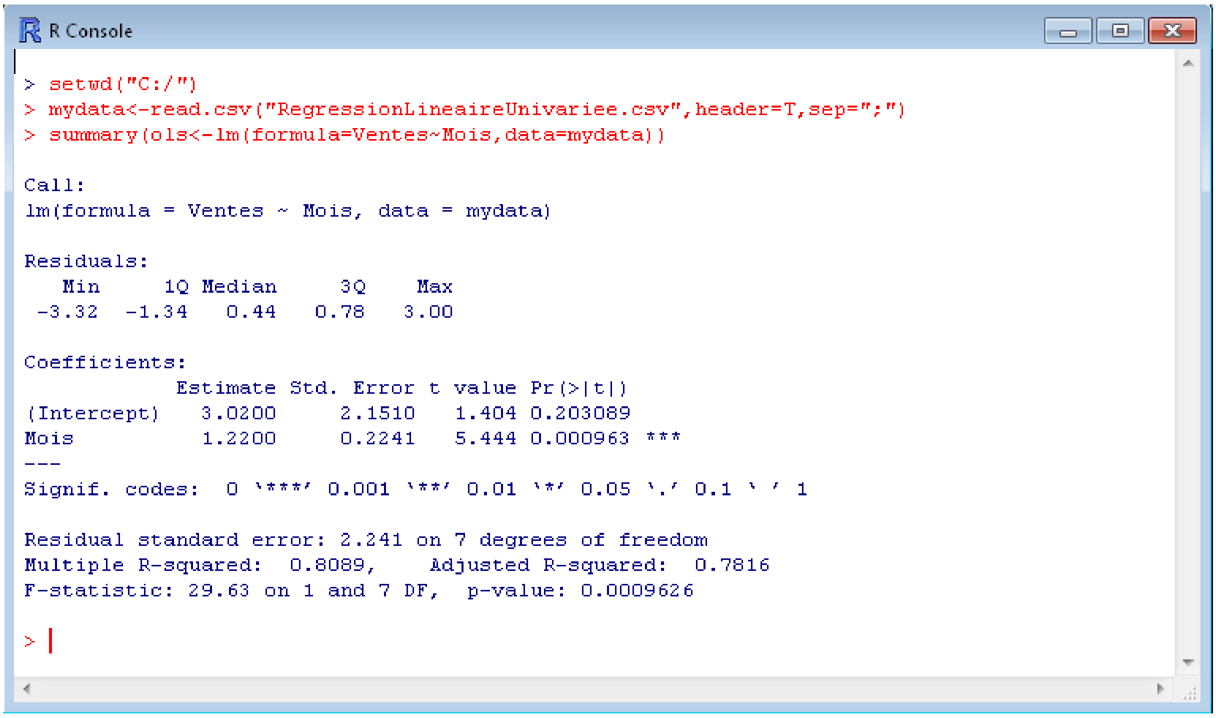
En général quand vous ferez des calculs dans R vous voudrez une certaine précision des calcules. Pour ce faire, vous pouvez utiliser la commande **options(digits=)**:



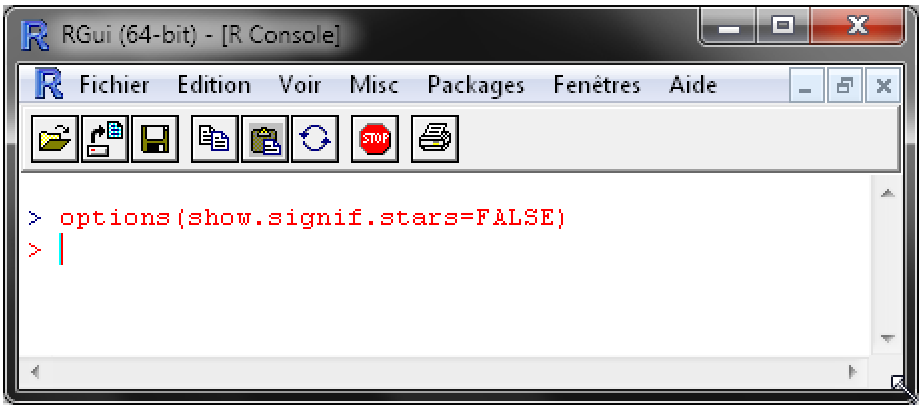
Remarque: Notez que souvent aurez des scripts dans lesquels vous modifierez des paramètres systèmes de R et, où la fin, vous souhaiterez les restaurer! Vous pouvez alors utiliser la fonctions **options( )** générale comme ci-dessous:



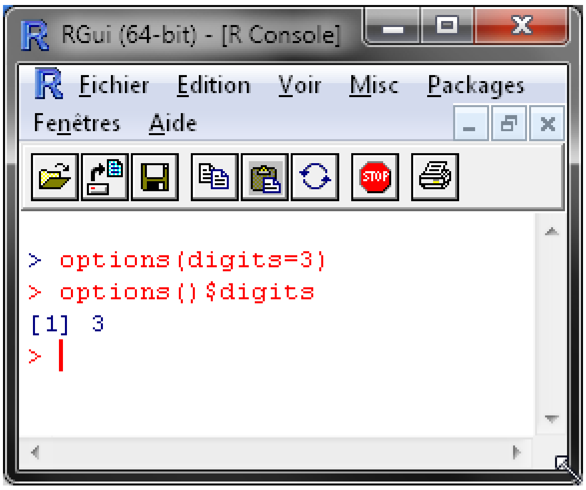
Parfois pour certains tests statistiques quand la *p*-value est très petite, R renvoie des étoiles (*) comme nous le verrons lors de notre étude de la régression linéaire univariée (par exemple...):



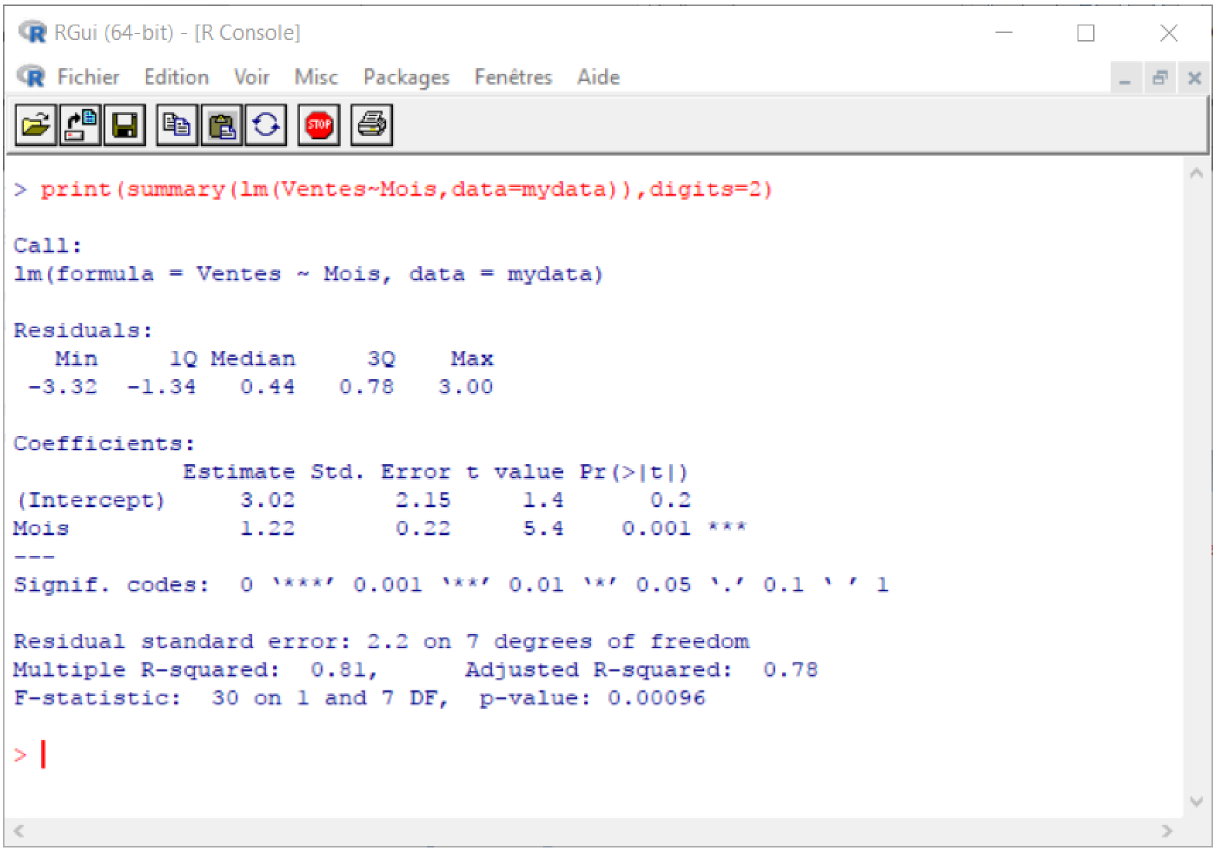
Pour désactiver cela, il suffit dans R, d'écrire:



Enfin, certaines analyses comme l'ANOVA ci-dessus le paramétrage:



n'aura aucun effet! Il faut alors par exemple écrire:

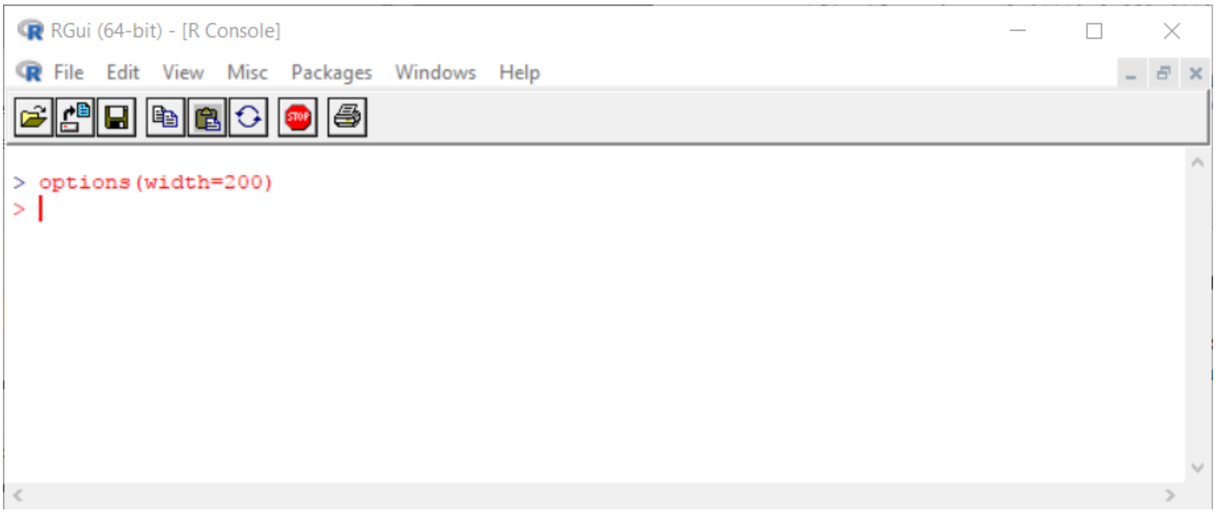


A noter que cela ne marche pas pour tout…! Dans des cas compliqués ont doit parfois même utiliser des syntaxes du type suivant (voir l'exemple à la page 1701):

noquote(format(round(truc à formater,3), format="f"))

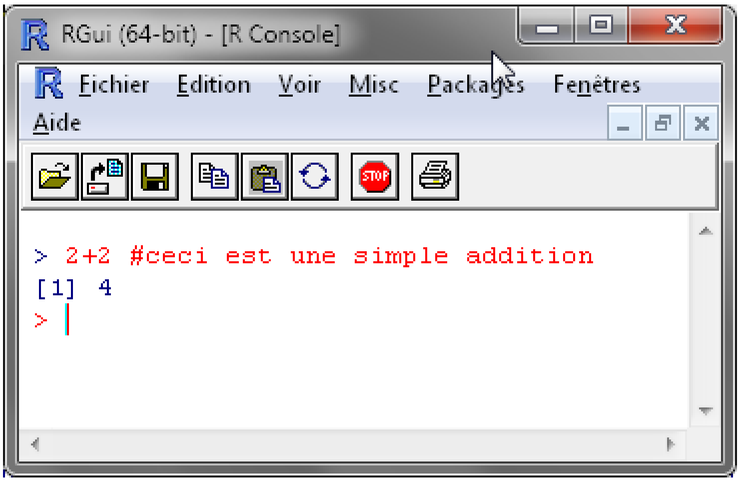
Définir la largeur de la sortie

Parfois certaines personnes ne sont pas satisfaites de la largeur de sortie des tables qu'elles affichent dans la console R. Pour changer la largeur de sortie on utilisera la commande **options(width=…)** où le nombre après le signe égal représente la largeur en caractères:

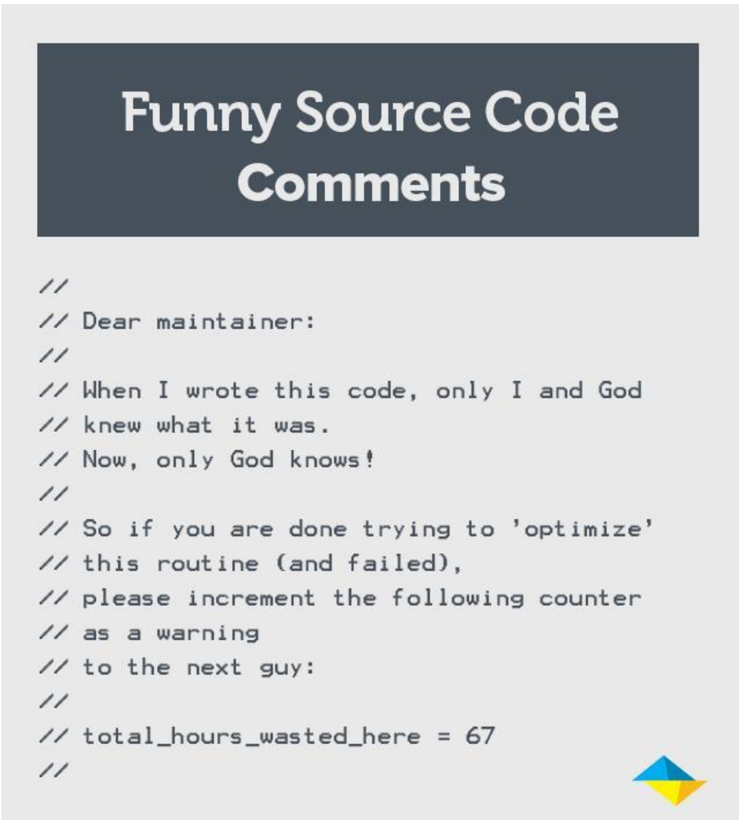


Ajouter des commentaires

Lorsque vous écrivez un paquet de commandes il peut être très utile de commenter vos manipulations (l'usage est d'avoir normalement un commentaire au moins pour trois lignes de commandes). Par exemple:



Les commentaires du code est un point très important du développement que l'on comprend souvent seulement avec l'expérience. Effectivement, les développeurs débutants n'ont dans un premier temps pas l'habitude de travailler très rigoureusement et très proprement et ce d'autant plus sur des codes rarement supérieurs à 1000 lignes et ne voient donc pas quelle est l'intérêt futur de bien commenter leur code. Cet état des faits a lieu chez la grande majorité des développeurs.



Ne pas commenter est une énorme erreur pour le développeur lui-même et tous ceux qui seraient amenés à intervenir ou à poursuivre son travail.

Certaines règles sont à mettre en place il convient immédiatement de mettre en pratique dès que l'on commence à rédiger un code. Voici ces règles:

R1. Toute procédure, fonction, classe, doit être accompagnée d'une cartouche de description telle que dans l'exemple ci-dessous

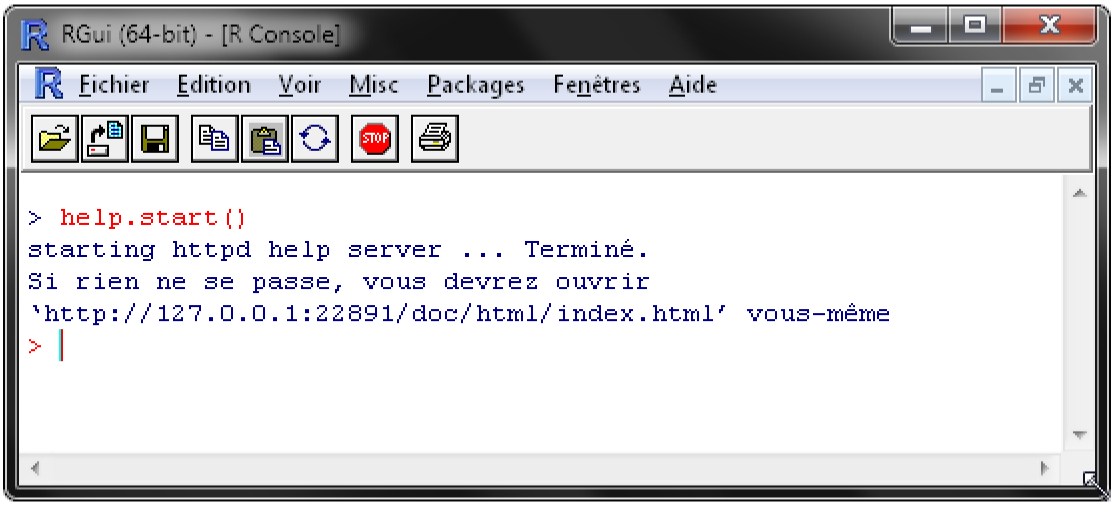
R2. Chaque ligne de code doit être commentée avec indication en initiales du commentateur et de la date de création du commentaire tel que dans l'exemple ci-dessous

R3. Au besoin, un schéma procédural doit être fait dans un logiciel adapté (Microsoft Visio suffit) pendant le travail afin de savoir qui appelle quoi en faisant usage de quelles variables

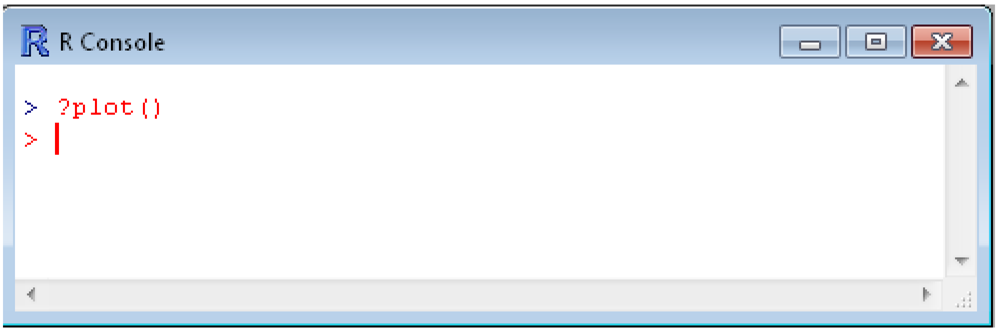
Obtenir de l'aide sur une commande ou sur un package

Contrairement à des logiciels comme Microsoft Excel, Microsoft Word, Microsoft PowerPoint, Microsoft Outlook, Minitab, SPSS et autres classiques du genre, il y a beaucoup trop d'outils dans R pour que quiconque puisse les retenir tous (un expert connaître maximum 4'000 à 5'000 commandes par coeur).

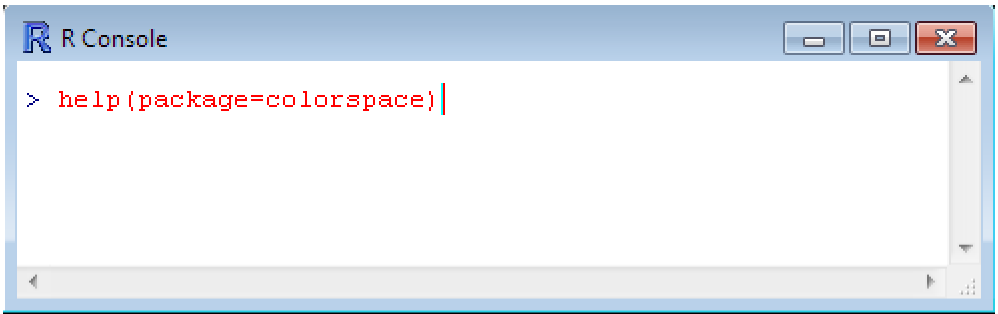
Nous pouvons lancer l'aide générale ave la commande **help.start( )** qui ouvrira la page web correspondante:



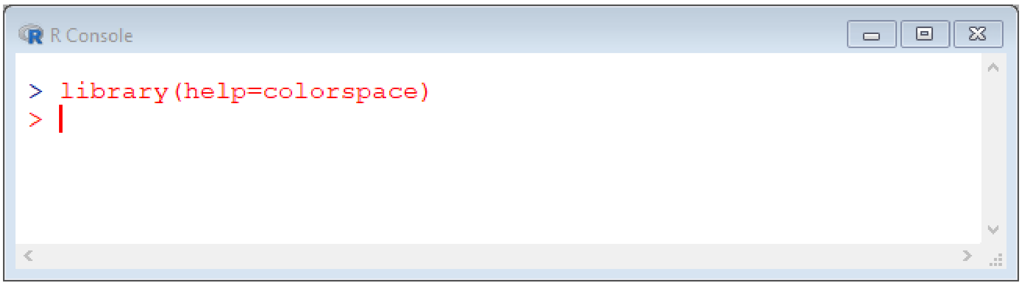
Pour obtenir de l'aide sur une commande qui vous semble mystérieuse, voici comment faire (du moins quand il y a une documentation...):



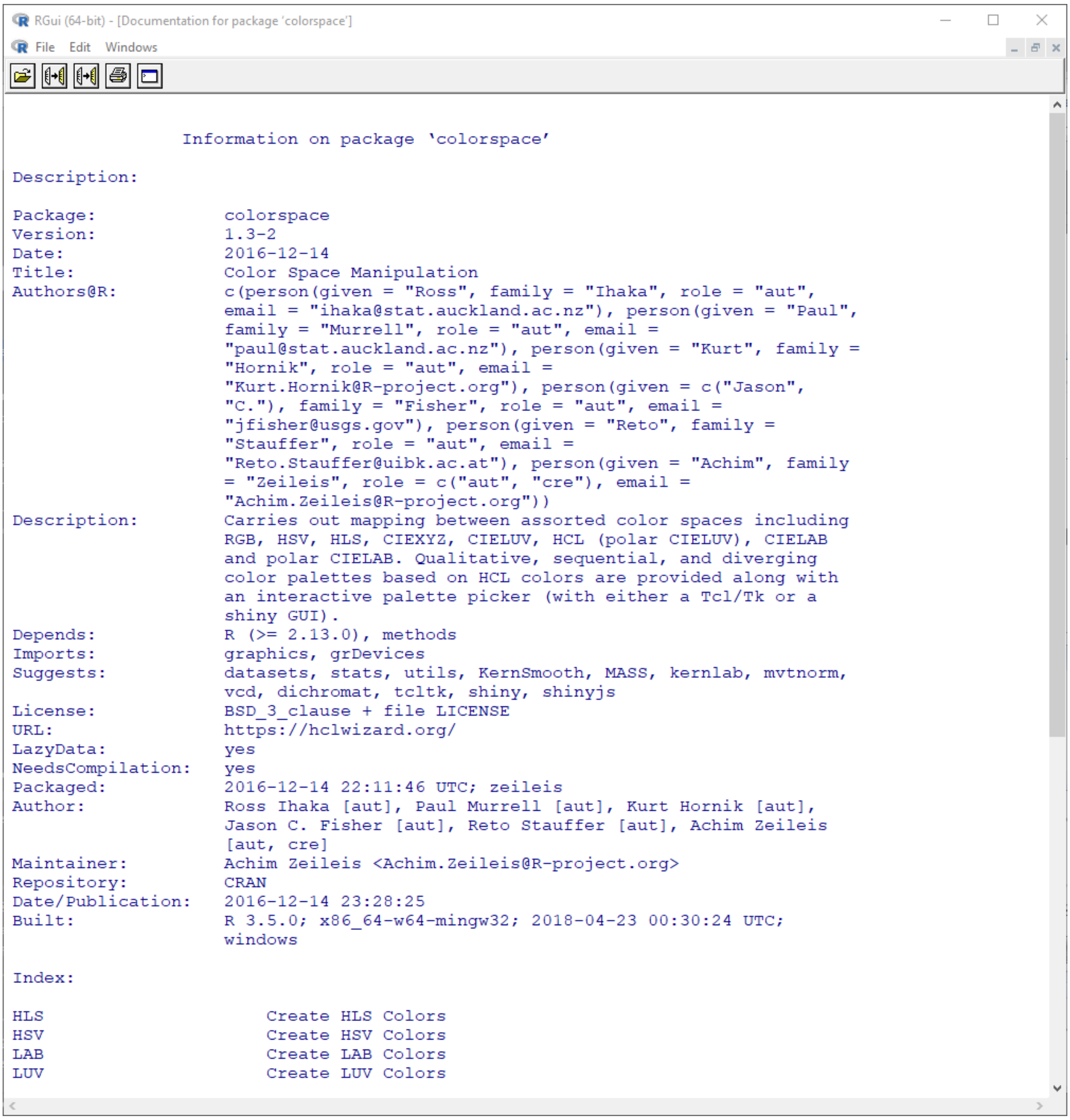
et pour un package on utilisera la commande **help( )**:



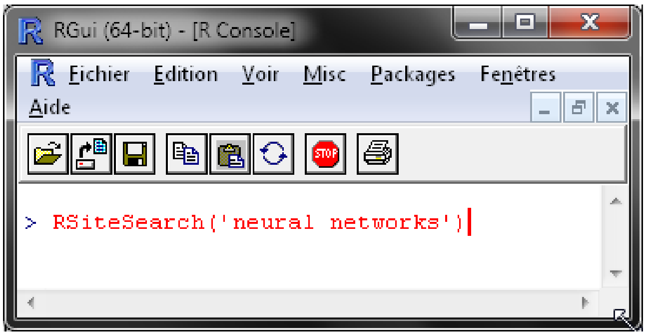
Qui va ouvrir l'aide locale dans le navigateur Internet. Si vous n'aimez pas vous pouvez utiliser:



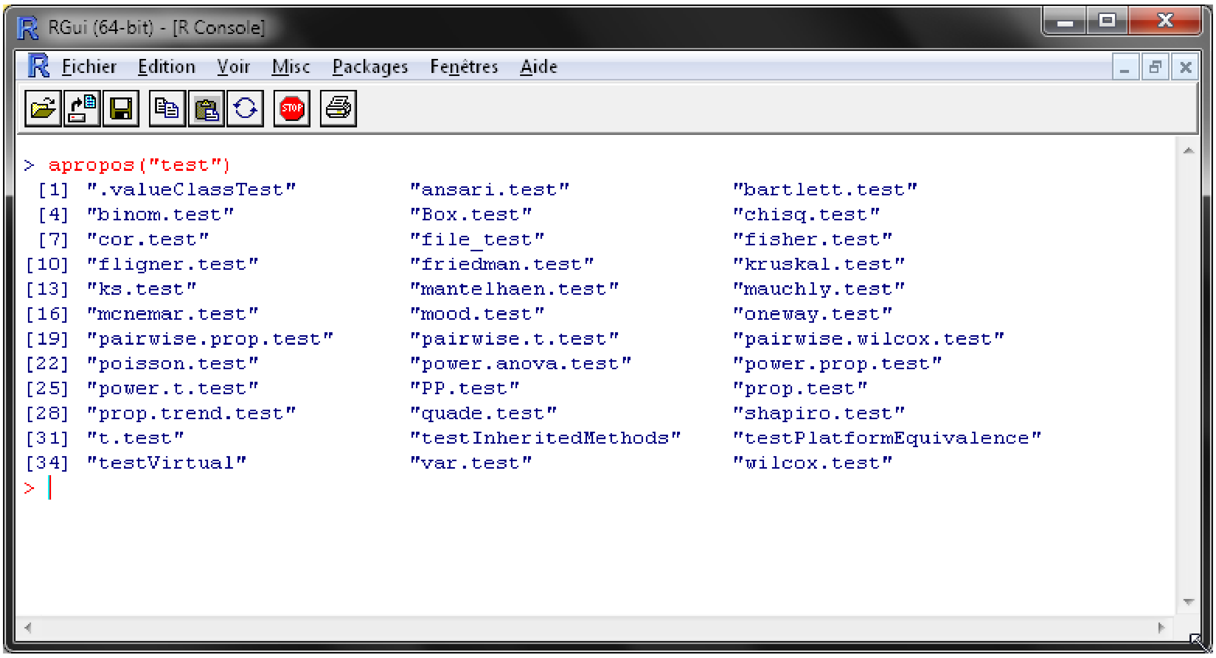
Qui ouvrira l'aide dans une fenêtre graphique de R comme visible ci-dessous:



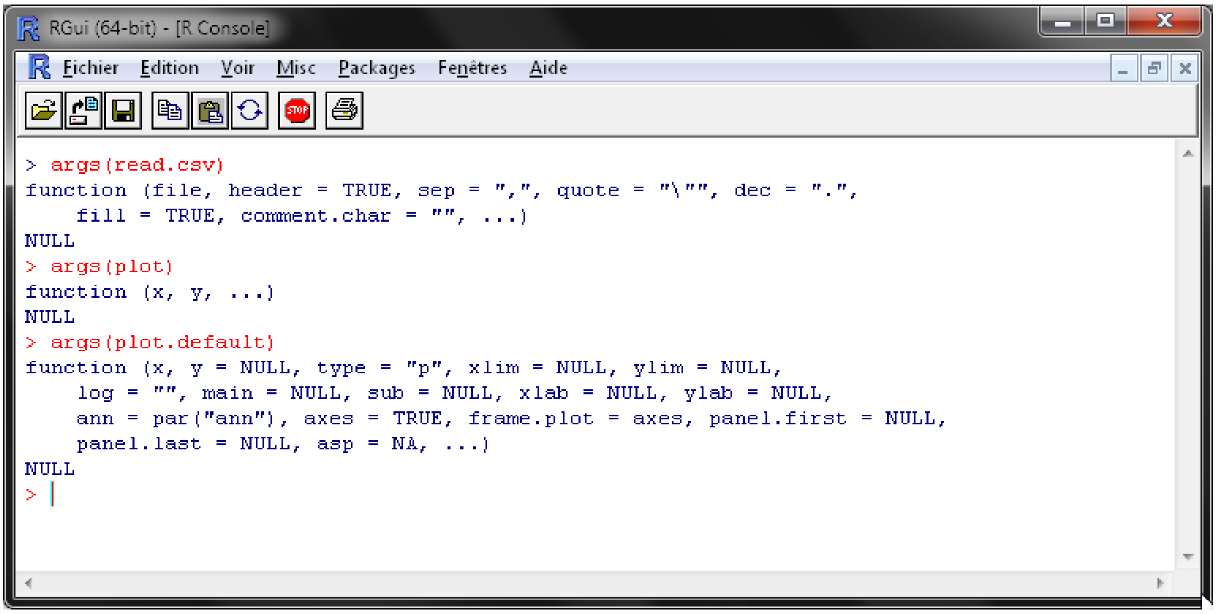
Ou pour chercher une expression dans la base de connaissances, on utilisera la fonction **RSiteSearch( )**:



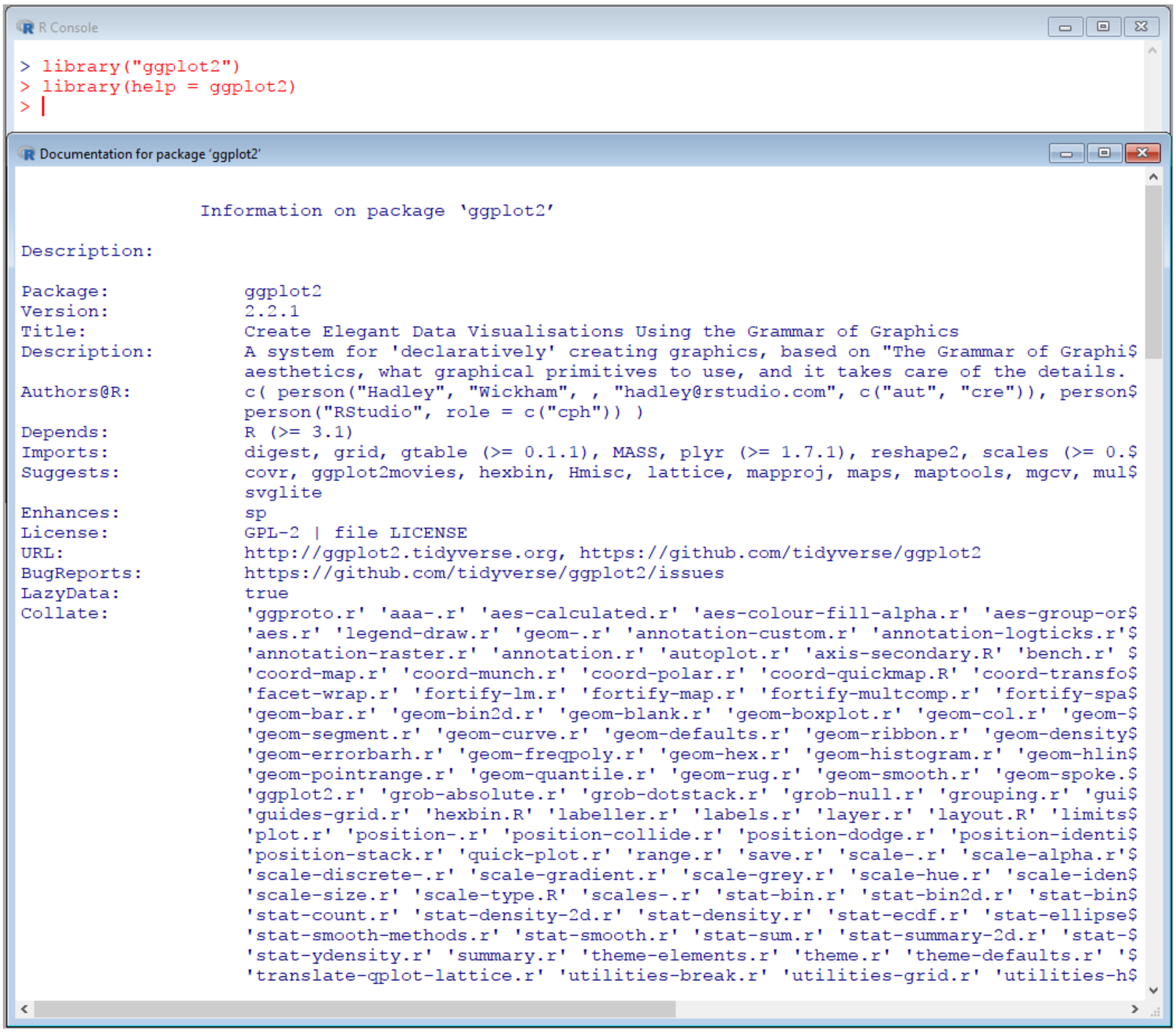
ou obtenir la liste de toutes les commandes qui contiennent un mot particulier avec la fonction **apropos( )**:



Nous pouvons aussi obtenir un rappel (aide) des arguments de certaines commandes et fonctions avec **args( )**:



Si on charge un package on peut aussi avoir plein d'informations en écrivant:



```
R Console
> library("ggplot2")
> library(help = ggplot2)
>

Documentation for package 'ggplot2'

                Information on package 'ggplot2'

Description:

Package:            ggplot2
Version:            2.2.1
Title:              Create Elegant Data Visualisations Using the Grammar of Graphics
Description:        A system for 'declaratively' creating graphics, based on "The Grammar of Graphi$
                    aesthetics, what graphical primitives to use, and it takes care of the details.
Authors@R:          c( person("Hadley", "Wickham", , "hadley@rstudio.com", c("aut", "cre")), person$
                    person("RStudio", role = c("cph")) )
Depends:            R (>= 3.1)
Imports:            digest, grid, gtable (>= 0.1.1), MASS, plyr (>= 1.7.1), reshape2, scales (>= 0.$
Suggests:           covr, ggplot2movies, hexbin, Hmisc, lattice, mapproj, maps, maptools, mgcv, mul$
                    svglite
Enhances:           sp
License:            GPL-2 | file LICENSE
URL:                http://ggplot2.tidyverse.org, https://github.com/tidyverse/ggplot2
BugReports:         https://github.com/tidyverse/ggplot2/issues
LazyData:           true
Collate:            'ggproto.r' 'aaa-.r' 'aes-calculated.r' 'aes-colour-fill-alpha.r' 'aes-group-or$
                    'aes.r' 'legend-draw.r' 'geom-.r' 'annotation-custom.r' 'annotation-logticks.r'$
                    'annotation-raster.r' 'annotation.r' 'autoplot.r' 'axis-secondary.R' 'bench.r' $
                    'coord-map.r' 'coord-munch.r' 'coord-polar.r' 'coord-quickmap.R' 'coord-transfo$
                    'facet-wrap.r' 'fortify-lm.r' 'fortify-map.r' 'fortify-multcomp.r' 'fortify-spa$
                    'geom-bar.r' 'geom-bin2d.r' 'geom-blank.r' 'geom-boxplot.r' 'geom-col.r' 'geom-$
                    'geom-segment.r' 'geom-curve.r' 'geom-defaults.r' 'geom-ribbon.r' 'geom-density$
                    'geom-errorbarh.r' 'geom-freqpoly.r' 'geom-hex.r' 'geom-histogram.r' 'geom-hlin$
                    'geom-pointrange.r' 'geom-quantile.r' 'geom-rug.r' 'geom-smooth.r' 'geom-spoke.$
                    'ggplot2.r' 'grob-absolute.r' 'grob-dotstack.r' 'grob-null.r' 'grouping.r' 'gui$
                    'guides-grid.r' 'hexbin.R' 'labeller.r' 'labels.r' 'layer.r' 'layout.R' 'limits$
                    'plot.r' 'position-.r' 'position-collide.r' 'position-dodge.r' 'position-identi$
                    'position-stack.r' 'quick-plot.r' 'range.r' 'save.r' 'scale-.r' 'scale-alpha.r'$
                    'scale-discrete-.r' 'scale-gradient.r' 'scale-grey.r' 'scale-hue.r' 'scale-iden$
                    'scale-size.r' 'scale-type.R' 'scales-.r' 'stat-bin.r' 'stat-bin2d.r' 'stat-bin$
                    'stat-count.r' 'stat-density-2d.r' 'stat-density.r' 'stat-ecdf.r' 'stat-ellipse$
                    'stat-smooth-methods.r' 'stat-smooth.r' 'stat-sum.r' 'stat-summary-2d.r' 'stat-$
                    'stat-ydensity.r' 'summary.r' 'theme-elements.r' 'theme.r' 'theme-defaults.r' '$
                    'translate-qplot-lattice.r' 'utilities-break.r' 'utilities-grid.r' 'utilities-h$
```

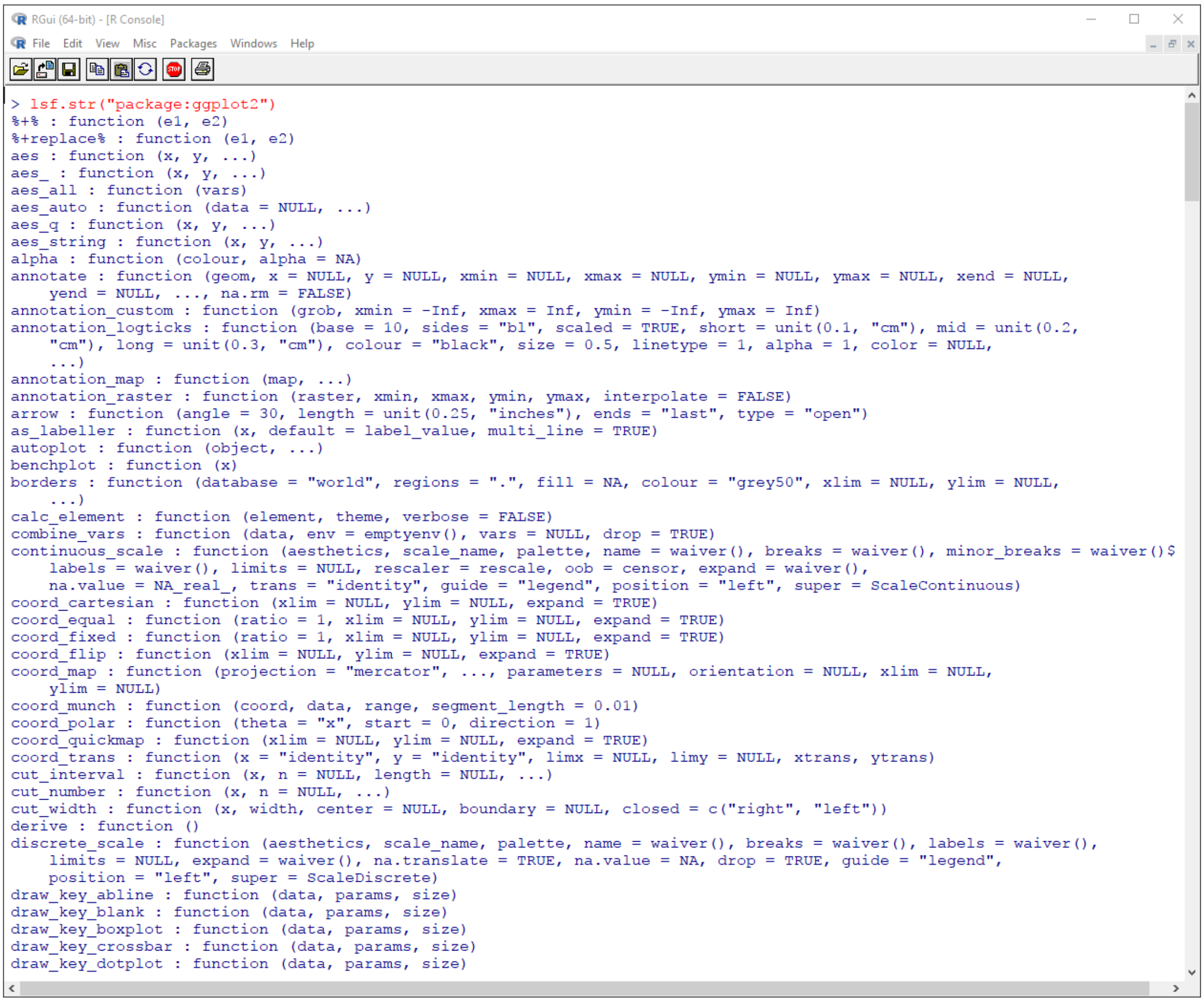
Ce qui donne aussi si on défile vers le bas, la liste de toutes les fonctions du package en question:

```
Documentation for package 'ggplot2'

Index:

+.gg                     Add components to a plot
aes                      Construct aesthetic mappings
aes_                     Define aesthetic mappings programatically
aes_colour_fill_alpha    Colour related aesthetics: colour, fill and
                         alpha
aes_group_order          Aesthetics: grouping
aes_linetype_size_shape
                         Differentiation related aesthetics: linetype,
                         size, shape
aes_position             Position related aesthetics: x, y, xmin, xmax,
                         ymin, ymax, xend, yend
annotate                 Create an annotation layer
annotation_custom        Annotation: Custom grob
annotation_logticks      Annotation: log tick marks
annotation_map           Annotation: a maps
annotation_raster        Annotation: high-performance rectangular tiling
autoplot                 Create a complete ggplot appropriate to a
                         particular data type
borders                  Create a layer of map borders
coord_cartesian          Cartesian coordinates
coord_fixed              Cartesian coordinates with fixed "aspect ratio"
coord_flip               Cartesian coordinates with x and y flipped
coord_map                Map projections
coord_polar              Polar coordinates
coord_trans              Transformed Cartesian coordinate system
cut_interval             Discretise numeric data into categorical
diamonds                 Prices of 50,000 round cut diamonds
economics                US economic time series
expand_limits            Expand the plot limits, using data
facet_grid               Lay out panels in a grid
facet_wrap               Wrap a 1d ribbon of panels into 2d
faithfuld                2d density estimate of Old Faithful data
fortify                  Fortify a model with data.
geom_abline              Reference lines: horizontal, vertical, and
                         diagonal
geom_bar                 Bars charts
geom_bin2d               Heatmap of 2d bin counts
```

Ou encore plus direct, avec la commande **lsf.str( )** on peut obtenir la liste de toutes les fonctions directement avec leur signature:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> lsf.str("package:ggplot2")
%+% : function (e1, e2)
%+replace% : function (e1, e2)
aes : function (x, y, ...)
aes_ : function (x, y, ...)
aes_all : function (vars)
aes_auto : function (data = NULL, ...)
aes_q : function (x, y, ...)
aes_string : function (x, y, ...)
alpha : function (colour, alpha = NA)
annotate : function (geom, x = NULL, y = NULL, xmin = NULL, xmax = NULL, ymin = NULL, ymax = NULL, xend = NULL,
    yend = NULL, ..., na.rm = FALSE)
annotation_custom : function (grob, xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf)
annotation_logticks : function (base = 10, sides = "bl", scaled = TRUE, short = unit(0.1, "cm"), mid = unit(0.2,
    "cm"), long = unit(0.3, "cm"), colour = "black", size = 0.5, linetype = 1, alpha = 1, color = NULL,
    ...)
annotation_map : function (map, ...)
annotation_raster : function (raster, xmin, xmax, ymin, ymax, interpolate = FALSE)
arrow : function (angle = 30, length = unit(0.25, "inches"), ends = "last", type = "open")
as_labeller : function (x, default = label_value, multi_line = TRUE)
autoplot : function (object, ...)
benchplot : function (x)
borders : function (database = "world", regions = ".", fill = NA, colour = "grey50", xlim = NULL, ylim = NULL,
    ...)
calc_element : function (element, theme, verbose = FALSE)
combine_vars : function (data, env = emptyenv(), vars = NULL, drop = TRUE)
continuous_scale : function (aesthetics, scale_name, palette, name = waiver(), breaks = waiver(), minor_breaks = waiver()$
    labels = waiver(), limits = NULL, rescaler = rescale, oob = censor, expand = waiver(),
    na.value = NA_real_, trans = "identity", guide = "legend", position = "left", super = ScaleContinuous)
coord_cartesian : function (xlim = NULL, ylim = NULL, expand = TRUE)
coord_equal : function (ratio = 1, xlim = NULL, ylim = NULL, expand = TRUE)
coord_fixed : function (ratio = 1, xlim = NULL, ylim = NULL, expand = TRUE)
coord_flip : function (xlim = NULL, ylim = NULL, expand = TRUE)
coord_map : function (projection = "mercator", ..., parameters = NULL, orientation = NULL, xlim = NULL,
    ylim = NULL)
coord_munch : function (coord, data, range, segment_length = 0.01)
coord_polar : function (theta = "x", start = 0, direction = 1)
coord_quickmap : function (xlim = NULL, ylim = NULL, expand = TRUE)
coord_trans : function (x = "identity", y = "identity", limx = NULL, limy = NULL, xtrans, ytrans)
cut_interval : function (x, n = NULL, length = NULL, ...)
cut_number : function (x, n = NULL, ...)
cut_width : function (x, width, center = NULL, boundary = NULL, closed = c("right", "left"))
derive : function ()
discrete_scale : function (aesthetics, scale_name, palette, name = waiver(), breaks = waiver(), labels = waiver(),
    limits = NULL, expand = waiver(), na.translate = TRUE, na.value = NA, drop = TRUE, guide = "legend",
    position = "left", super = ScaleDiscrete)
draw_key_abline : function (data, params, size)
draw_key_blank : function (data, params, size)
draw_key_boxplot : function (data, params, size)
draw_key_crossbar : function (data, params, size)
draw_key_dotplot : function (data, params, size)
```

Nous pouvons aussi demander des exemples de démos sur une commande avec **example( )**:

```
> example(cor)

cor> var(1:10)  # 9.166667
[1] 9.166667

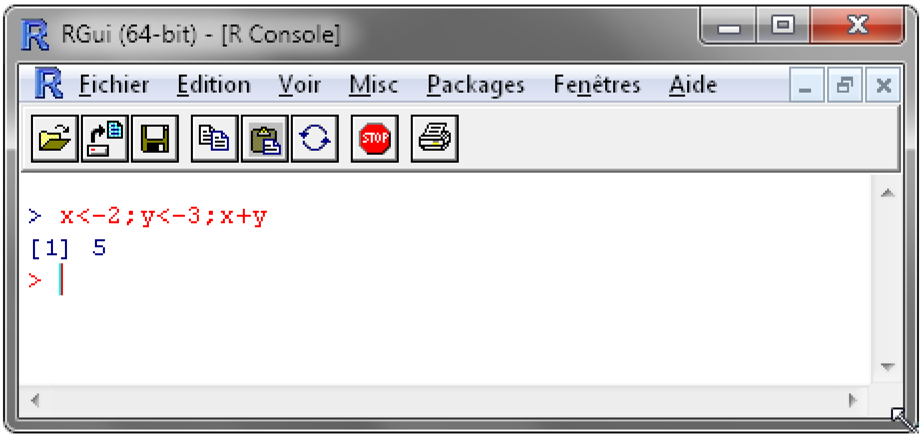
cor> var(1:5, 1:5) # 2.5
[1] 2.5

cor> ## Two simple vectors
cor> cor(1:10, 2:11) # == 1
[1] 1

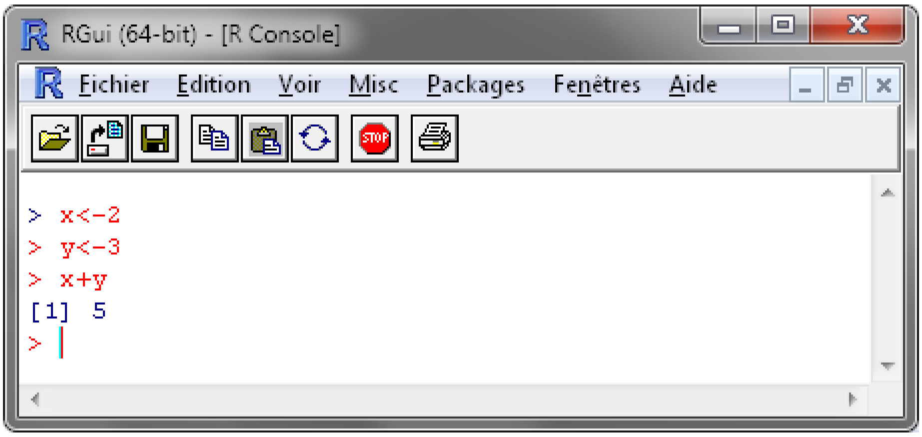
cor> ## Correlation Matrix of Multivariate sample:
cor> (Cl <- cor(longley))
             GNP.deflator       GNP Unemployed Armed.Forces Population
GNP.deflator    1.0000000 0.9915892  0.6206334    0.4647442  0.9791634
GNP             0.9915892 1.0000000  0.6042609    0.4464368  0.9910901
Unemployed      0.6206334 0.6042609  1.0000000   -0.1774206  0.6865515
Armed.Forces    0.4647442 0.4464368 -0.1774206    1.0000000  0.3644163
Population      0.9791634 0.9910901  0.6865515    0.3644163  1.0000000
Year            0.9911492 0.9952735  0.6682566    0.4172451  0.9939528
Employed        0.9708985 0.9835516  0.5024981    0.4573074  0.9603906
                  Year  Employed
GNP.deflator 0.9911492 0.9708985
GNP          0.9952735 0.9835516
Unemployed   0.6682566 0.5024981
Armed.Forces 0.4172451 0.4573074
```

Plusieurs commandes sur une ligne

Même si ce n'est pas recommandé pour des questions de confort de lecture, il est possible d'écrire plusieurs commandes sur une seule ligne si l'on veut faire vite... Voilà un bête exemple:



au lieu de:



Installer et charger/décharger des packages

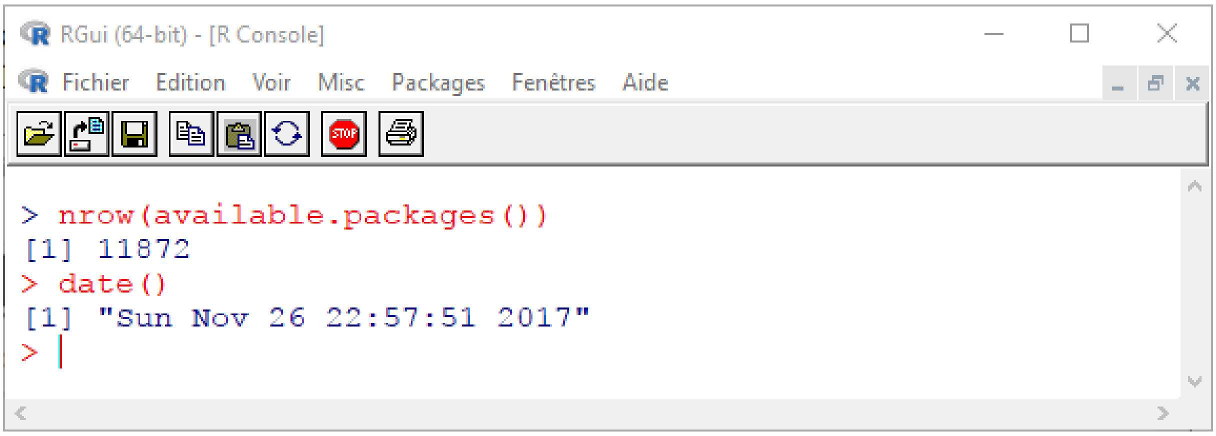
L'intérêt majeur de R ce sont ces dizaines milliers de packages spécialisés qui sont pour l'immense majorité gratuits et open source!

Ce qu'il faut d'abord savoir c'est que lors de la première installation, R vient avec les pages suivants:

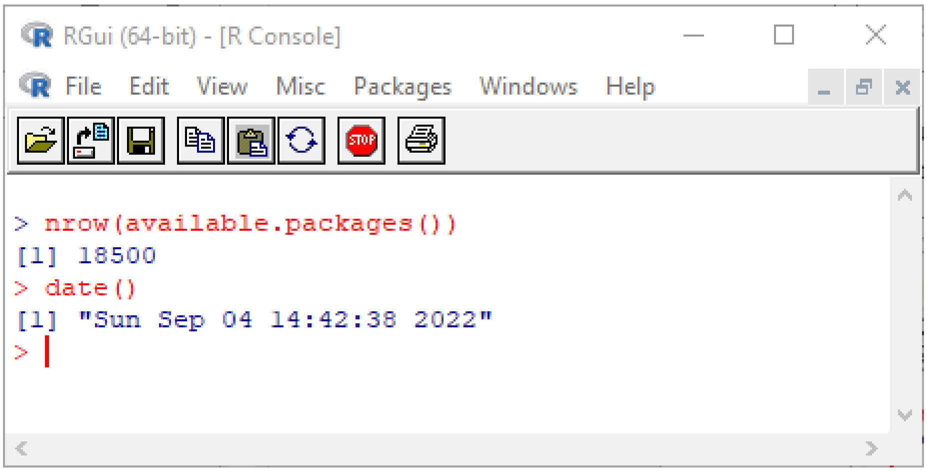


Qui sont le noyau dur de R (leur nom suffit à les décrire).

Voyons d'abord comment obtenir le nombre de packages actuellement existant sur CRAN étant donné que c'est une question fréquente en utilisant la fonction **avalaible.packages( )**:



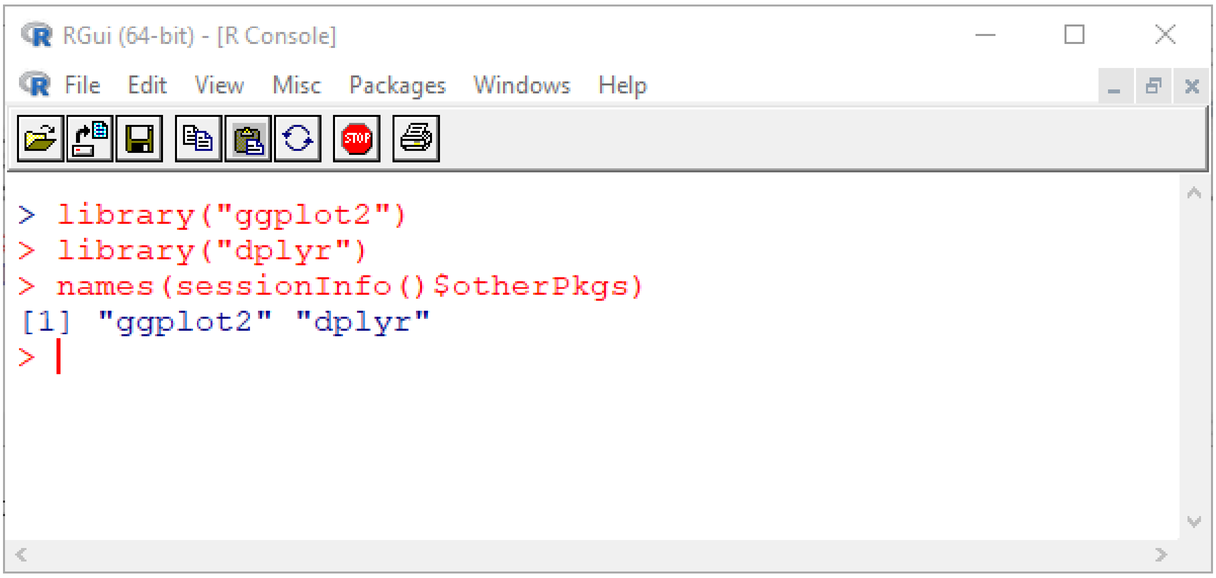
Et environ 5 ans plus tard:



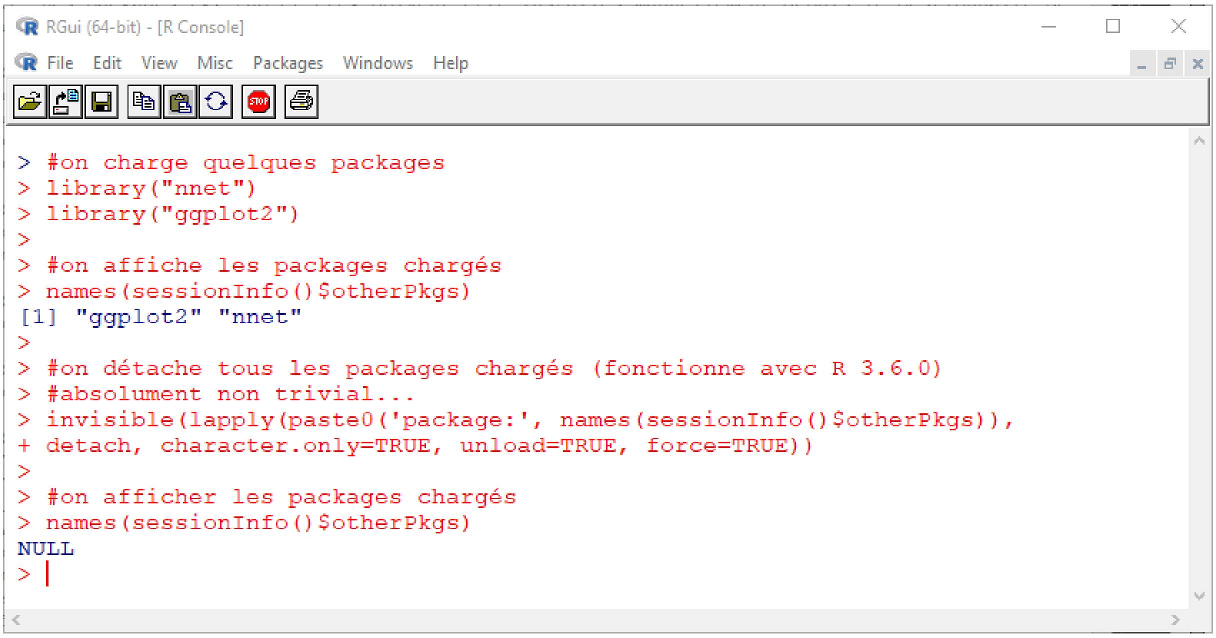
Ensuite, voyons avant toute chose comment obtenir la liste des package installés par défaut dans R avec la commande **getOption("defaultPackages")**:



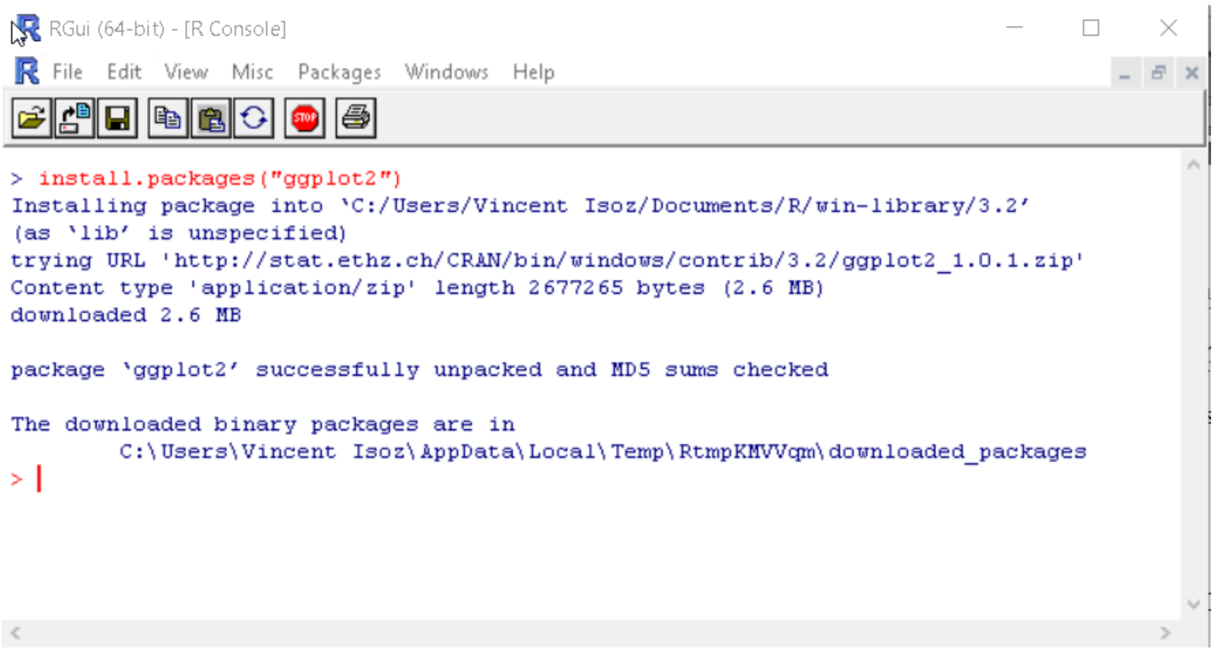
Sinon pour avoir la liste des packages chargés autres que ceux par défaut:



On peut détacher de façon générale et non triviale de la manière suivante un package:



Installer un package est très simple (dans le cas normal) en utilisant la commande **install.packages( )**:



ou pour installer depuis GitHub le lecteur pourra se référer la page 1091 (**mais attention!!! Sous Microsoft Windows, l'installation se passe souvent mal si le compte utilisateur contient des espaces!**).

Pour installer les packages les plus utilisés dans ce support de cours, copier/collez la commande ci-dessous dans la console R validez par ENTER:

```
install.packages(c("remotes","readxl","googlesheets","haven","readr","rio",
"Hmisc","sqldf","jsonlite","XML","httr","quantmod","tidyquant","rvest","dpl
yr","purrr","reshape2","tidyr","magrittr","validate","testthat","data.table
","stringi","stringr","lubridate","zoo","editR","knitr","officer","listview
er","DT","ggplot2","ggiraph","dygraphs","googleVis","metricsgraphics","RCol
```

```
orBrewer","sf","leaflet","ggmap","tmap","tmaptools","mapsapi","tidycensus",
"glue","rga","RSiteCatalyst","roxygen2","shiny","flexdashboard","openxlsx",
"gmodels","janitor","car","rcdimple","foreach","scales","plotly","highchart
er","profvis","tidytext","diffobj","Prophet","feather","fst","googleAuthR",
"cloudyR","shiny","car","checkpoint","CHAID","CCA","caret","bootstrap","Bio
StatR","bibtex","bayestestR","AnomalyDetection","ADGofTest","actuar","model
time","tidymodels","tidyverse","animation","RCurl","forecast","NbClust","An
omalyDetection","audio","bayesAB","bayesboot","BayesFactor","bayesplot","br
oom","C50","CircStats","copula","correlation","data.table","plyr","desc","D
escTools","desirability","deSolve","docxtractr","DiagrammeR","ecd","ellipse
","epiR","FactoMineR","dbscan","Rcpp","forecast","GGally","reshape","geofac
et","quantmod","corrplot","RSelenium","netstat","pingr","desc","e1071","eva
luate","geogrid","gridExtra","jpeg","lifecycle","prettyunits","quadprog","R
cpp","RODBC","survival","tseries","xts","zoo","clipr","scales","htmlwidgets
","xml","arules","arulesViz","grDevices","grid","methods","ISLR","TSP","fpc
","gplots","arules","seriation","NMF","scatterplot3d","VennDiagram","fitdis
trplus","dbscan","car","fpc","foreign","rpivotTable","ggvoronoi","voronoiTr
eemap","survminer","latex2exp","VIM","ggpubr"))
```

Il y a des cas plus compliqués, surtout pour ceux qui travaillent en génétique et en biologie moléculaire avec le package BioConductor comme l'illustre le code suivant:

```
1  install.packages("BiocManager")
2  BiocManager::install("MassSpecWavelet")
3  BiocManager::install("impute")
```

On peut choisir également d'installer un package dans un dossier spécifique et le recharger du même dossier spécifique:


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> install.packages("zoo",lib="c:/tmp/")
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/zoo_1.8-1.zip'
Content type 'application/zip' length 915692 bytes (894 KB)
downloaded 894 KB

package 'zoo' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Vincent Isoz\AppData\Local\Temp\Rtmpwd4X01\downloaded_packages
> library("zoo",lib="c:/tmp/")

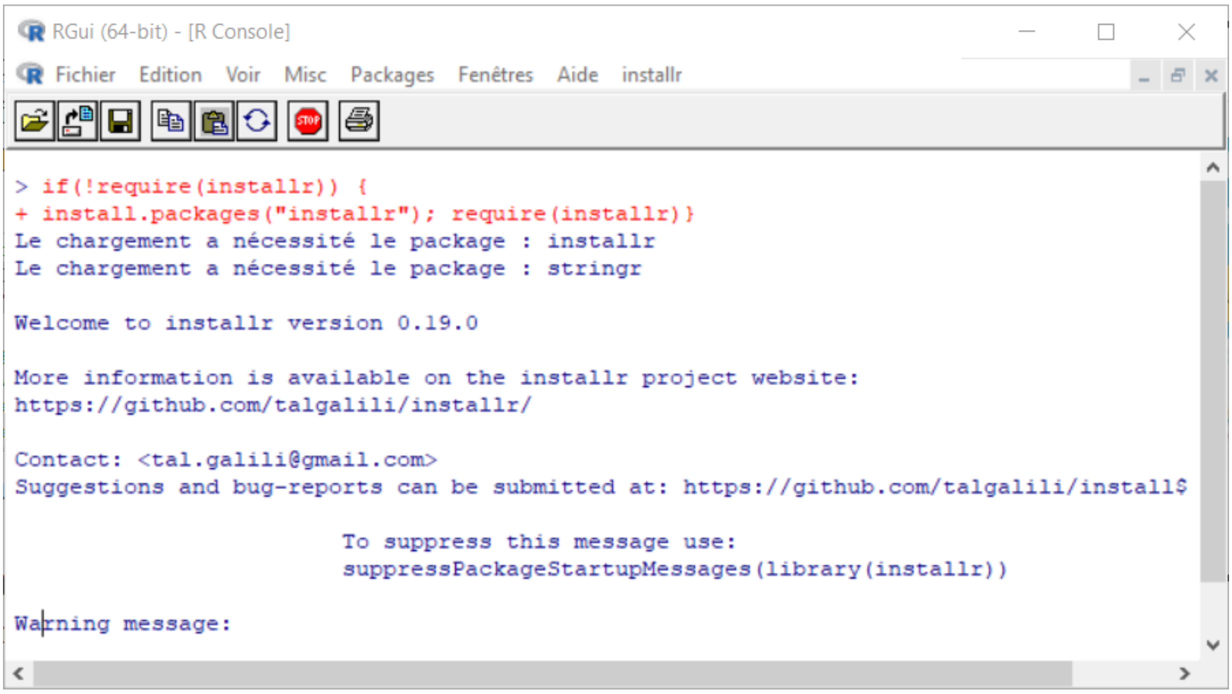
Attaching package: 'zoo'

The following objects are masked from 'package:base':

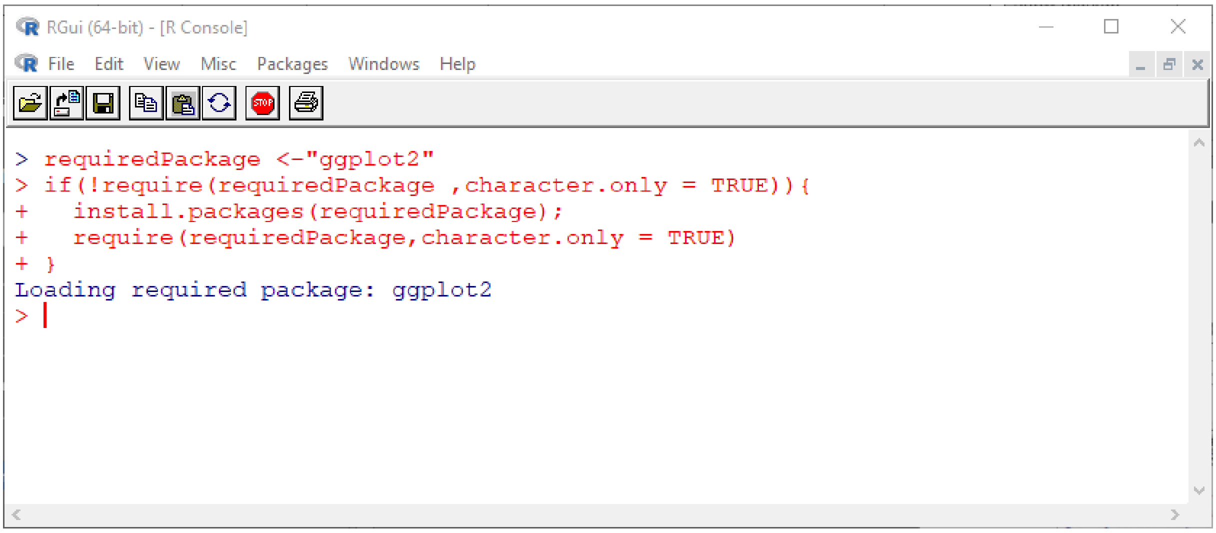
    as.Date, as.Date.numeric

> 
```

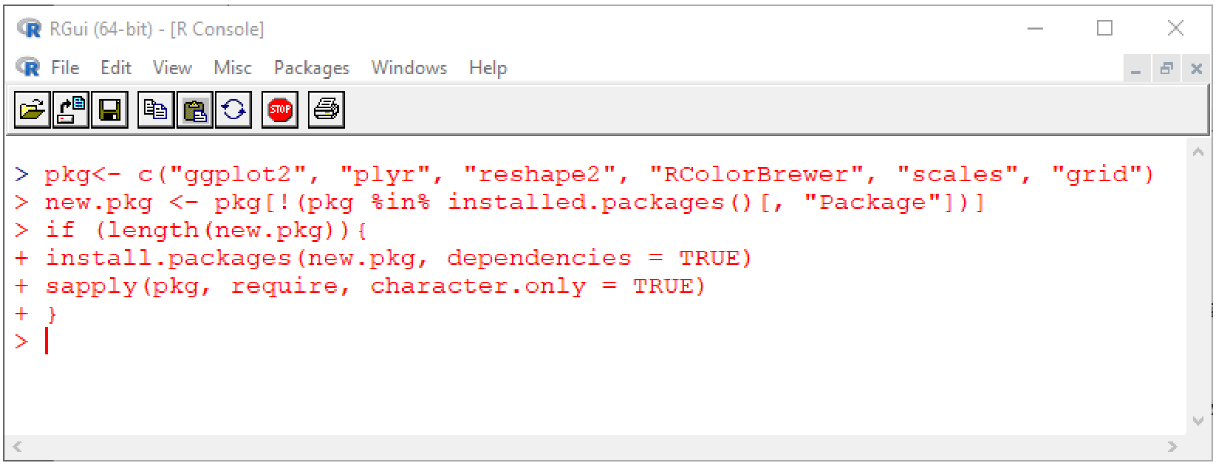
Je ne connais aucune commande native sur <u>une ligne unique</u> permettant (à ce jour…!) de vérifier qu'un package est déjà installé, le cas échéant de le télécharger et installer plusieurs packages d'un seul coup à ce jour (bien sûr un script est faisable mais ce n'est pas le sujet ici!!!). Le mieux que je connaisse est:



Si vous souhaitez passer le nom du package via une variable il faudra changer le script ci-dessus comme visible dans la figure suivante:

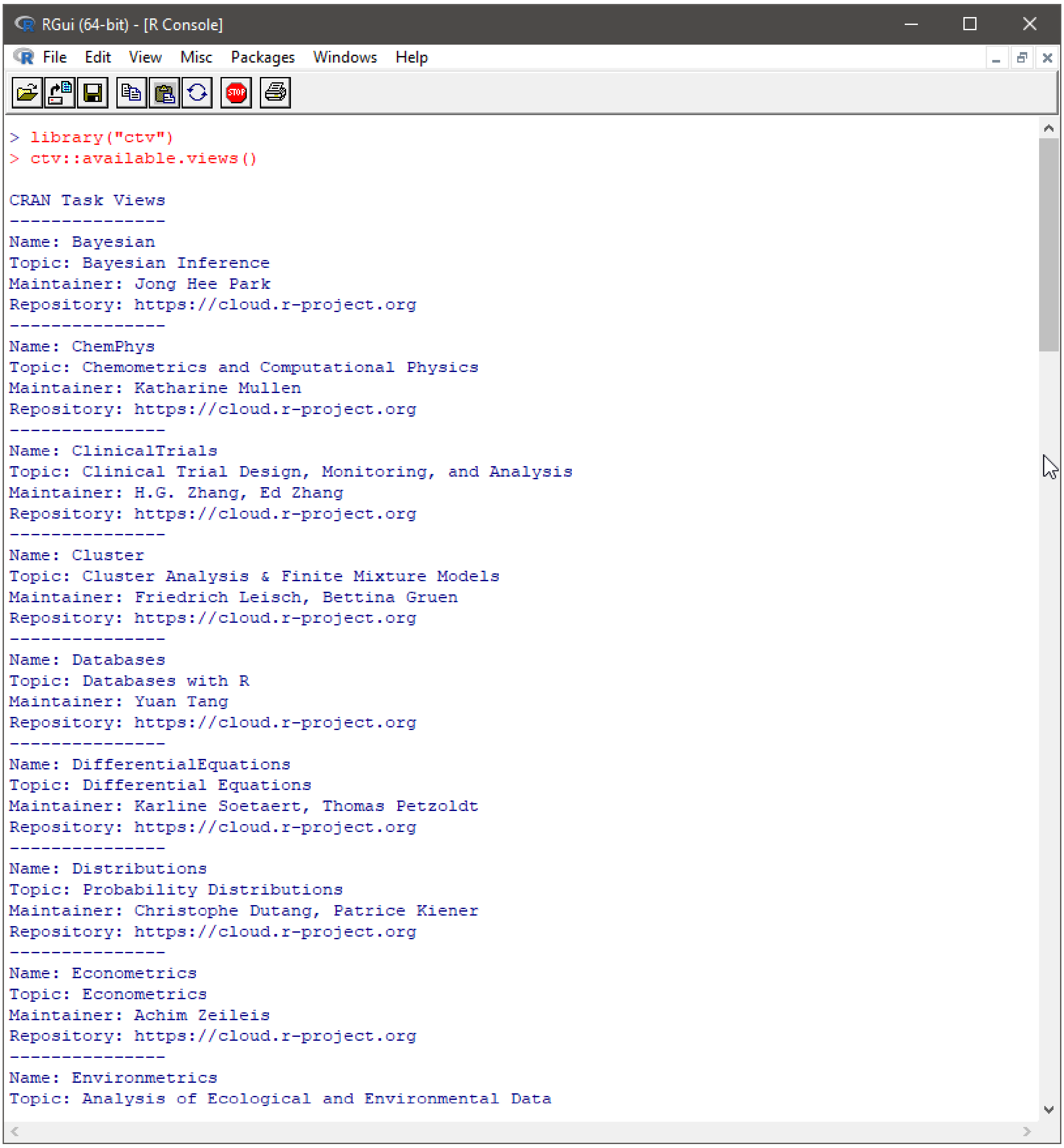


Nous pouvons généraliser cela à un vecteur de packages:



Cependant comme nous allons le voir plus bas il existe une package nommé **pacman** faisant tout le travail.

Pour installer une "vue" (voir les détails des vues sur https://cran.r-project.org/web/views/) on utilisera le package **ctv**:

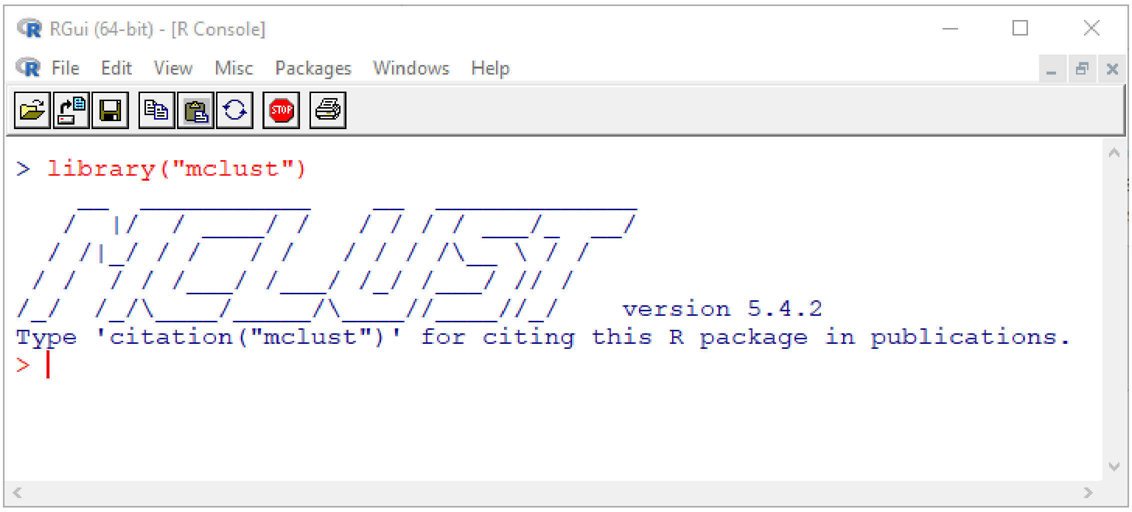


Et ensuite la commande **install.views(<nom de la vue>)**.

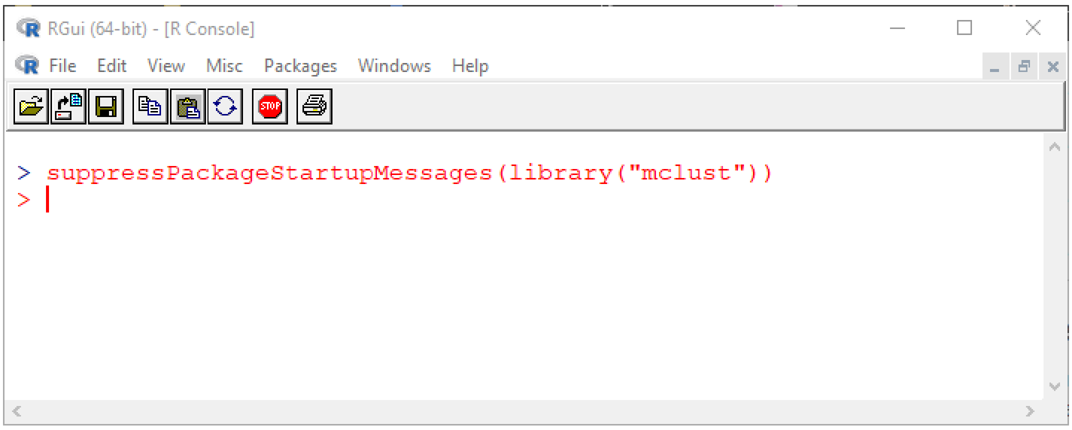
Pour charger un package c'est très simple en utilisant la commande **library("***nom_package***")**:



Si le package affiche un long message comme par exemple le package **mclust**:

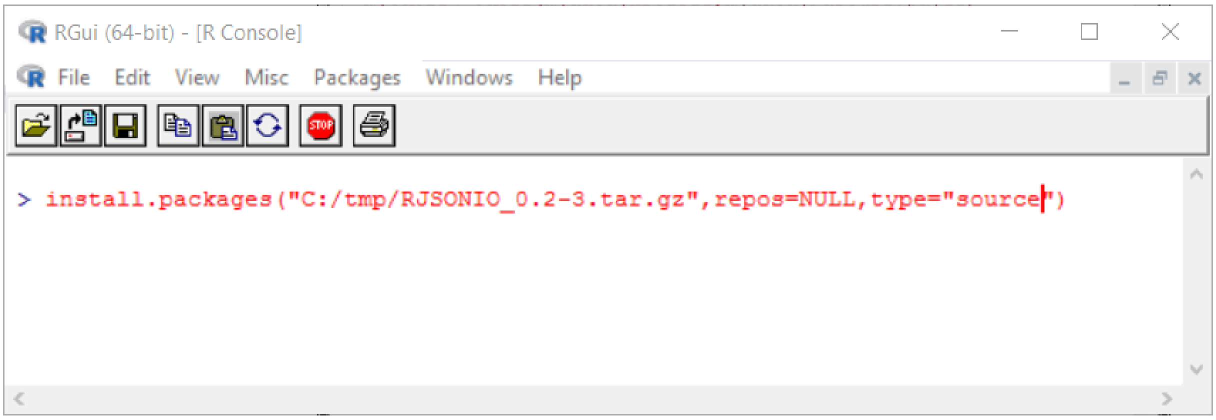


vous pouvez utiliser la fonction **suppressPackageStartupMessages( )**:

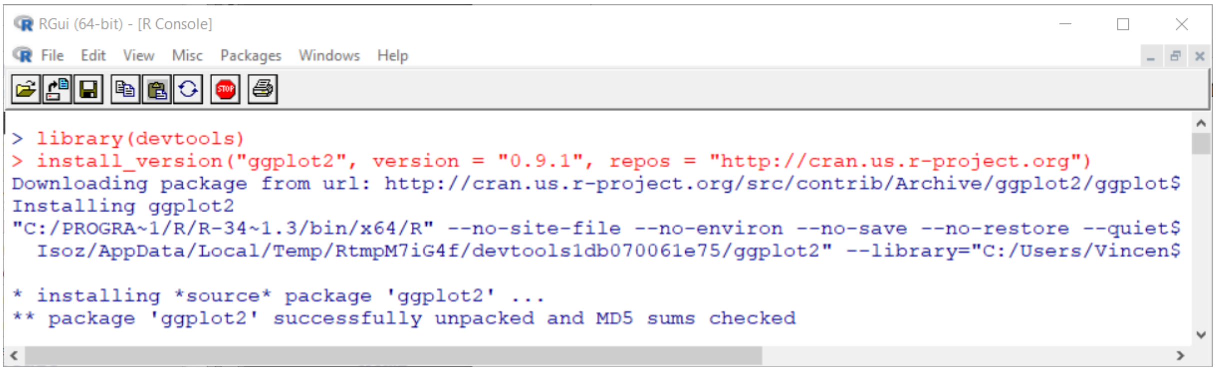


En passant, il faut savoir que l'utilisation de la fonction **require(***nom_package***)** à la place de **library(***nom_package***)** pour charger les packages fait la même chose à une différence près: c'est que **require(***nom_package***)** retourne une valeur booléenne invisible **TRUE** si le package a été retrouvé dans la bibliothèque sinon **FALSE**, ce qui nous permet de l'utiliser dans des scripts avec les structure conditionnelles **if**… **else**.

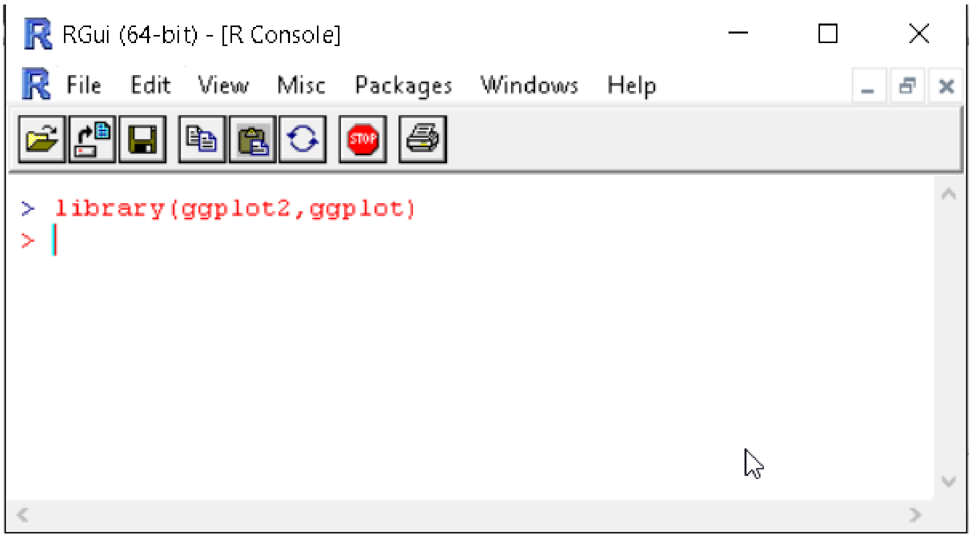
Si le package a été téléchargé ou fourni au format *.tar.gz car votre employeur (typiquement: banque, assurance, gouvernement) a bloqué la connexion Internet, voici comment procéder…:



Si on a besoin de réinstaller un vieux package pour des raisons de **reproductibilité**, on utiliser le package **devtools** qui contient la commande très utile **install_version( )**:



Pour charger plusieurs packages en même temps c'est très simple:



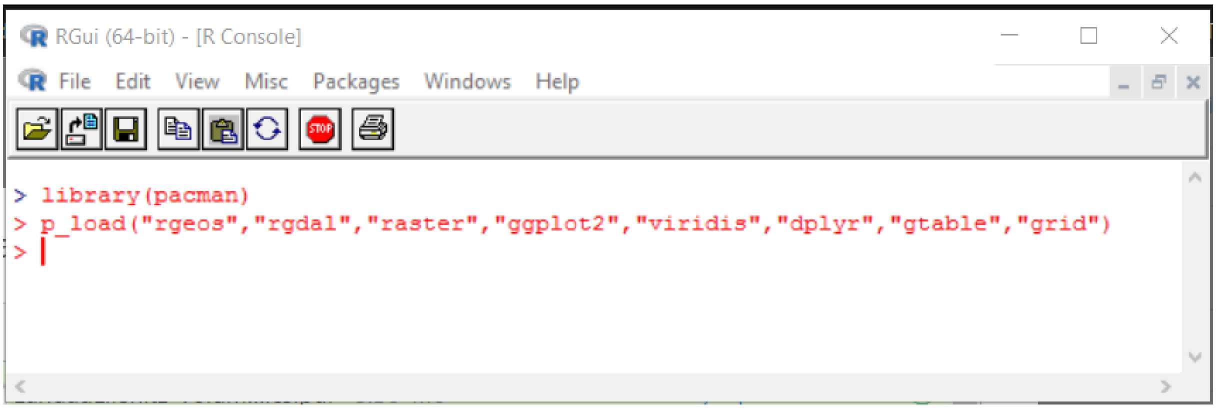
Pour décharger un package (ce qui est utile lors d'essais ou conflits de fonctions), sans redémarrer R, on utilisera la fonction **detach( )**:



Ou pour décharger TOUS les packages chargés:



Maintenant voyons comment utiliser la package **pacman** qui va permettre de vérifier si une série de packages est installée et dans le cas contraire les télécharger et les installer. D'abord nous installons le package comme indiqué précédemment, et ensuite nous pouvons le tester:

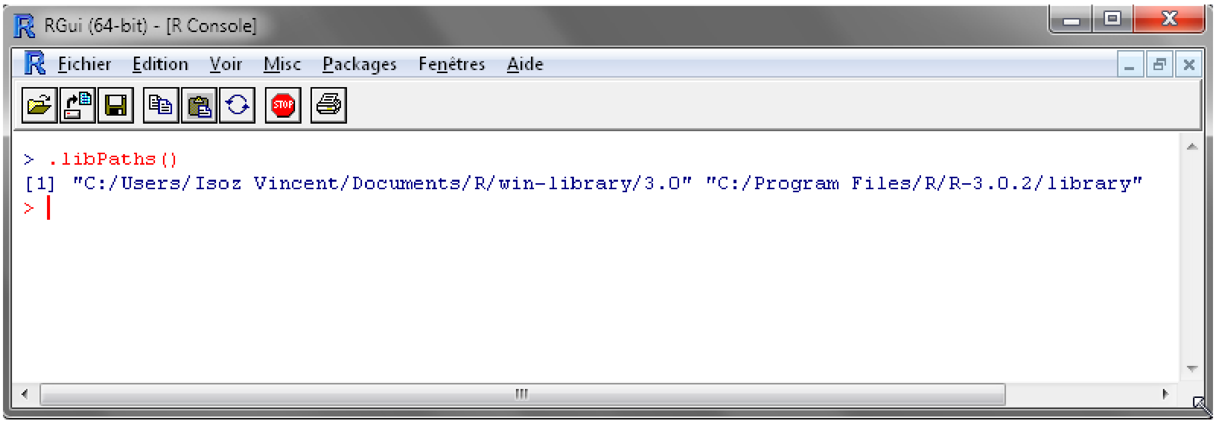


et comme nous pouvons le voir, tout se passe en arrière-plan et est transparent pour l'utilisateur final!!! Donc ce package est très intéressant pour des scripts développées en production!

Changer le dossier d'installation par défaut des packages

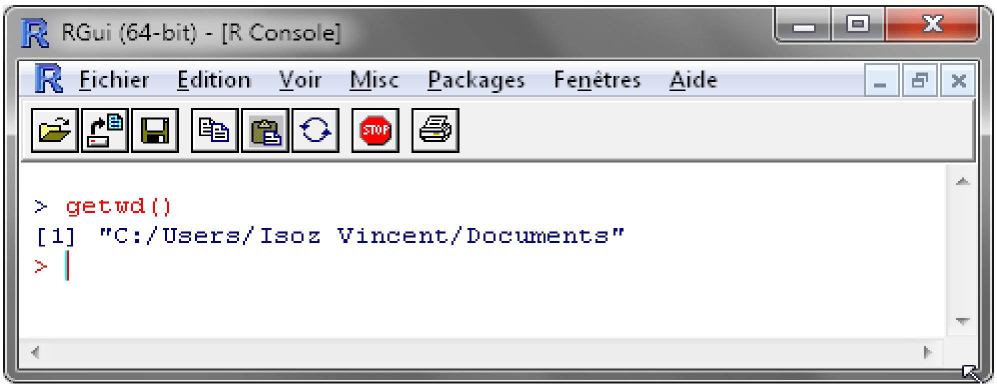
Il se peut en entreprise que le département IT vous fasse changer d'ordinateur (logique...) ou migrent de nouvelles versions de l'OS. Dès lors, vous n'allez probablement pas vouloir

réinstaller tous les packages. Pour déterminer où ils sont installés, vous pouvez utiliser la commande **.libPaths( )**:

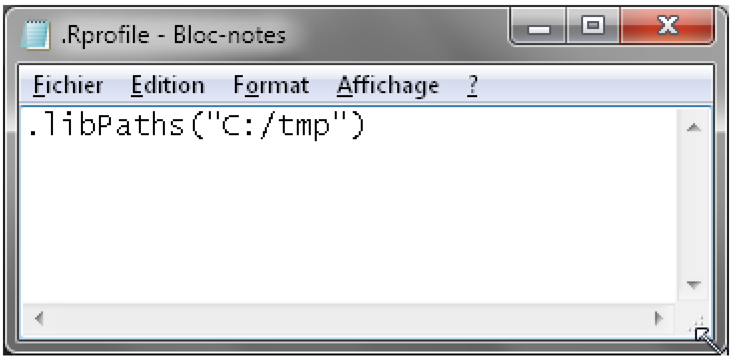


Le premier dossier est le dossier où R vous avait demandé où installer le premier package (du moins si vous en avez déjà installé un...) et le deuxième est le dossier des packages par défaut de R (il y en a une trentaine qui sont déjà installés comme vous pourrez le vérifier en vous y rendant).

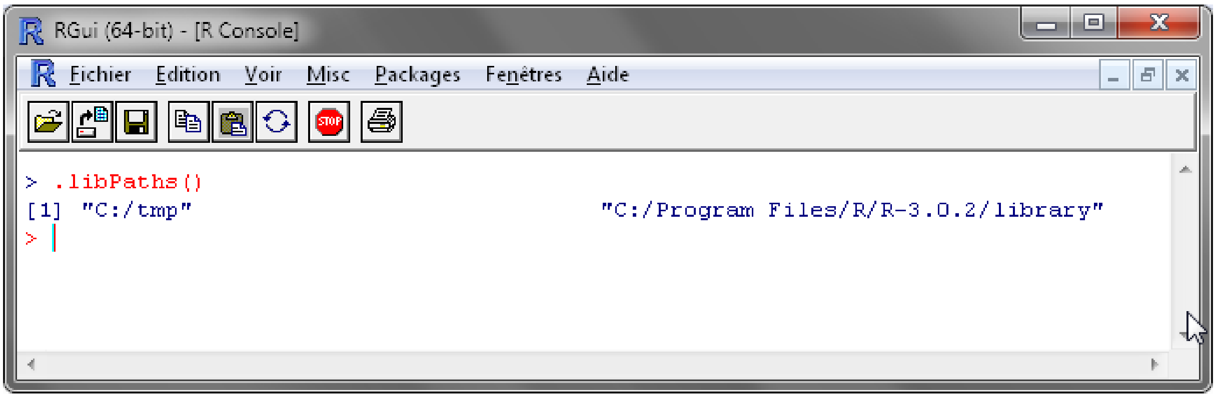
Pour changer le dossier par défaut d'installation d'abord déterminez votre dossier de travail par défaut en faisant typiquement la commande **getwd( )**:



Une fois ceci fait, dans ce dossier créez le fichier .Rprofile avec le dossier par défaut désiré pour les packages:



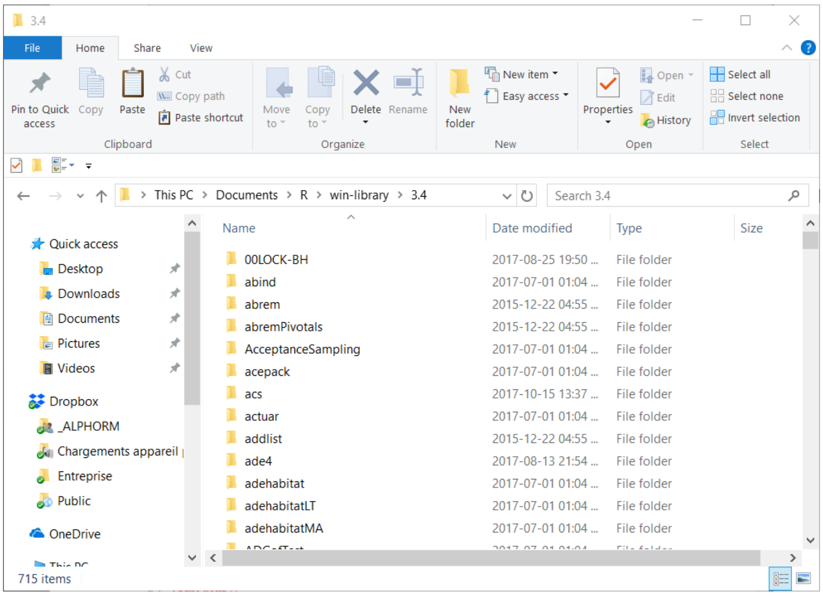
et après avoir fermé et rouvert R vous aurez bien:



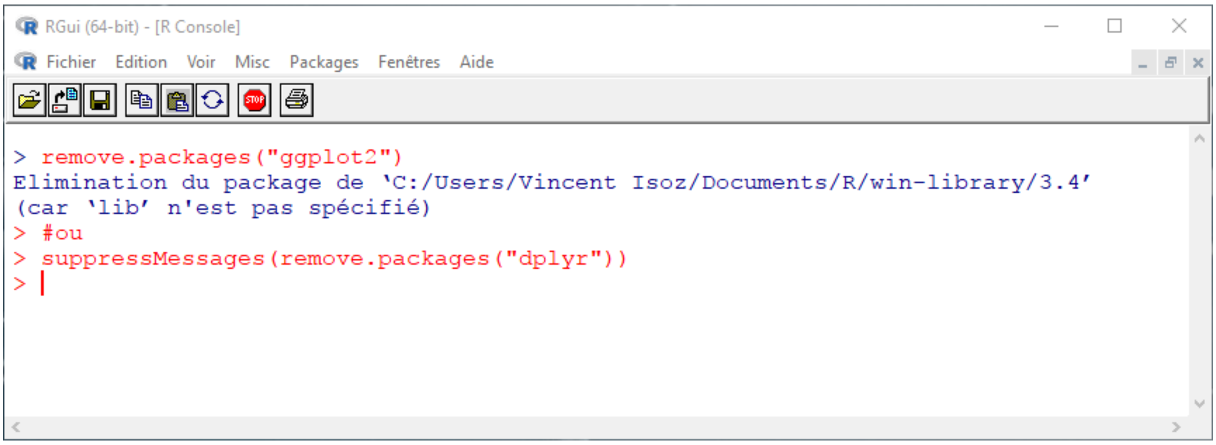
bingo! C'est très pratique lorsque l'on a plusieurs dossiers avec des packages (personnels, de notoriété publique ou simplement validés par le CRAN).

Désinstaller un package

Pour désinstaller un package, soit on va simplement dans le dossier des packages pour le supprimer:



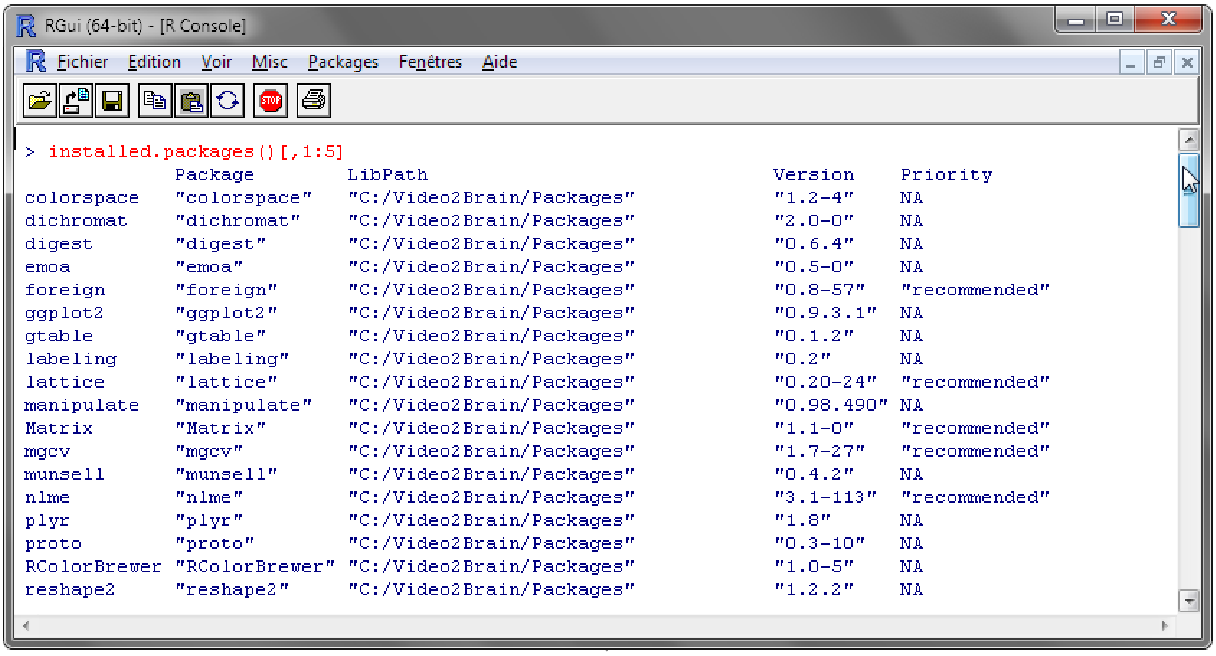
soit on utilise la commande **remove.packages( )**:



Obtenir les versions des packages et dépendances

Lorsque vous publiez un article scientifique, outre le fait de devoir mentionner la version détaillée du logiciel, il vous faudra également détailler la version détaillée du package utilisé.

Pour obtenir une liste détaillée sur les packages installés voici une solution simple (nous avons limité le retour des informations à 5 colonnes mais n'hésitez pas à découvrir par vous même ce qu'il y au-delà) utilisant la commande **installed.packages( )**:

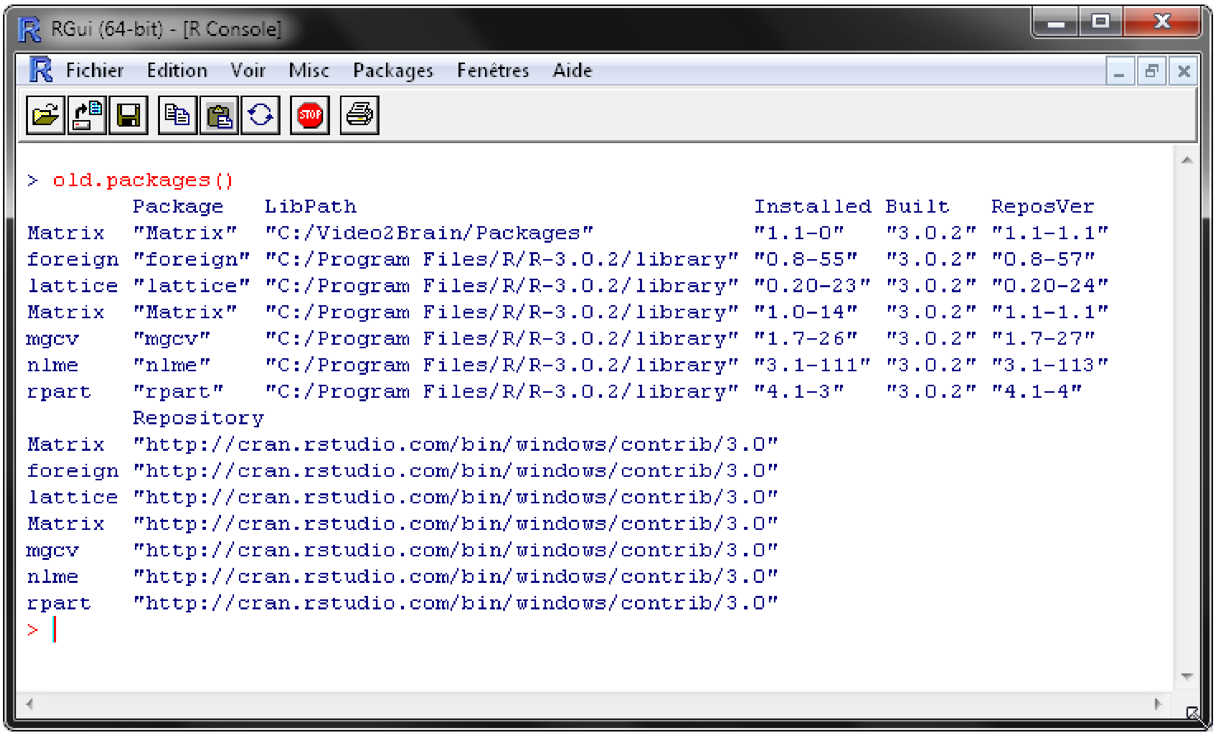


Et pour obtenir la liste des packages chargés dans la session R en cours on utiliser la commande non-triviale **(.packages()):**

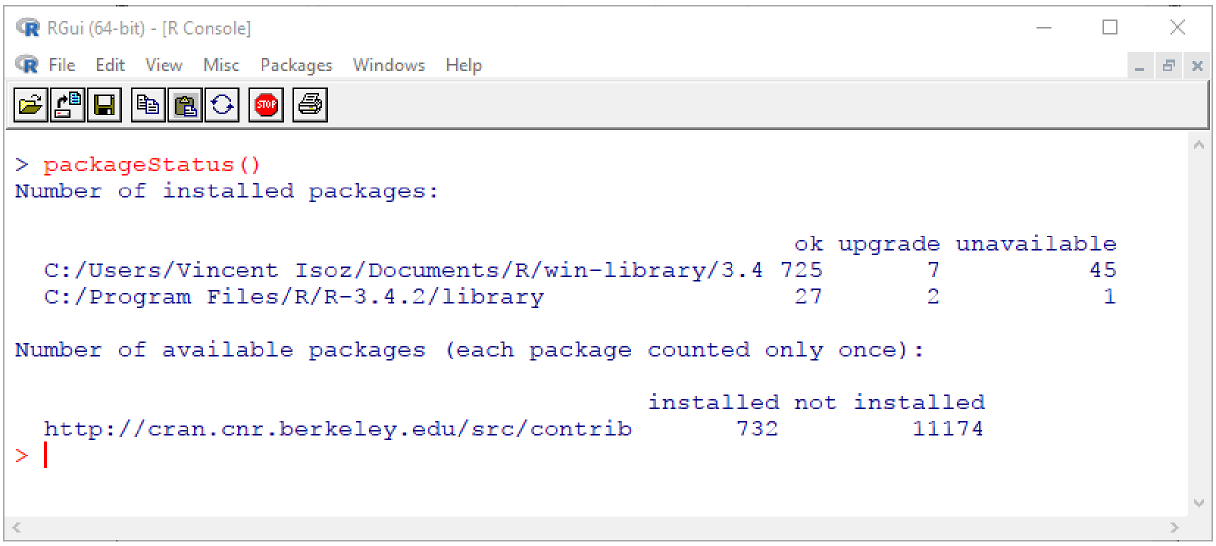


Vérifier si les packages sont à jour

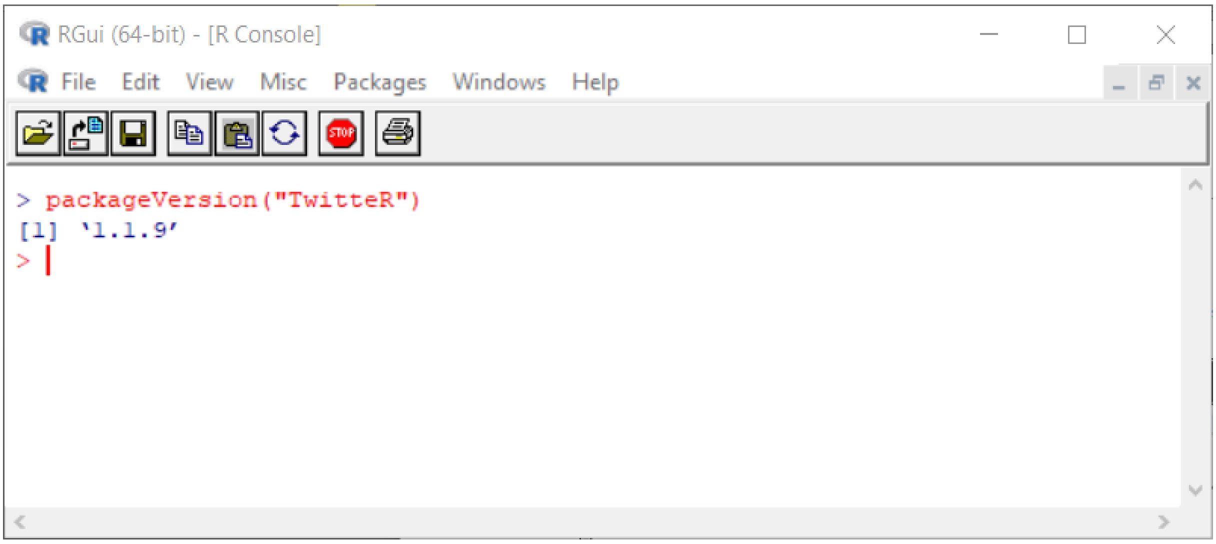
Pour vérifier si les packages sont à jour, nous utiliserons la commande **old.packages( )**:



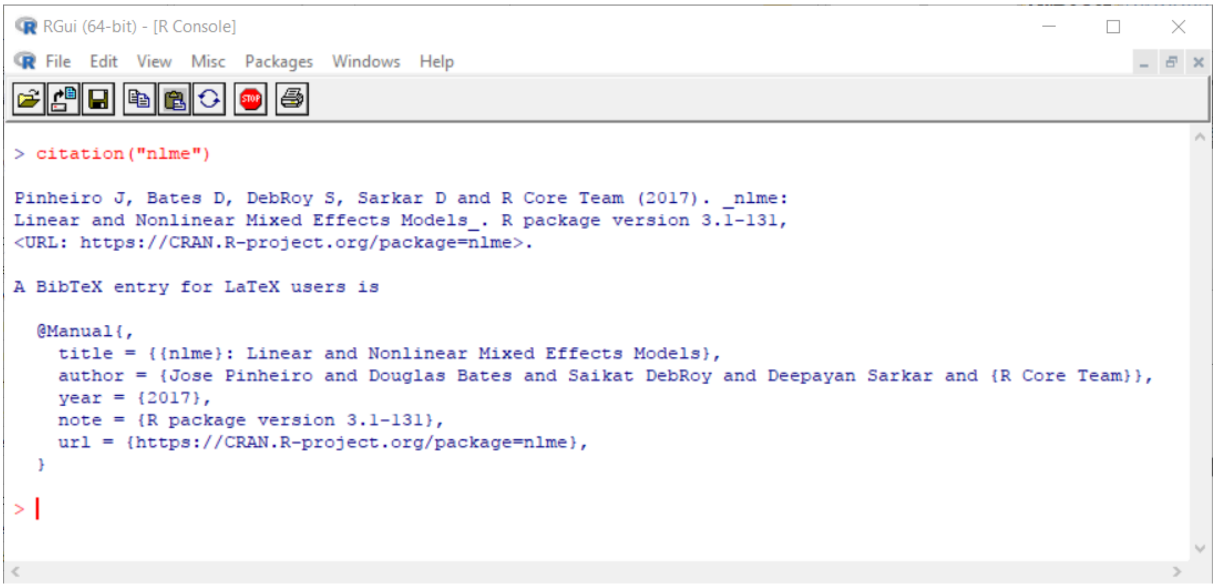
Il y a aussi la commande **packageStatus( )** qui donne un résumé intéressant de l'état des choses:



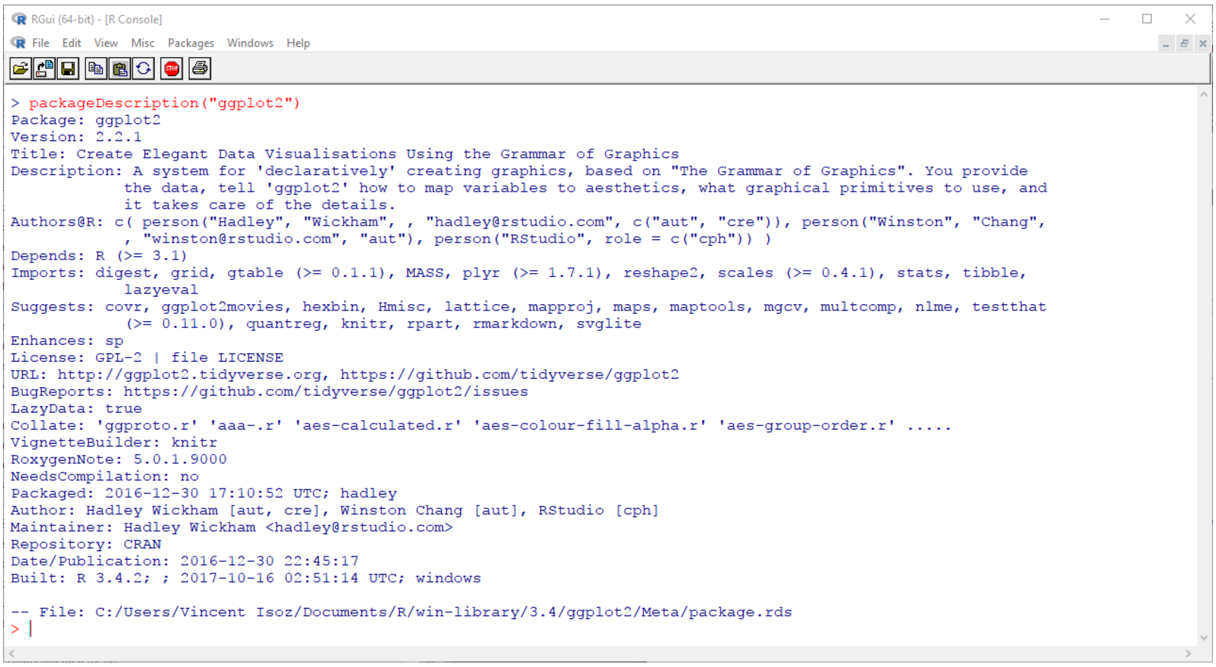
Pour vérifier la version d'un package on utilisera la commande **packageVersion( )**:



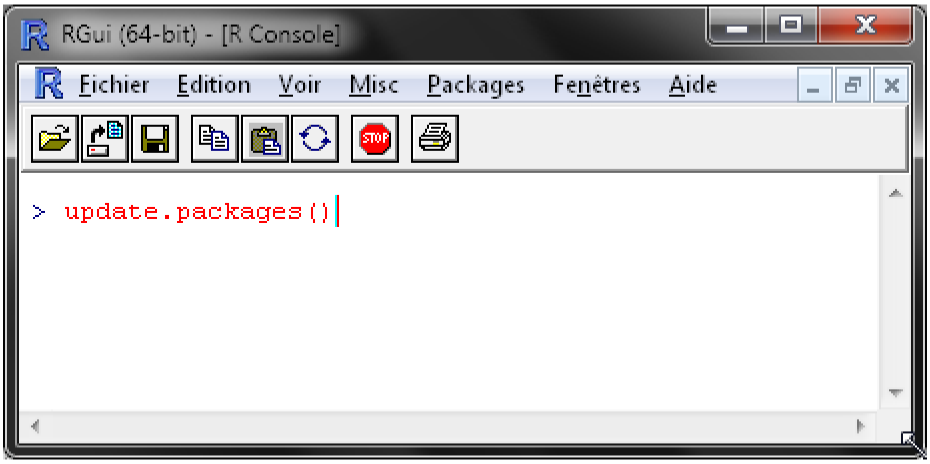
ou on peut obtenir la version aussi dans la fonction **citation( )** du package:



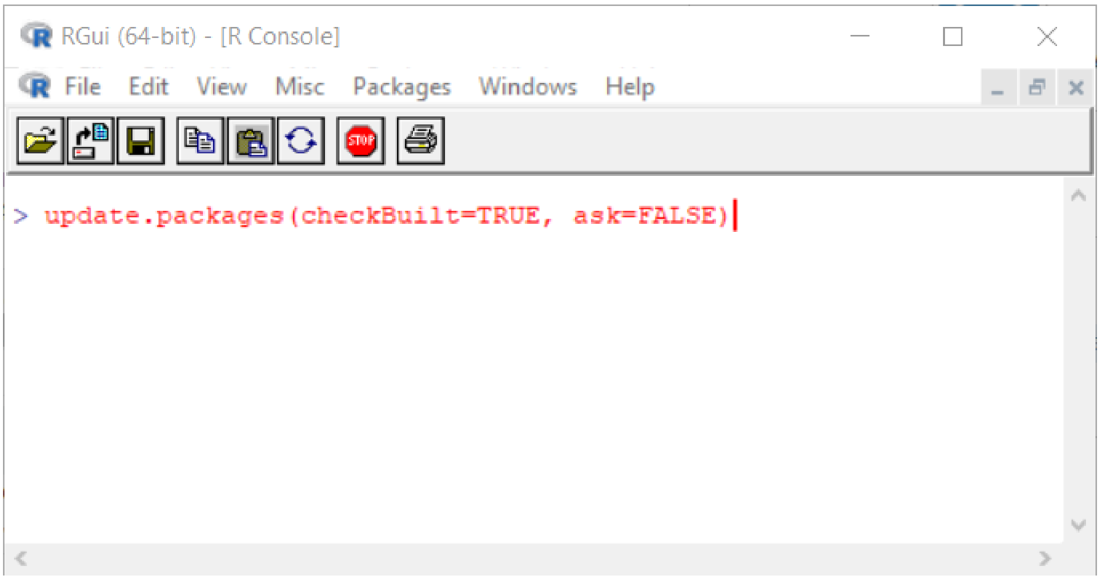
Ou encore avec la fonction **packageDescription( )**:



Et pour mettre à jour rapidement plutôt que de passer par le menu nous utiliserons la commande **update.packages( )**:



Et si nous voulons que R ne demande pas quels packages à installer:



Attention!!! Cette commande est très utile lorsque l'on veut utiliser l'automatisation de la mise à jour de script dans des packages vendus en tant que société de service!

Il n'existe pas à ce jour de commande pour mettre à jour un seul et unique package à notre connaissance. Il faut simplement le réinstaller avec la commande **install.packages( )**.

Liste des packages utilisés dans ce support

Pour ne pas vous casser la tête vous pouvez déjà installer les packages suivants (d'autres packages connexes s'installeront par nécessité que nous n'avons pas listé ci-dessous… il y en aurait sinon un peu plus de **1530** à lister… et si les installer 1 par 1 vous pose problème vous pouvez les télécharger – et donc les installer – en un seul coup en téléchargeant le fichier *.zip suivant http://www.sciences.ch/R_Packages.zip de 2.53 GB....):

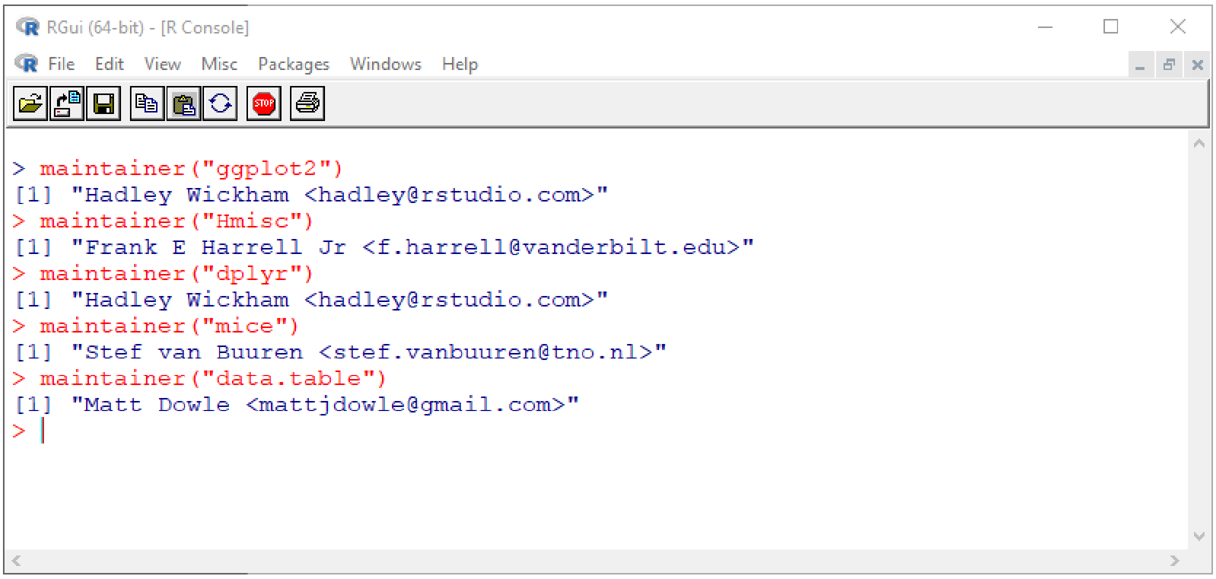
1. AcceptanceSampling
2. ade4
3. ADGofTest
4. Akima
5. alr3
6. aplpack
7. animation
8. aqfig
9. basetheme
10. Benchmarking
11. bio3d
12. bnlean
13. boot
14. broom
15. bupaR
16. C50
17. car
18. caret
19. catools
20. cartography
21. CHAID
22. changepoint
23. chron
24. circlize
25. class
26. classInt
27. climatol
28. cluster
29. coin
30. compiler
31. ConvergenceConcepts
32. Copula
33. dabestr
34. data.frame
35. data.tree
36. deepdep
37. dendextend
38. DescTools
39. devtools
40. dggridR
41. diagram
42. DiagTest3Grp
43. distr
44. distrEx
45. dplyr
46. doBy
47. docxtractr
48. DoE.base
49. e1071
50. earth
51. ellipse
52. emmeans
53. Epi
54. epiR
55. effects
56. equatiomatic
57. exactmeta
58. factoMineR
59. facoextra
60. fastICA
61. ff
62. fGarch
63. fields
64. FinancialMath
65. fImport
66. fitdistrplus
67. forecast
68. foreign
69. fOption
70. fpc
71. fPortfolio
72. fpp
73. FrF2
74. FuzzyR
75. genderizeR
76. ggbiplot
77. ggbump
78. ggdendro
79. ggforce
80. ggjoy
81. ggmap
82. ggOceanMaps
83. ggparliament
84. ggplot2
85. ggtree
86. ggvoronoi
87. gmodels
88. GPArotation
89. Gplots
90. ggpubr
91. grid
92. gridExtra
93. gtools
94. gtrendsR
95. GUIDE
96. hexabin
97. HH
98. hht
99. Hmisc
100. HSAUR
101. ineq
102. installr
103. irr
104. itunesr
105. jrvFinance
106. Kendall
107. KernSmooth
108. LaF
109. lattice
110. latticeExtra
111. lawstat
112. leaflet
113. leaps
114. lifecontingencies
115. likert
116. linnprog
117. lhs
118. lmtest
119. lsmeans
120. lubridate
121. mailR
122. maps
123. maptools
124. markovchain
125. MASS
126. Matrix
127. Mblm
128. mekko
129. memoise
130. MethComp
131. mice
132. Migest
133. MLmetrics
134. modeest
135. modeltime
136. moments
137. mratios
138. ncdf4
139. neuralnet
140. NMOF
141. nnet
142. nortest
143. numDeriv
144. OpenImageR
145. outliers
146. pacman
147. pastecs
148. patchwork

149. performance
150. PerformanceAnalytics
151. permuco
152. pheatmap
153. phytools
154. plot3D
155. plotKML
156. plot_ly
157. PlotPrjNetworks
158. plotrix
159. plyr
160. pmr
161. png
162. Portfolio
163. pracma
164. pROC
165. processmapR
166. profr
167. prob
168. propagate
169. psych
170. phonTools
171. pwr
172. qa
173. qcc
174. qdap
175. qqplotr
176. quantmod
177. quantreg
178. qualityTools
179. queueing
180. randtoolbox
181. rattle
182. rbenchmark
183. Rcmdr
184. RColorBrewer
185. readr
186. ResourceSelection
187. reshape2
188. RExcel
189. rgdal
190. rgdl
191. rgeos
192. rgl
193. Rgraphviz
194. RiskPortfolios
195. R.matlab
196. rms
197. RODBC
198. rootSolve
199. rpart
200. Rpdb
201. RoughSets
202. rjson
203. RSwissMaps
204. RQuantLib
205. RWeka
206. rworldmap
207. sampling
208. scales
209. scatterplot3d
210. SCperf
211. seewave
212. sensitivity
213. sentiment
214. seqinr
215. signal
216. sm
217. smovie
218. SnowballC
219. sp
220. spatstat
221. sqldf
222. SuperLearner
223. SuppDists
224. Survival
225. tcltk2
226. TeachingDemos
227. Tesseract
228. textcat
229. tidymodels
230. tidyverse
231. tidyxl
232. timeDate
233. timetk
234. timevis
235. tm
236. tinytex
237. topogRam
238. treemap
239. tseries
240. TSP
241. twilio
242. tuber
243. tuneR
244. twitteR
245. urca
246. unpivotr
247. usingR
248. vasicekreg
249. vcd
250. vegan
251. vioplot
252. waffle
253. waterfall
254. wordcloud
255. xesreadR
256. XLConnect
257. XML
258. xtable

Contacter la personne maintenant un package

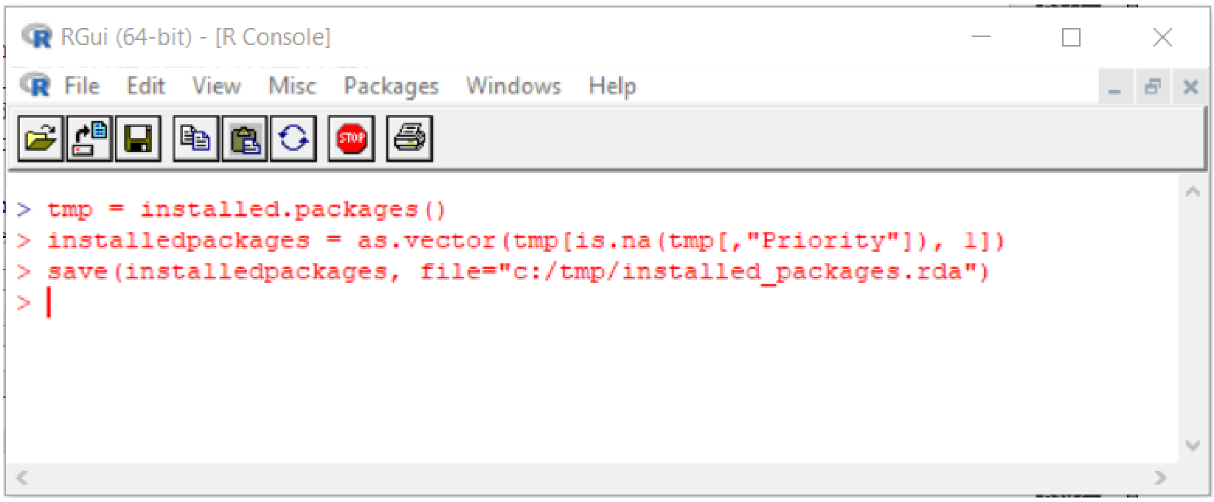
Certes on peut avoir cette information directement dans le PDF du package ou sur la page web du package. Mais quand on développe des applications R, cela peut être utile pour automatisr le rapport de bugs.

Pour cela il suffit d'utiliser la fonction **maintainer( )**:

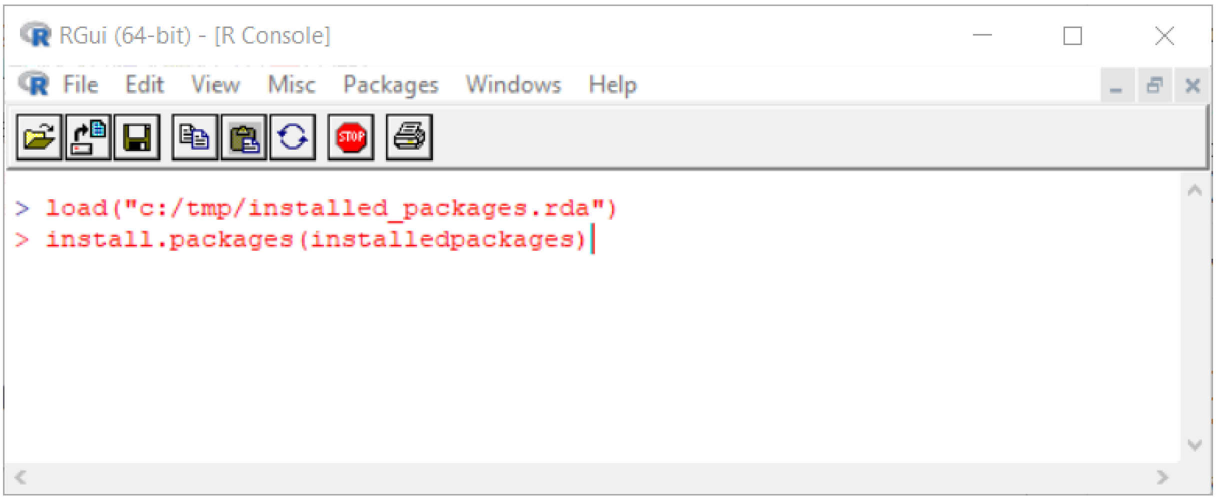


Déplacer les packages vers un autre ordinateur

Le but ici (fréquent pour les enseignants et entre élèves) est de pouvoir rapidement réinstaller tous les packages d'un ordinateur sur un autre (vous pouvez utiliser le format *.RData au lieu de l'ancien *.rda si vous le voulez!):

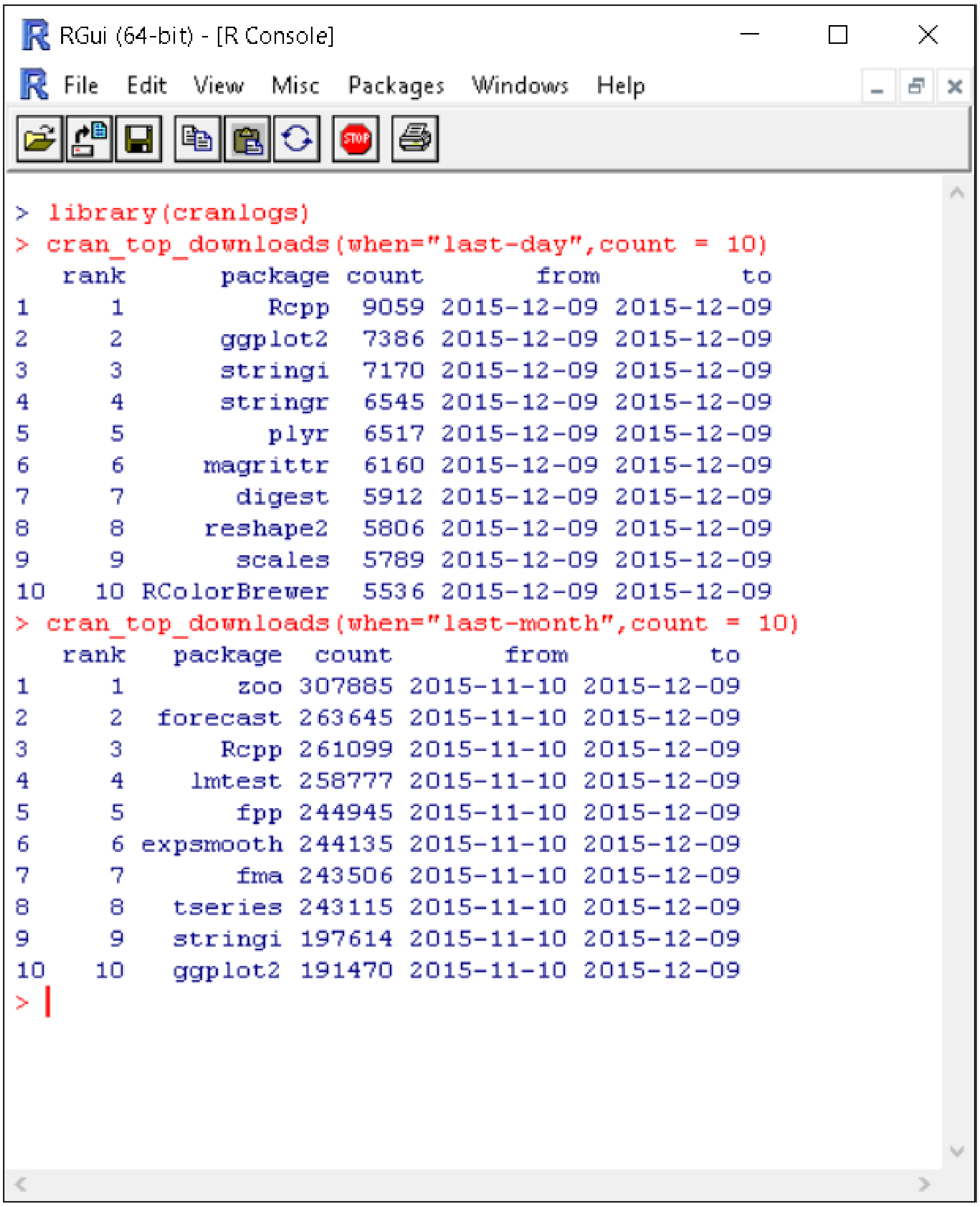


Ensuite sur l'autre ordinateur, on charge le fichier correspondant et on installe selon les commandes suivantes:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> load("c:/tmp/installed_packages.rda")
> install.packages(installedpackages)

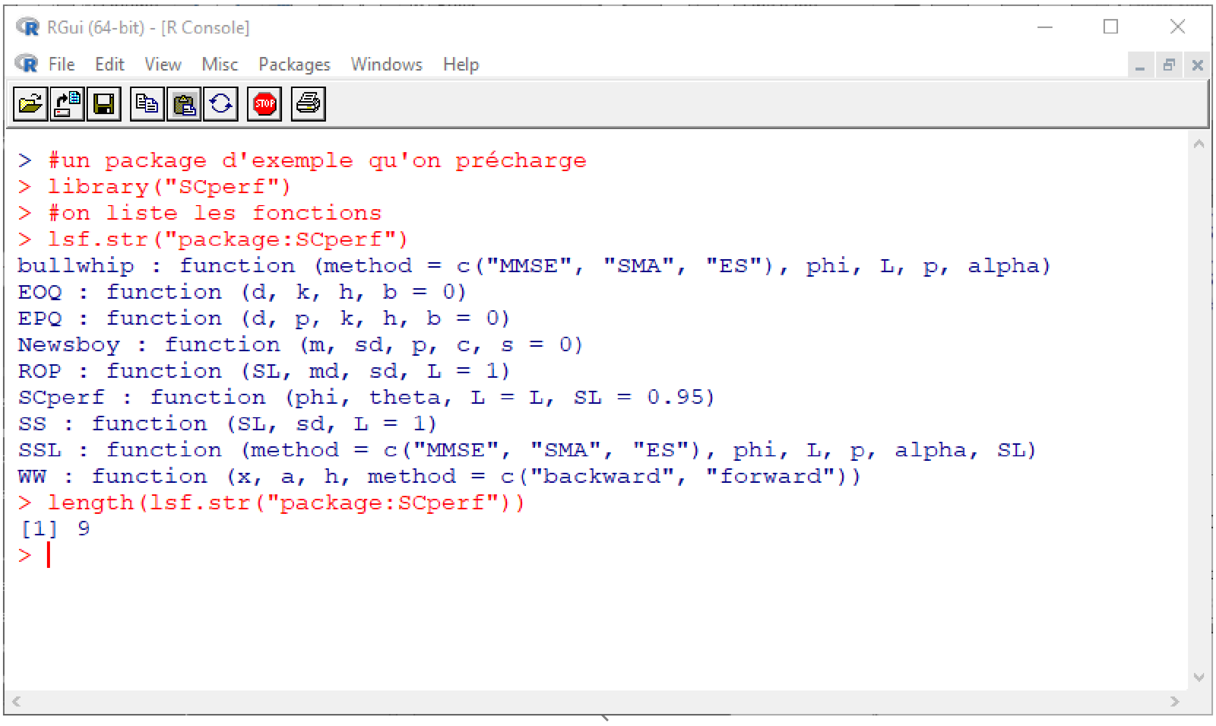
Consulter la popularité des packages

Pour consulter la popularité des packages dans un temps relativement proche c'est assez simple en utilisant la fonction **cran_top_downlaods( )** du package **cranlogs**:



Consulter la liste des fonctions d'un package et les compter

Pour ceux qui sont intéressés à savoir combien de fonctions contient un package dans sa totalité, ou en faisant une boucle, de savoir combien de fonctions tous les packages installés sur leur ordinateur contiennent, voici comment procéder en utilisant la fonction native **lsf.str( )**:



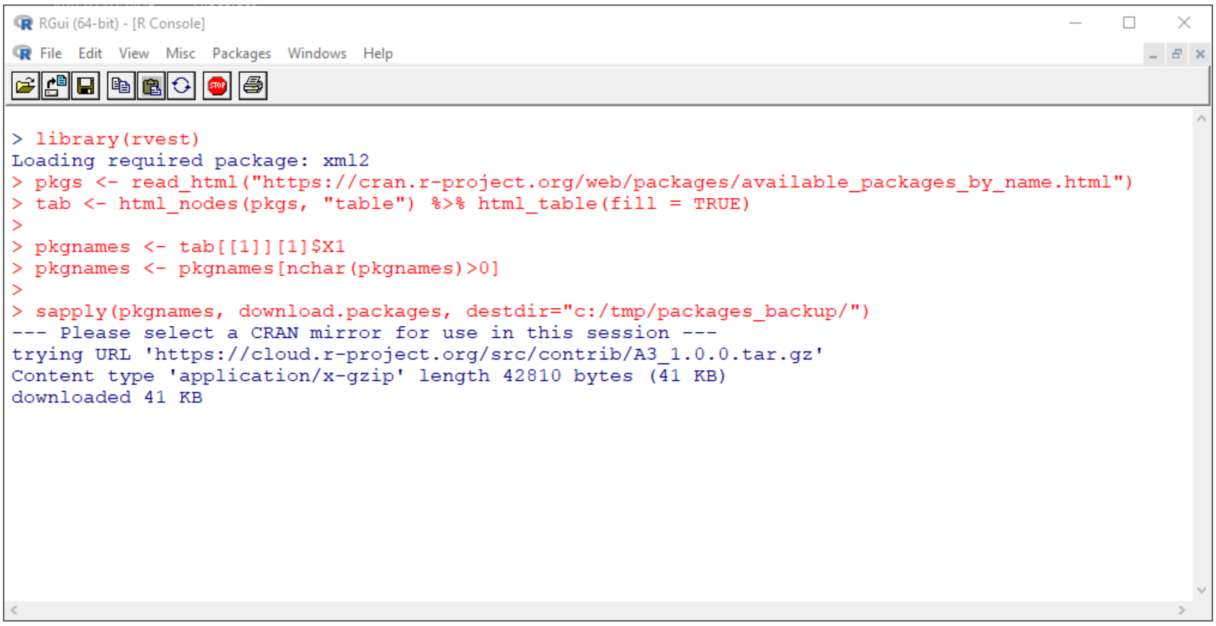
ou:



Sauvegarder toutes les packages sources du CRAN

Nous avons vu plus haut comment déterminer le nombre total de packages disponibles sur le CRAN. Comme vous le savez peut-être déjà, ce qui fait de R un outil précieux pour la communauté scientifique, c'est que le code source de la grande majorité des ces packages est accessible et utilisable gratuitement (sous les conditions habituelles de citer l'auteur qui est à l'origine du code qu'on aura réutilisé).

Cependant, une crainte majeure, c'est de ne plus avoir accès à ces sources. Donc si vous êtes comme quoi et que vous souhaitez télécharger les sources (compressées en *.tar.gz) des plus de 10'000 packages du serveur CRAN, voici le code qu'il vous faut qui utilise le package **rvest** et ses fonctions **read_html( )** et **html_nodes( )** pour "webscrapper" la page web du CRAN contenant un tableau de la liste de tous les packages disponibles:



Attention… prévoyez quelques dizaines de GB de libre sur votre ordinateur!

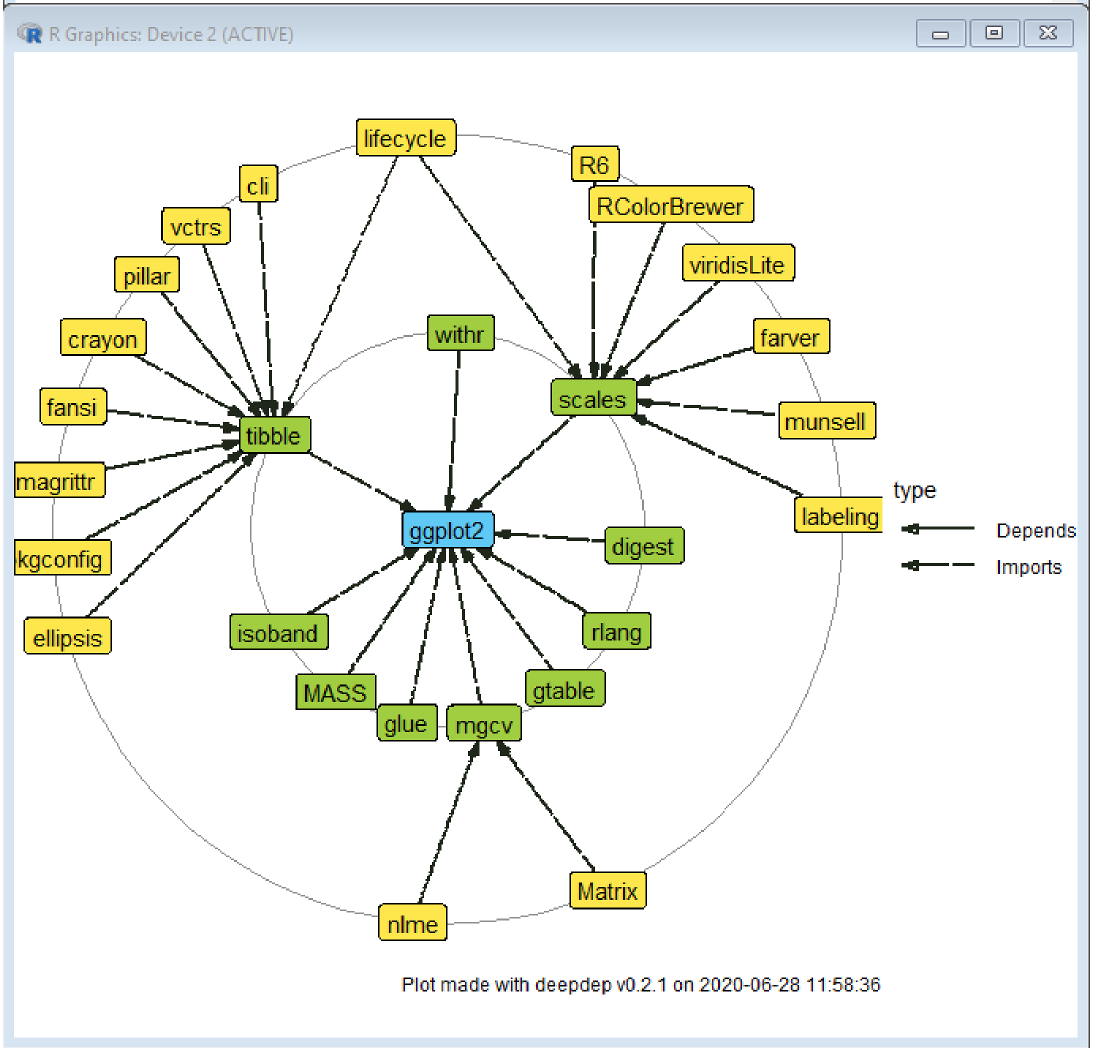
Analyser les dépendences des packages

Il existe un formidable package avec un grand potentiel. Il se nomme **deepdep** et sa documentation (vignette) est disponible ici:

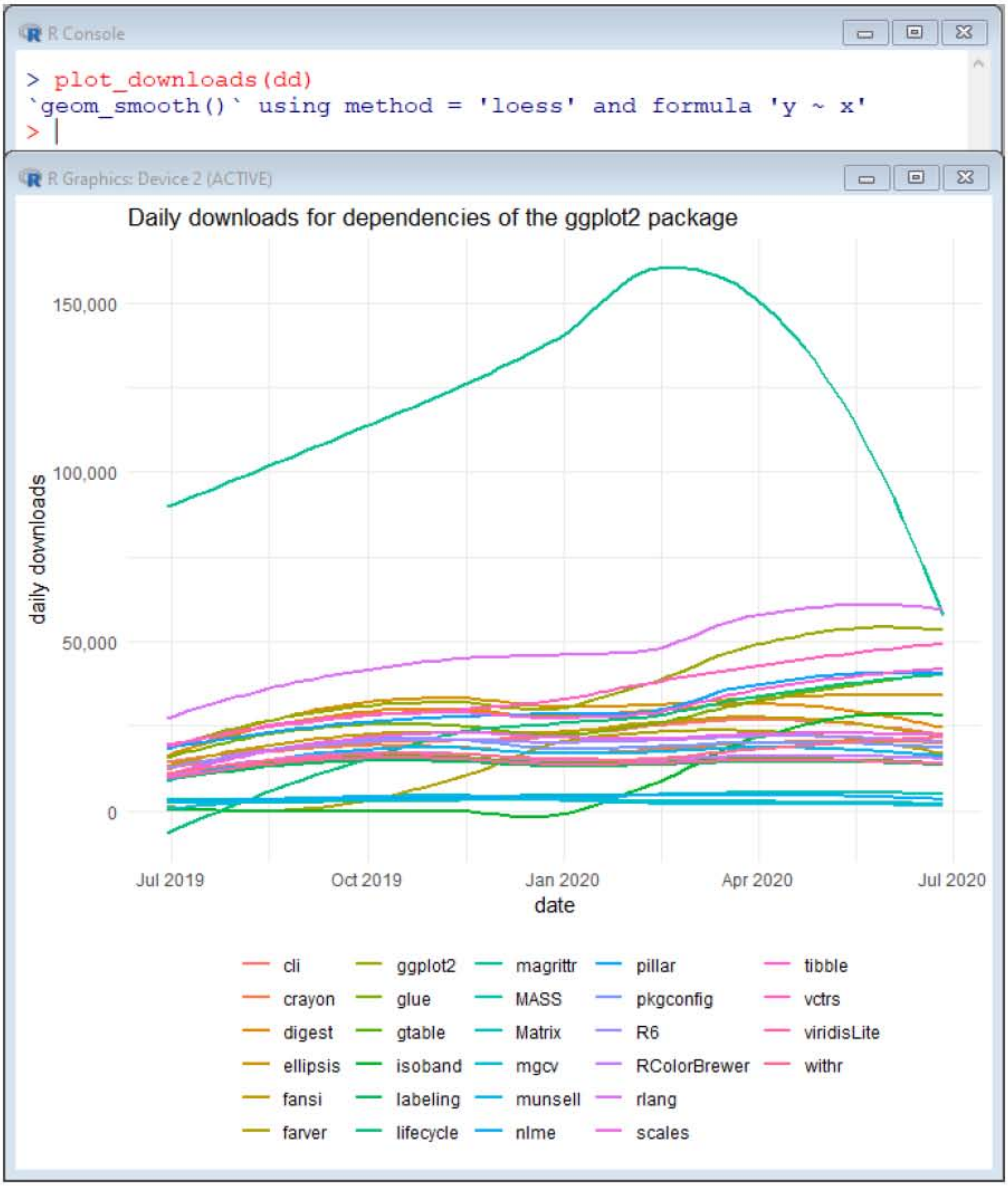
https://dominikrafacz.github.io/deepdep/

Voyons un exemple avec le package ggplots comme dans la vignette:

```
> library(deepdep)
> dd <- deepdep("ggplot2", depth = 2)
> head(dd)
   origin    name   version    type origin_level dest_level
1 ggplot2  digest      <NA> Imports            0          1
2 ggplot2    glue      <NA> Imports            0          1
3 ggplot2  gtable >= 0.1.1 Imports            0          1
4 ggplot2 isoband      <NA> Imports            0          1
5 ggplot2    MASS      <NA> Imports            0          1
6 ggplot2    mgcv      <NA> Imports            0          1
> plot_dependencies(dd, "circular")
> plot_dependencies(dd, "circular")
> 
```

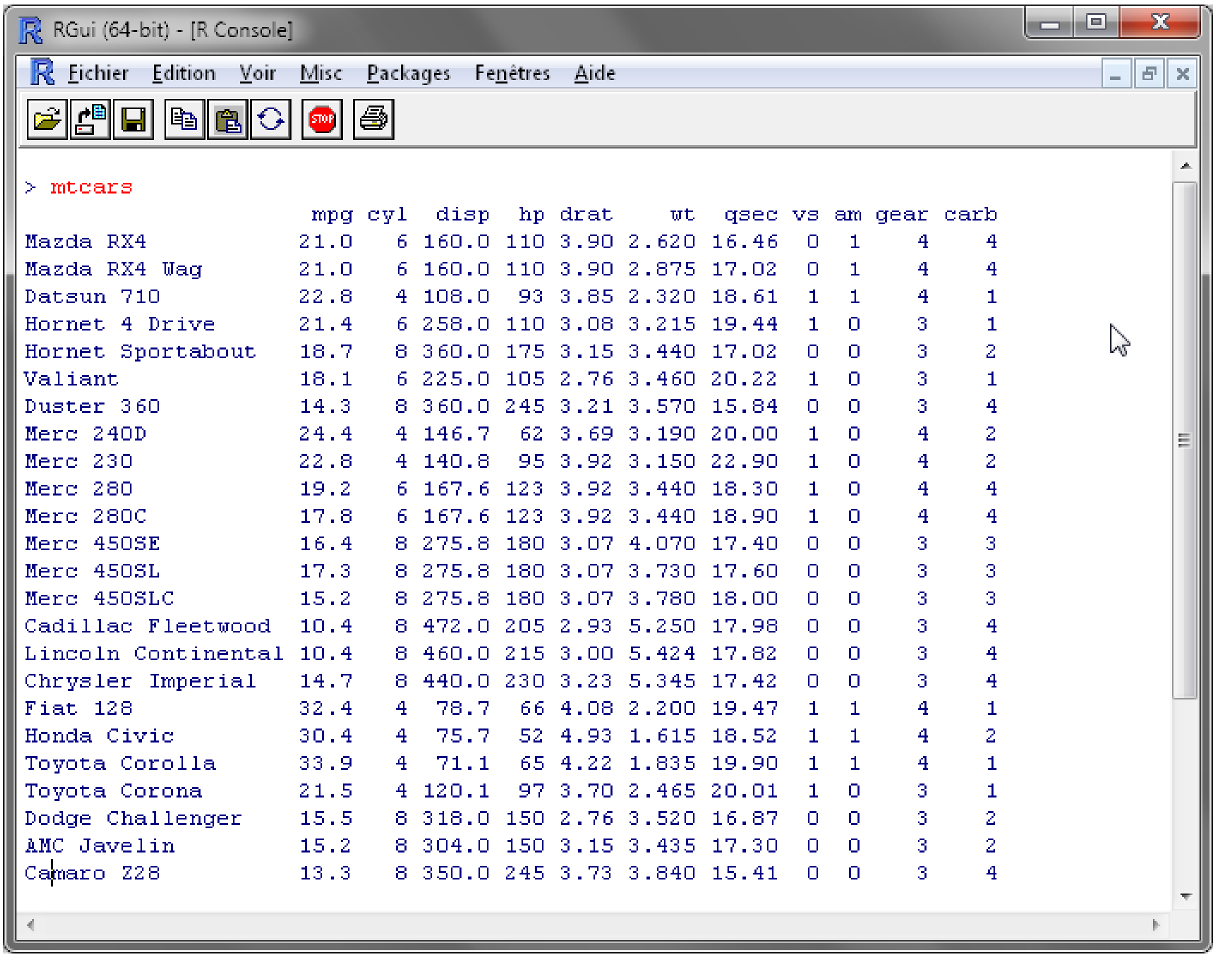


On peut plotter les téléchargements aussi:



Quelques data sets

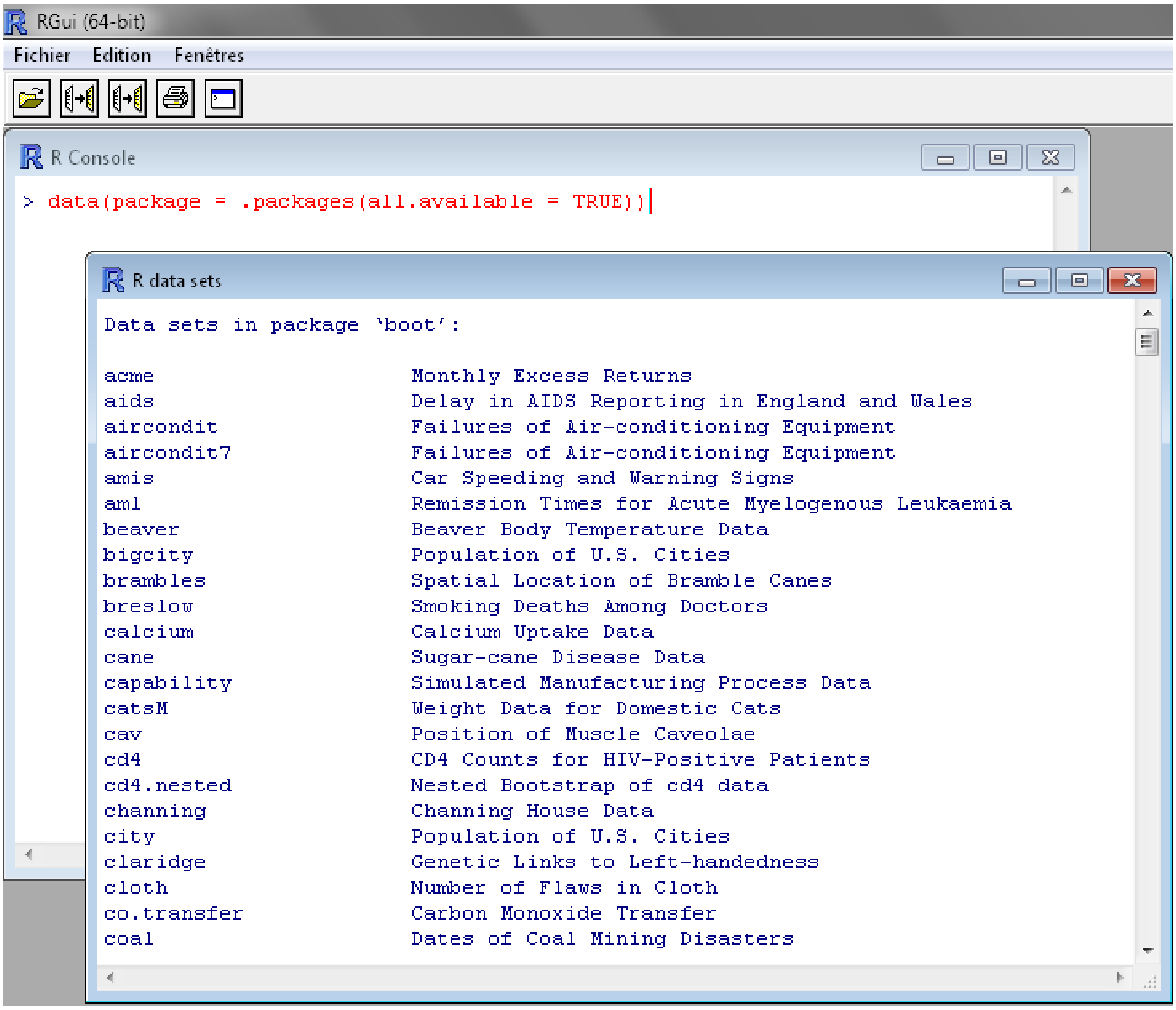
Voici un exemple data set pour se faire la main:




```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4

mais en réalité pour voir tous les data set de tous les packages, vous pouvez utiliser la commande **data(package = .packages(all.available = TRUE))**:



Mais alors pourquoi est-ce que je vais majoritairement utiliser mes propres sets de données par la suite vous demandez-vous peut-être???

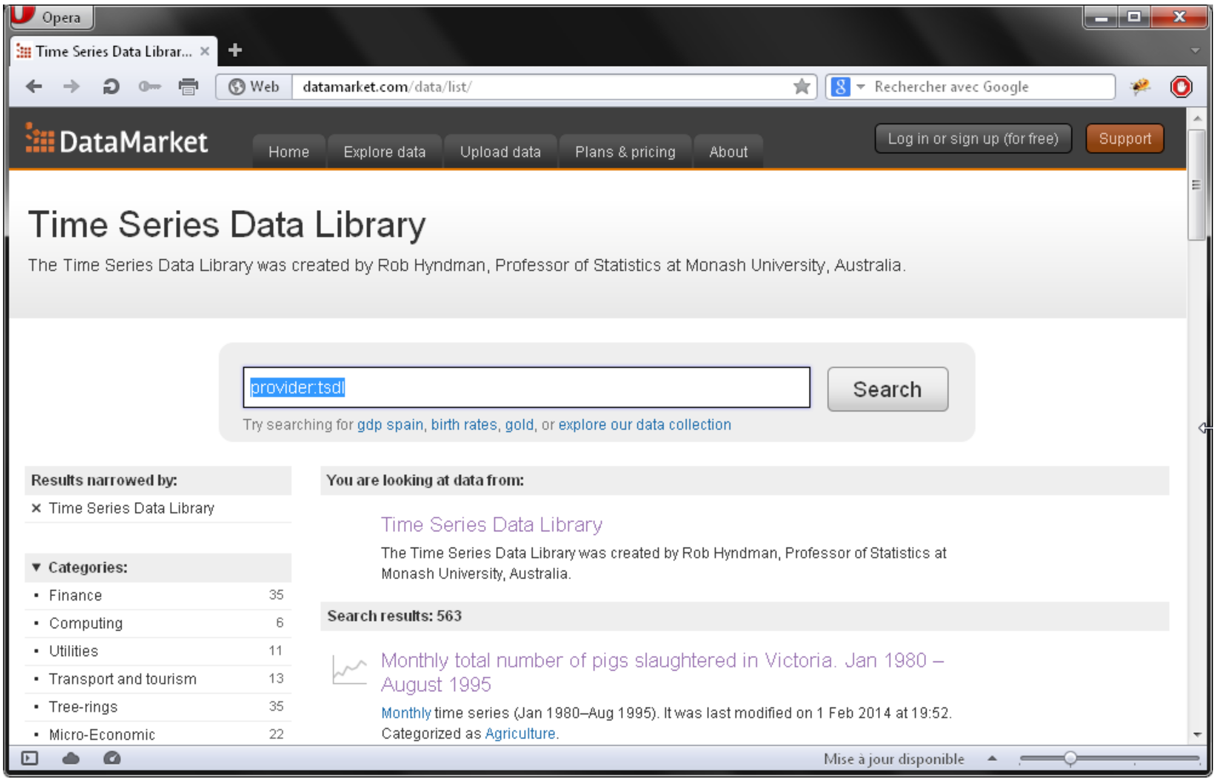
Simplement parce que mes formations Excel, Minitab, SPSS, Tanagra et R ont pour objectif d'être homogènes... voilà simplement l'explication.

Cependant dans la majorité des cas nous enlèverons les accents et les espaces de la ligne de titre (légende des colonnes) car cela pose problème à R (puisque non conforme aux normes) et que cela nous ferait perdre du temps de les renommer avec les commandes de R (le temps étant précieux en formation continue...).

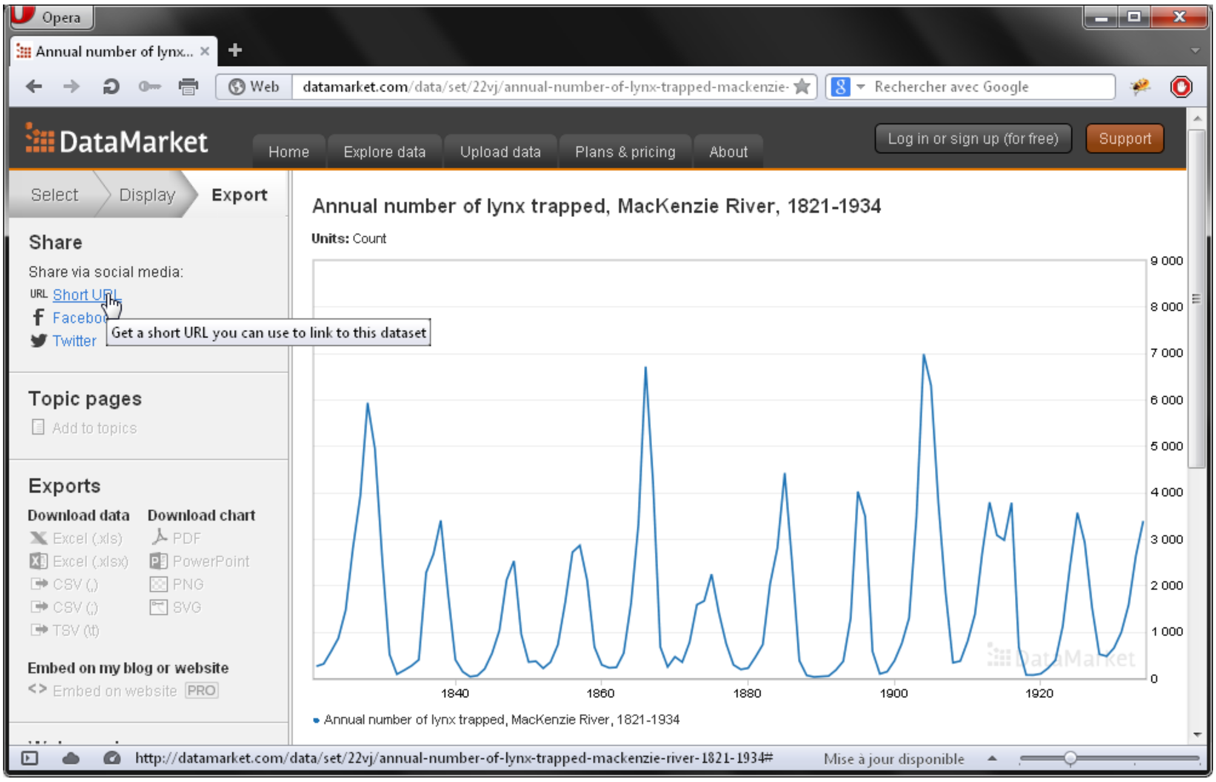
Signalons également la possibilité de se connecter à d'excellents dépôts de données. Comme par exemple à l'adresse suivante:

http://datamarket.com/data/list/?q=provider:tsdl

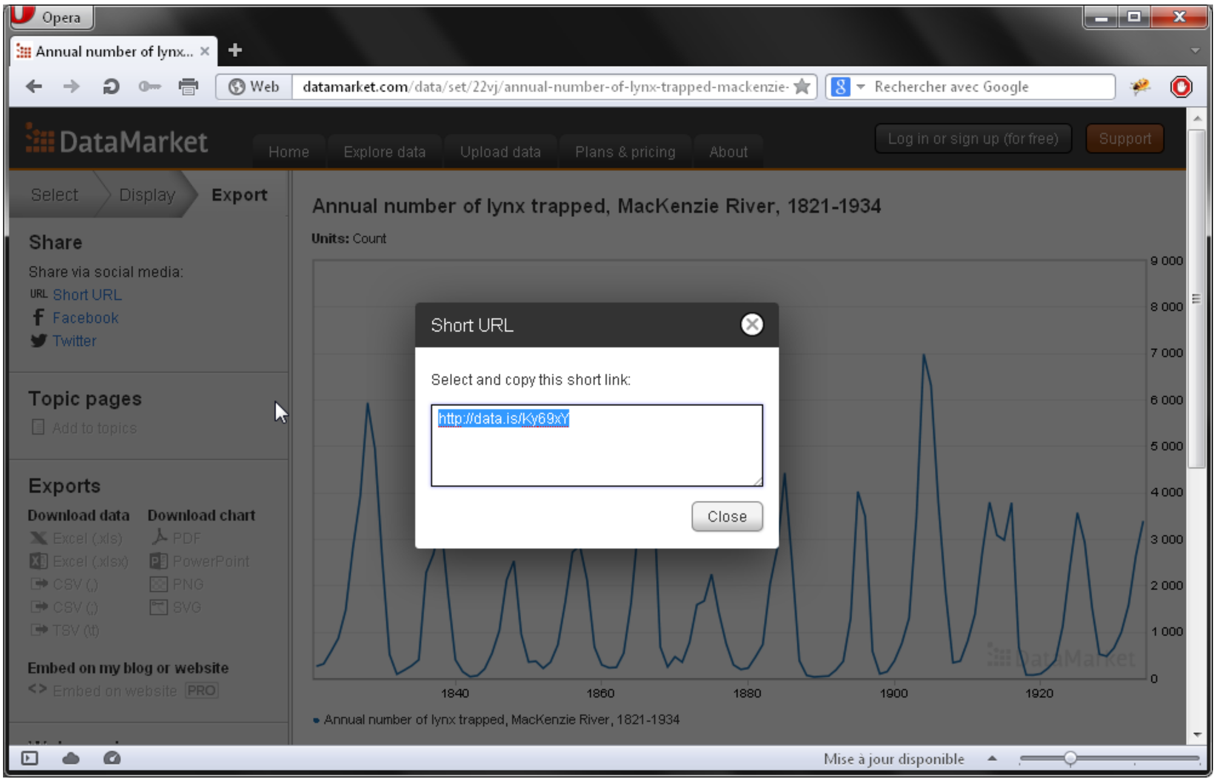
Ce qui vous amènera sur la page suivante:



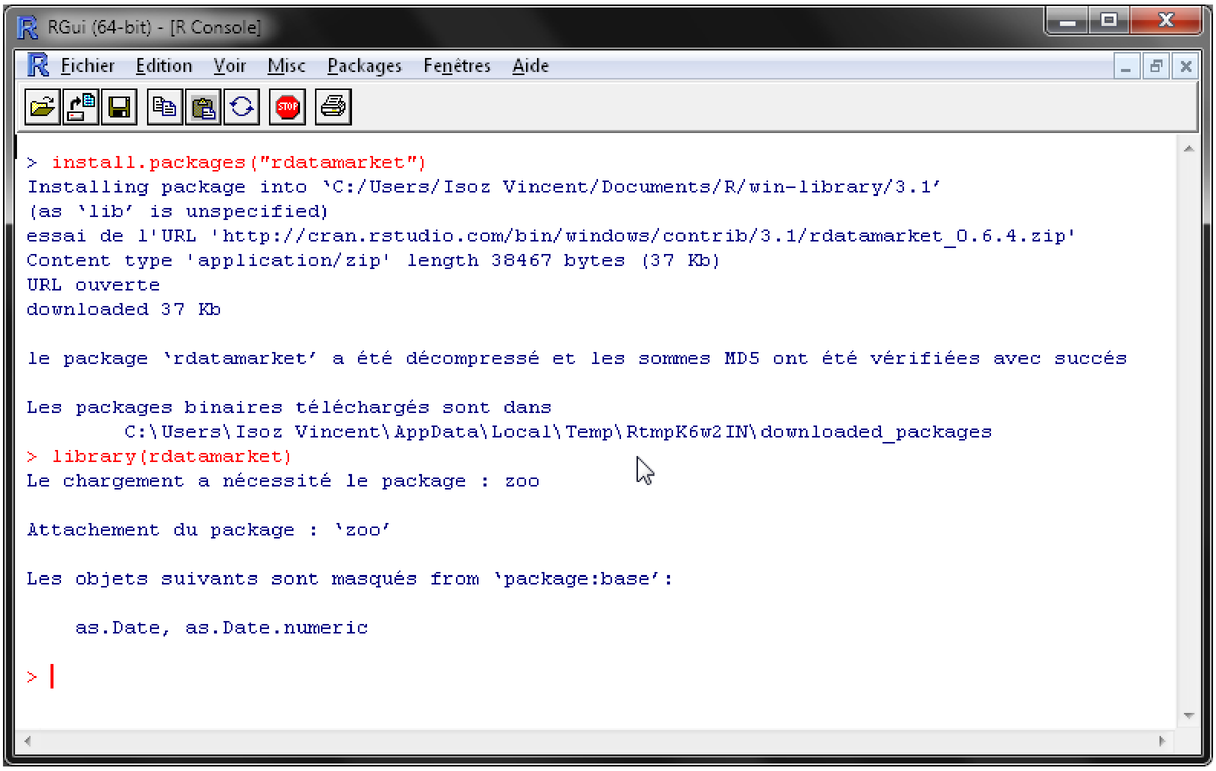
Vous pouvez ensuite choisir un jeu de données en cliquant dessus, en allant dans **Export**:



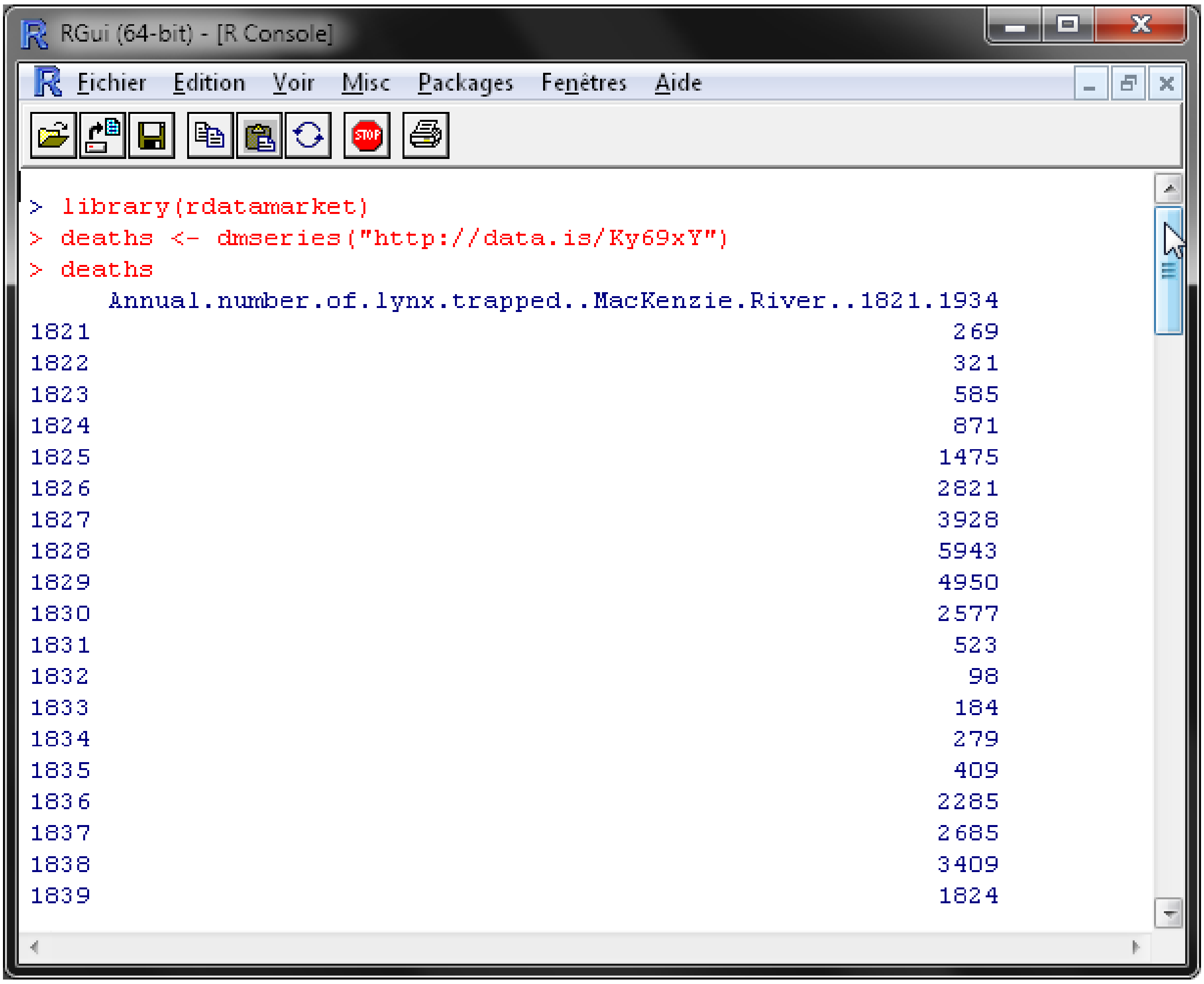
et ensuite en cliquant sur **Short URL**, vous obtiendrez le lien direct d'accès à la source de données:



Il ne reste plus qu'à installer et charger le package R permettant d'y accéder:



et ensuite:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(rdatamarket)
> deaths <- dmseries("http://data.is/Ky69xY")
> deaths
Annual.number.of.lynx.trapped..MacKenzie.River..1821.1934
1821 269
1822 321
1823 585
1824 871
1825 1475
1826 2821
1827 3928
1828 5943
1829 4950
1830 2577
1831 523
1832 98
1833 184
1834 279
1835 409
1836 2285
1837 2685
1838 3409
1839 1824

et voilà!

Contrôler la mémoire

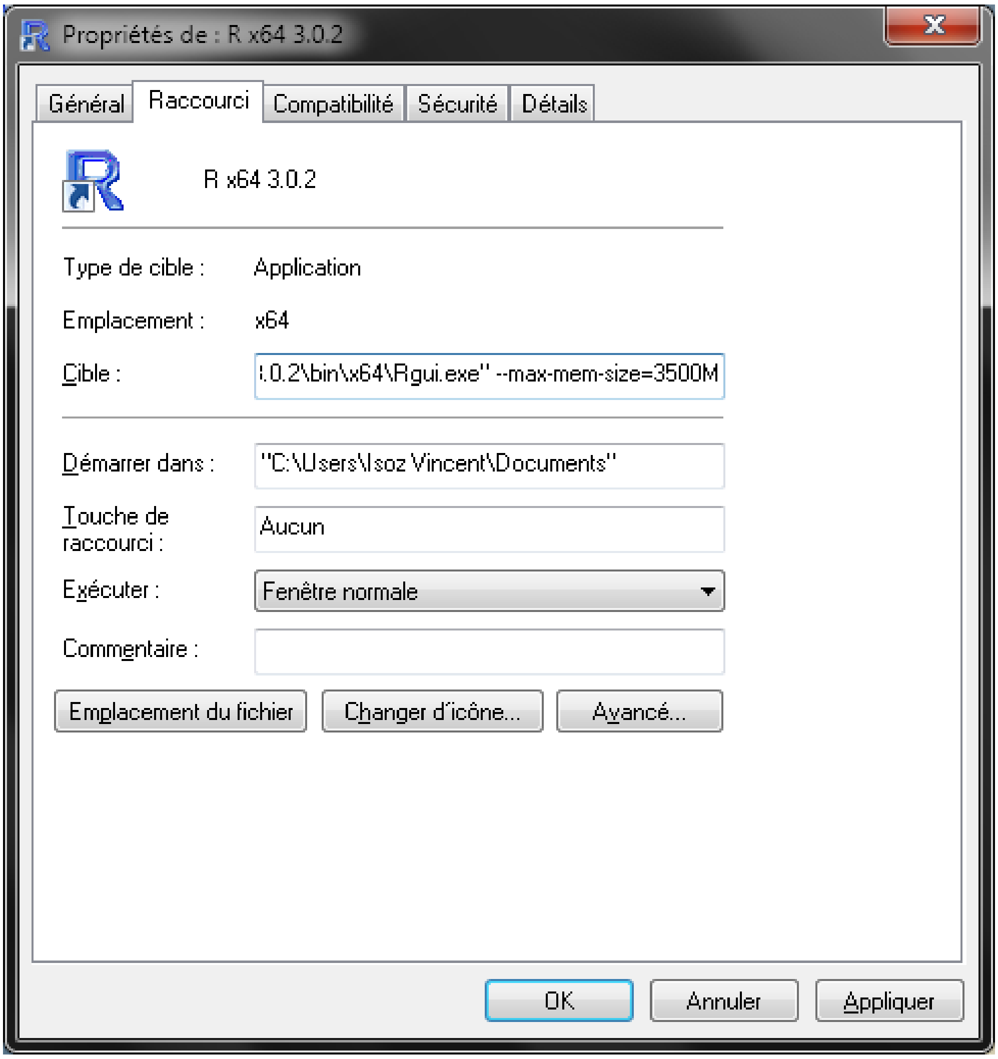
Lorsque vous travaillez sur des jeux de données énormes, il va vous falloir maîtriser la mémoire que R utilise. Nous verrons plus tard quand nous manipulerons des variables scalaires, vectorielles, matricielles ou autres comment supprimer celles-ci de la mémoire mais ici nous nous intéressons vraiment à un cas plus général.

D'abord deux infos:

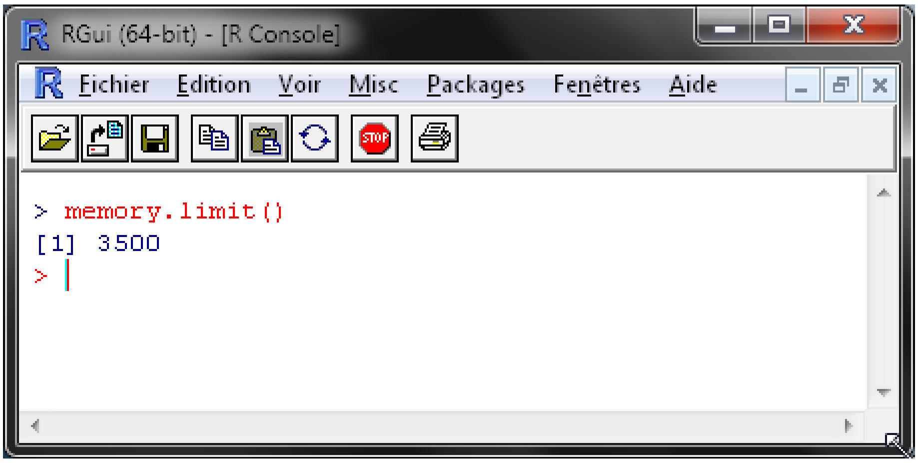
- **R 32 bits** est limité **4GB** (en pratique c'est plutôt 3GB car l'OS a besoin de fonctionner)

- **R 64 bits** est limité à **128TB** sous LINUX (en pratique il faut trouver de la place pour mettre tout cela...) et **8TB** sous Microsoft Windows…

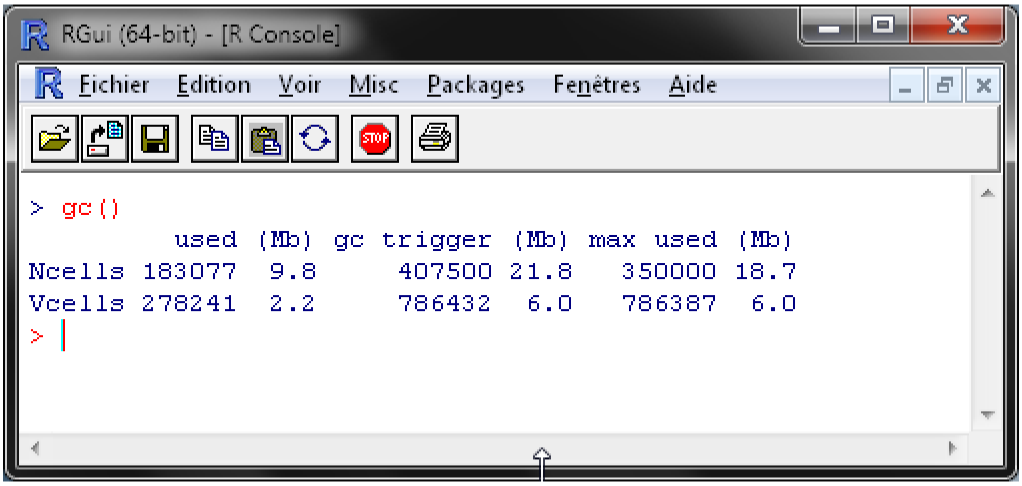
Pour étendre la mémoire disponible à R ou la limiter, vous allez simplement dans les propriétés du raccourci de R pour rajouter en Mb la taille désirée:



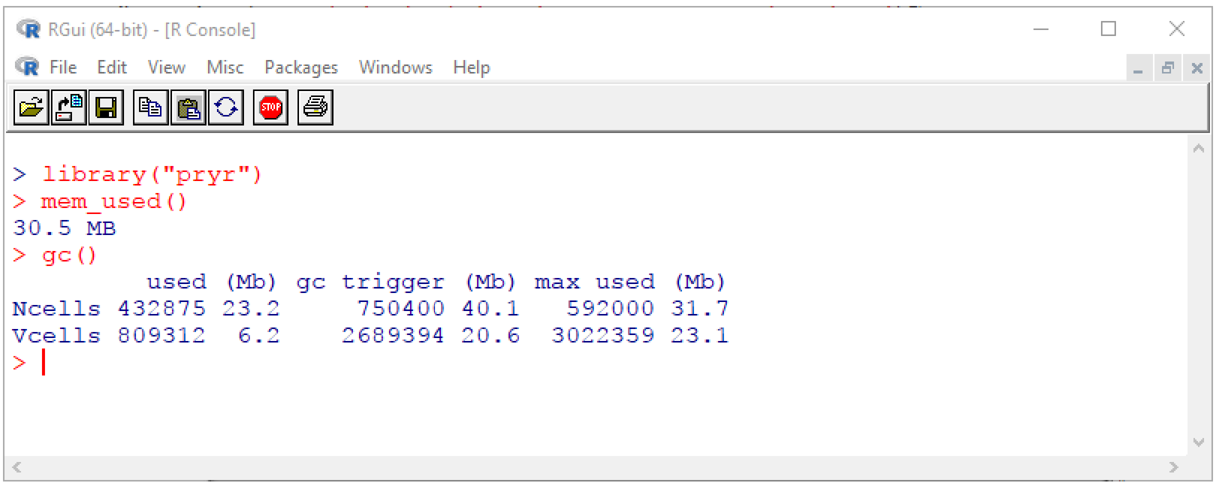
Vous pourrez alors contrôler après l'ouverture de R:



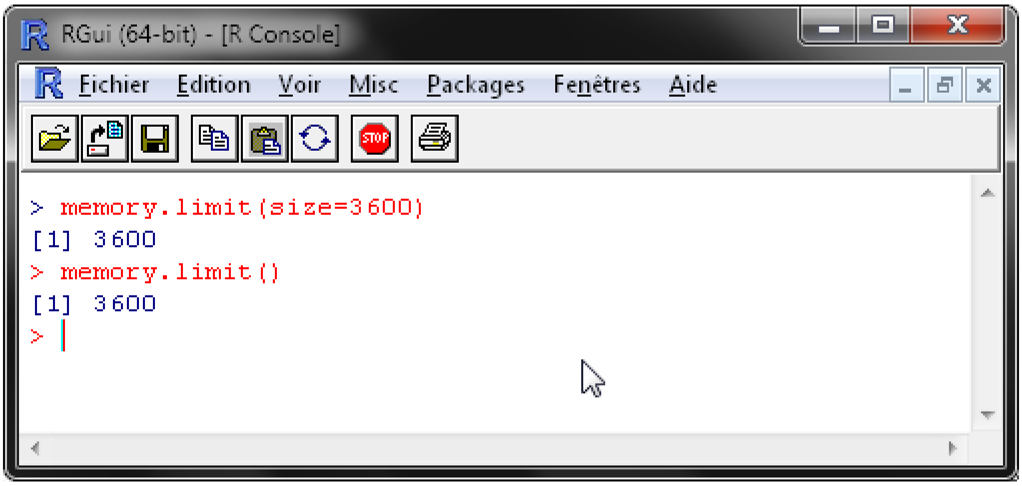
Ensuite, on peut faire un état des lieux avec la commande du garbage collector qui au passage détruit les objets inutiles dans la mémoire **gc( )**:



ou en comparant avec la fonction **mem_used( )** du package **pryr**:

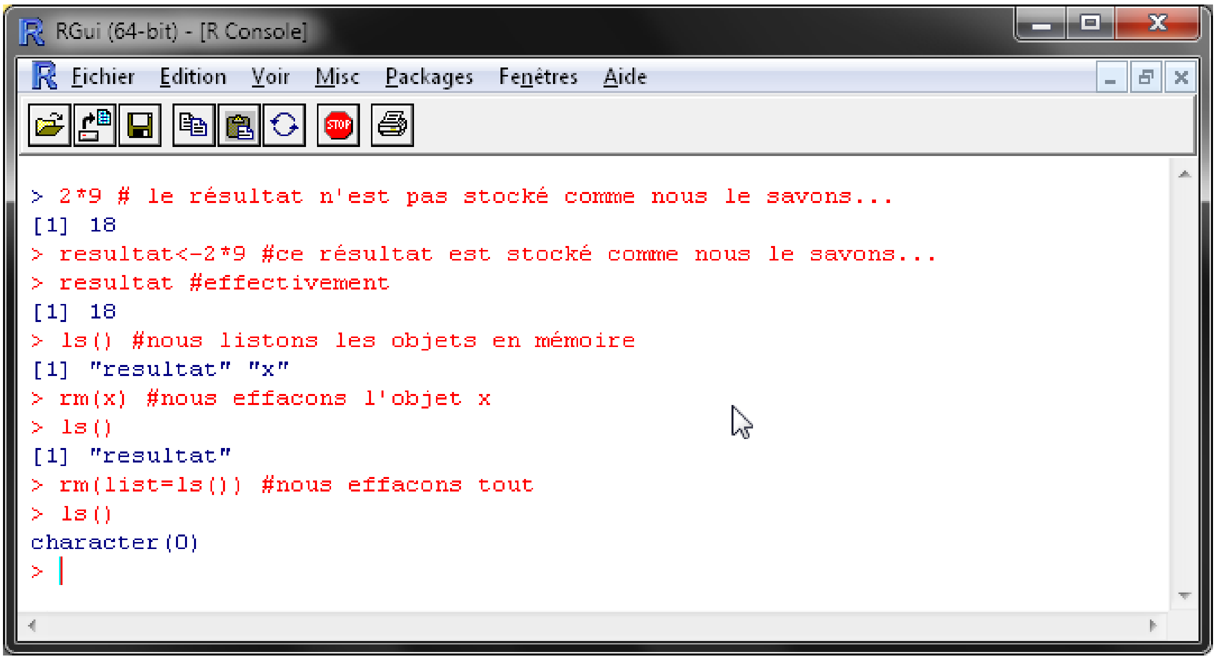


Sinon on peut aussi changer la mémoire allouée à la volée directement dans R plutôt qu'à l'ouverture (cependant cela n'est pas toujours permis par le système!):

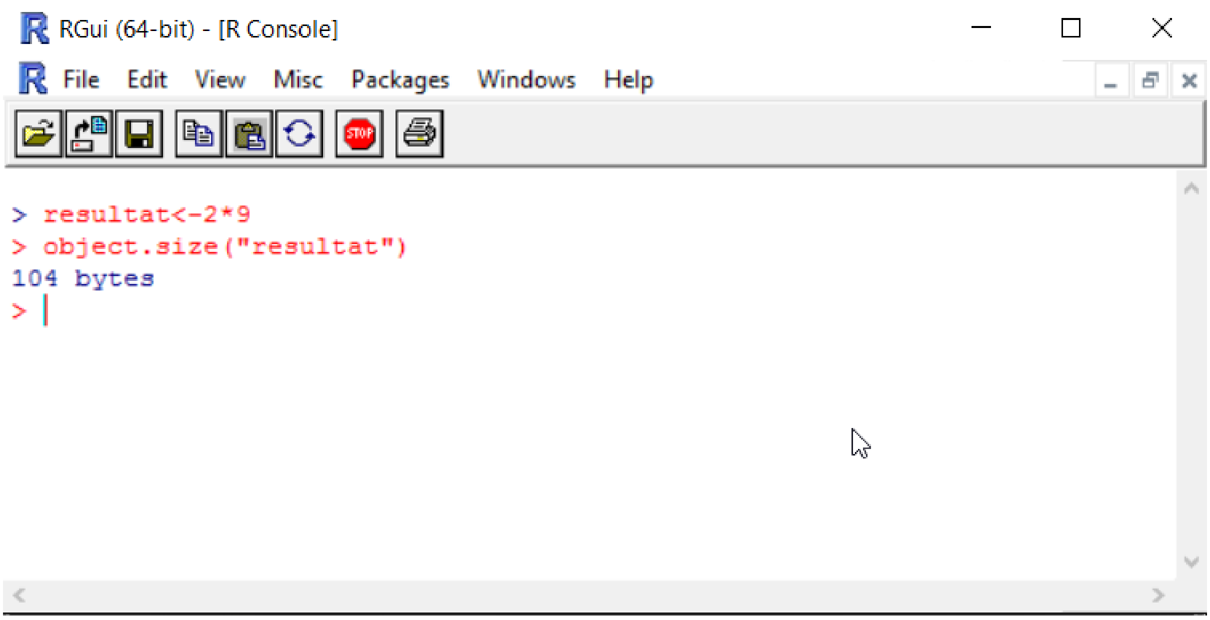


Gérer les variables dans la mémoire

Pas grand chose à dire car c'est assez simple. Il suffit de lire pour comprendre comment utiliser les options de base des commandes **ls( )** et **rm( )** et supprimer ou lister les objets en mémoire:



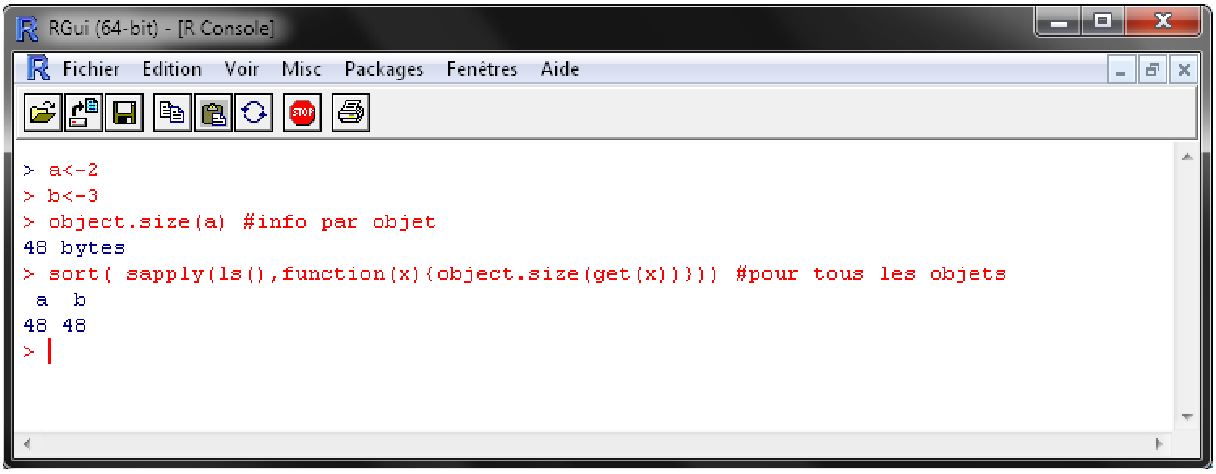
Avec la fonction **object.size( )** nous pouvons aussi obtenir la taille mémoire d'une variable:



À remarquer que nous pouvons obtenir une vision plus structurée des variables créées à l'aide de la commande **ls.str( )** :



Nous pouvons aussi voir la taille d'un objet ou de plusieurs dans la mémoire:



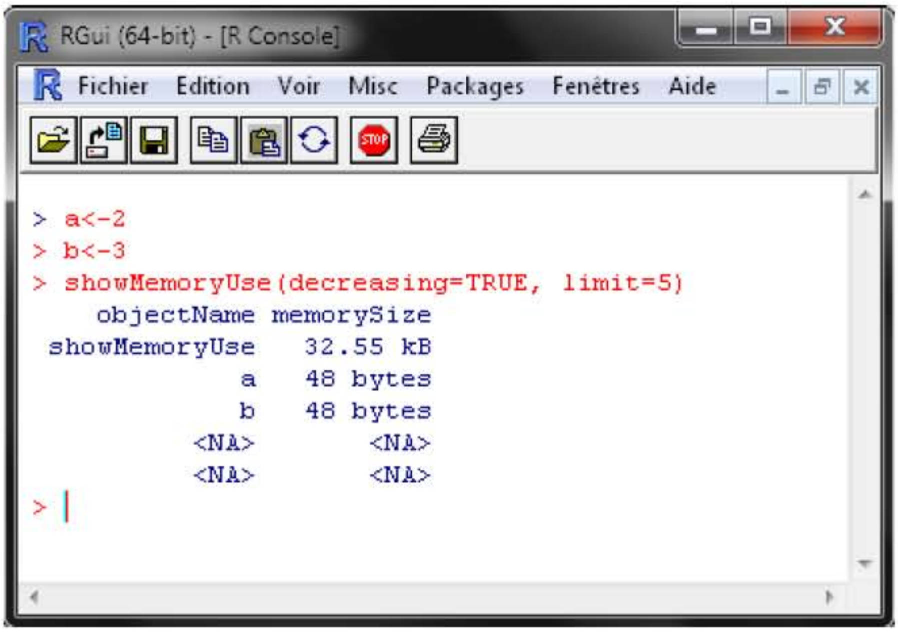

```
> a<-2
> b<-3
> object.size(a) #info par objet
48 bytes
> sort( sapply(ls(),function(x){object.size(get(x))})) #pour tous les objets
 a  b
48 48
>
```

Il y en a qui ont fait encore plus fort dans la communauté R:



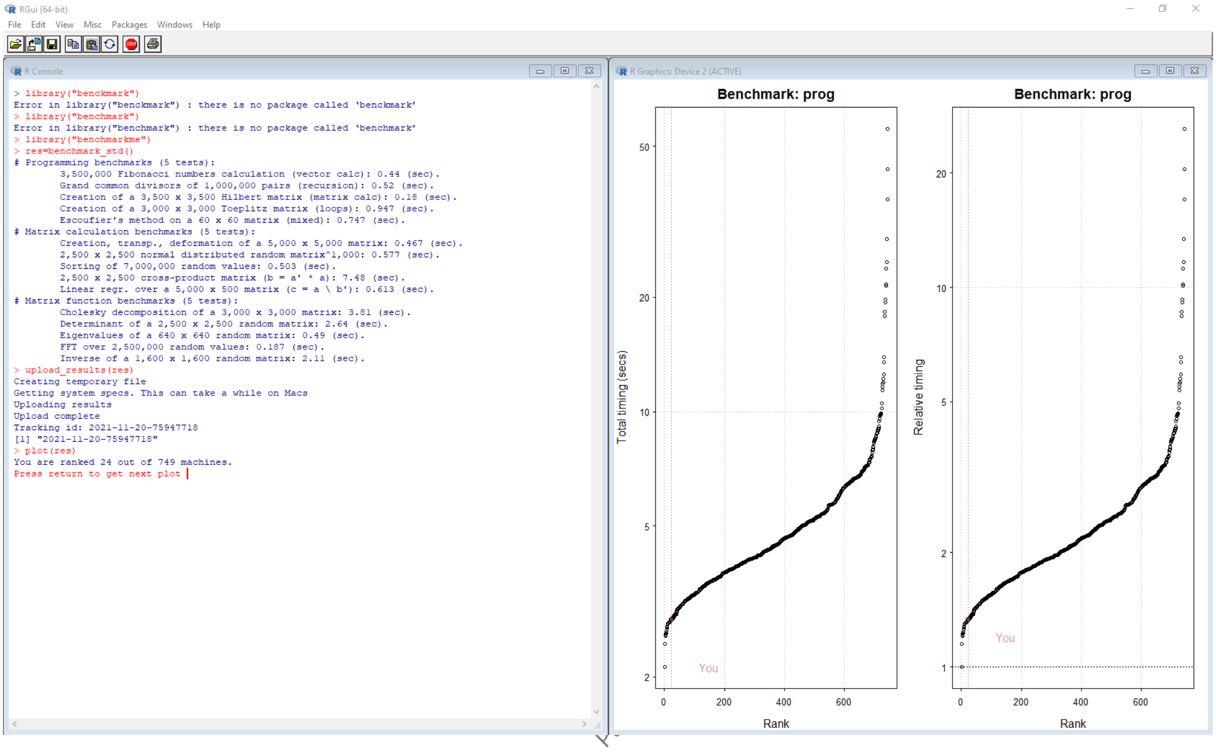

```
> showMemoryUse <- function(sort="size", decreasing=FALSE, limit) {
+
+   objectList <- ls(parent.frame())
+
+   oneKB <- 1024
+   oneMB <- 1048576
+   oneGB <- 1073741824
+
+   memoryUse <- sapply(objectList, function(x) as.numeric(object.size(eval(parse(text=x)))))
+
+   memListing <- sapply(memoryUse, function(size) {
+         if (size >= oneGB) return(paste(round(size/oneGB,2), "GB"))
+         else if (size >= oneMB) return(paste(round(size/oneMB,2), "MB"))
+         else if (size >= oneKB) return(paste(round(size/oneKB,2), "kB"))
+         else return(paste(size, "bytes"))
+       })
+
+   memListing <- data.frame(objectName=names(memListing),memorySize=memListing,row.names=NULL)
+
+   if (sort=="alphabetical") memListing <- memListing[order(memListing$objectName,decreasing=decreasing),]
+   else memListing <- memListing[order(memoryUse,decreasing=decreasing),] #will run if sort not specified or "size"
+
+   if(!missing(limit)) memListing <- memListing[1:limit,]
+
+   print(memListing, row.names=FALSE)
+   return(invisible(memListing))
+ }
>
```

Ce qui donne:

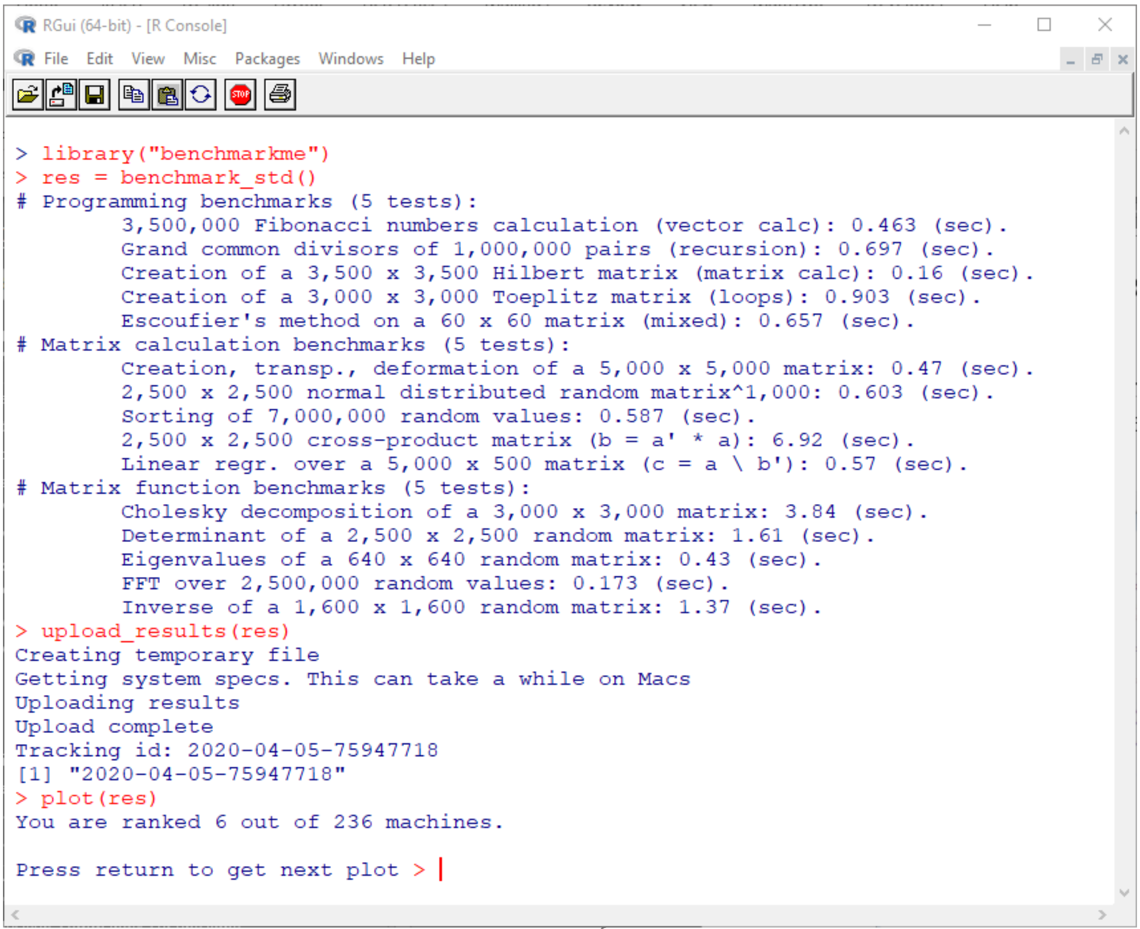

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> a<-2
> b<-3
> showMemoryUse(decreasing=TRUE, limit=5)
objectName memorySize
showMemoryUse 32.55 kB
a 48 bytes
b 48 bytes
<NA> <NA>
<NA> <NA>
>

Benchmarker son ordinateur

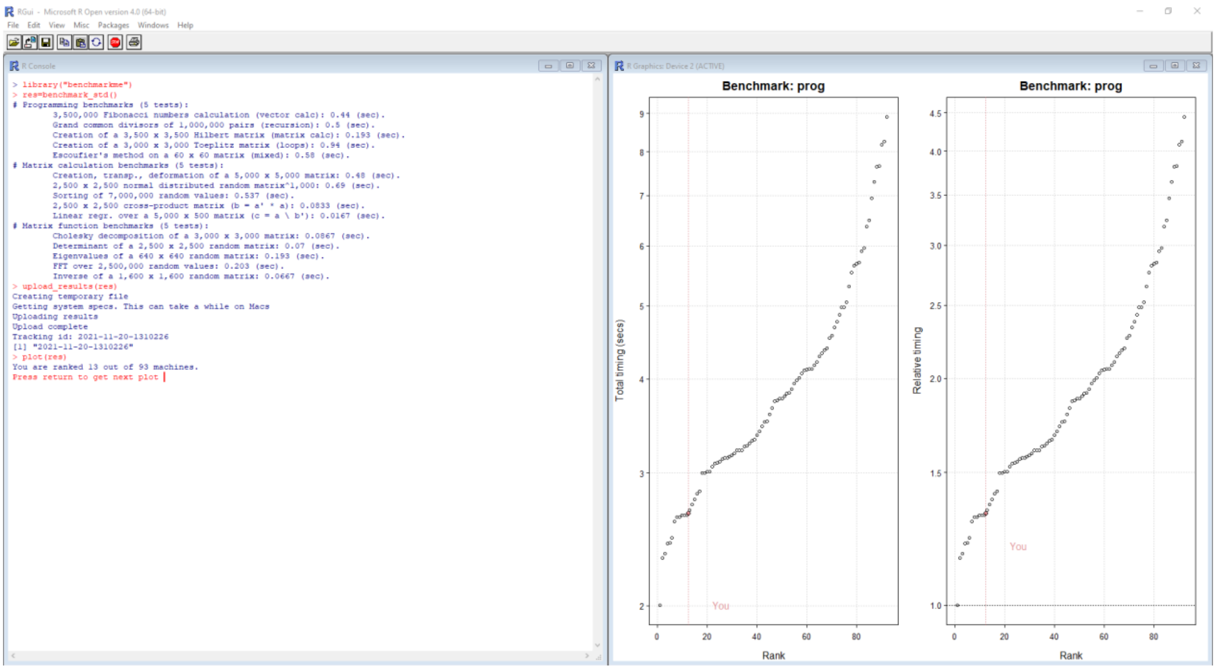
Comme dans le cours MATLAB, au début de celui-ci, voyons comment nous pouvons benchmarker la performance de notre ordinateur relativement à ses performancs avec R en utilisant le package **benchmarkme**:



avec un zoom sur la partie script au cas où ce serait trop petit:



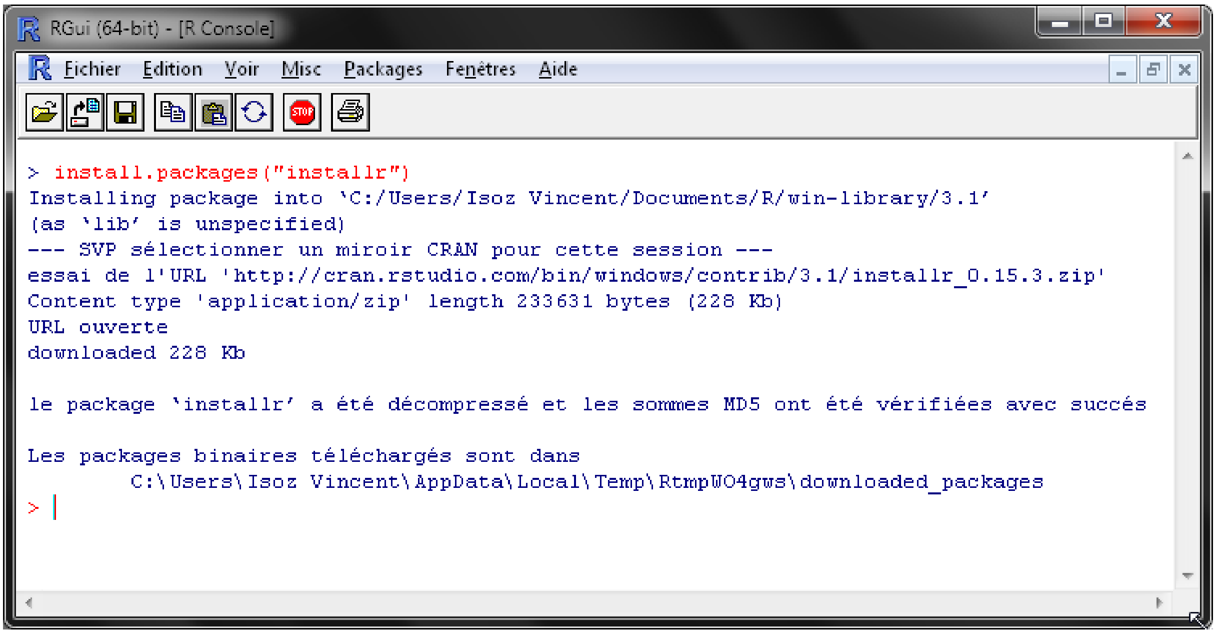
C'est intéressant de comparer ce résultat avec celui obtenu avec Microsoft R Open:



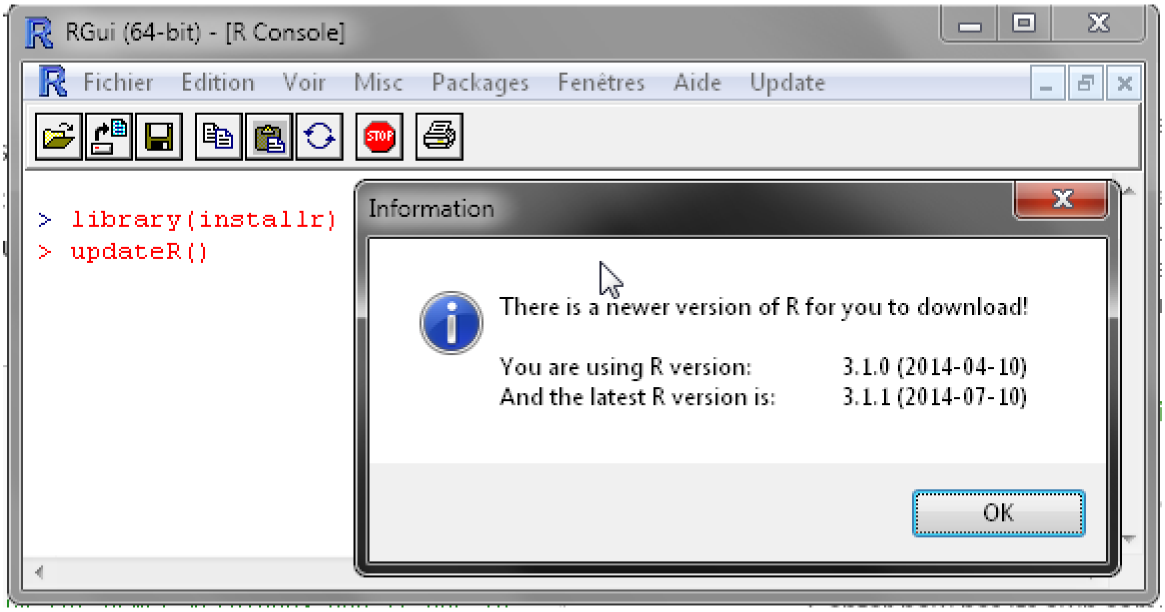
Mettre à jour R

Mettre à jour R reste à ce jour un peu pénible car il faut aller sur le site web du CRANS, télécharger, etc. etc. etc.

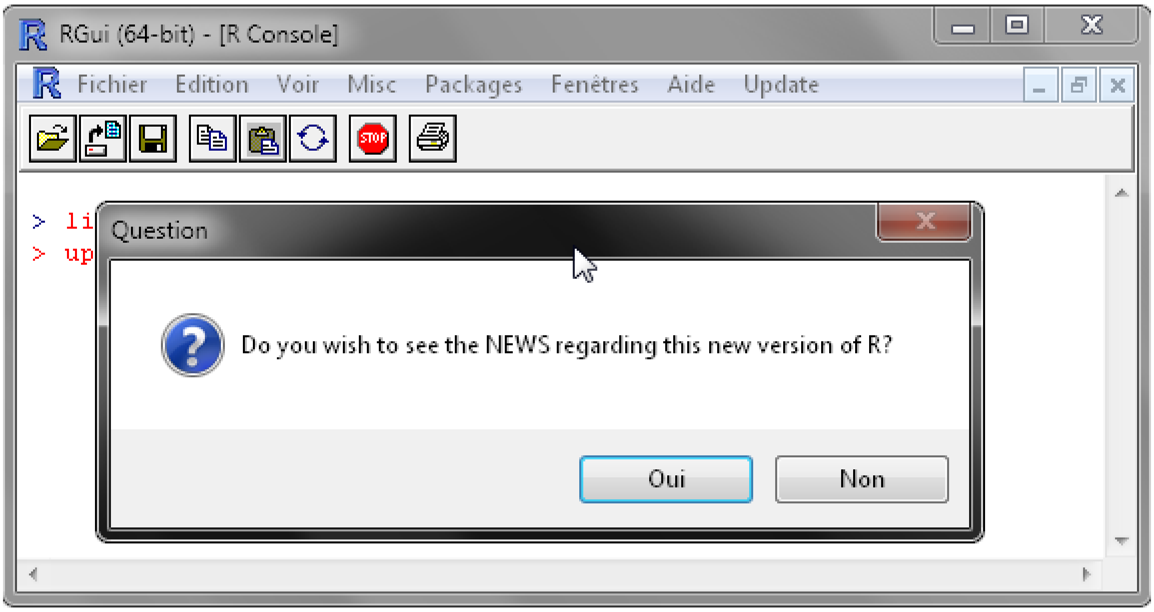
Heureusement un package **installr** a été développé pour faciliter l'opération:



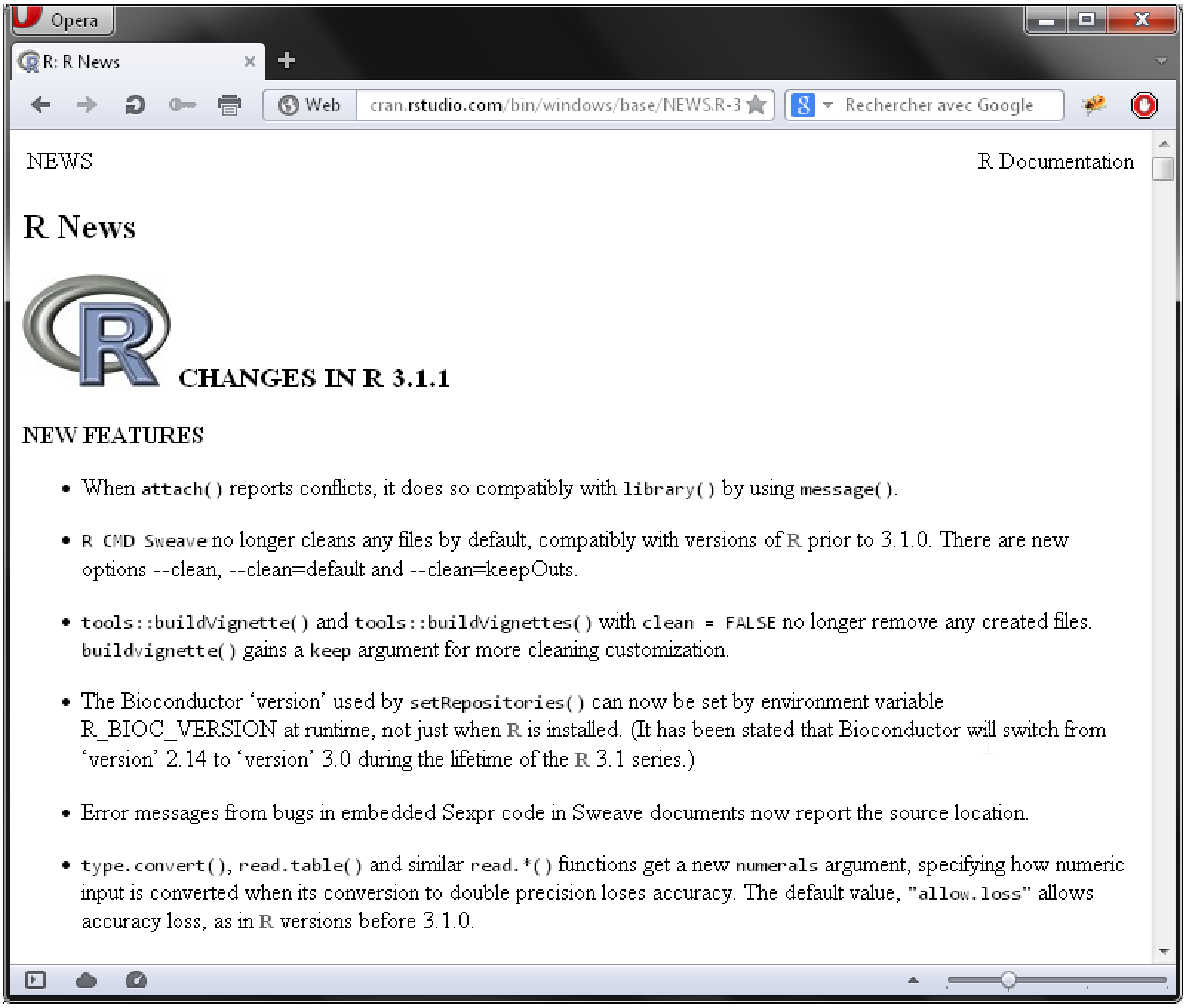
Une fois installé, nous le chargeons et l'exécutons:



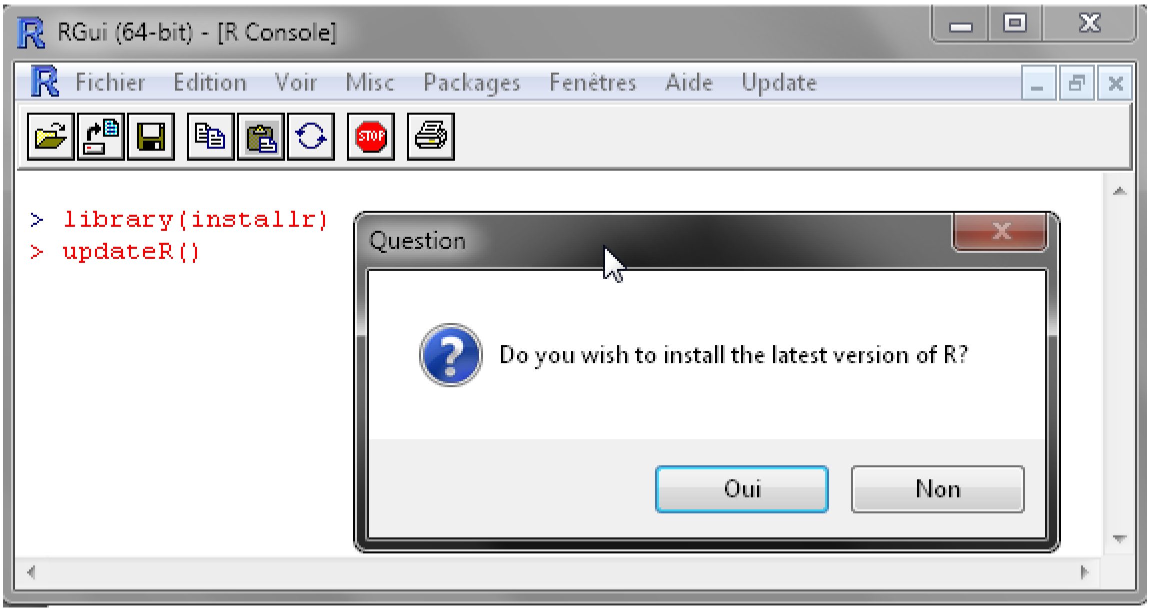
Nous validons par **OK**:



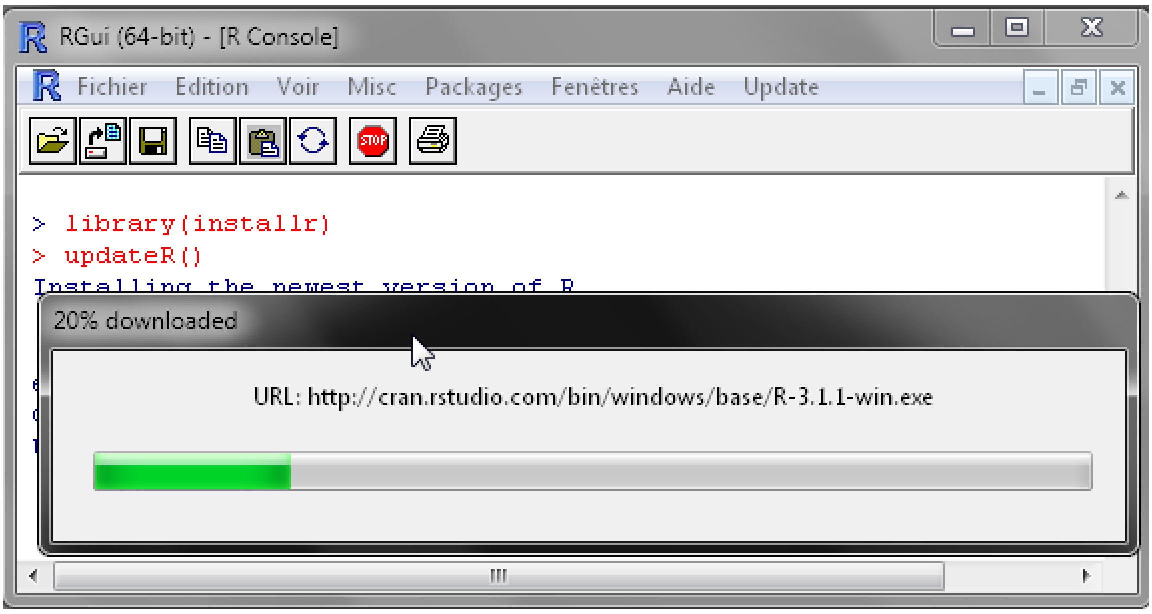
Si nous cliquons sur **Oui**, nous avons:



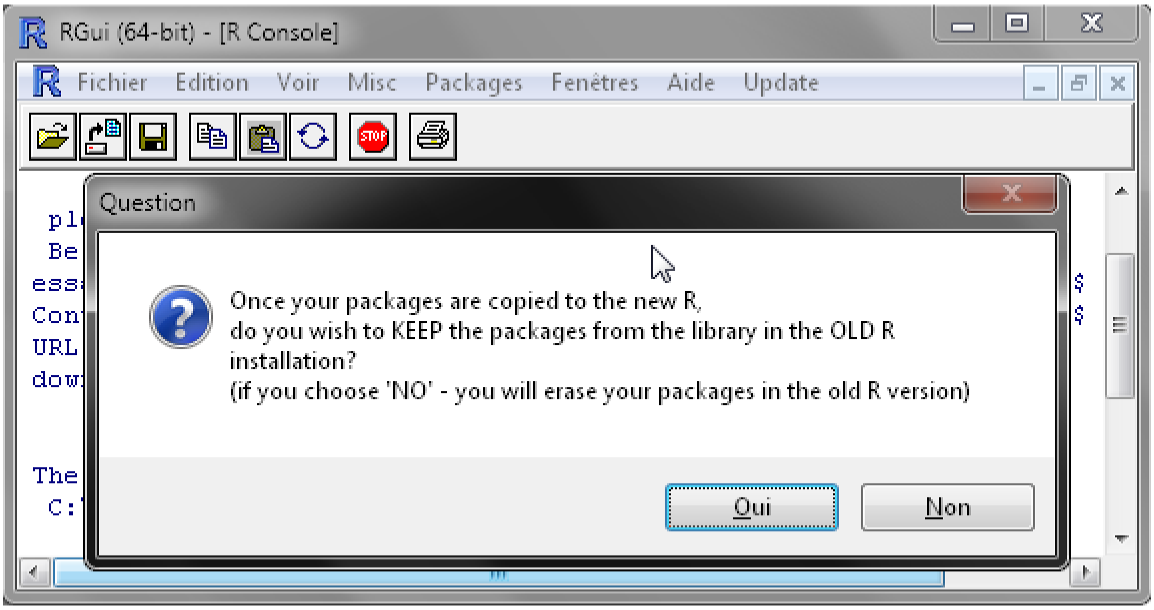
Il vient ensuite:



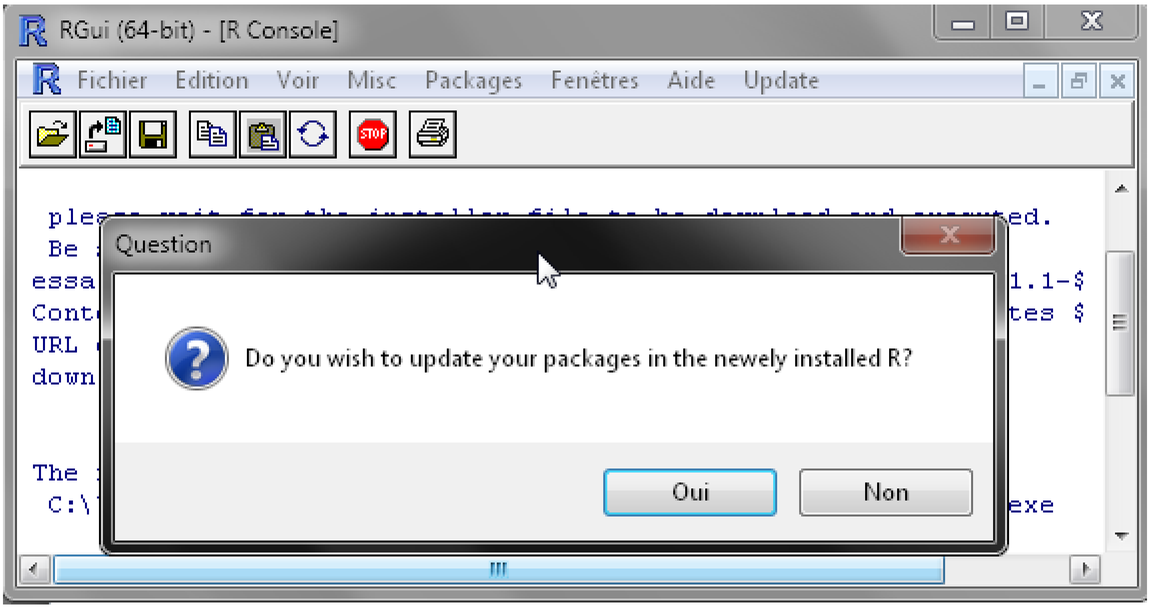
Nous cliquons sur **Oui** et attendons:



Une fois le téléchargement terminé, l'installation classique se fait et une fois celle-ci terminée, nous avons:



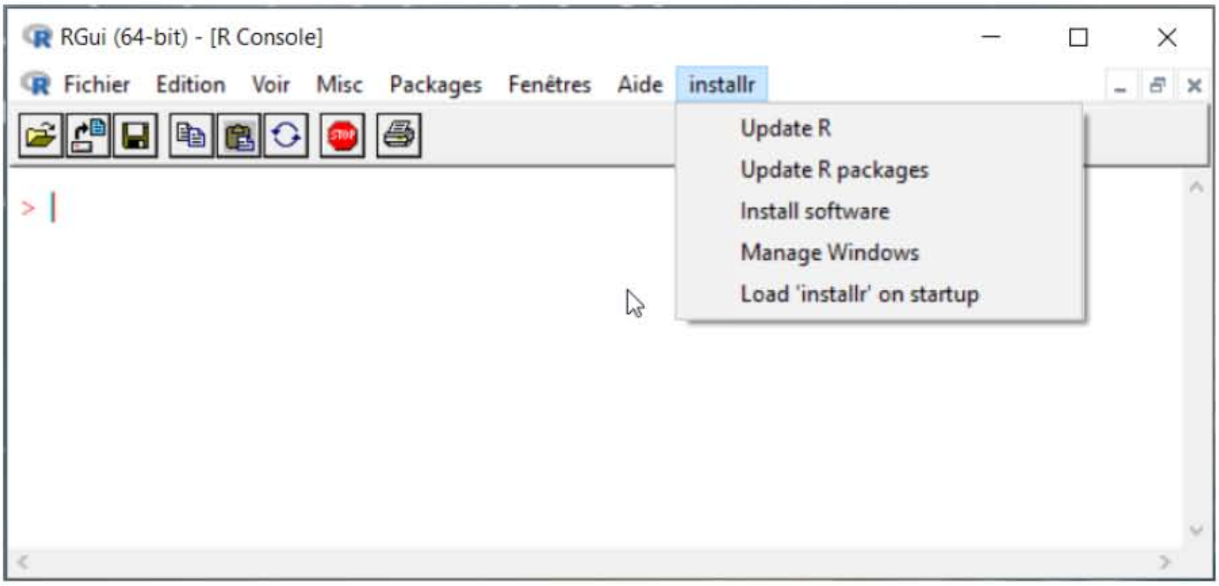
Et ensuite:



Et si vous cliquez sur **Oui** c'est terminé.

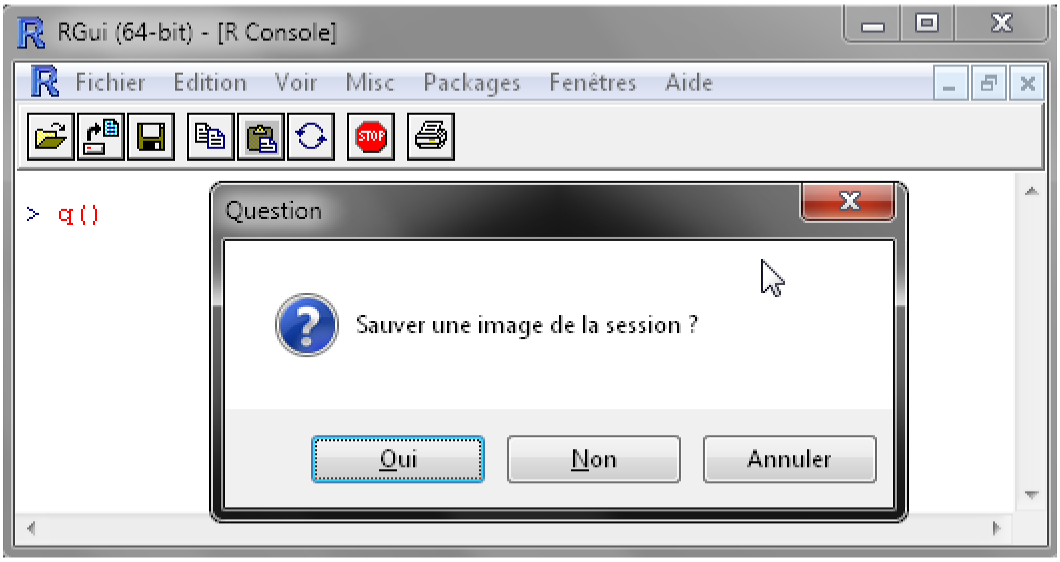
Il faut savoir que même si cela installe la nouvelle version, cela ne supprime pas l'ancienne!

Notez enfin que ce package ajoute le menu suivant à R:

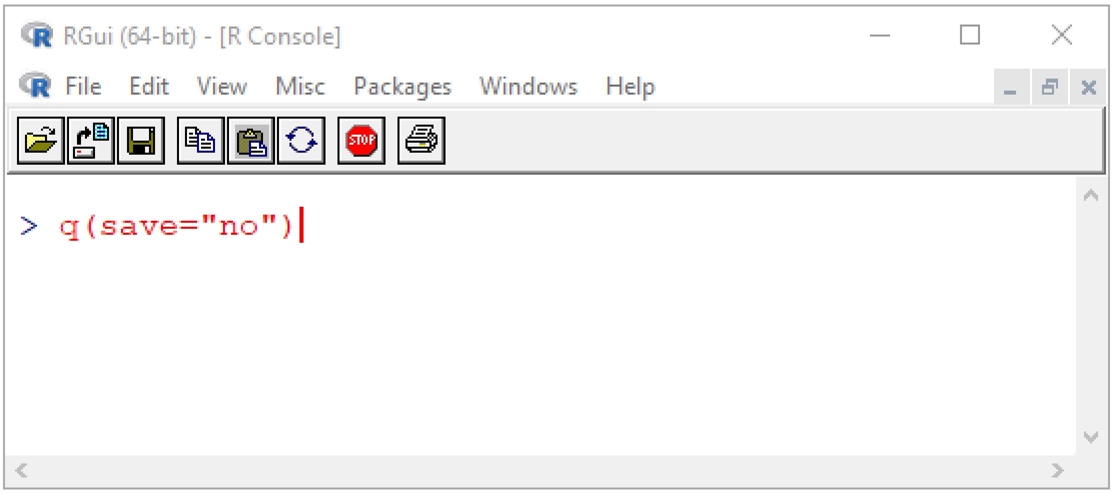

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide installr
Update R
Update R packages
Install software
Manage Windows
Load 'installr' on startup

Fermer R

Quand vous ferrez du scripting, il vous sera utile de connaître la commande qui ferme R. Soit vous utilisez alors la commande **q( )** seule qui aura l'effet suivant:



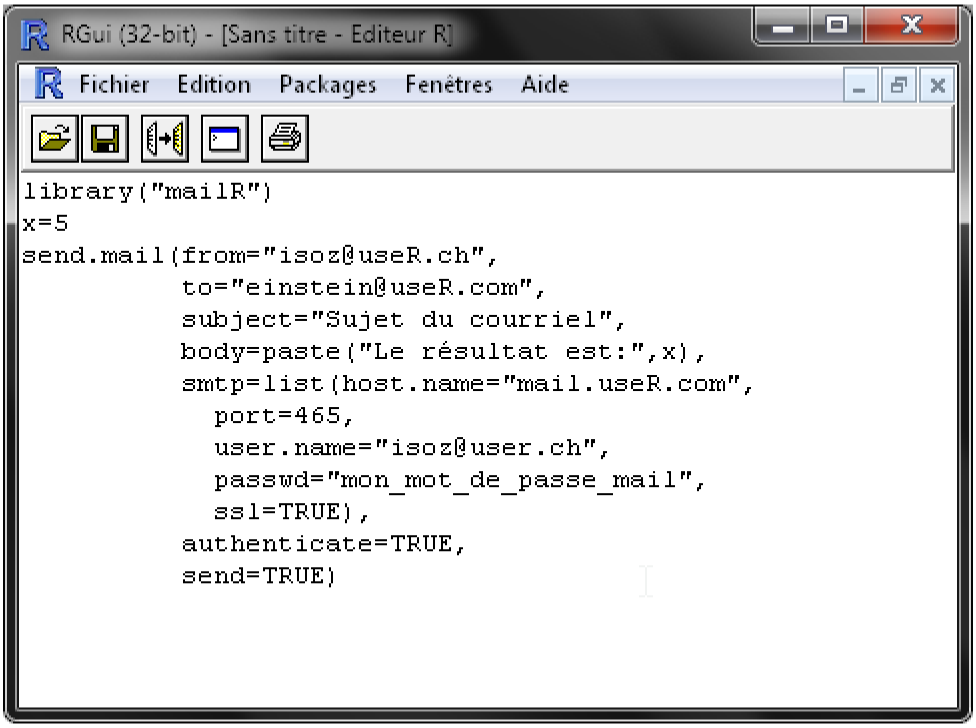
soit vous savez d'avance qu'il ne faut pas enregistrer et dès lors vous passez la valeur *no* en paramètre:



Envoyer un courriel (email) via R

Voici une méthode utilisant la fonction **send.mail( )** du package **mailR** pour envoyer par courriel des résultats de calculs ne nécessitant pas de passer par du VBScript ou par GMail (ici seul un exemple au format texte brut est fait, je développerai si un client me demande un cas plus compliqué):

Évidemment il faut changer dans le script ci-dessous l'adresse e-mail de l'expéditeur, du récepteur, le protocole, le nom d'utilisateur, le mot de passe, bref tous les paramètres techniques (mais j'ai testé et cela marche!):



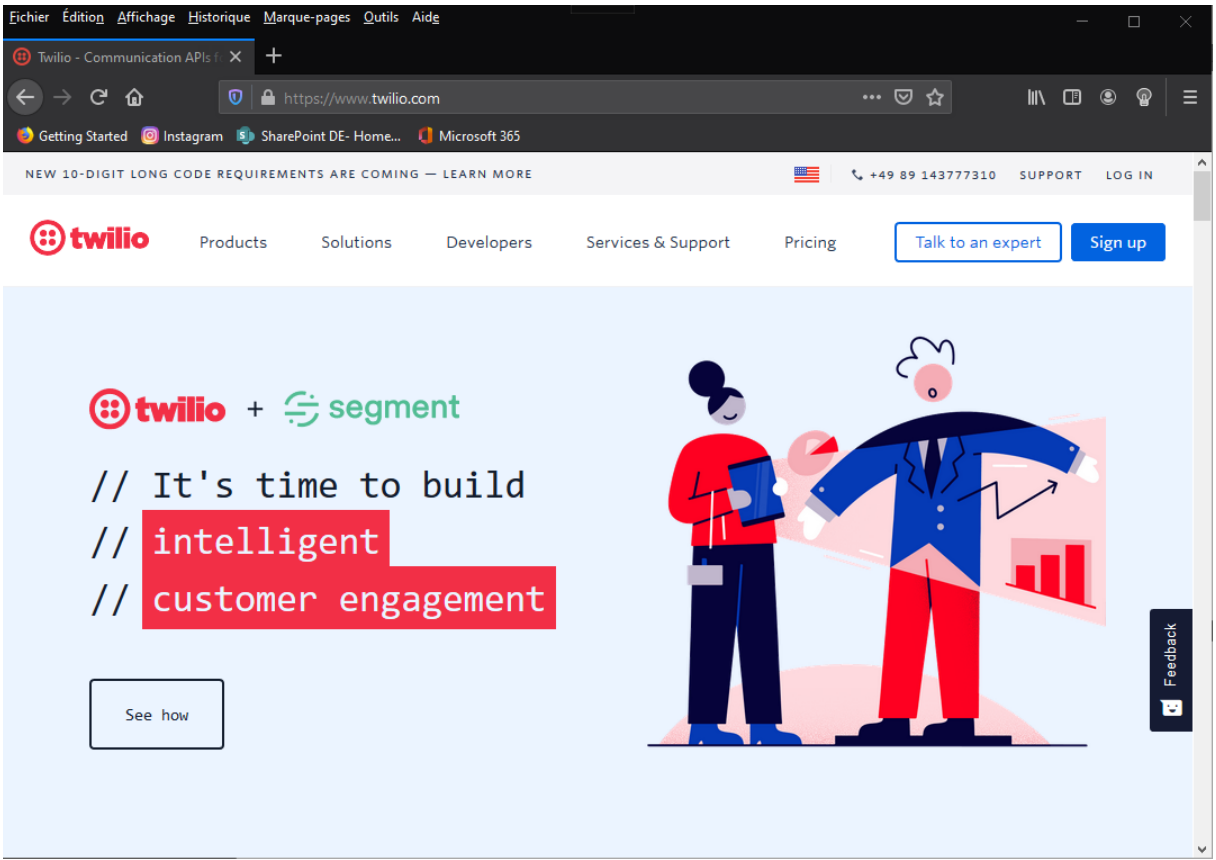
Pour envoyer des beaux e-mails au format HTML, il existe un package nommé **blastula** que je n'ai pas eu le temps de tester complétement.

Envoyer un SMS via R

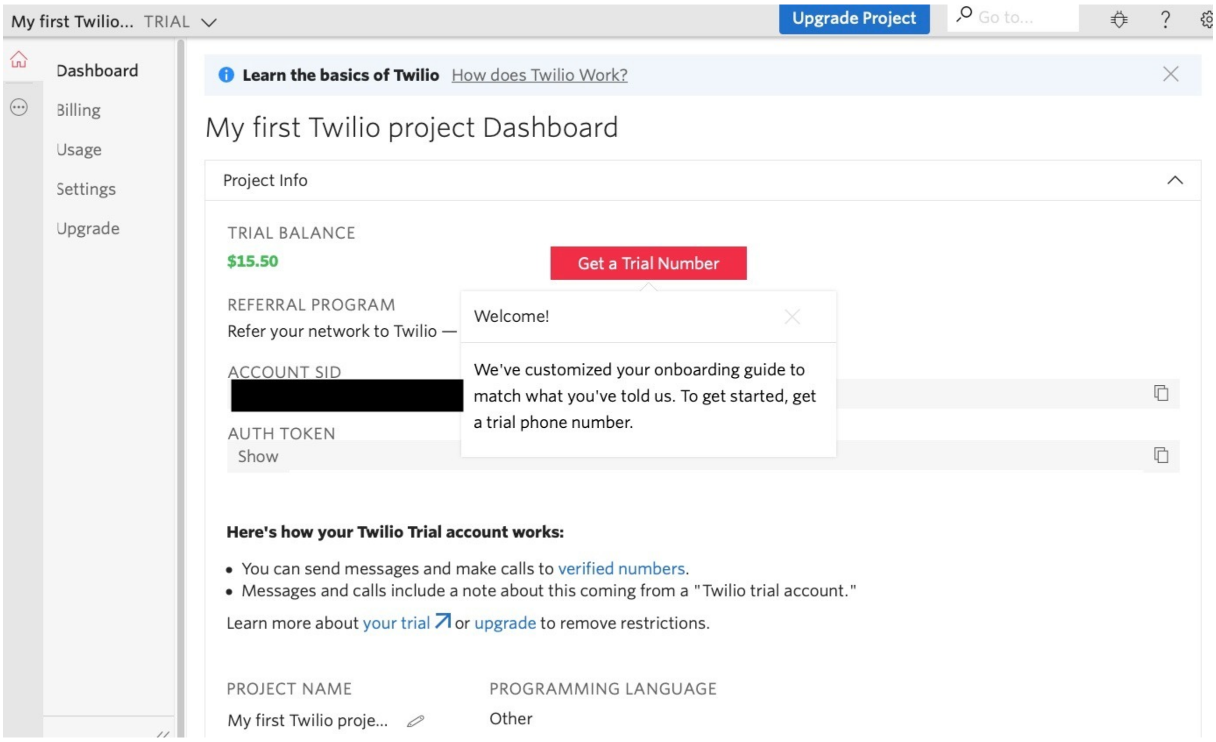
Très utile quand on ne veut pas attendre devant sont ordinateur le résultat d'un calcul pendant des heures… (et évidemment d'autres applications sont possibles!).

D'abord créez un compte sur Twilio:

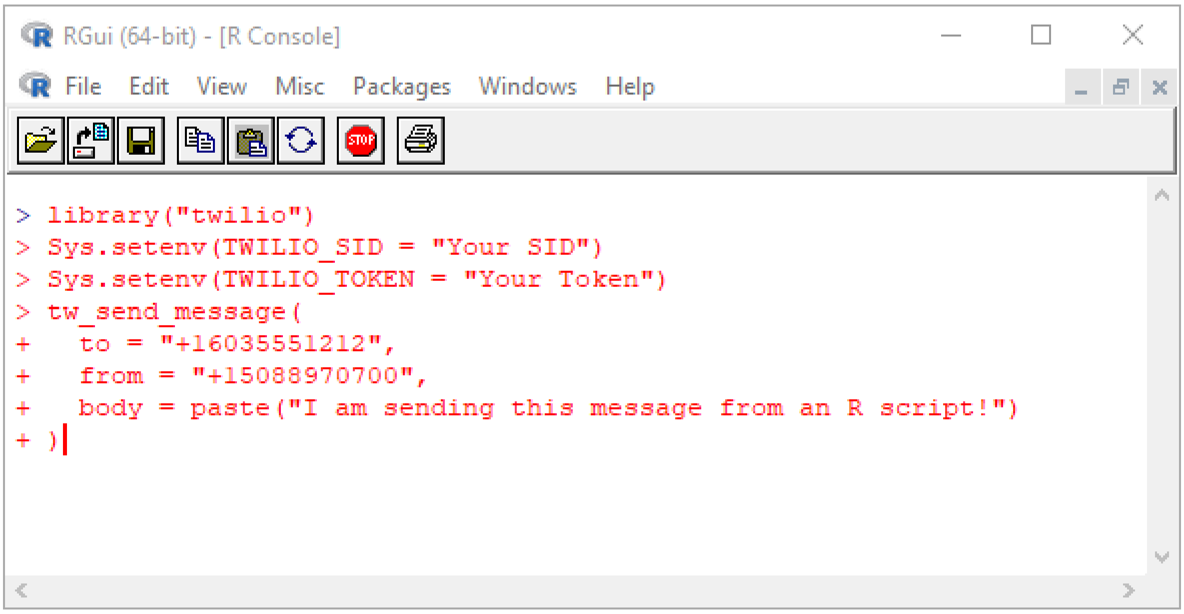
https://www.twilio.com/



Une fois le compte crée vous pourrez accéder à votre SID et TOKEN:



Ensuite pour envoyer le SMS on utilisera:



5. Fichiers R

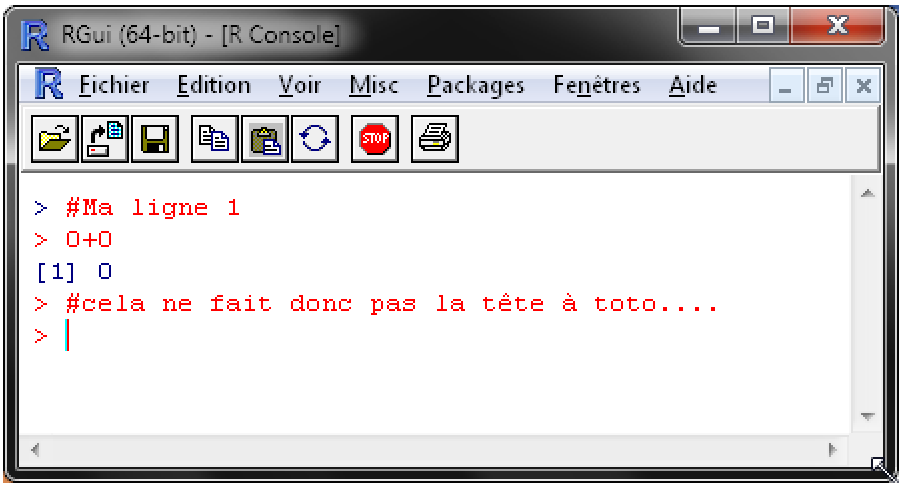
Il existe trois types de fichier dans R:

- Les fichiers: *.Rhistory
- Les fichiers: *.RData
- Les fichiers: *.R pour les scripts (voir page 2894)
- Le fichier de config: *.Rprofile (voir page 83 ou 105)

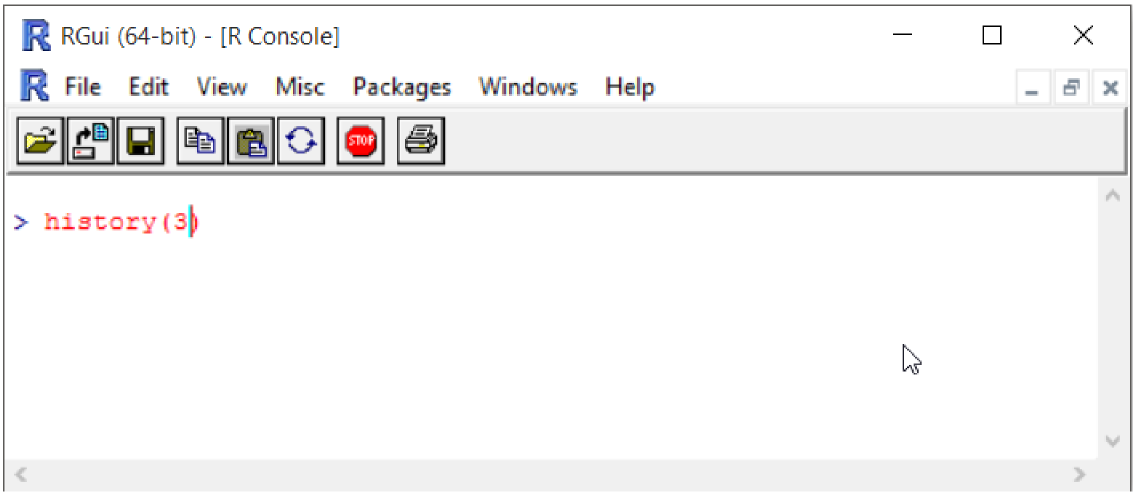
Historique et fichiers *.Rhistory

Nous allons commencer par découvrir le type le plus trivial de fichier qui est le *.Rhistory.

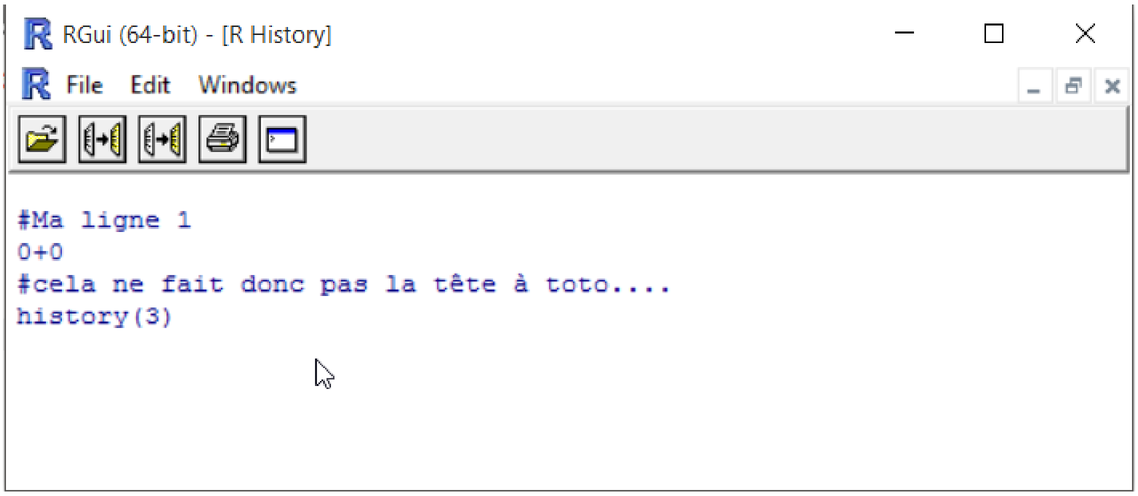
Il s'agit simplement d'un type de fichier qui va sauvegarder l'historique de vos commandes. Pour voir cela, nous nous ouvrons R et écrivons:



D'abord faisons un **Ctrl+L** pour effacer la console et utilisons la fonction **history( )**:

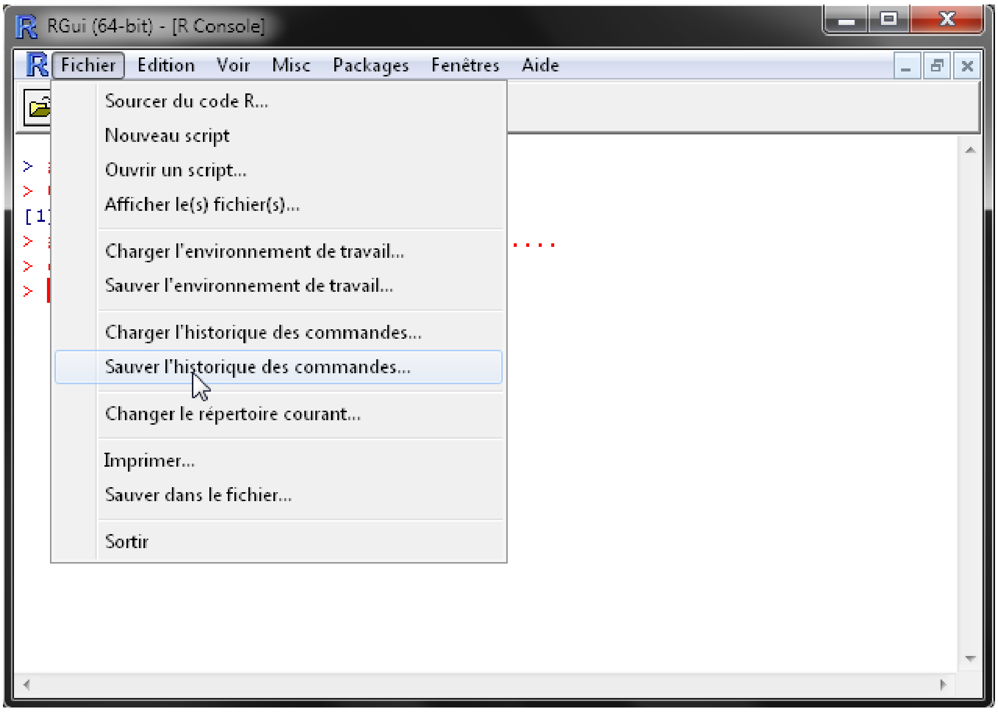


Cela donne:

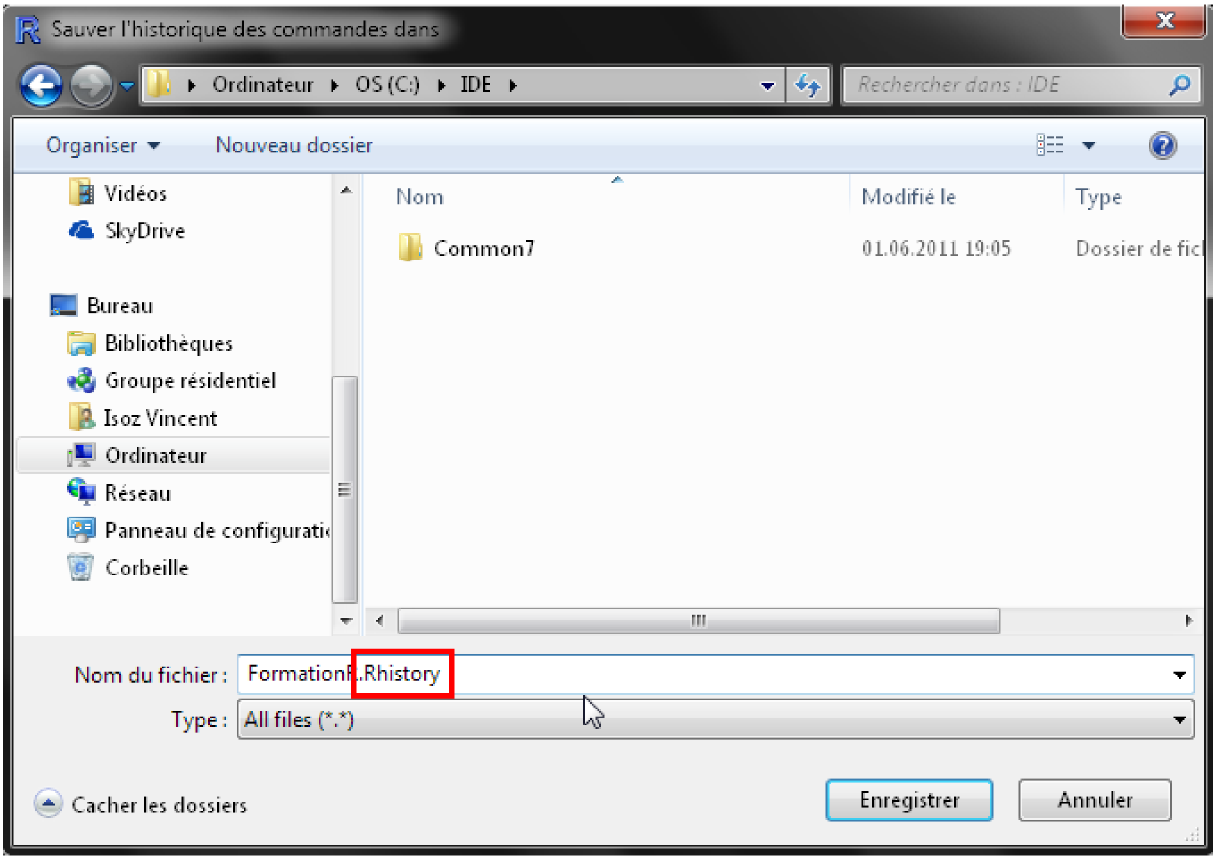


et ouvre donc le fichier R History uniquement pour le nombre de commandes demandées!

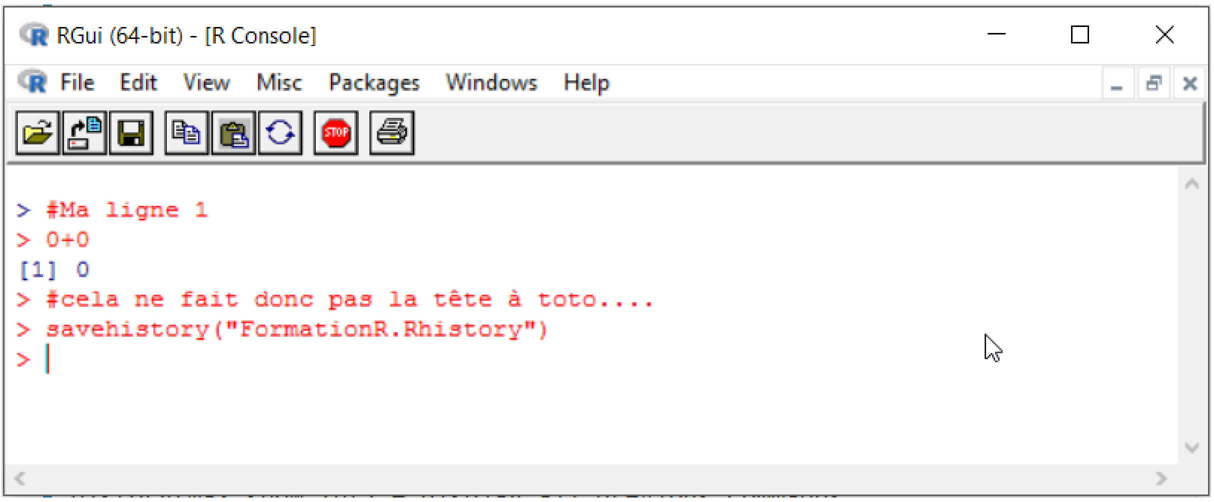
Maintenant, nous allons dans le menu **Fichier/Sauver l'historique des commandes**:



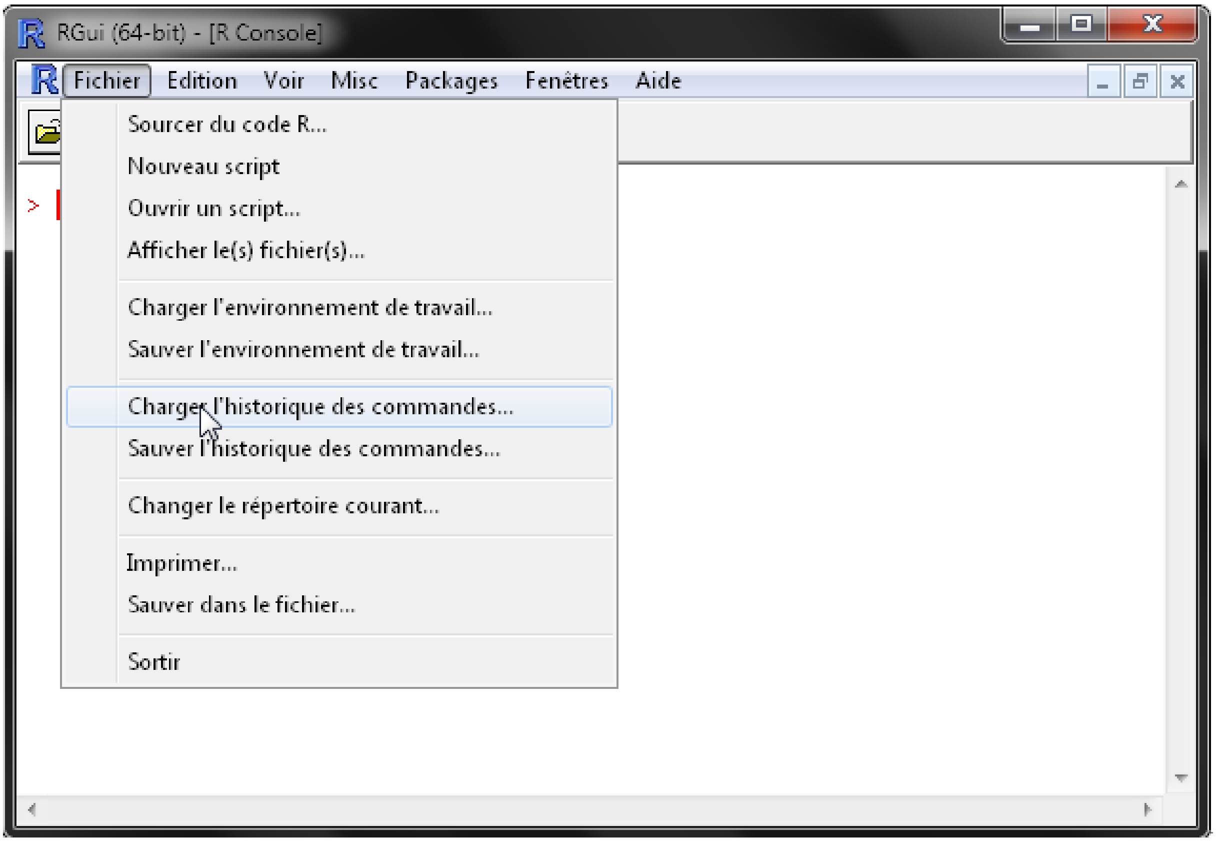
Et il vient:



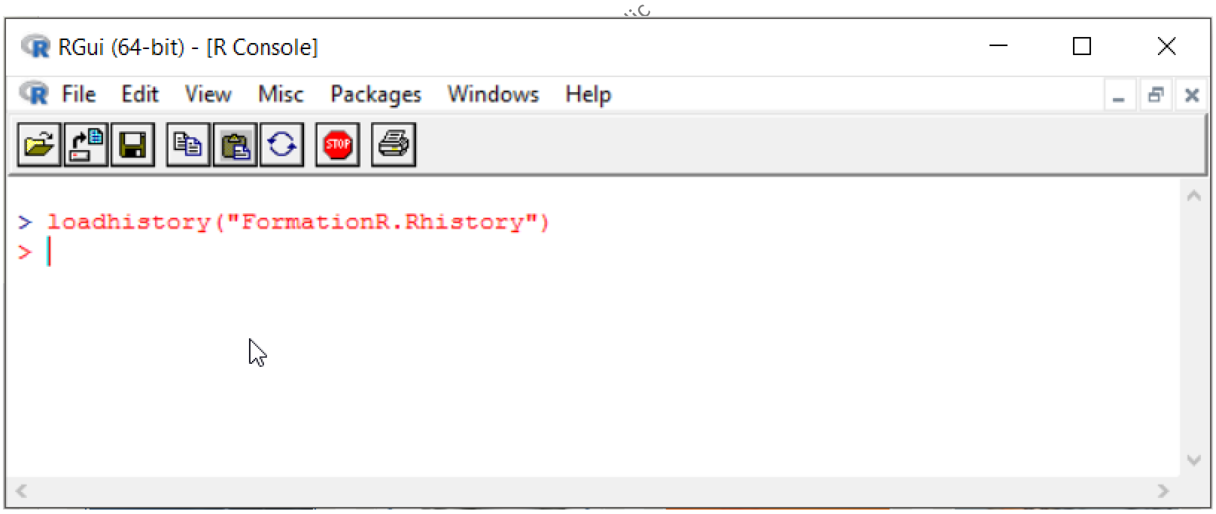
où nous avons déjà mis le nom désiré devant l'extension du fichier. Nous pouvons utiliser aussi la commande **savehistory( ):**



Si nous fermons R et le rouvrons, nous pouvons recharger les commandes passées en allant dans **Fichier/Charger l'historique des commandes**:



ou avec la fonction **loadhistory( )**:



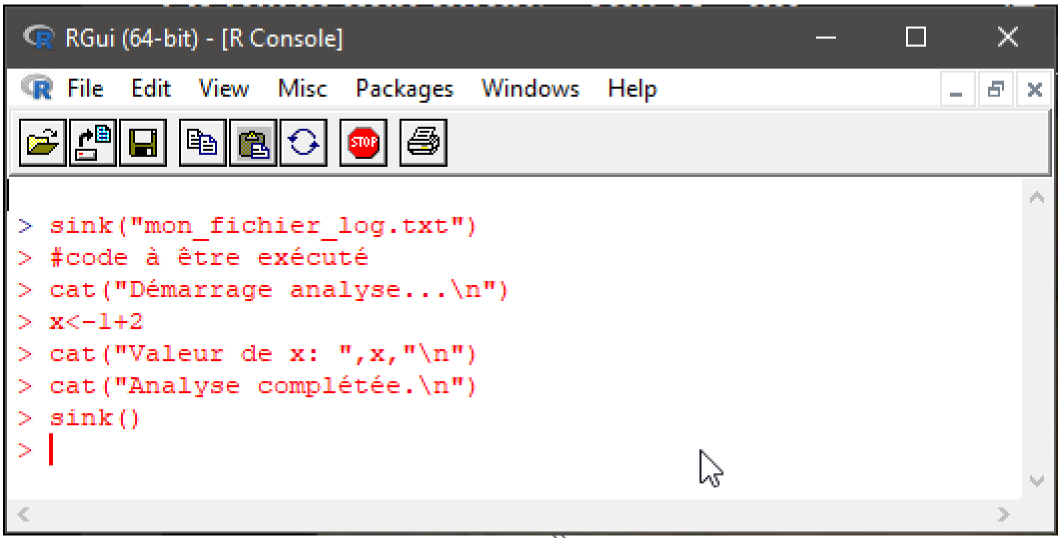
Mais rien n'apparaîtra quand vous chargerez votre fichier *.Rhistory. Effectivement pour retrouver vos commandes, il vous faudra jouer avec les flèches du clavier ↑ ou ↓ pour remonter ou descendre dans l'historique des commandes.

On comprend alors mieux l'intérêt de sauvegarder l'historique des commandes dans un fichier script *.R (voir page 2682 pour rappel).

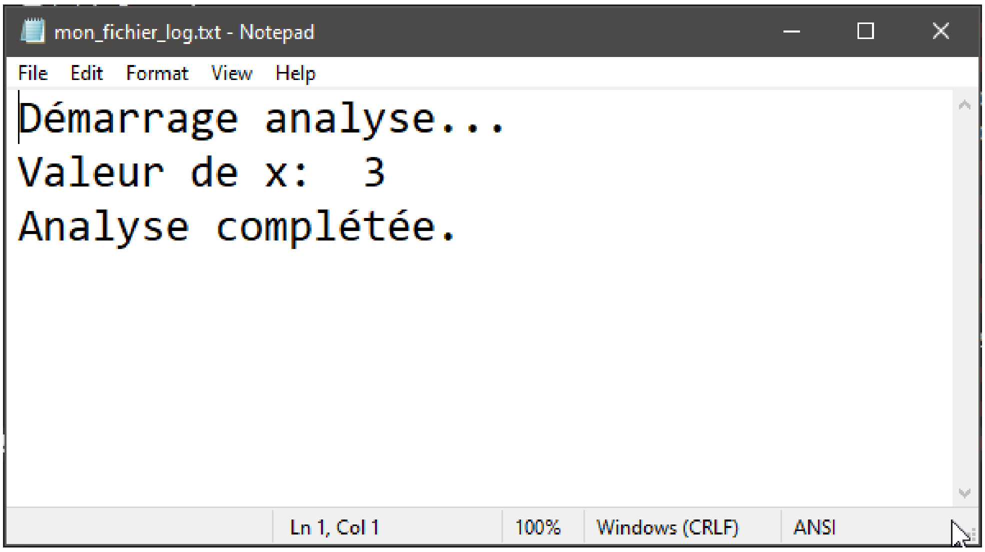
Journalisation

Alors que les fichiers de journalisation SAS sont considérés comme essentiels dans les essais cliniques, R fournit une approche alternative pour générer des journaux via la fonction de sink( ). La fonction sink( ) dans R permet à l'utilisateur de rediriger la sortie de la console vers un fichier, créant ainsi un fichier journal. La fonction de puits est personnalisable et peut capturer une large gamme de sorties, y compris des avertissements et des erreurs. De plus, le fichier journal peut être examiné et analysé après les faits.

Voici un exemple de création d'un fichier journal simple à l'aide de sink( ) :



Ce qui donnera:



La fonction de sink( ) de R n'est peut-être certes pas être aussi complète que les fichiers journaux de SAS. Cela vaut également la peine de préciser que R a des packages tels que **logging** et **log4r** qui fournissent des capacités de journalisation plus avancées.

Fichiers *.Rdata

Les fichiers *.Rdata (fichiers binaires très rapides à charger et à enregistrer) vont stocker toutes les variables, dates frames, listes, matrices, vecteurs, etc. que vous avez créés ou modifiés pendant une session de travail R.

Cela peut paraître a priori inutile mais en réalité c'est très utile lorsque l'on fait du calcul massif (big data).

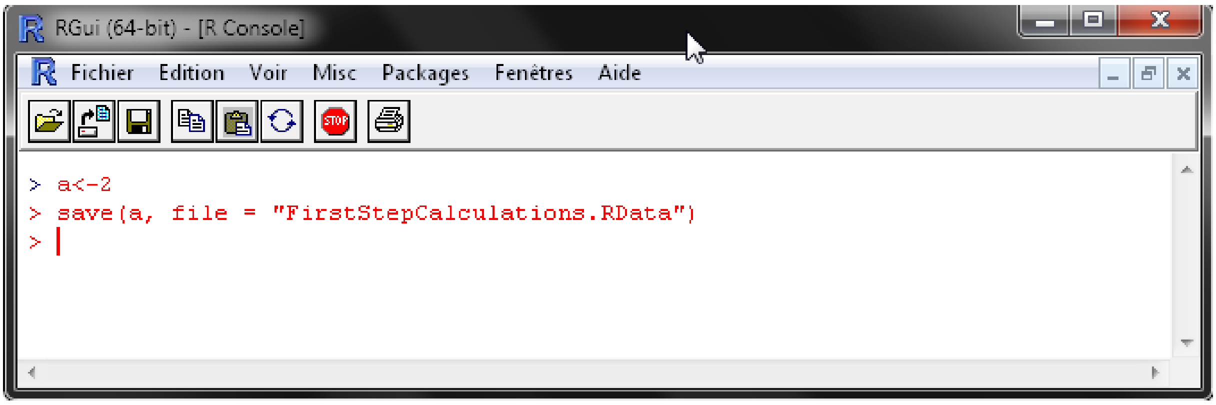
Effectivement, si vous êtes sur une machine dont la mémoire vive est limitée pour des raisons de budgets il est alors possible de jouer de façon subtile avec ces fichiers pour libérer temporairement de la mémoire.

Voyons une démonstration simple de cela.

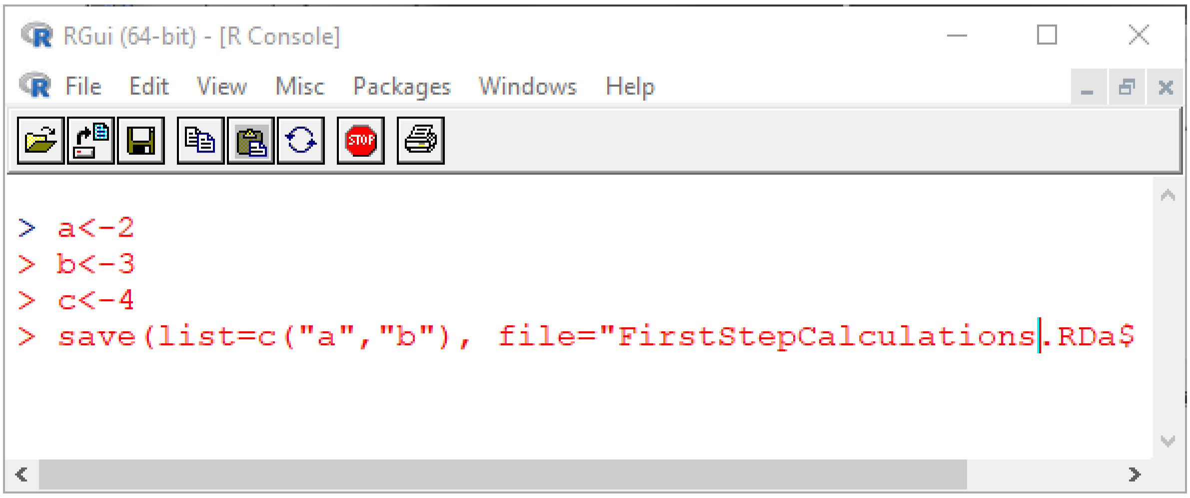
Nous créons une simple variable à laquelle nous affectons une valeur scalaire:



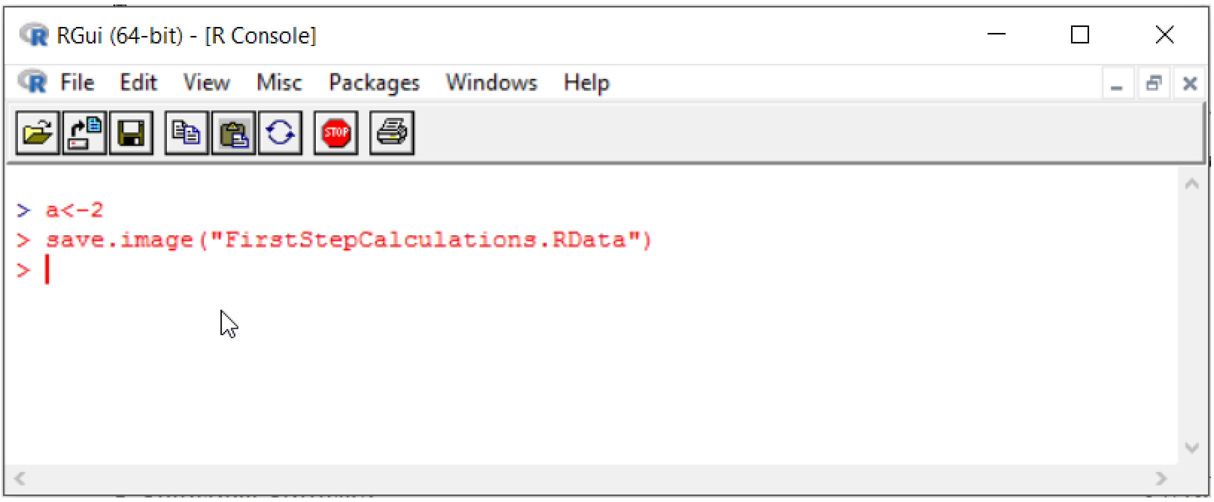
Nous pouvons créer un fichier *.RData en quittant R mais ici le but est d'avoir un contrôle total sous forme de lignes de commandes (pour intégrer cela plus tard dans des scripts). Pour enregistrer **une variable de la mémoire** de l'espace de travail en cours de R nous utilisons la fonction **save( )** (si nous ne spécifions pas le dossier donc il prendra l'espace de travail par défaut actif!):



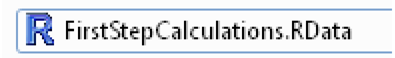
Ou pour deux variables parmi plusieurs:



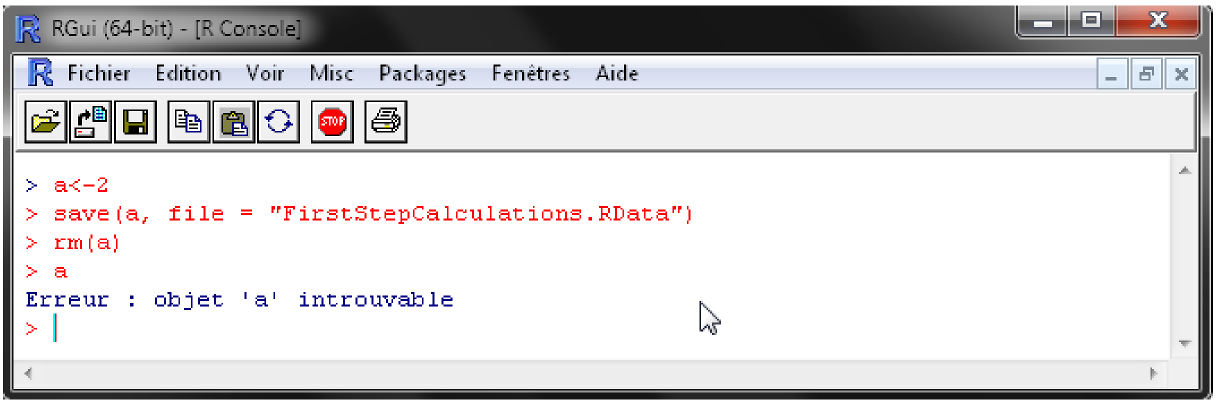
ou pour toutes les variables de l'environnement active, on utilisera **save.image( )**:



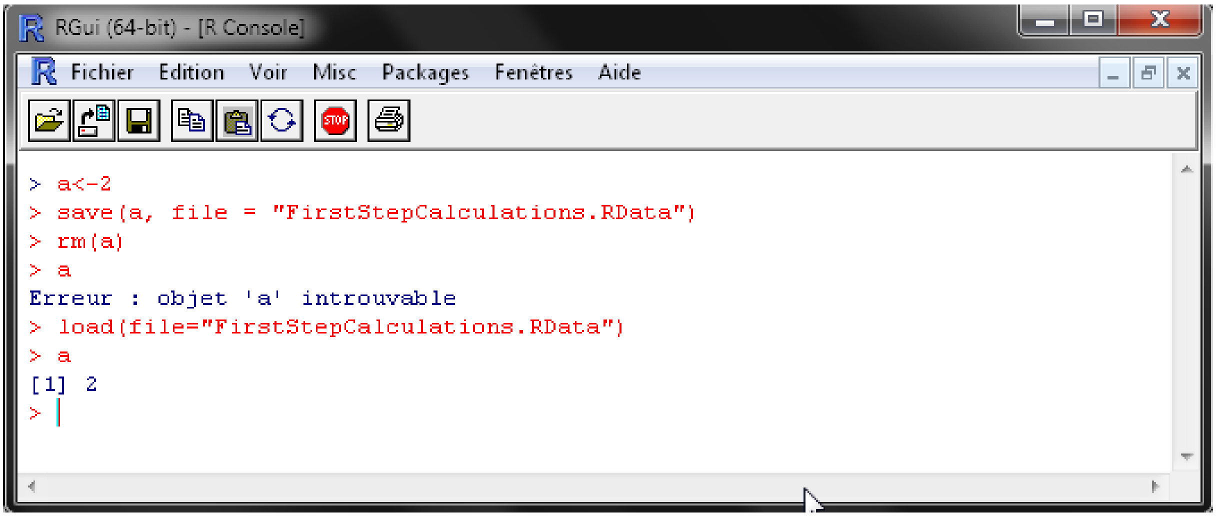
Ce qui donnera sur votre disque un petit fichier *.RData:



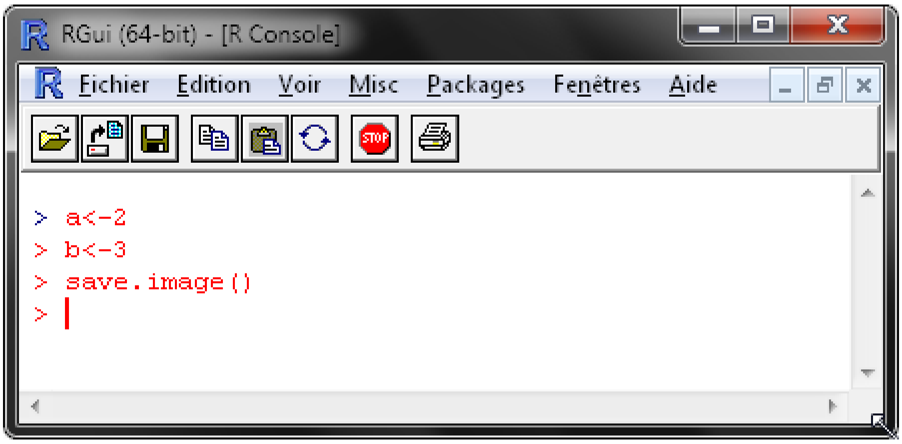
Maintenant nous effaçons cette variable de la session en cours et vérifions qu'elle n'existe plus:



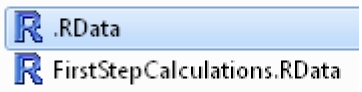
Mais nous pouvons la rappeler avec la commande **load( )**:



Si vous utilisez la syntaxe suivante, il va enregistrer dans le fichier *.RData par défaut de votre installation de R:



Effectivement:

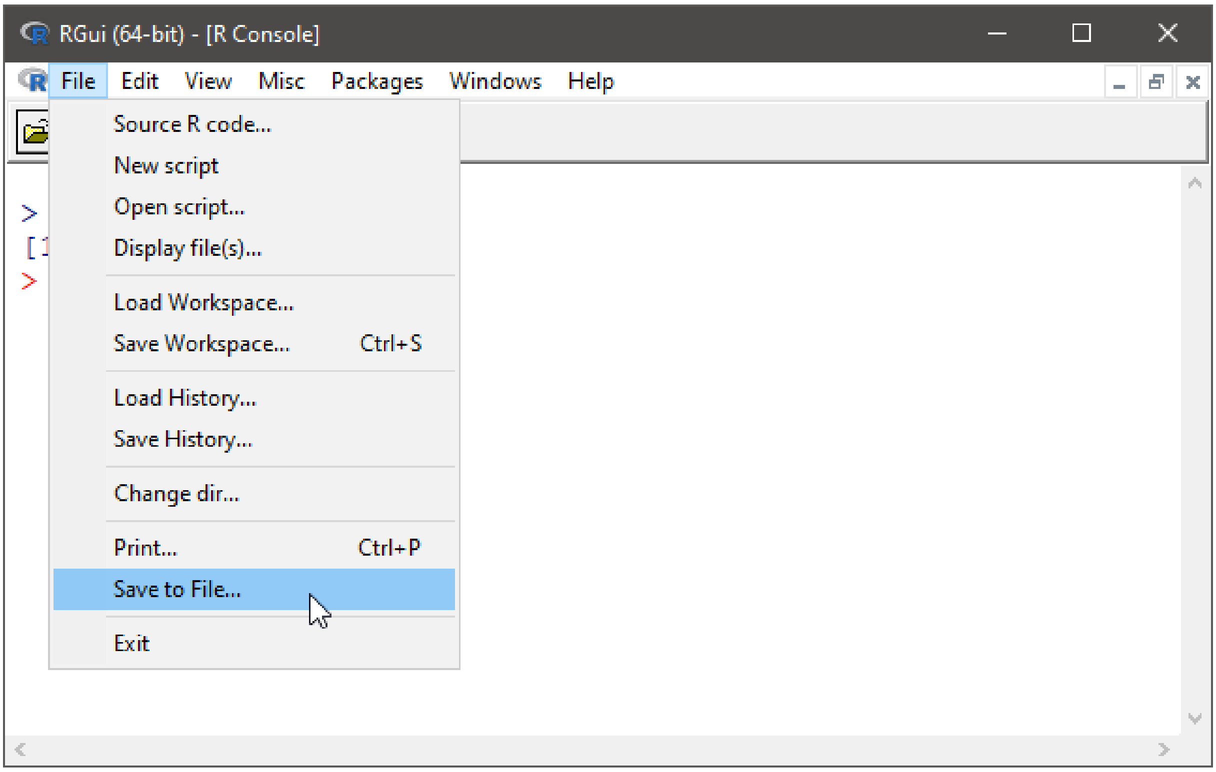


Nous voyons alors vite la possibilité de créer autant de fichiers *.RData que désirés et de les décharger ou recharger à loisir quand cela est nécessaires (dans les grosses applications on aura typiquement un fichier *.RData pour chaque gros data frame de manipulé).

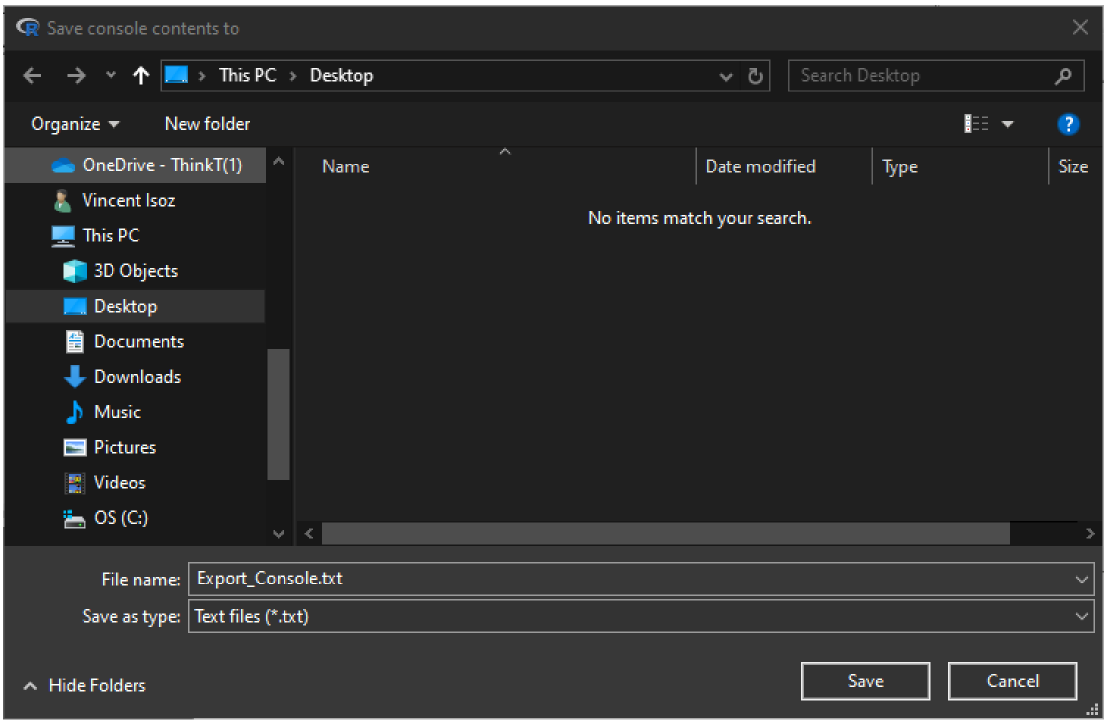
Petit truc à savoir. Si vous sauvegardez tout un espace de travail et en rechargez tout le contenu de suite après, cela peut faire gagner parfois un peu plus de 50% de la mémoire utilisée.

Fichiers *.txt

Il est possible dans R d'exporter les lignes de commande et leurs résultats dans un fichier texte manuellement simplement soit par copier/coller ou par un menu. Pour ceci on ira dans le menu **Fichier/Enregistrer le fichier sous…**:



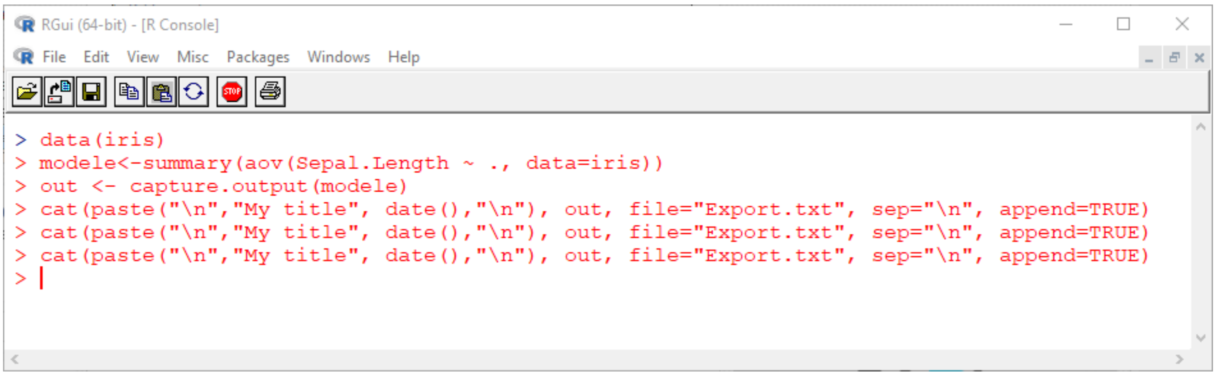
Ce qui ouvrira la boîte de dialogue habituelle:



Le résultat donnera dans le cas présent:



Dans une certaine mesure, on peut exporter automatiquement avec un script. Illustrons cela avec l'exemple suivant, faisant usage de la fonction **capture.output( )**, qui a une importance particulière pour ceux travaillent avec Microsoft Power BI et R comme nous le verrons plus loin, ainsi que la fonction **paste( )** pour concaténer des chaînes caractères et la fonction **cat( )** qui concatène (au maximum deux chaînes de caractères!) et qui peut exporter en même temps:



6. Calculs arithmétiques élémentaires

Contenu

Exemple 1.: Opérations sur scalaires

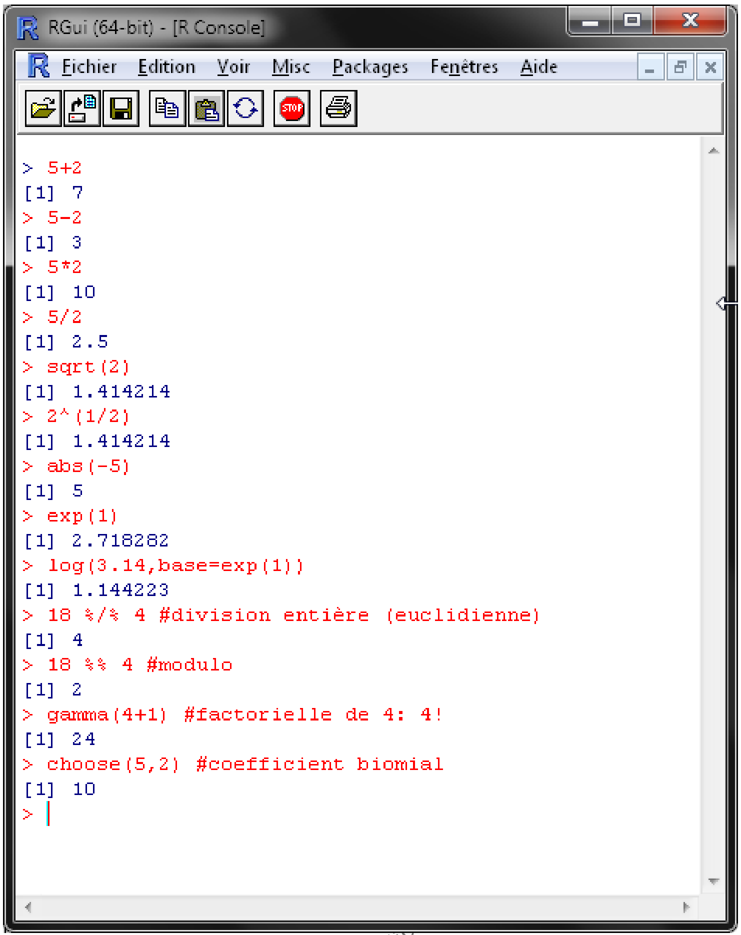
R 3.0.2

Le but final de R est sur le plus ou moins long terme, lorsque l'on est utilisateur, d'écrire des scripts qui vont automatiser un travail répétitif de manipulation de données, de calculs et tests/analyses statistiques (comme le VBA dans la suite Microsoft Office par exemple).

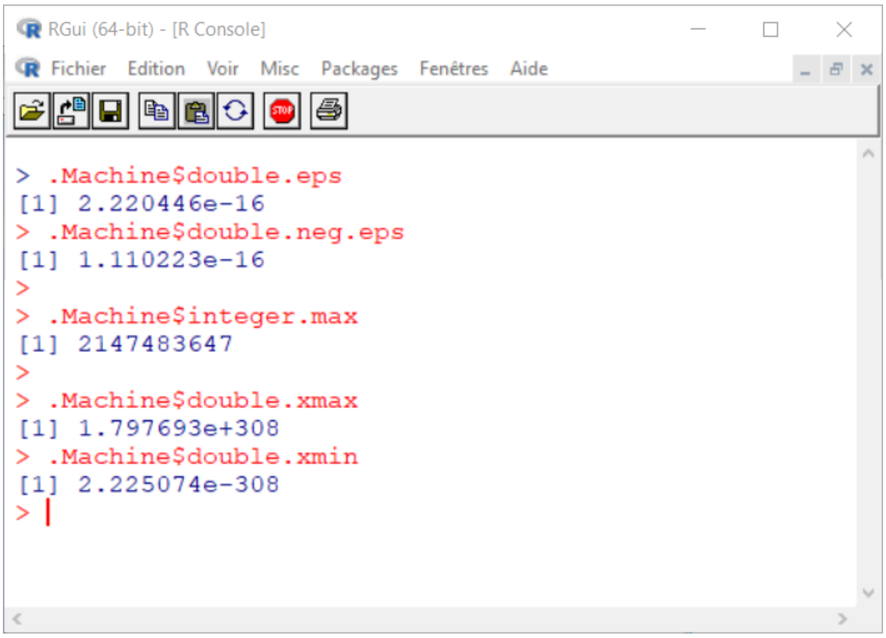
Dès lors, au même titre qu'un utilisateur Excel aura des difficultés à faire du code s'il ne connaît les structures et opérateur algébrique de base, il en sera aussi souvent de même pour l'utilisateur R.

Donc nous allons commencer par les mathématiques élémentaires ceci dans le but de préparer bien plus tard un terrain propice au scripting de haut vol.

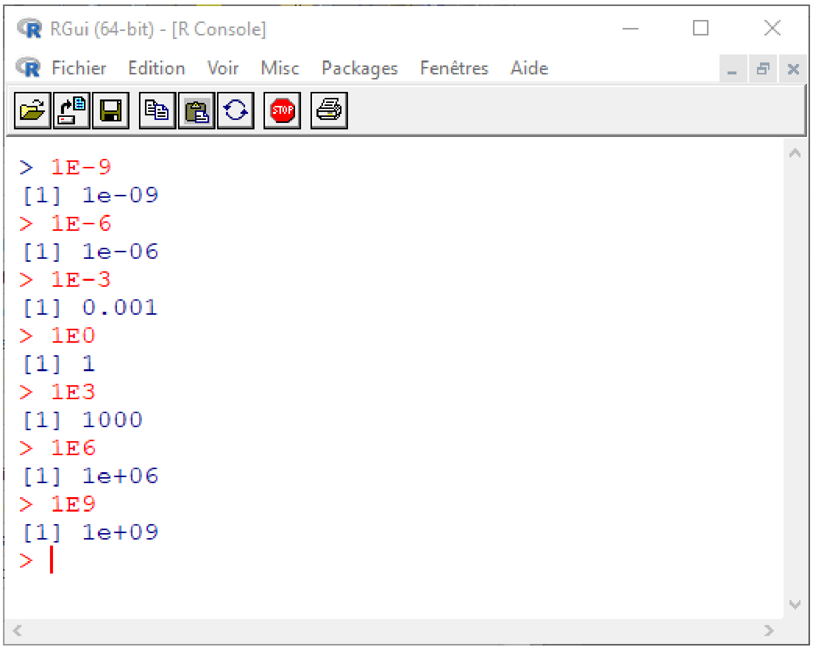
D'abord commençons par quelques opérations algébriques élémentaires comme l'addition, la soustraction, la division, la multiplication, le logarithme naturel avec la commande **log( )**, la racine carrée avec la commande **sqrt( )**, la valeur absolue avec la commande **abs( )** ou encore l'exponentielle avec la commande **exp( )**, la factorielle avec **gamma( )** et le coefficient binomial avec la commande **choose( )** ou encore le modulo avec **%%**:



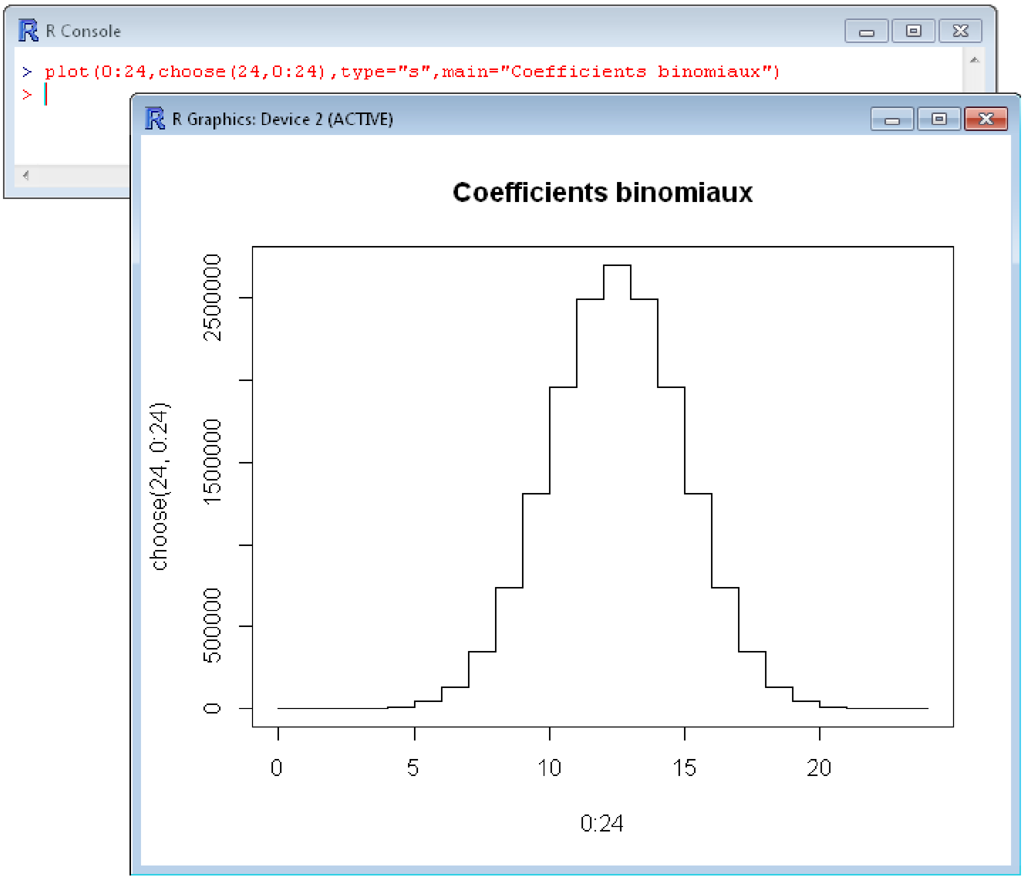
Pour ceux qui veulent savoir les valeurs les réelles ou entières les plus petites ou plus grandes gérées par R on peut utiliser la fonction d'environnement **.Machine**:



Sinon, pour créer des nombres avec la notation d'ingénierie (on peut aussi les créer en utilisant un "e" minuscule au lieu du "E" majuscule bien évidemment!):



Avec le coefficient binomial nous pouvons reprendre l'exemple du cours d'Arithmétique en se demandant combien il y a de manières de créer un mot de passe de *n* caractères (*n* allant de 0 à 24) sachant qu'on peut utiliser seulement un alphabet de 24 lettres (sans remise). Cela nous donne:

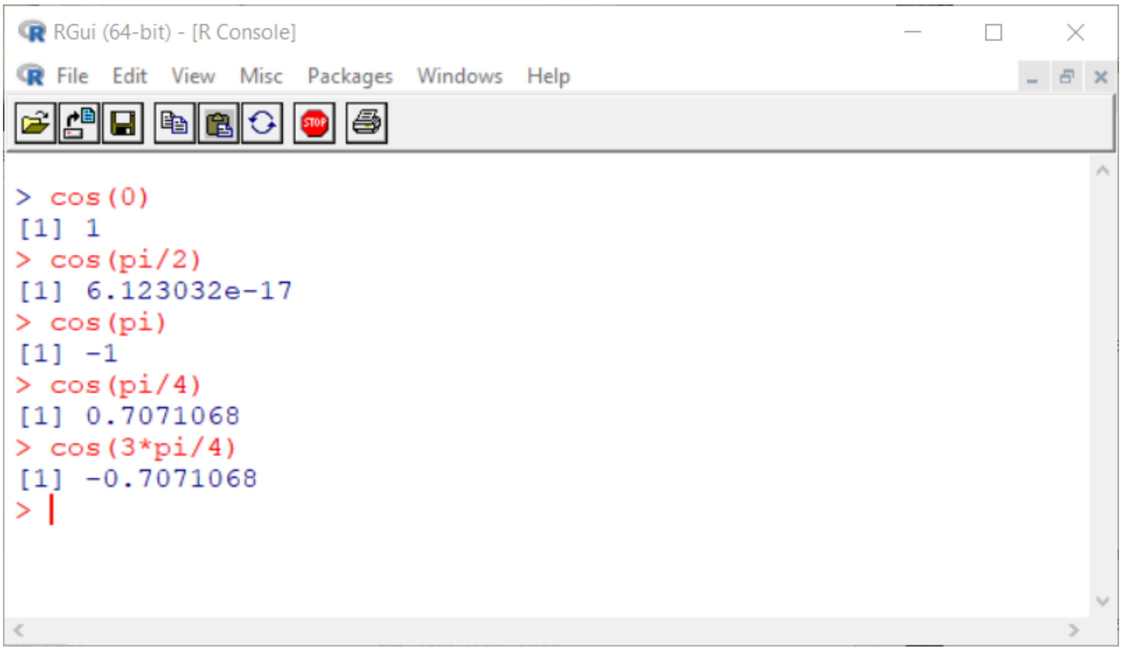


Exemple 2.: Fonctions trigonométriques

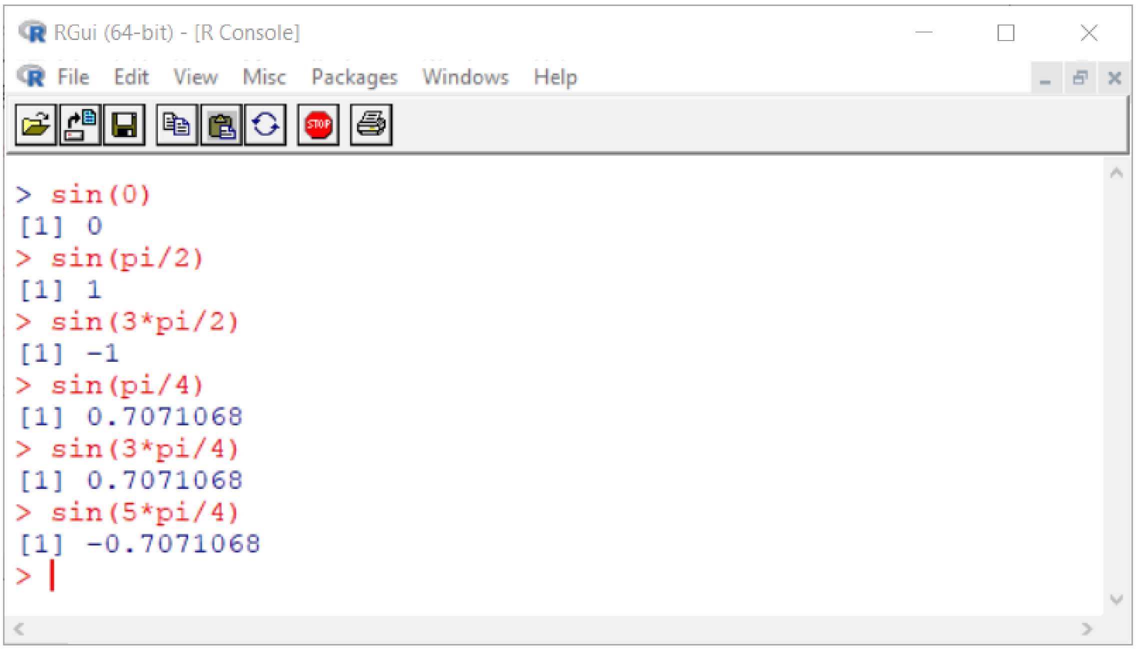
R 3.4.2

Nous savons quelle est l'importance des fonctions trigonométriques dans tous les domaines des maths, de l'ingénierie, de la finance et de la Data Science. Voyons alors comme utiliser ces dernières.

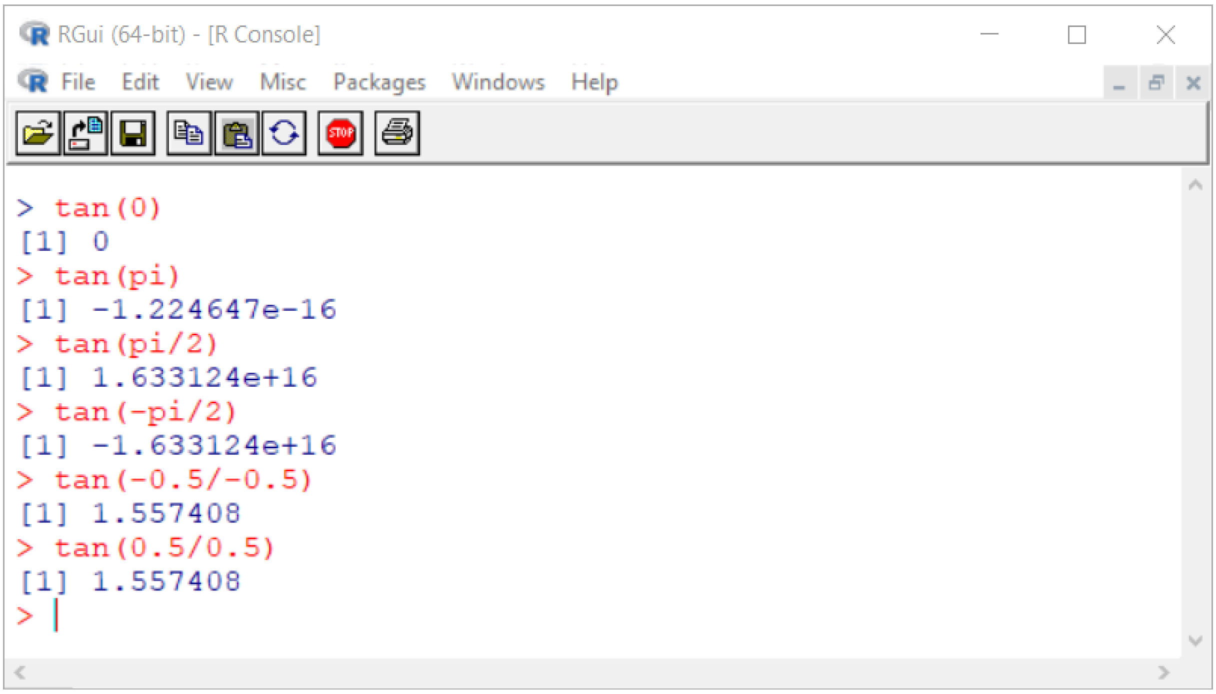
Commençons avec la fonction **cos( )** et quelques valeurs remarquables:



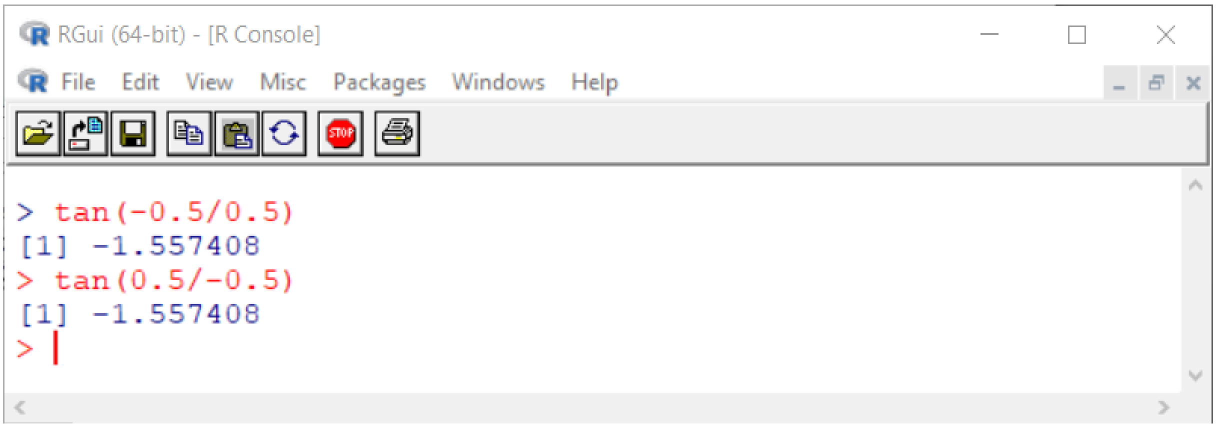
Et pour la fonction **sin( )**:



Et pour la fonction **tan( )**:

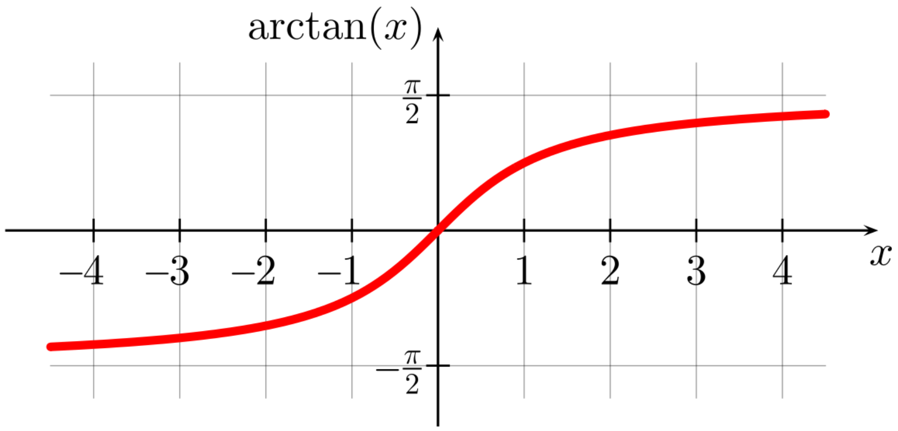


Maintenant gardez bien en tête cet exemple:

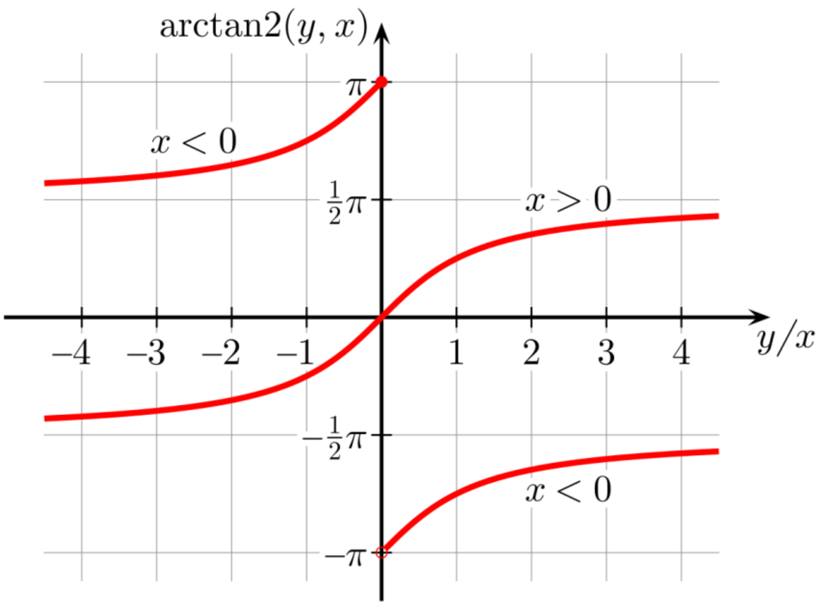


Et maintenant introduisons la fonction inverse **atan( )** et **atan2( )**[3] dont les graphs correspondants sont:

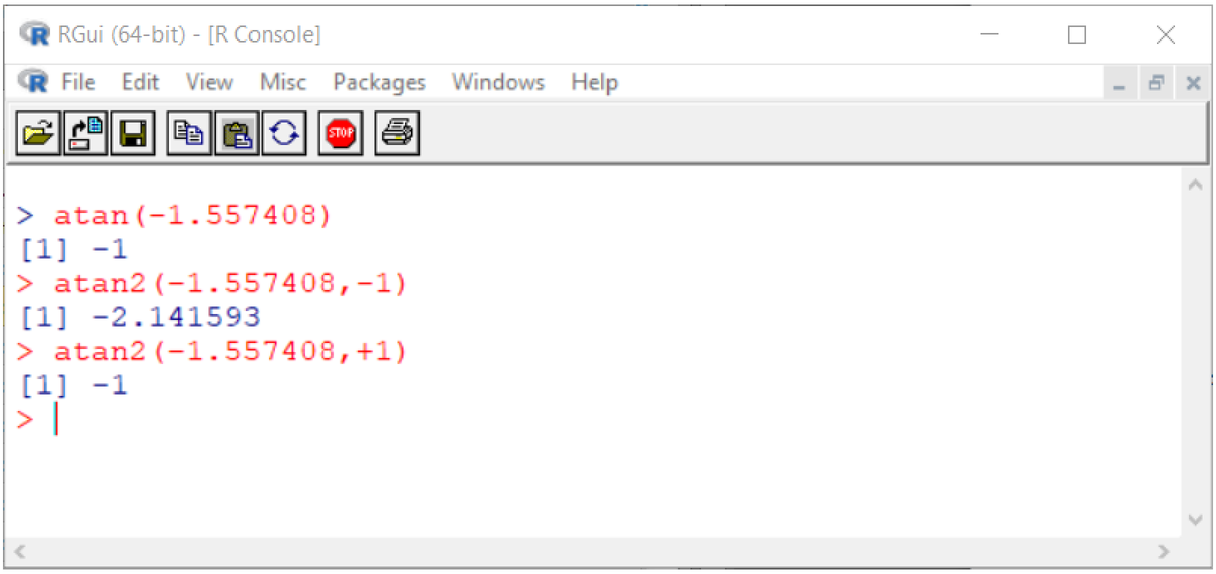
[3] Puisque nous savons que pour cos( ) et sin( ) il n'y a pas le problème de dénominateur qui peut être positif ou négatif puisque l'hypoténuse est toujours positive.



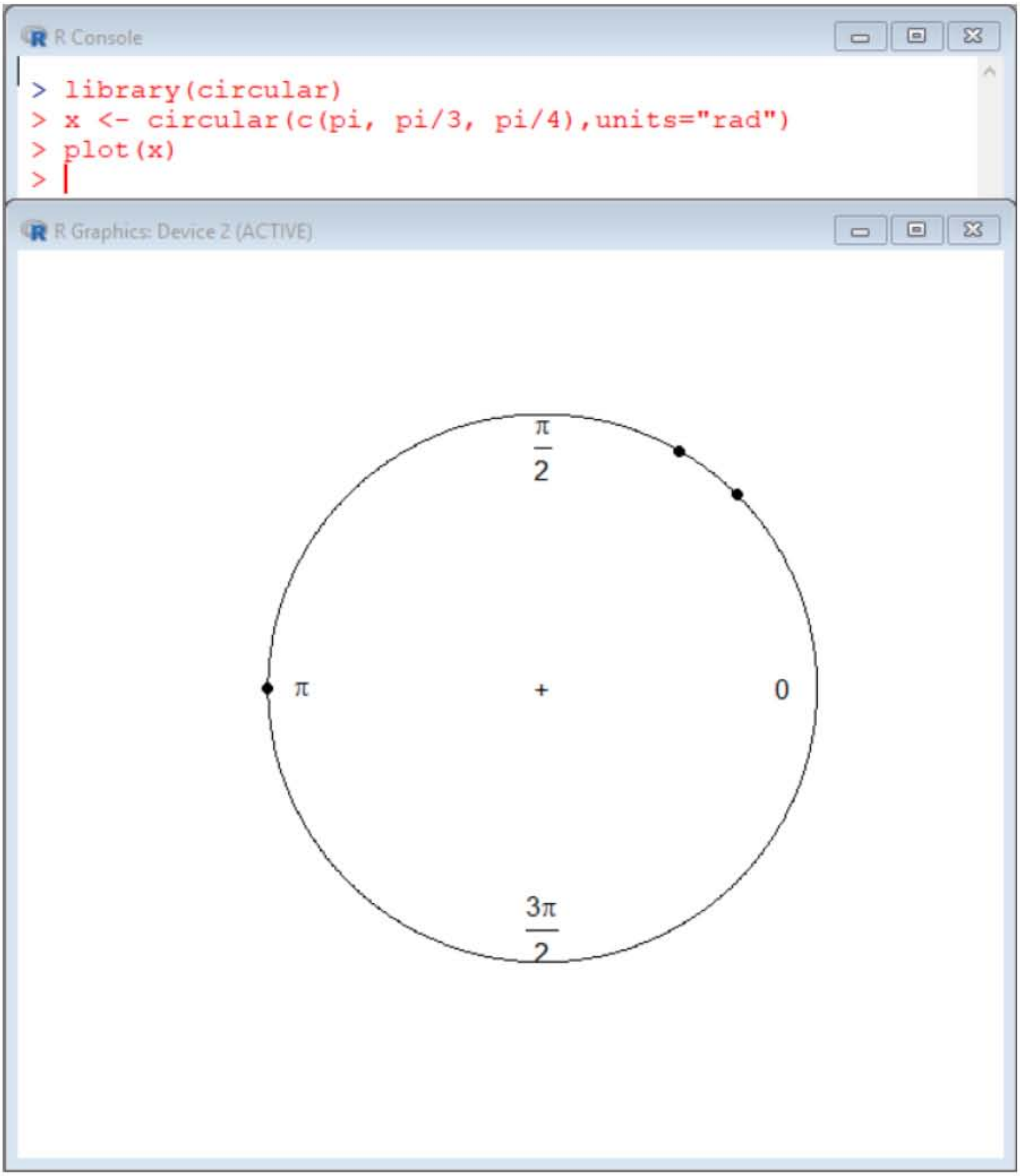
et pour arctan2:



Donc nous avons:



Enfin, une parenthèse pour les profs qui souhaiteraient plotter le cercle trigonométrique avec certains angles spécifiques visibles, il est possible d'utiliser le package **circular**:

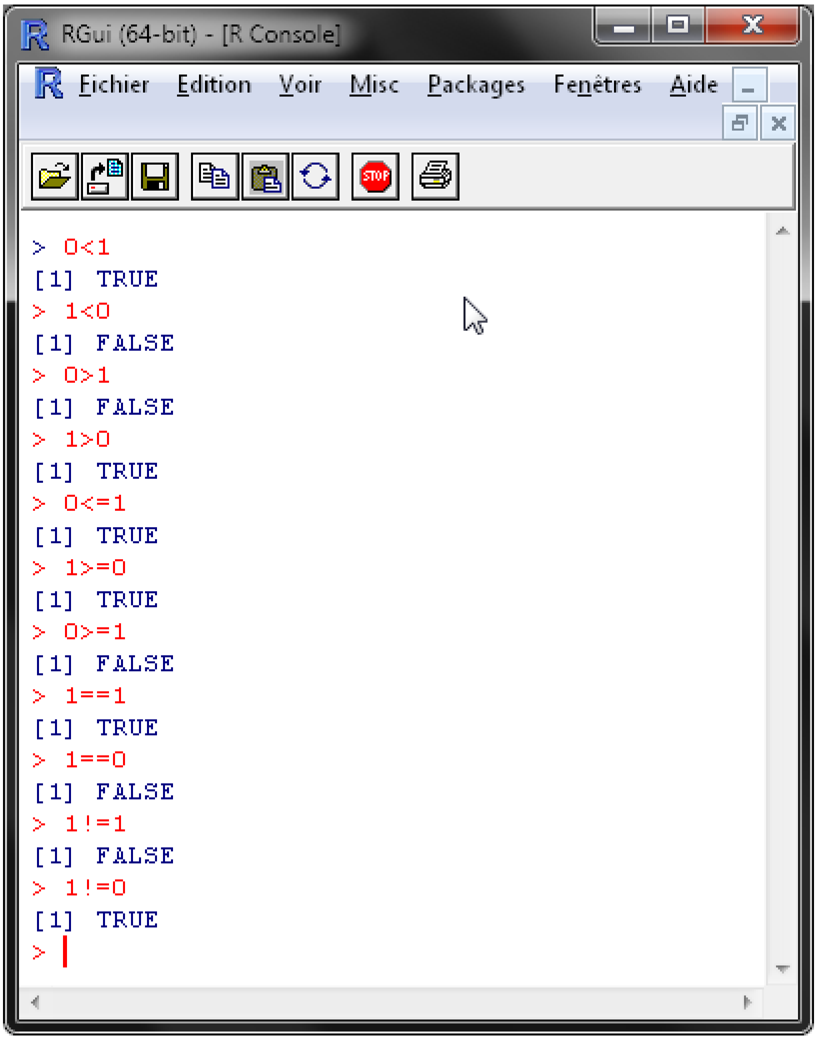


Exemple 3.: Opérateurs booléens et relations d'ordre

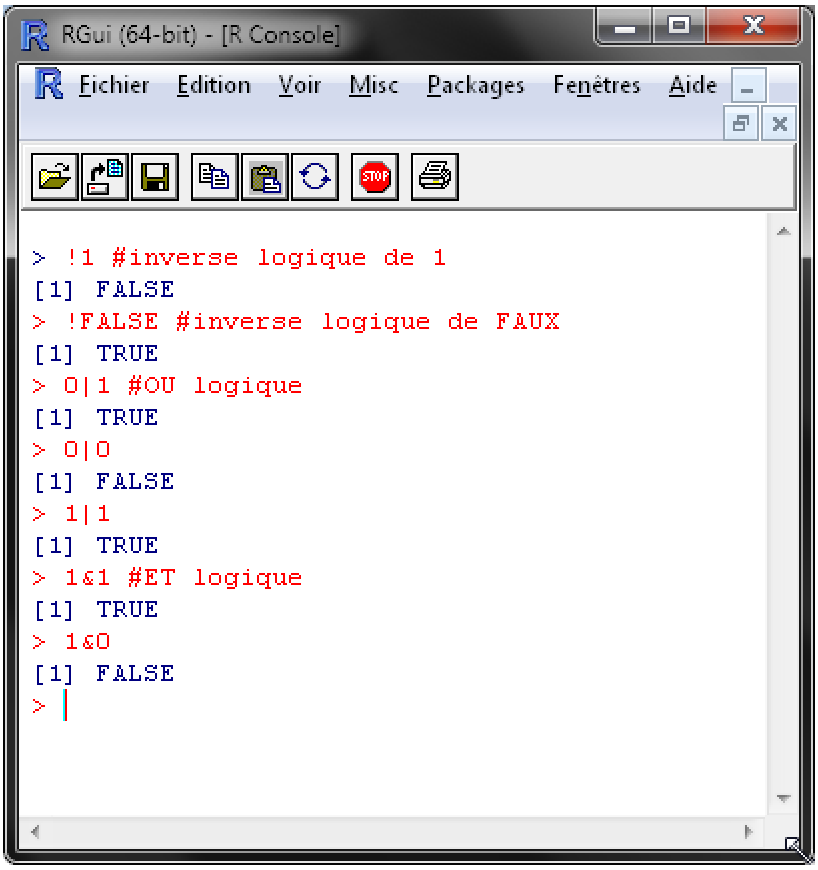
R 3.0.2

Lorsque vous allez faire du scripting, vous aurez parfois besoin de faire des tests logiques utilisant les valeurs booléennes. Il est donc important de revoir les cas les plus courants.

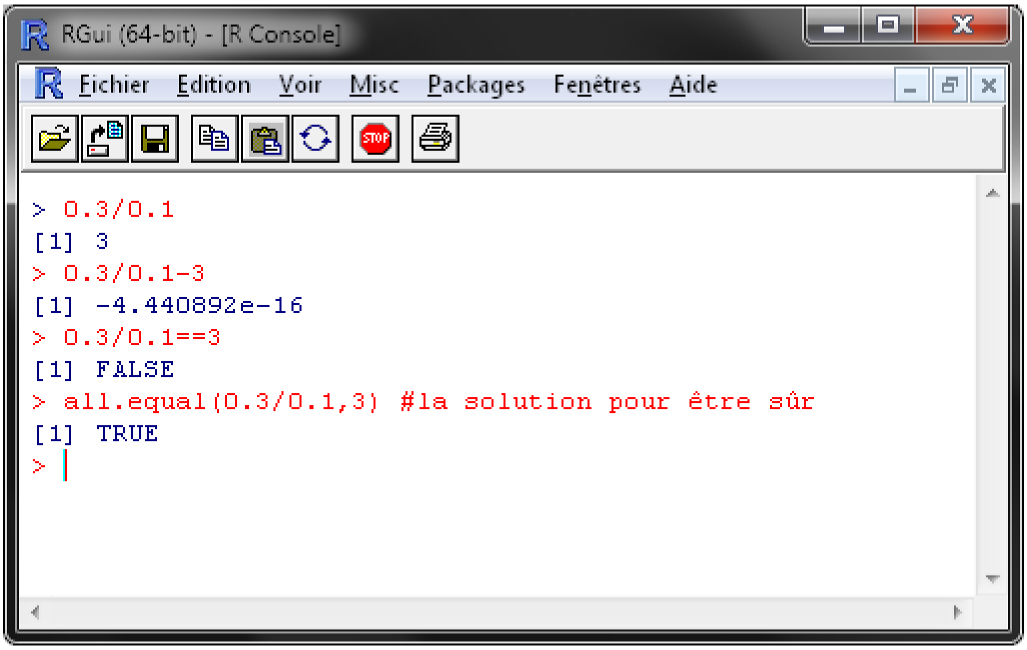
Nous avons d'abord la syntaxe suivante pour les relations d'ordre les plus courants:



et un peu d'algèbre booléenne élémentaire:

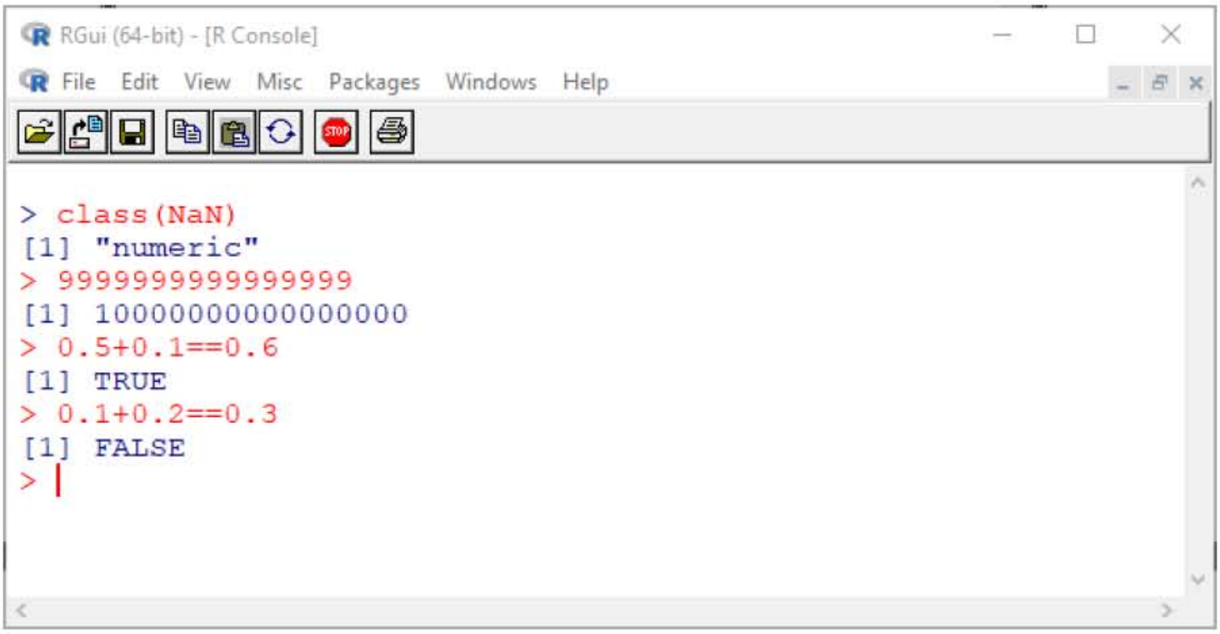


Signalons également le piège suivant à cause de la manière dont fonctionnent les ordinateurs avec le binaire lorsque nous comparons des calculs avec un résultat étalon:

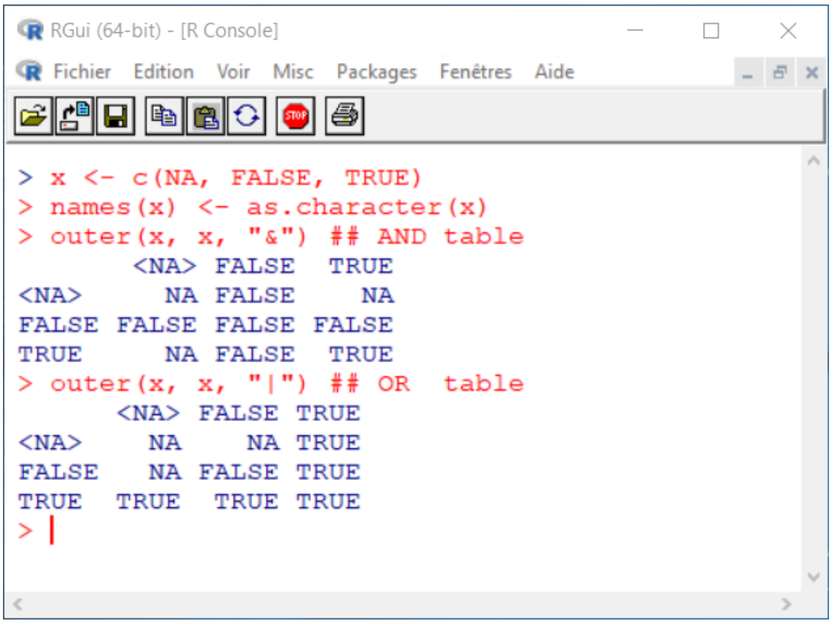


Donc le problème se résout à l'aide de la fonction **all.equal( )**.

Sinon une liste de pseudo-curiosités (bon on a encore de la marge par rapport au Javascript…):


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> class(NaN)
[1] "numeric"
> 9999999999999999
[1] 10000000000000000
> 0.5+0.1==0.6
[1] TRUE
> 0.1+0.2==0.3
[1] FALSE
>

Et pour les profs qui souhaitent construire une table logique dans R avec leurs élèves en utilisant la fonction **outer( )** :

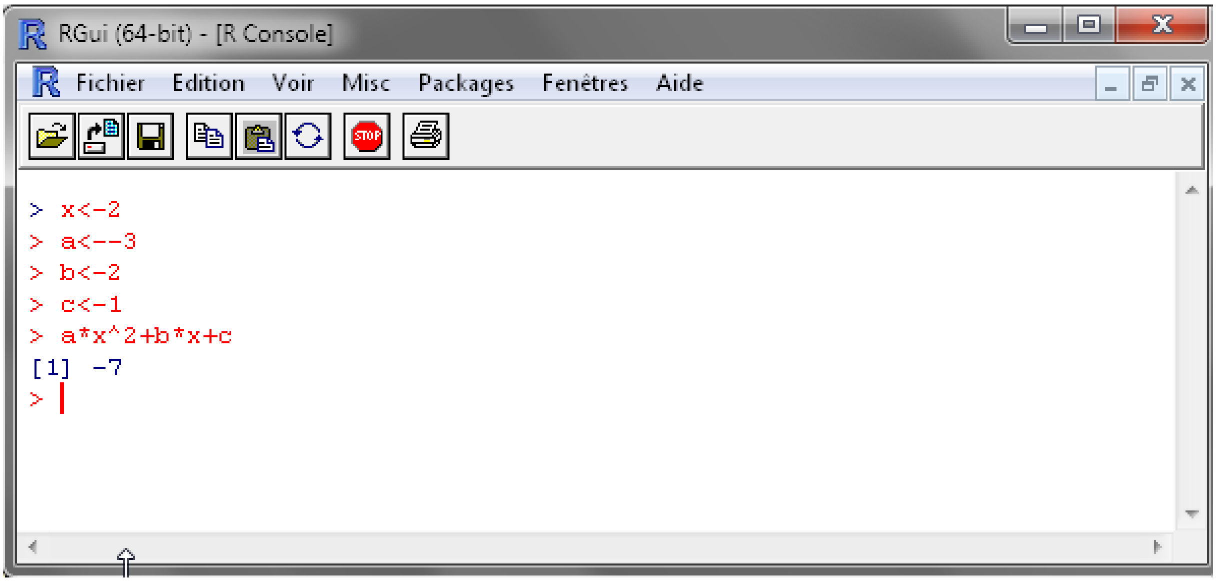


Rappelons que **outer( )** correspond au "produit dyadique" ou "outer product" et qu'il a une place très importante en statistiques avancées comme nous l'avons démontré dans le cours théorique (estimation de paramètres via la méthode de vraisemblance et particulièrement via la matrice de Fisher).

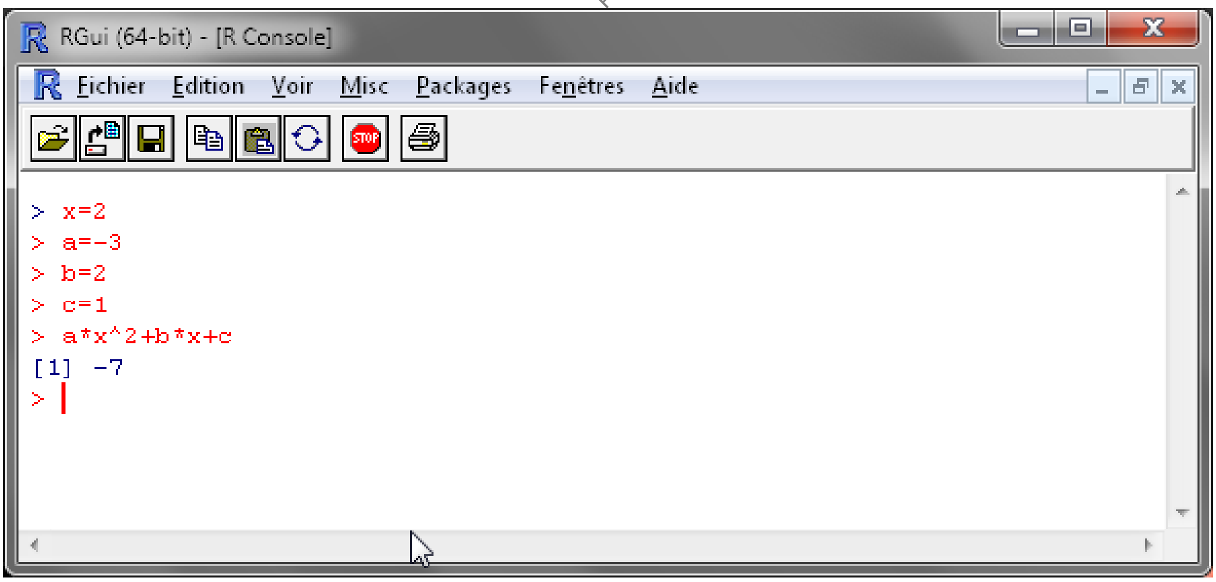
Exemple 4.: Manipulations de variables scalaires

R 3.0.2

Pas grand chose à dire ici tellement c'est élémentaire (mais attention! si possible respectez dans la pratique industrielle la nomenclature de Leszinsky-Reddick pour le nommage de vos variables):



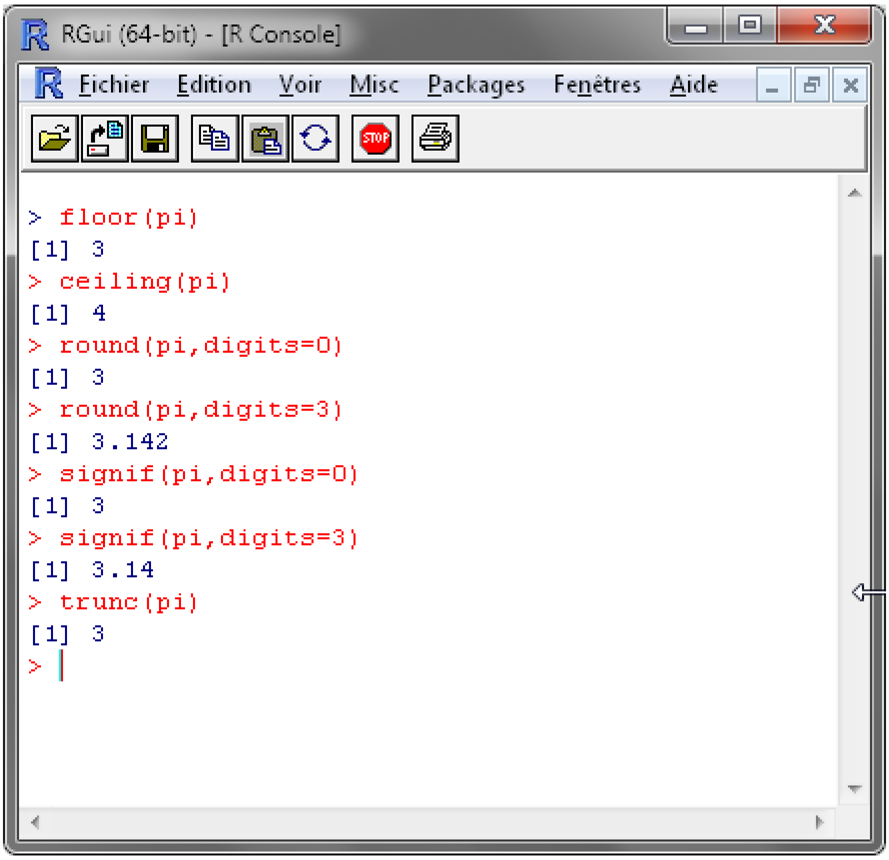
Autre manière d'écrire cependant parfois plus claire pour les petites classes:



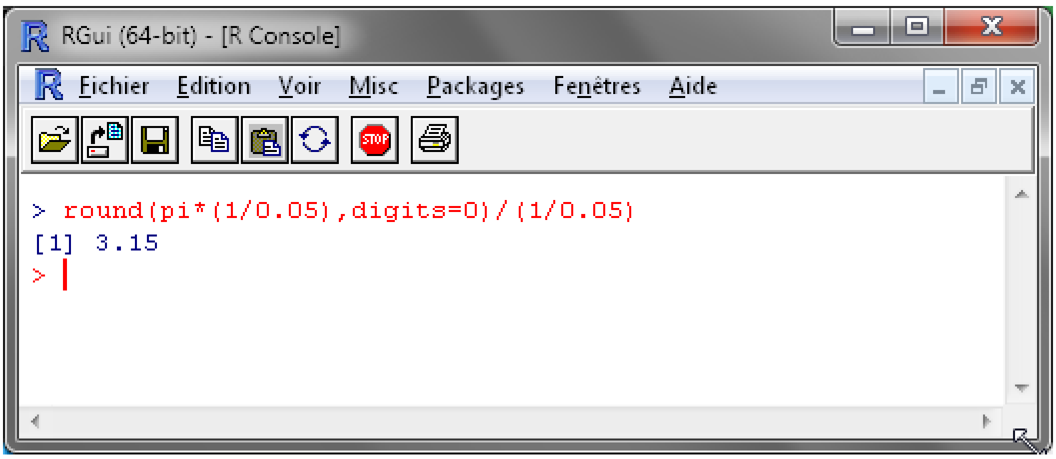
Exemple 5.: Arrondir des calculs

R 3.0.2

Là encore une fois le but est de faire un petit parallèle avec un tableur. Voyons que nous pouvons effectivement arrondir les résultats de calculs avec les commandes natives **floor( )**, **ceiling( )**, **round( )**, **signif( )** et **trunc( )**:



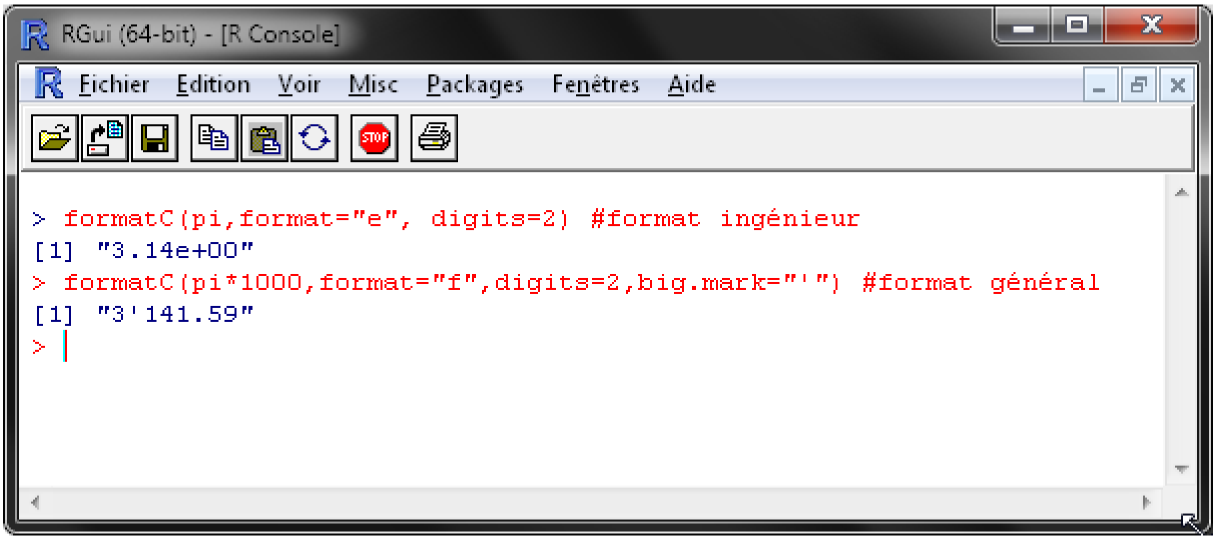
Sinon pour arrondir au 5 centimes près, il faudra passer par la même astuce qu'on utilisait dans les très vieux tableurs:



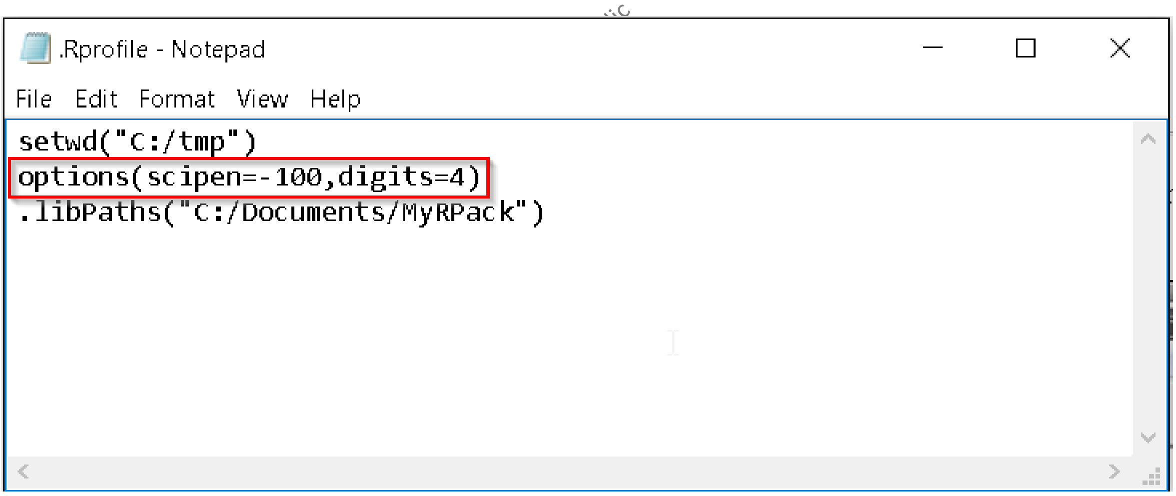
Exemple 6.: Options de formatage des nombres

R 3.0.2

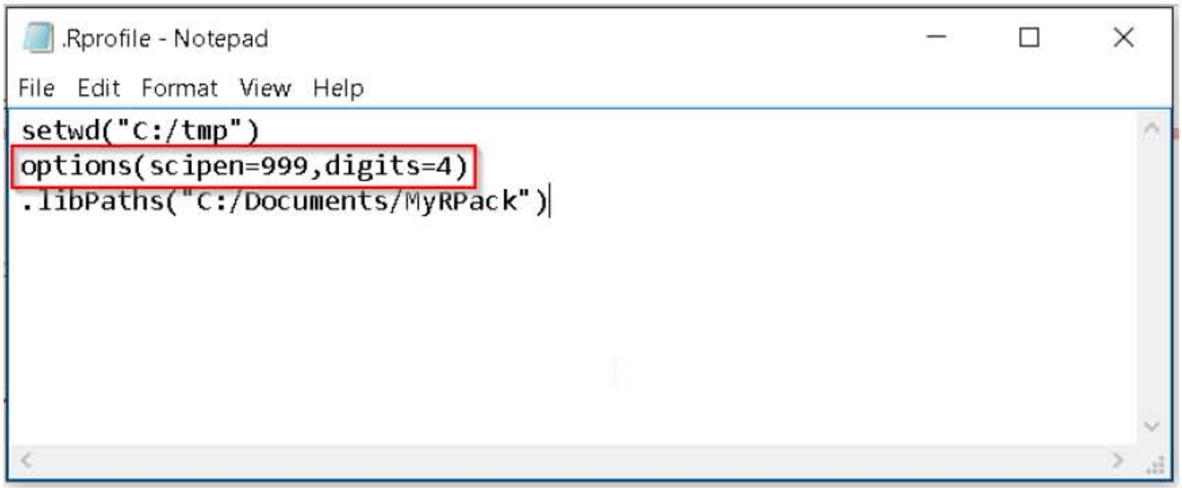
Quelques classiques pour afficher les nombres comme le proposent les calculatrices en utilisant la commande **formatC( )**:



Pour mettre par défaut le format ingénierie au démarrage de R vous pouvez mettre l'option **scipen=-100** dans votre fichier **.Rprofile**:



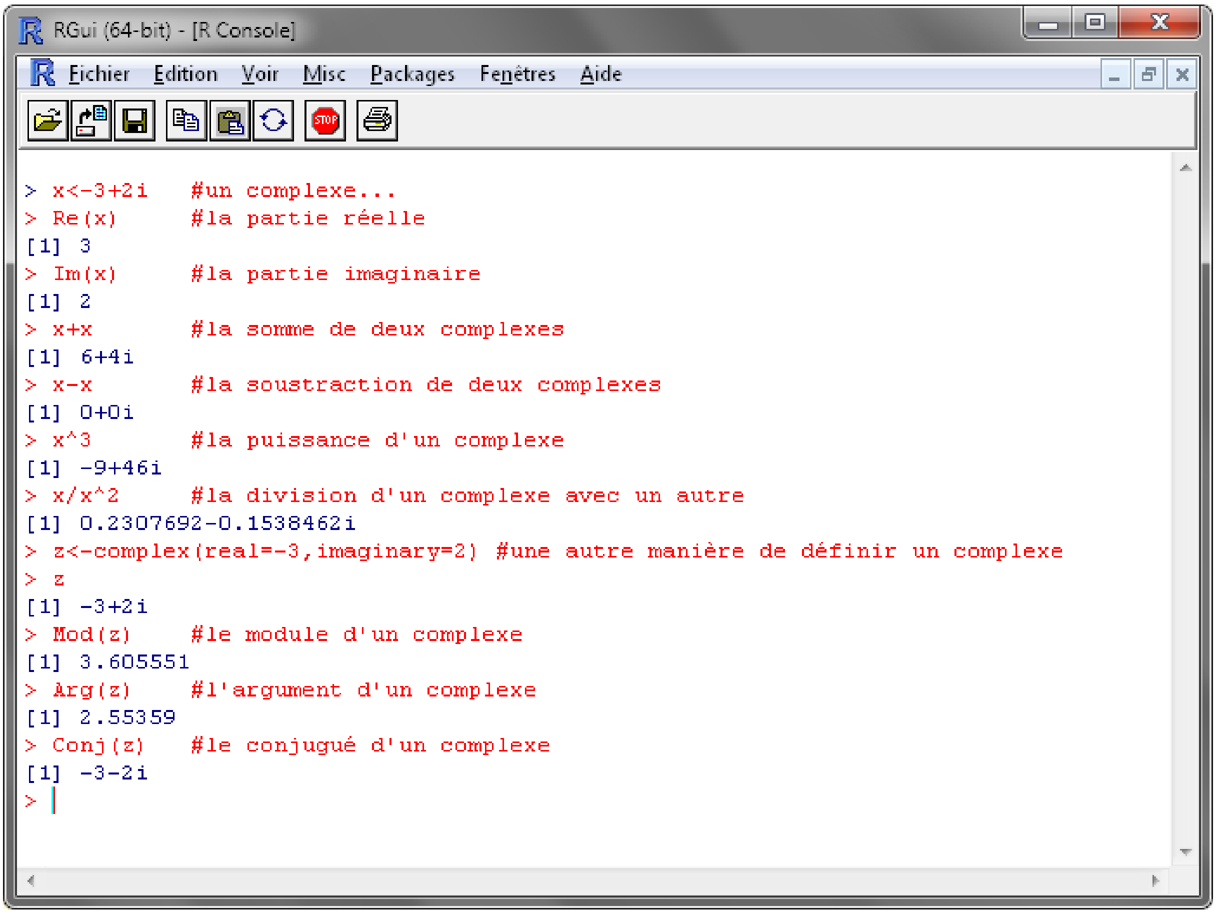
Inversement pour éliminer le format d'ingénierie dans toutes les sorties de type ANOVA et régressions linéaires vous pouvez mettre:


.Rprofile - Notepad
File Edit Format View Help
setwd("C:/tmp")
options(scipen=999,digits=4)
.libPaths("C:/Documents/MyRPack")

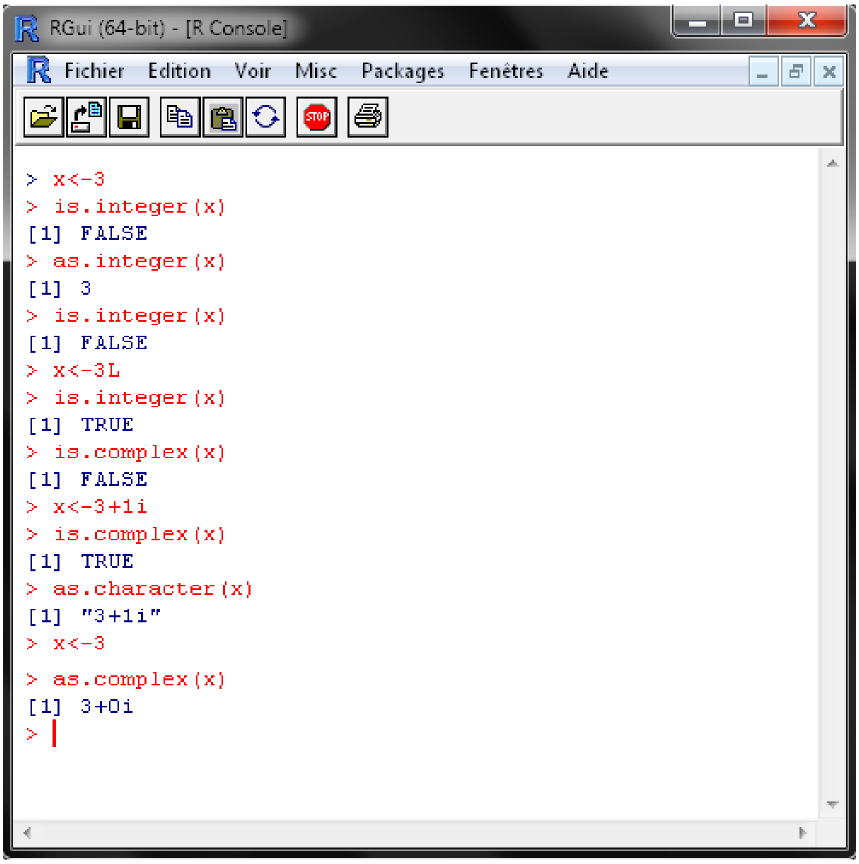
Exemple 7.: Nombres et variables complexes

R 3.0.2

On retrouvera dans la pratique les nombres complexes dans certains modèles de fractales ou également dans la transformée de Fourier. Donc vérifions que les éléments de base sont bien là comme dans n'importe quel tableur:



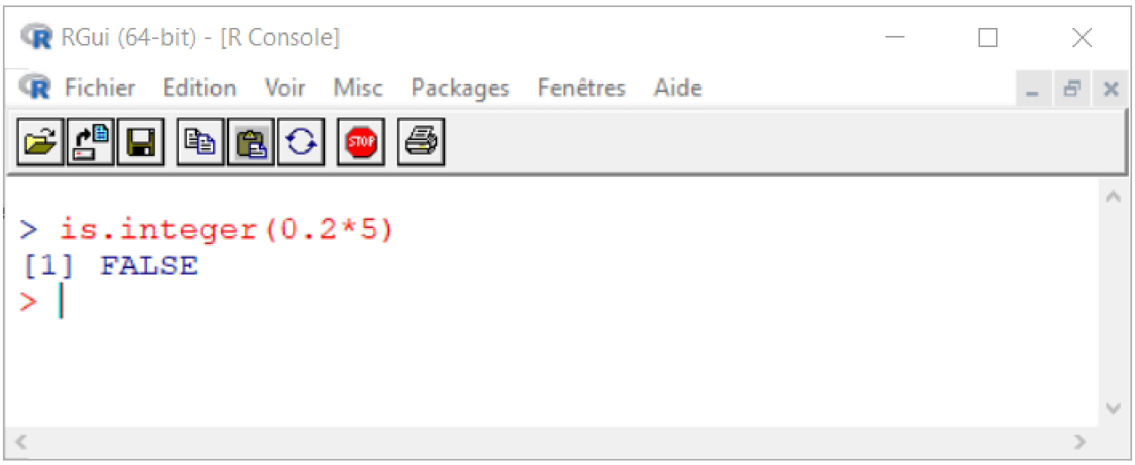
Faisons quelques tests de typage pour voir comment réagit R:



Actuellement dans R on a:

- is.logical
- is.integer
- is.numeric
- is.complex
- is.character

Il faut être très prudent quant à l'utilisation de **is.integer( )** comme on le voit dans l'exemple ci-dessous:



Exemple 8.: Séries arithmétiques ou géométriques ou autre

R 3.4.1

Dans le cours théorique nous avons démontré la série fameuse suivante:

$$\sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

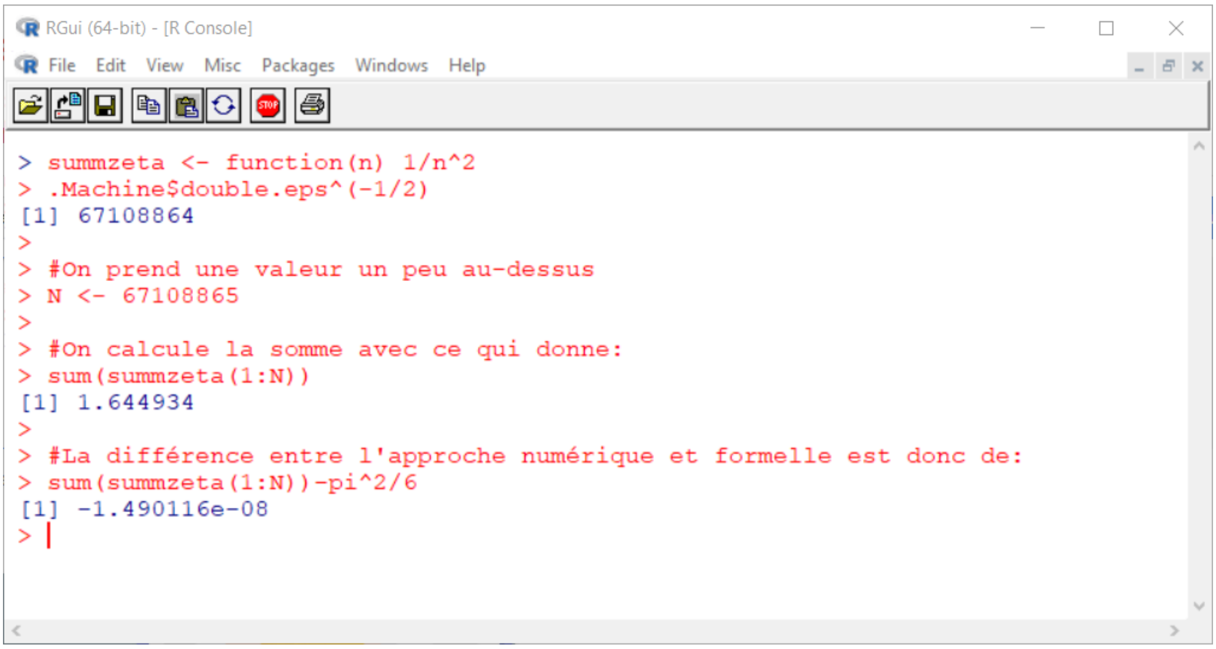
La convergence numérique sera obtenue quand :

$$\frac{1}{k^2} < \varepsilon$$

Soit quand:

$$k > \frac{1}{\sqrt{\varepsilon}} = \varepsilon^{-1/2}$$

Voyons cela avec R:



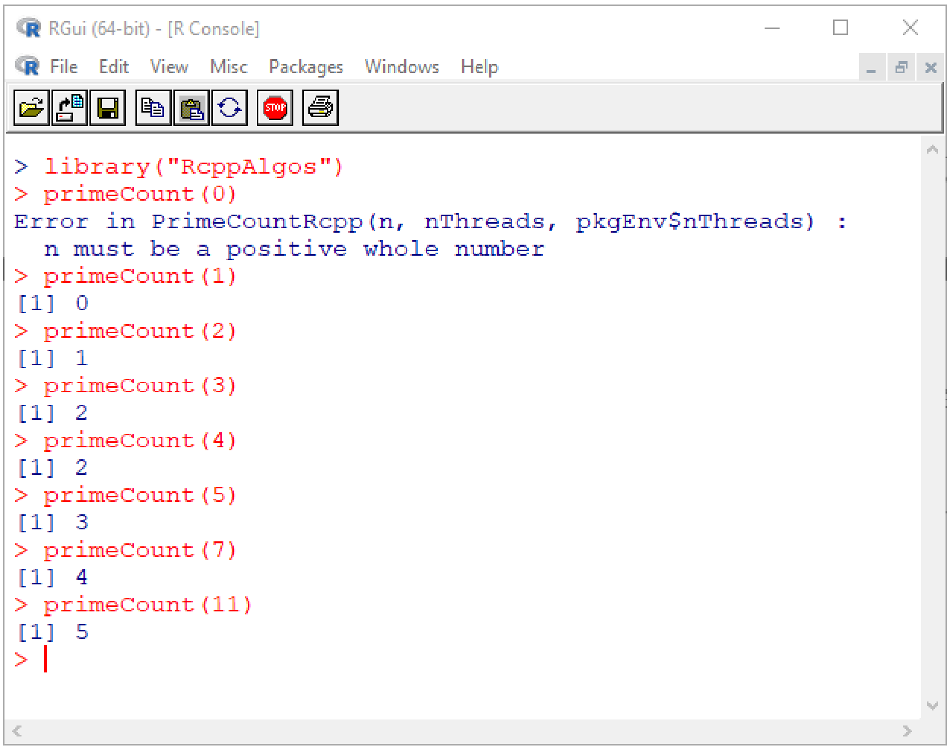
```
> summzeta <- function(n) 1/n^2
> .Machine$double.eps^(-1/2)
[1] 67108864
>
> #On prend une valeur un peu au-dessus
> N <- 67108865
>
> #On calcule la somme avec ce qui donne:
> sum(summzeta(1:N))
[1] 1.644934
>
> #La différence entre l'approche numérique et formelle est donc de:
> sum(summzeta(1:N))-pi^2/6
[1] -1.490116e-08
>
```

Cette différence est beaucoup plus grande que la précision numérique.

Exemple 9.: Nombres premiers

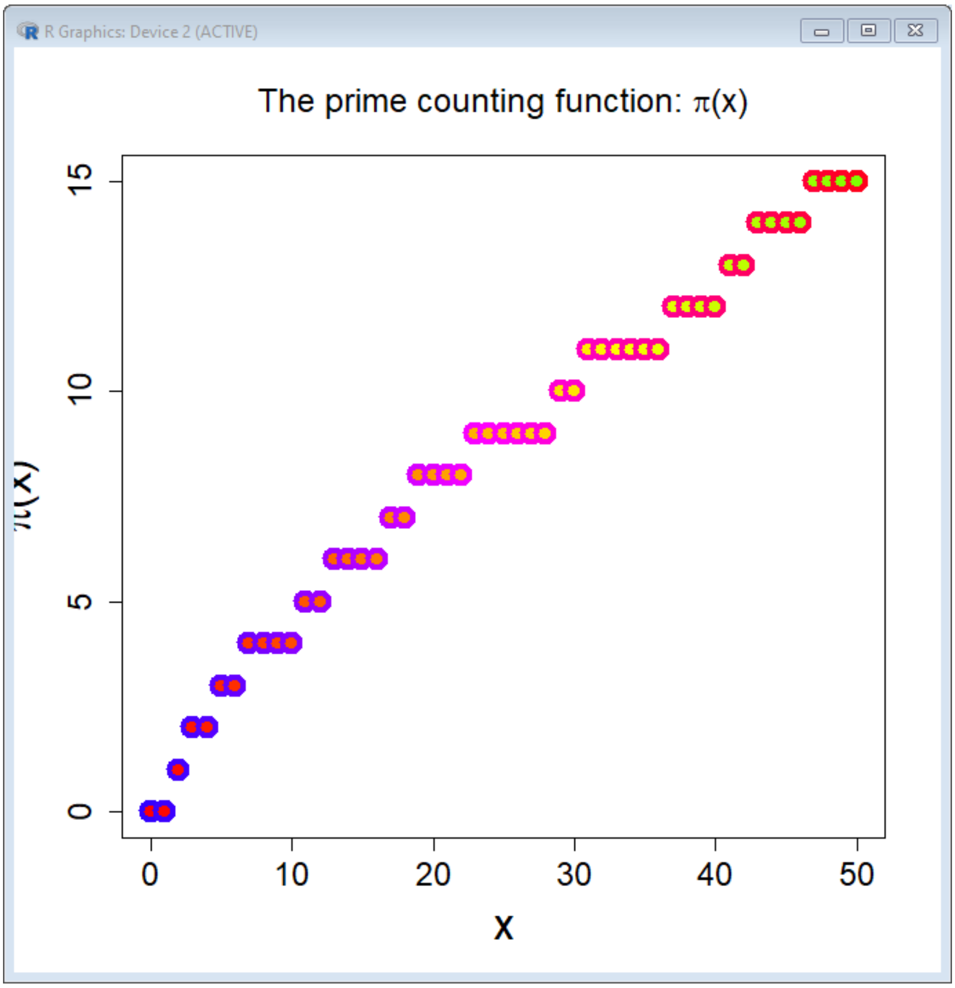
R 3.6.2

Comme dans le cours théorique, voyons la fonction de comptage de primalité (quantité de nombres premiers inférieurs ou égaux à un nombre donné) à l'aide du package RcppAlgos et sa fonction **primeCount( )**:

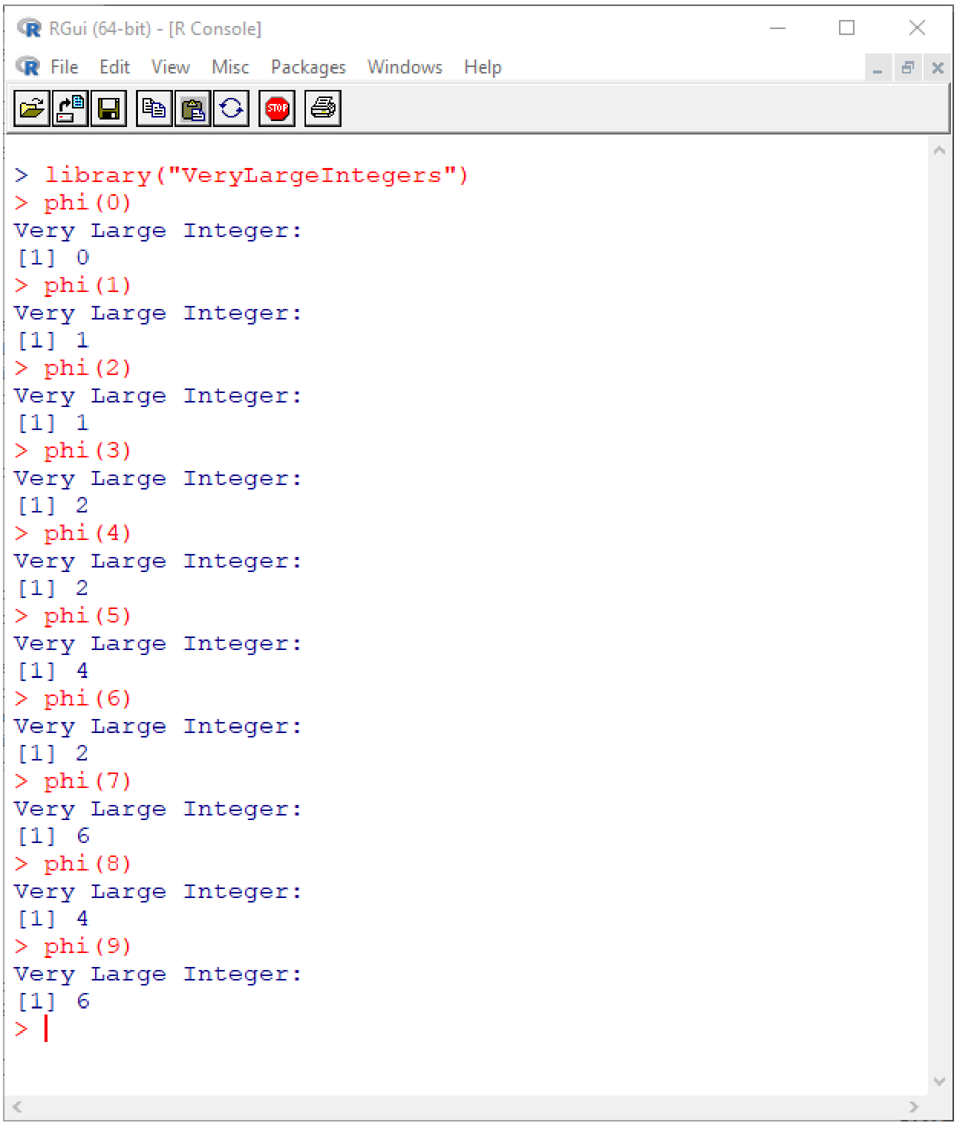


Nous pouvons aussi en faire un graphique à l'aide des packages **numbers**, **data.table** et **latex2exp**:

```
> library("numbers")
> library("data.table")
> library("latex2exp")
> x <- data.table(A=c(0:50))
> x$B <- as.numeric(isPrime(x$A))
> x[ , C := cumsum(B)]
> head(x,10)
    A B C
 1: 0 0 0
 2: 1 0 0
 3: 2 1 1
 4: 3 1 2
 5: 4 0 2
 6: 5 1 3
 7: 6 0 3
 8: 7 1 4
 9: 8 0 4
10: 9 0 4
> plot(x$A,x$C,pch=21,
+ main=TeX("The prime counting function: $\\pi(x)$"),
+ xlab=TeX("$x$"), ylab=TeX("$\\pi(x)$"), lwd=4,
+ col=rainbow(start=.7, 51), bg=rainbow(200),
+ cex=2, cex.lab=2, cex.main=1.5, cex.axis=1.5)
> 
```



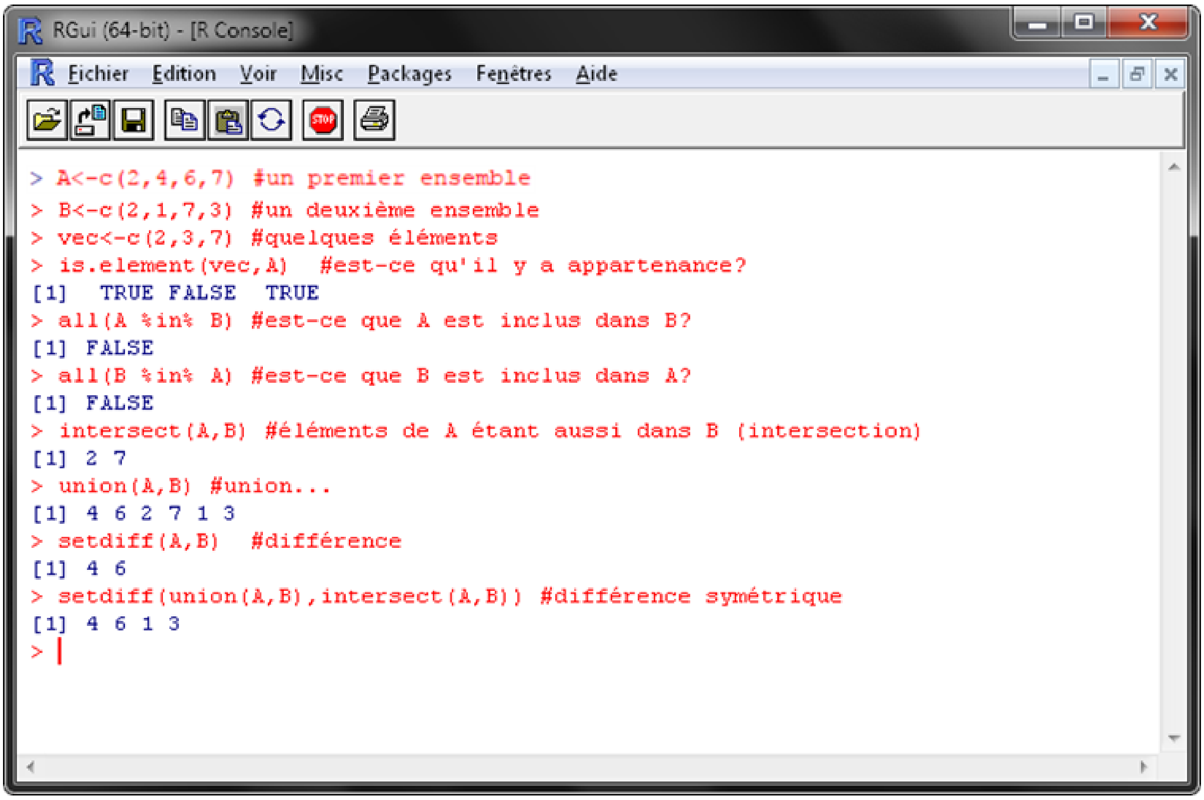
Fonction indicatrice phi-d'Euler (qui donne la quantité de nombre relativement premier à un nombre donné) à l'aide de la fonction **phi( )** du package **VeryLargeIntegers**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("VeryLargeIntegers")
> phi(0)
Very Large Integer:
[1] 0
> phi(1)
Very Large Integer:
[1] 1
> phi(2)
Very Large Integer:
[1] 1
> phi(3)
Very Large Integer:
[1] 2
> phi(4)
Very Large Integer:
[1] 2
> phi(5)
Very Large Integer:
[1] 4
> phi(6)
Very Large Integer:
[1] 2
> phi(7)
Very Large Integer:
[1] 6
> phi(8)
Very Large Integer:
[1] 4
> phi(9)
Very Large Integer:
[1] 6
>

Exemple 10.: Manipulation d'ensembles (un peu de théorie des ensembles)

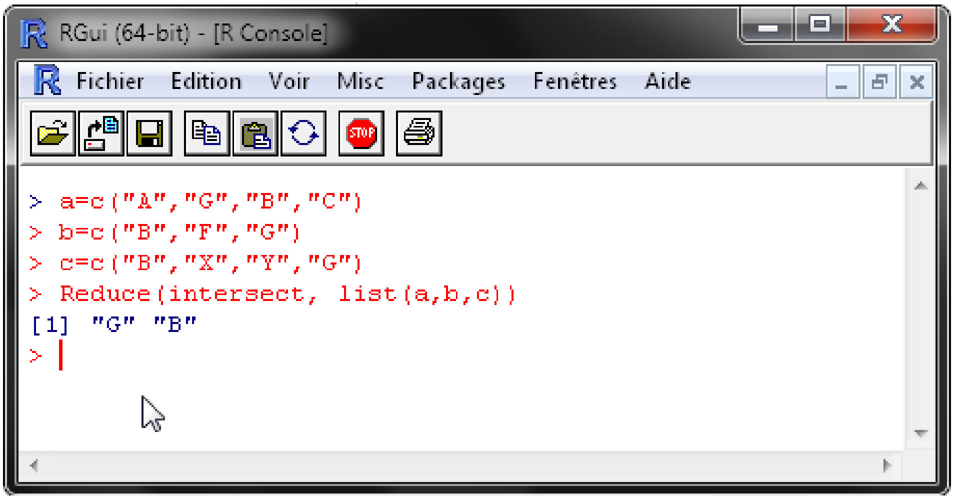
R 3.0.2

Voici quelques éléments de la théorie des ensembles qui rappelleront des souvenirs de la petite école à de nombreux lecteurs:

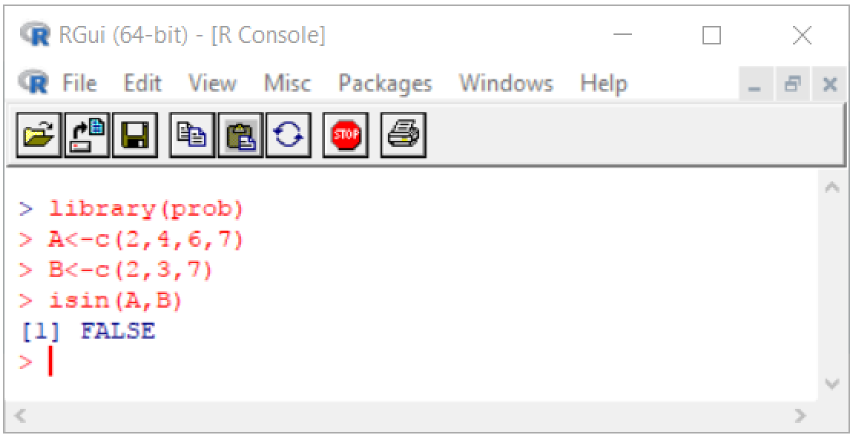


La commande **intersect( )** est particulièrement utilisée en finance lorsque nous possédons deux séries temporelles dont nous souhaitons extraire que les dates communes! Une fois que nous avons le vecteur des dates communes, nous pouvons utiliser les techniques de filtrage habituelles sur les jeux de données initiaux.

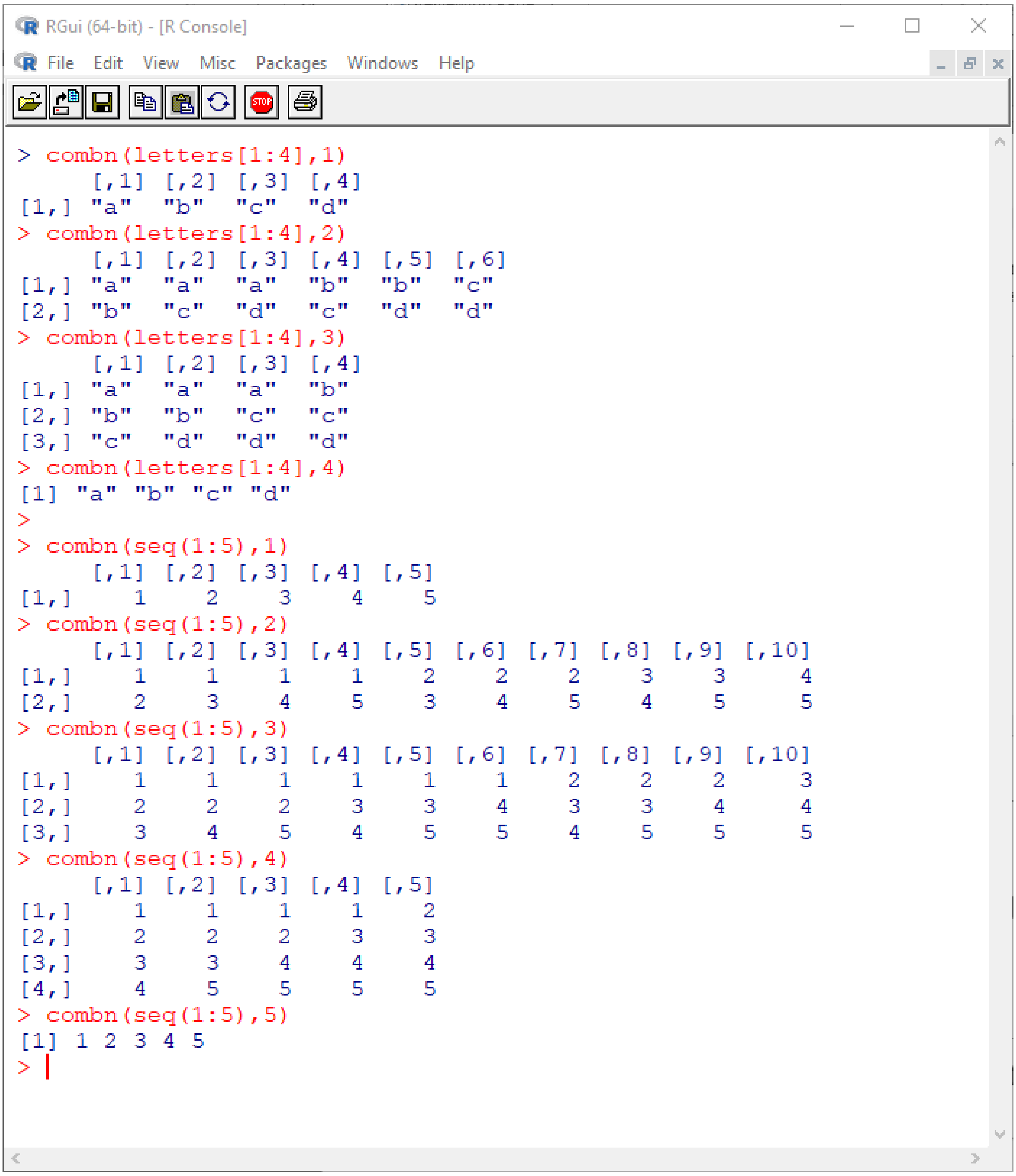
Comme **intersect( )** ne prend que deux éléments en entrée, une manière pour généraliser est d'utiliser la commande **reduce( )**:



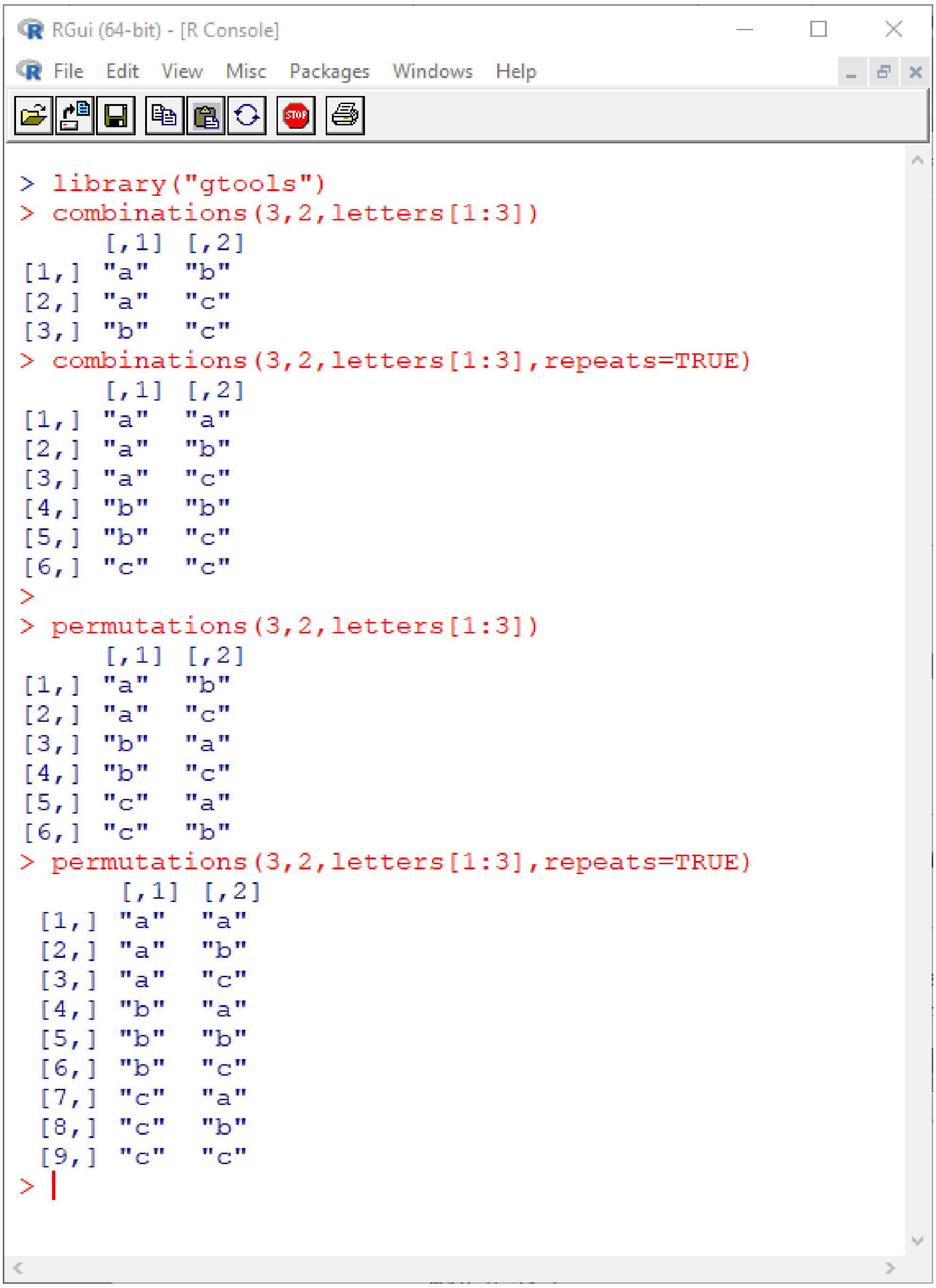
Il y a aussi la commande **isin( )** du package **prob** que nous aurions pu signaler au tout début (mais que nous avons oublié d'indiquer) qui permet de savoir si un ensemble (ou vecteur) est totalement inclus dans un autre ensemble (vecteur) et donc évite de faire un test (if) sur le résultant de l'usage de **%in%** comme vu précédemment:



Sinon il y aussi la fonction **combn( )** qui n'est pas qu'utile pour avoir toutes les combinaisons d'un certain ordre d'un ensemble mais aussi pour le calcul de certaines statistiques non paramétriques (dont la quasimédiane!).



Ou sinon au niveau des permutation et combinaisons on peut avoir les classiques avec le package **gtools** et les fonctions **permutations( )** et **combinations( )**:

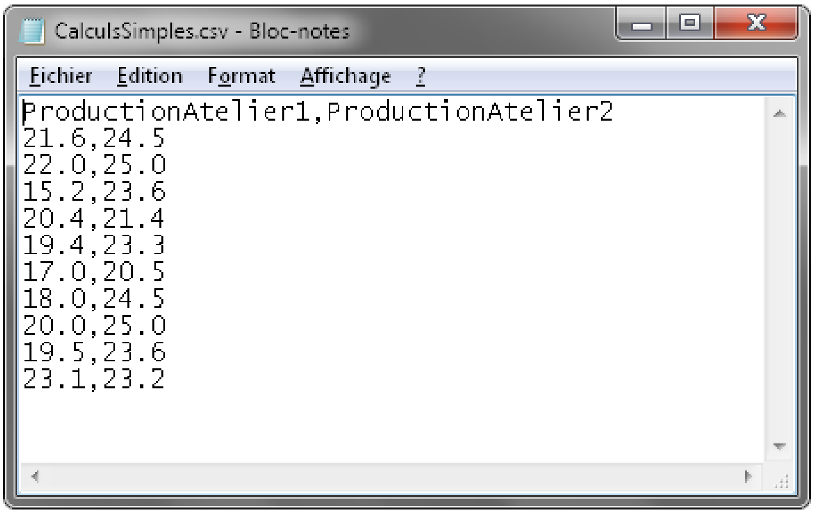


Mais le package **arrangements** (pour les curieux) est bien plus complet pour information!

Exemple 11.: Opérations avec des data frame

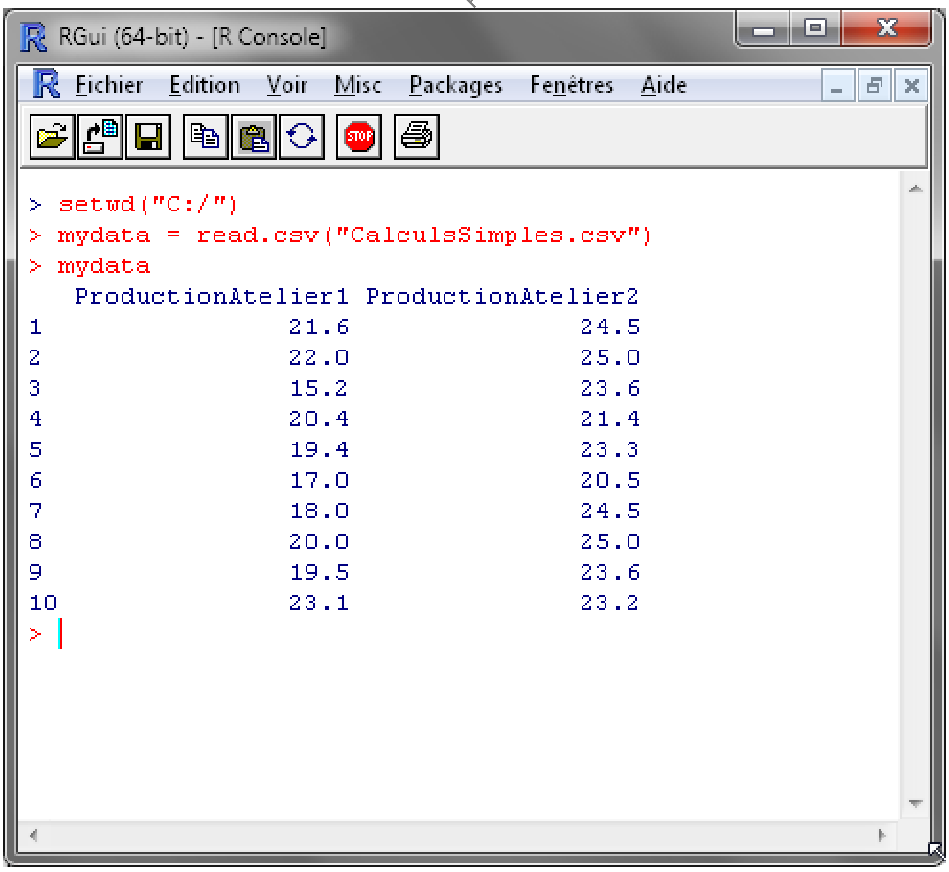
R 3.0.2

Il peut arriver dans R que nous ayons parfois besoin de faire des calculs arithmétiques élémentaires. Voyons un exemple. Ouvrez le fichier *CalculsSimple.csv*:

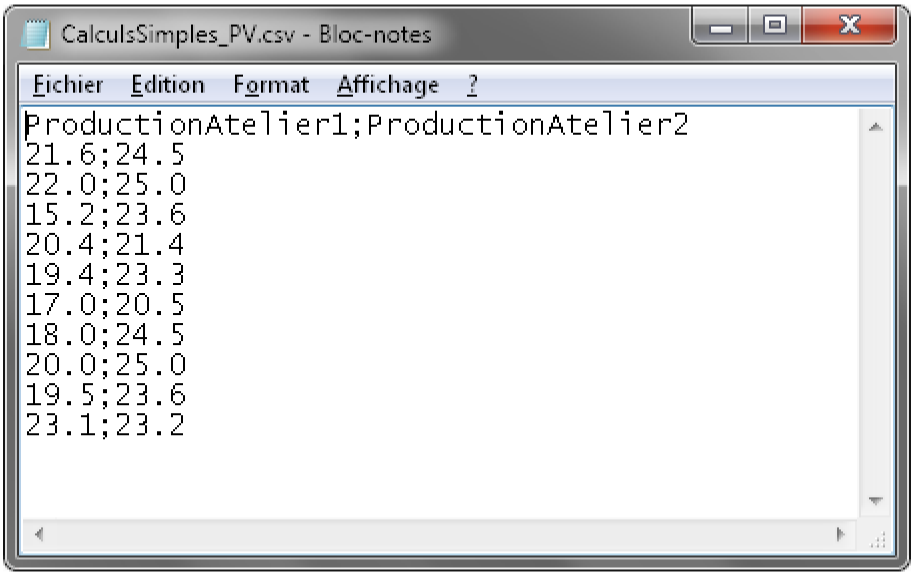


Remarquez bien qu'il n'y a pas d'espaces dans les titres et que le séparateur est bien la virgule et non le ";".

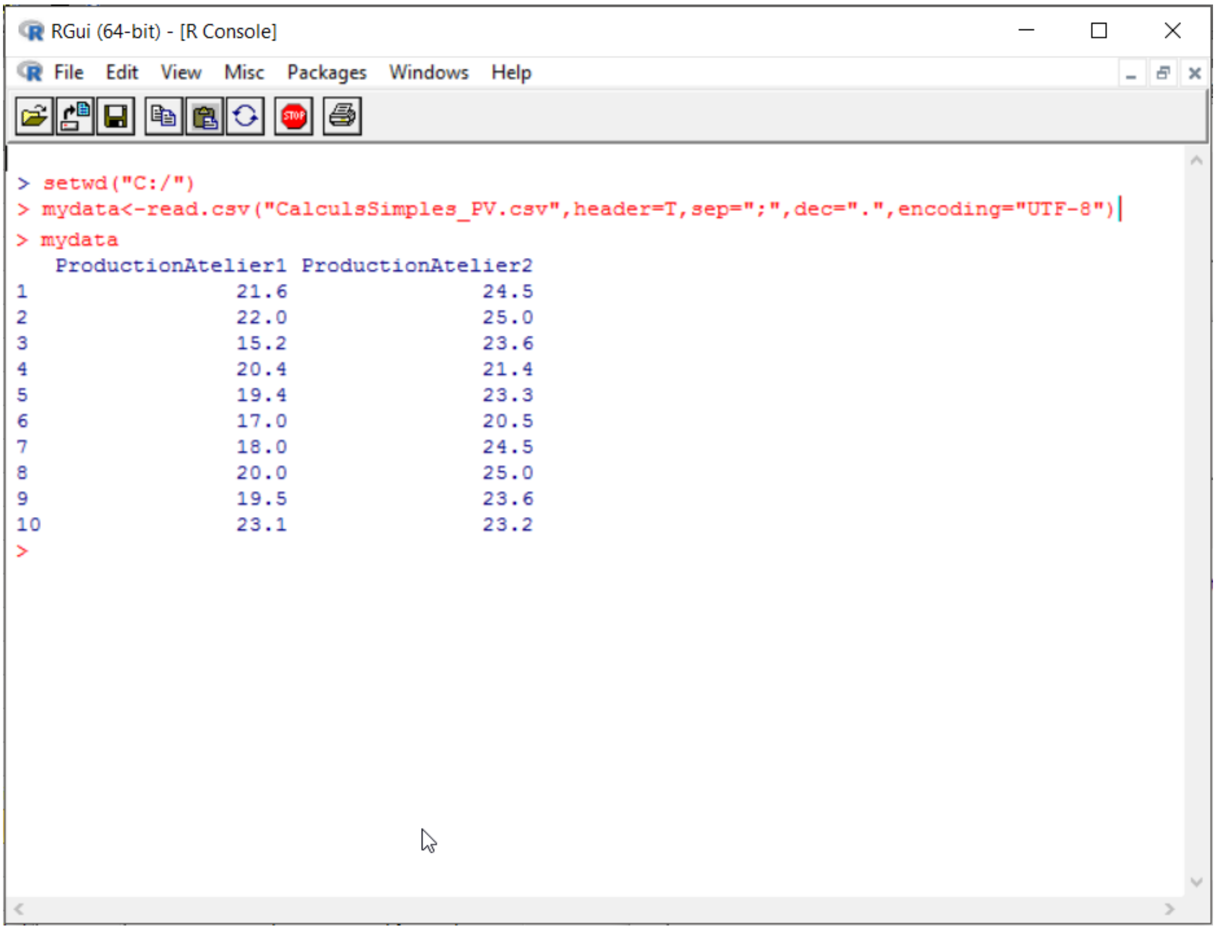
Nous allons d'abord le charger dans R avec la commande **read.csv( )** et afficher son contenu:



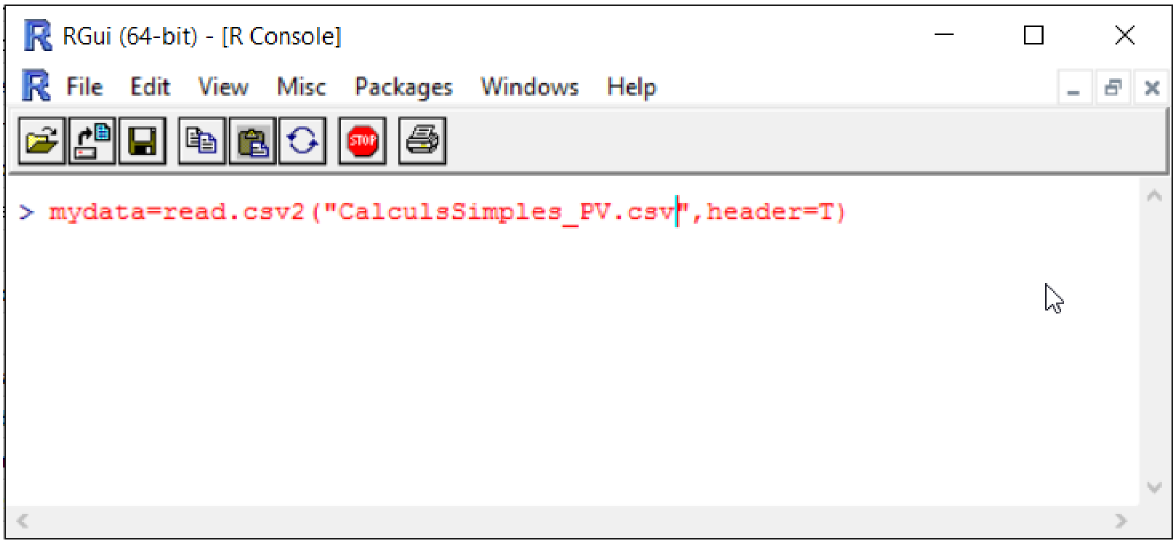
Remarquez que si le fichier avait eu la structure suivante:



nous aurions pu aussi l'importer ainsi en étant encore plus complet:

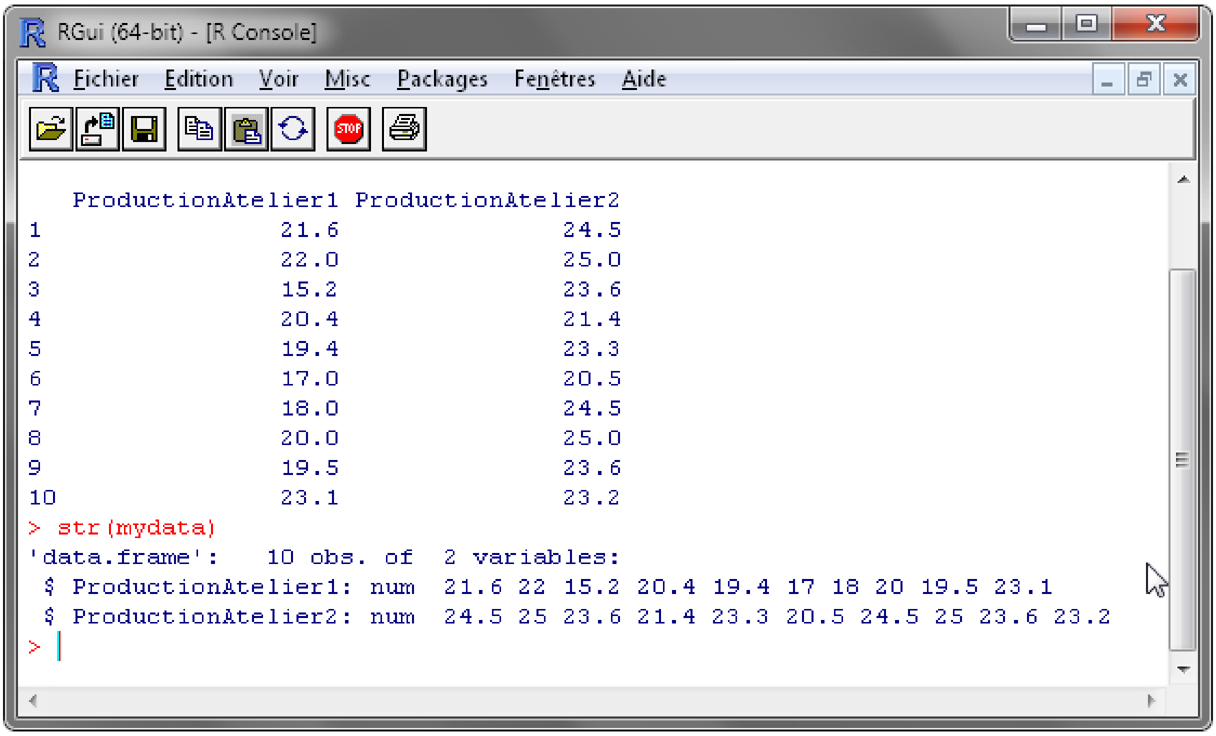


ou utiliser simplement:

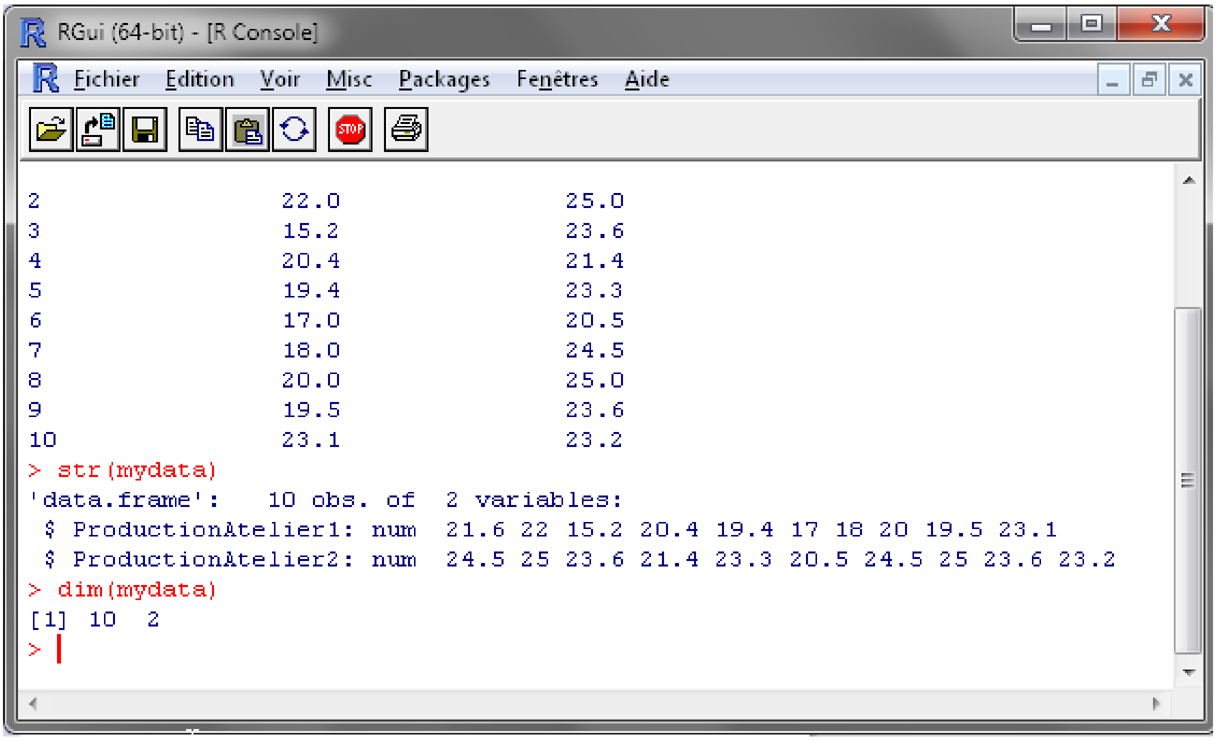


puisque la fonction **read.csv2( )** par défaut considère le séparateur comme un ";".

Et nous pouvons demander un petit résumé avec la commande **str**:

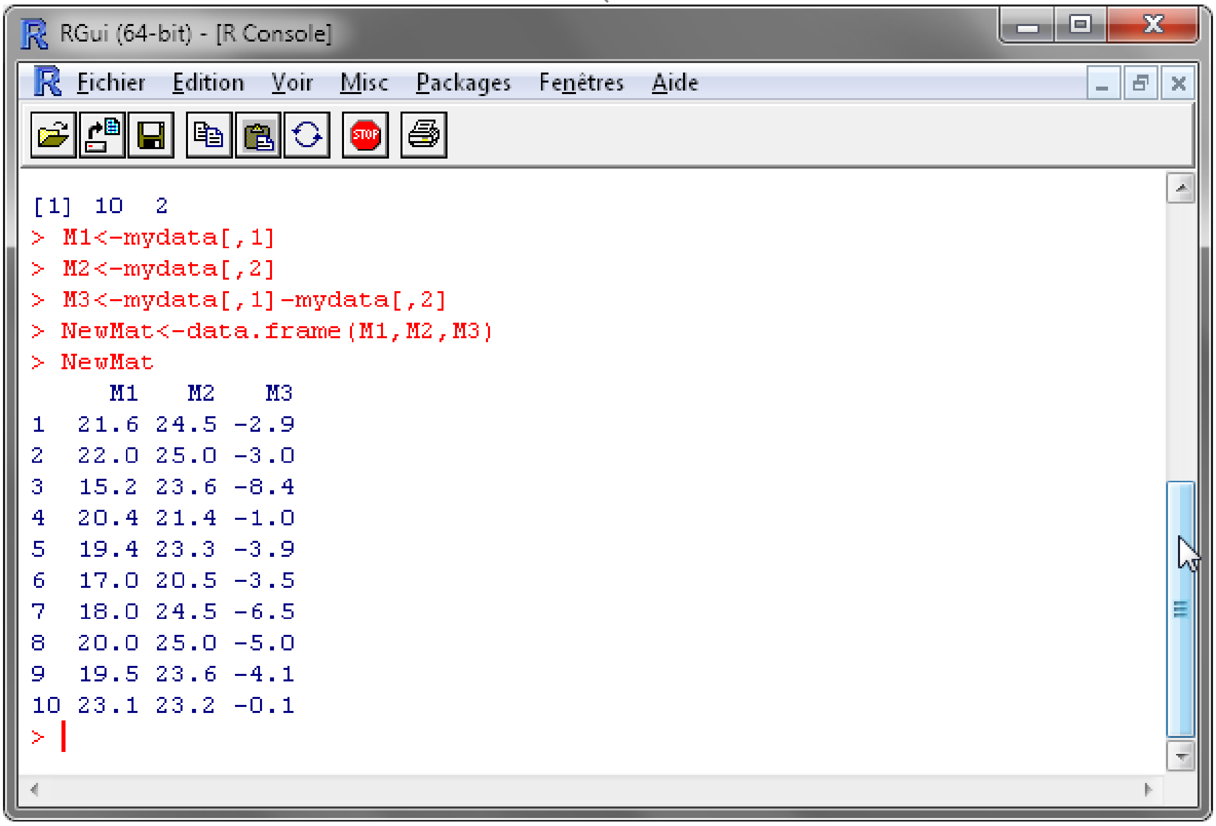


et demander d'afficher les dimensions avec la commande **dim( )** (lignes, colonnes):

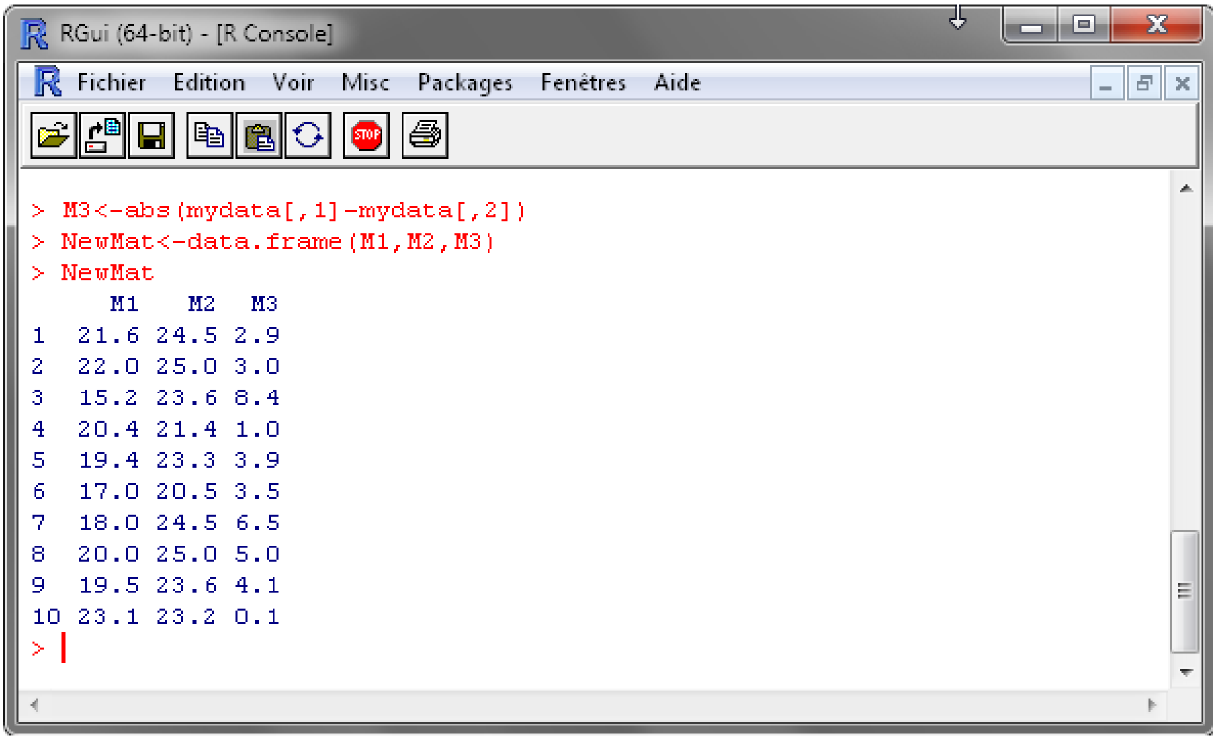


Remarque: **str( )** signifie "structure"

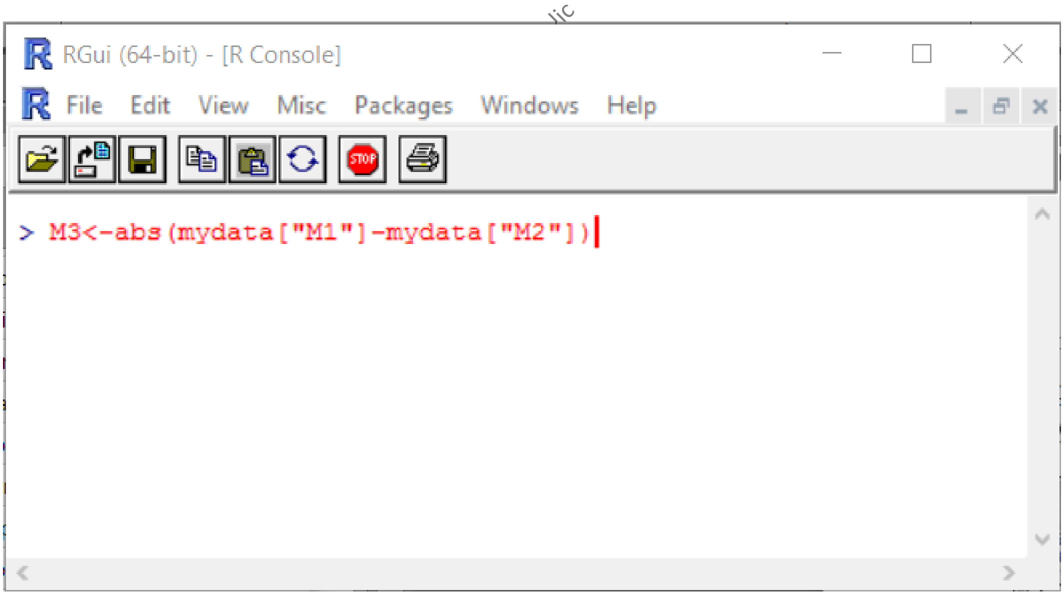
Maintenant, nous souhaiterions simplement une colonne qui contient la différence des deux. Pour se faire, nous allons utiliser la commande **data.frame( )** qui permet de recréer des arrays dans la mémoire:



ou pour faire comme dans le livre Minitab avec la valeur absolue en utilisation fonction **abs( )**:



À noter que la première ligne ci-dessus aurait aussi pu être écrite:

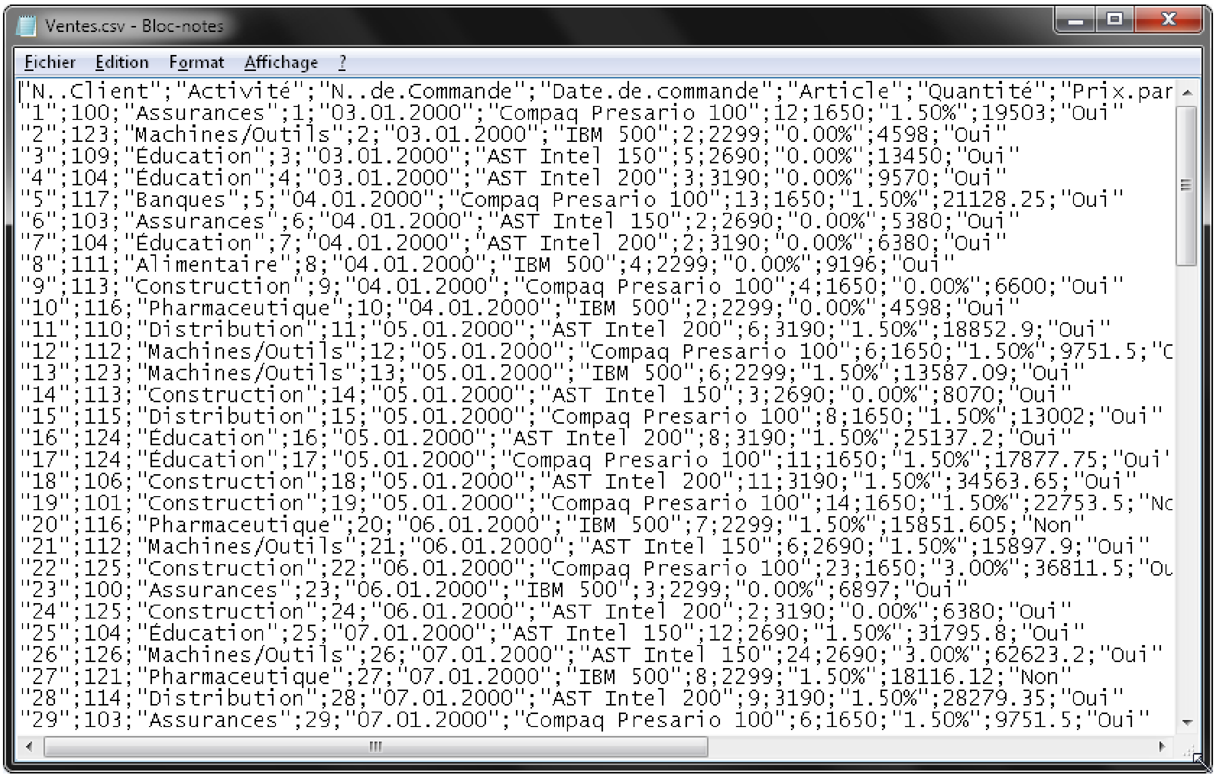


Nous reviendrons plus en détails sur la manipulation des data frames dans un chapitre qui sera consacré à ce sujet à la page 498.

Exemple 12.: Statistiques descriptives élémentaires univariées de data frame

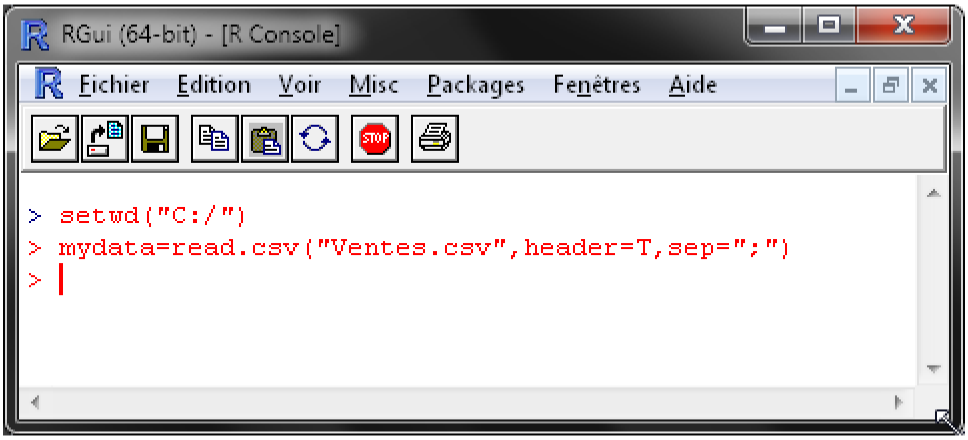
R 3.0.2

Nous importons le fichier suivant:



```
"N..Client";"Activité";"N..de.Commande";"Date.de.commande";"Article";"Quantité";"Prix.par
"1";100;"Assurances";1;"03.01.2000";"Compaq Presario 100";12;1650;"1.50%";19503;"Oui"
"2";123;"Machines/Outils";2;"03.01.2000";"IBM 500";2;2299;"0.00%";4598;"Oui"
"3";109;"Éducation";3;"03.01.2000";"AST Intel 150";5;2690;"0.00%";13450;"Oui"
"4";104;"Éducation";4;"03.01.2000";"AST Intel 200";3;3190;"0.00%";9570;"Oui"
"5";117;"Banques";5;"04.01.2000";"Compaq Presario 100";13;1650;"1.50%";21128.25;"Oui"
"6";103;"Assurances";6;"04.01.2000";"AST Intel 150";2;2690;"0.00%";5380;"Oui"
"7";104;"Éducation";7;"04.01.2000";"AST Intel 200";2;3190;"0.00%";6380;"Oui"
"8";111;"Alimentaire";8;"04.01.2000";"IBM 500";4;2299;"0.00%";9196;"Oui"
"9";113;"Construction";9;"04.01.2000";"Compaq Presario 100";4;1650;"0.00%";6600;"Oui"
"10";116;"Pharmaceutique";10;"04.01.2000";"IBM 500";2;2299;"0.00%";4598;"Oui"
"11";110;"Distribution";11;"05.01.2000";"AST Intel 200";6;3190;"1.50%";18852.9;"Oui"
"12";112;"Machines/Outils";12;"05.01.2000";"Compaq Presario 100";6;1650;"1.50%";9751.5;"C
"13";123;"Machines/Outils";13;"05.01.2000";"IBM 500";6;2299;"1.50%";13587.09;"Oui"
"14";113;"Construction";14;"05.01.2000";"AST Intel 150";3;2690;"0.00%";8070;"Oui"
"15";115;"Distribution";15;"05.01.2000";"Compaq Presario 100";8;1650;"1.50%";13002;"Oui"
"16";124;"Éducation";16;"05.01.2000";"AST Intel 200";8;3190;"1.50%";25137.2;"Oui"
"17";124;"Éducation";17;"05.01.2000";"Compaq Presario 100";11;1650;"1.50%";17877.75;"Oui'
"18";106;"Construction";18;"05.01.2000";"AST Intel 200";11;3190;"1.50%";34563.65;"Oui"
"19";101;"Construction";19;"05.01.2000";"Compaq Presario 100";14;1650;"1.50%";22753.5;"Nc
"20";116;"Pharmaceutique";20;"06.01.2000";"IBM 500";7;2299;"1.50%";15851.605;"Non"
"21";112;"Machines/Outils";21;"06.01.2000";"AST Intel 150";6;2690;"1.50%";15897.9;"Oui"
"22";125;"Construction";22;"06.01.2000";"Compaq Presario 100";23;1650;"3.00%";36811.5;"Ou
"23";100;"Assurances";23;"06.01.2000";"IBM 500";3;2299;"0.00%";6897;"Oui"
"24";125;"Construction";24;"06.01.2000";"AST Intel 200";2;3190;"0.00%";6380;"Oui"
"25";104;"Éducation";25;"07.01.2000";"AST Intel 150";12;2690;"1.50%";31795.8;"Oui"
"26";126;"Machines/Outils";26;"07.01.2000";"AST Intel 150";24;2690;"3.00%";62623.2;"Oui"
"27";121;"Pharmaceutique";27;"07.01.2000";"IBM 500";8;2299;"1.50%";18116.12;"Non"
"28";114;"Distribution";28;"07.01.2000";"AST Intel 200";9;3190;"1.50%";28279.35;"Oui"
"29";103;"Assurances";29;"07.01.2000";"Compaq Presario 100";6;1650;"1.50%";9751.5;"Oui"
```

Ce qui donne:

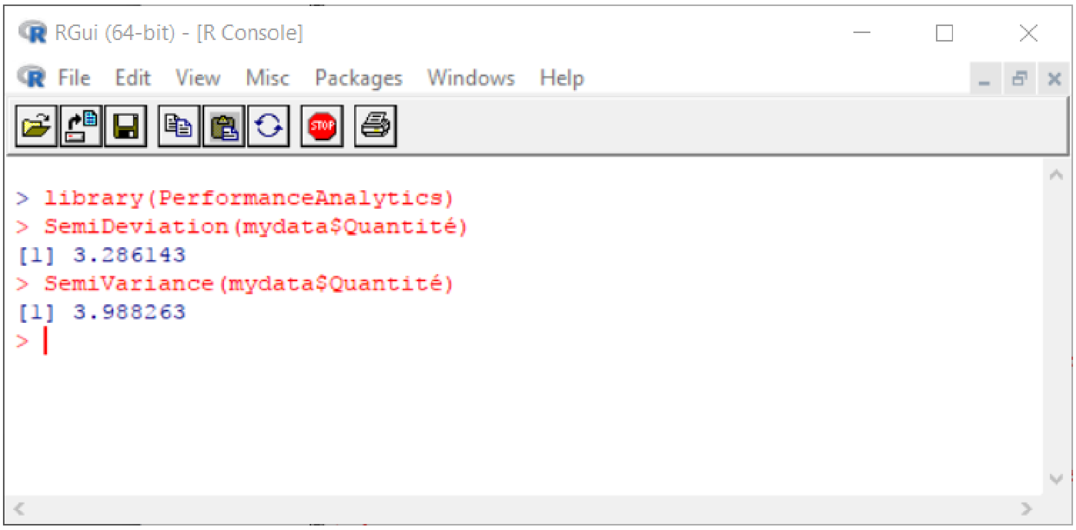


```
> setwd("C:/")
> mydata=read.csv("Ventes.csv",header=T,sep=";")
>
```

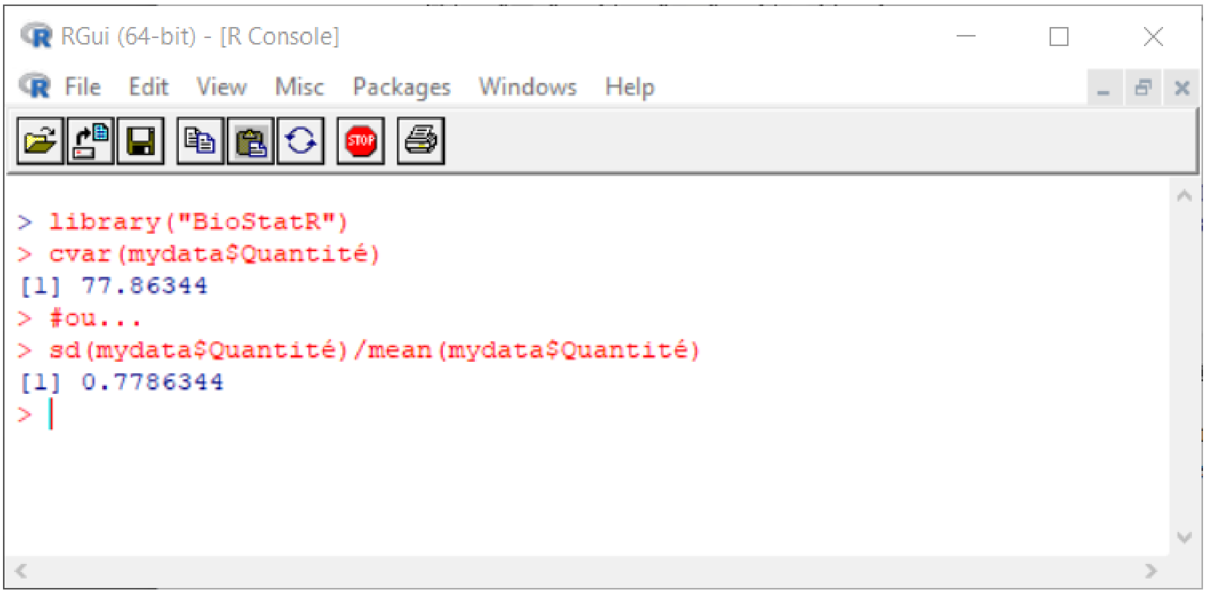
Nous demandons quelques statistiques descriptives élémentaires comme **max( )**, **sd( )** , **var( )** , **median( )** , **quantile( )** (attention!!! Se rappeler qu'il y neuf manières de calculer les quantiles à ce jour et donc les résultats différent d'un logiciel à un autre!), **iqr( )** , **mad( )** (respectivement: maximum, écart-type, variance, médiane, quantile – ou déciles, écart interquartile, déviation absolue de la médiane) d'une colonne particulière:

```
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=",")
> attach(mydata,warn.conflicts = FALSE) #pour faciliter la suite
>
> (n<-length(Quantite)) #comptage
[1] 109
> sum(Quantite) #somme
[1] 884
> mean(Quantite) #moyenne
[1] 8.110092
> max(Quantite) #maximum
[1] 33
> min(Quantite) #minimum
[1] 1
> range(Quantite) #étendue
[1]  1 33
> diff(range(Quantite)) #étendue calculée
[1] 32
> names(table(Quantite))[table(Quantite)==max(table(Quantite))] #valeur modale (sans package)
[1] "2"
> var(Quantite) #variance biaisée
[1] 39.87666
> var(Quantite)*n/(n-1) #variance non biaisée
[1] 40.24588
> sd(Quantite) #écart-type biaisé
[1] 6.314797
> sd(Quantite)*(n/(n-1))^2 #écart-type non biaisé
[1] 6.432279
>
> median(Quantite) #médiance
[1] 6
> quantile(Quantite,0.5) #quantile
50%
  6
> IQR(Quantite) #étendue interquartile
[1] 7
> mad(Quantite)  #déviation absolume moyenne à la médiane
[1] 4.4478
>
> sd(Quantite)/mean(Quantite) #coefficient de variation paramétrique
[1] 0.7786344
> mad(Quantite)/median(Quantite) #coefficient de variation non-paramétrique
[1] 0.7413
>
> abs(mean(Quantite)-mean(sample(Quantite,50))) #biais de la moyenne
[1] 0.4899083
> 
```

ou encore les semi-variance ou semi-écart-type très utilisé en finance:



Pour le coefficient de variation on peut aussi utiliser la fonction **cvar( )** du package BioStatR au besoin:



Les quantiles étant importants dans la pratique, investiguons un peu quelques syntaxes possibles:

```
> quantile(mydata$Quantité,probs=c(0.1,0.9))
 10%  90% 
 3.2 23.8 
> quantile(mydata$Quantité,probs=1:10/10)
 10%  20%  30%  40%  50%  60%  70%  80%  90% 100% 
 3.2  5.4  7.6  9.8 12.0 14.2 17.2 21.6 23.8 33.0 
> 
```

ou la commande **fivenum( )** qui renvoie respectivement le minimum, le 1er quartile de Tukey, la médiane (50ème centile), le 3ème quartile de Tukey (75ème centile) et le maximum:

```
> fivenum(mydata$Quantite)
[1]  1  4  6 11 33
> 
```

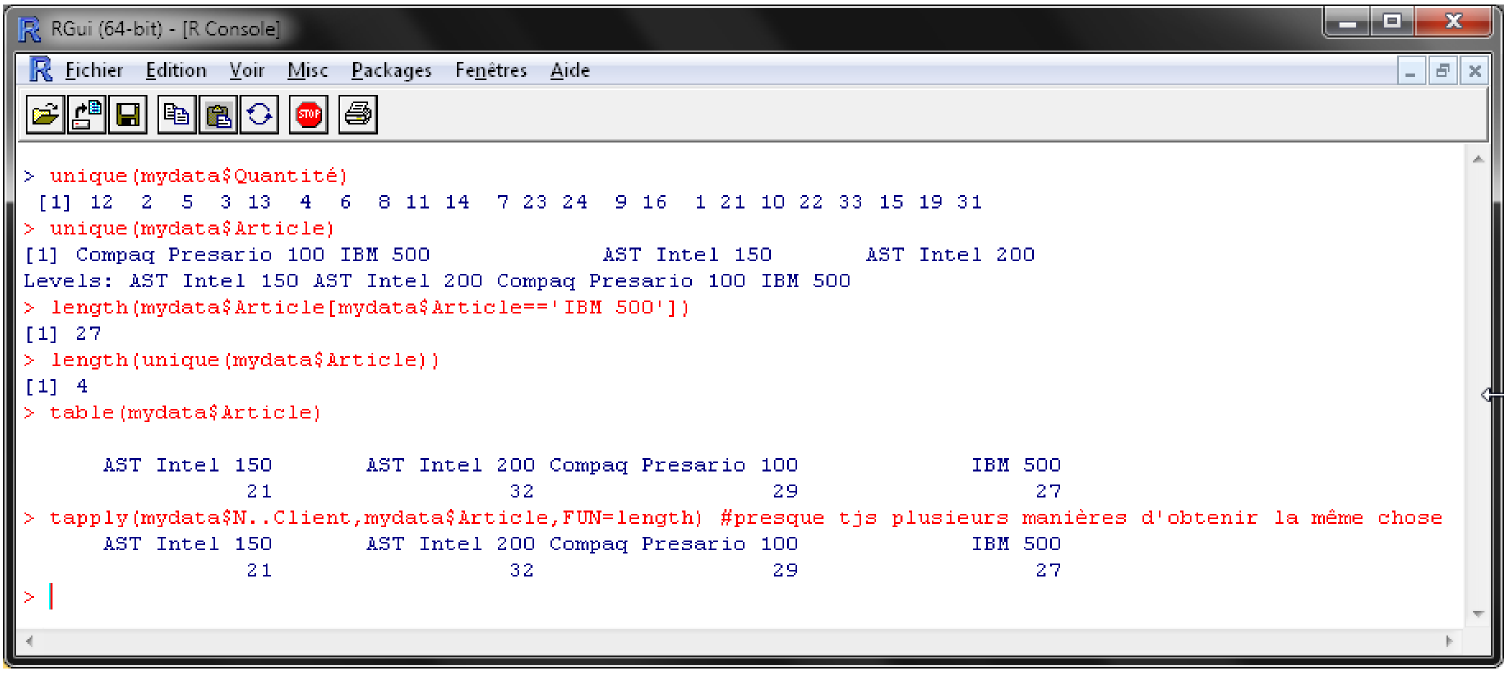
Il y a aussi la commande **summary( )** :

```
> summary(mydata$Quantité)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    4.00    6.00    8.11   11.00   33.00 
> summary(mydata[,1:6])
   N..Client                  Activité   N..de.Commande   Date.de.commande                  Article      Quantité    
 Min.   :100.0   Construction     :21   Min.   :  1     19.01.2000:10      AST Intel 150      :21   Min.   : 1.00  
 1st Qu.:105.0   Éducation        :19   1st Qu.: 28     05.01.2000: 9      AST Intel 200      :32   1st Qu.: 4.00  
 Median :113.0   Machines/Outils:18   Median : 55     28.01.2000: 9      Compaq Presario 100:29   Median : 6.00  
 Mean   :112.8   Distribution     :13   Mean   : 55     18.01.2000: 8      IBM 500            :27   Mean   : 8.11  
 3rd Qu.:120.0   Pharmaceutique   :12   3rd Qu.: 82     04.01.2000: 6                                3rd Qu.:11.00  
 Max.   :127.0   Assurances       :10   Max.   :109     07.01.2000: 6                                Max.   :33.00  
                 (Other)          :16                   (Other)   :61                                                
> 
```

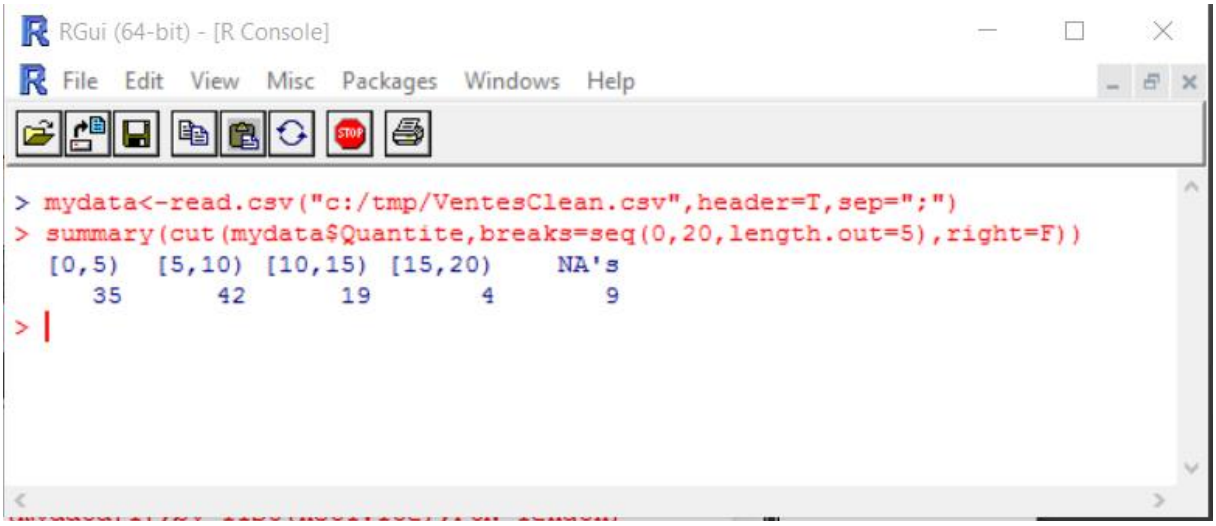
dont il est possible de voir toutes les méthodes puisqu'il s'agit d'une méthode the type "S3" (permettant de voir alors toutes les analyses sur lesquelles la commande **summary( )** s'applique par défaut dans R):

```
> methods(summary)
 [1] summary.aov                  summary.aovlist*
 [3] summary.aspell*              summary.check_packages_in_dir*
 [5] summary.connection           summary.data.frame
 [7] summary.Date                 summary.default
 [9] summary.ecdf*                summary.factor
[11] summary.glm                  summary.infl*
[13] summary.lm                   summary.loess*
[15] summary.manova               summary.matrix
[17] summary.mlm*                 summary.nls*
[19] summary.packageStatus*       summary.PDF_Dictionary*
[21] summary.PDF_Stream*          summary.POSIXct
[23] summary.POSIXlt              summary.ppr*
[25] summary.prcomp*              summary.princomp*
[27] summary.proc_time            summary.srcfile
[29] summary.srcref               summary.stepfun
[31] summary.stl*                 summary.table
[33] summary.tukeysmooth*
see '?methods' for accessing help and source code
> 
```

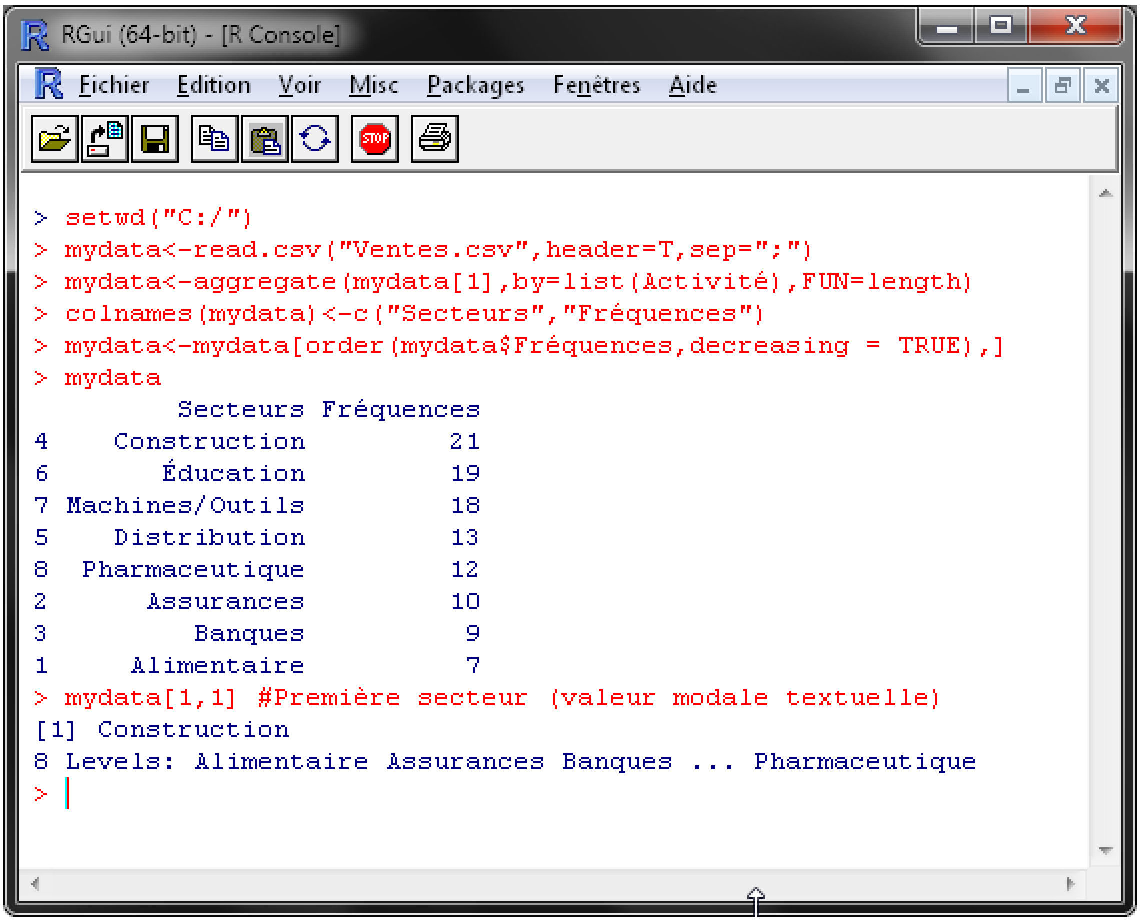
ou encore avec les commandes **unique( )** et **length( )** :



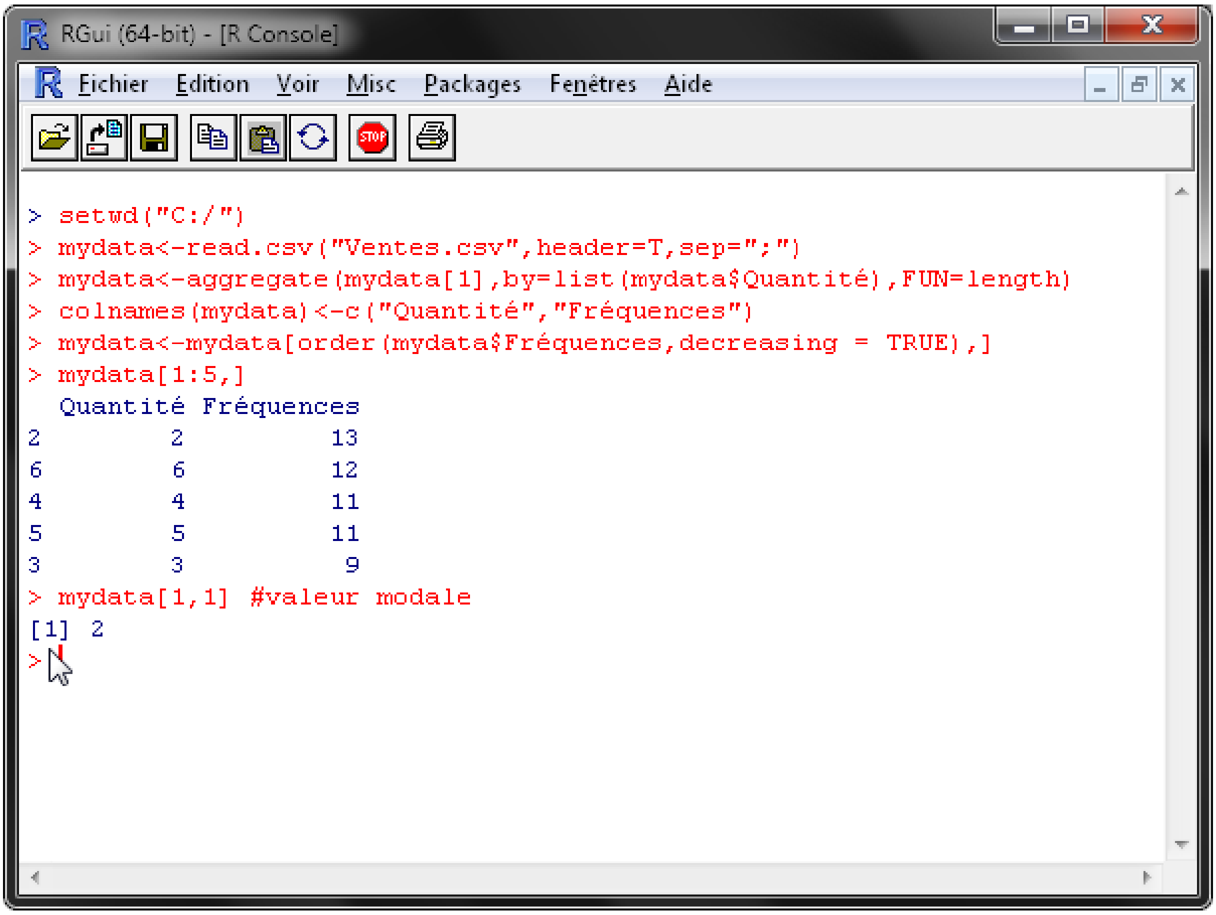
ou encore:



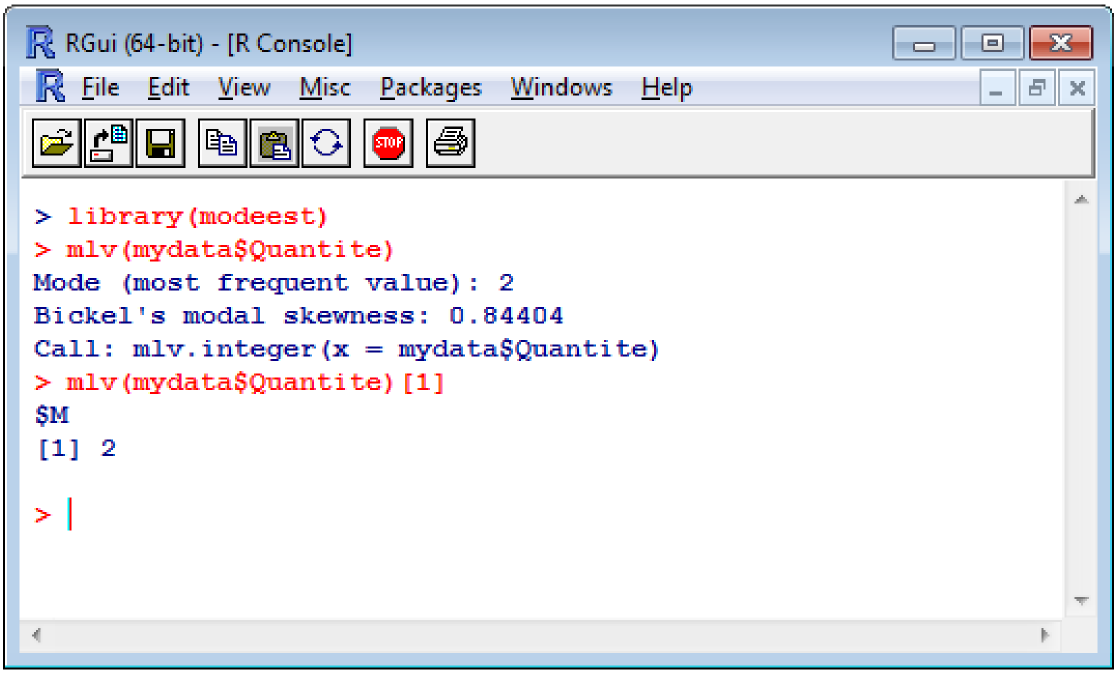
ou encore pour avoir la catégorie modale textuelle (il existe de nombreuses techniques) en d'autres termes… une table des fréquences:



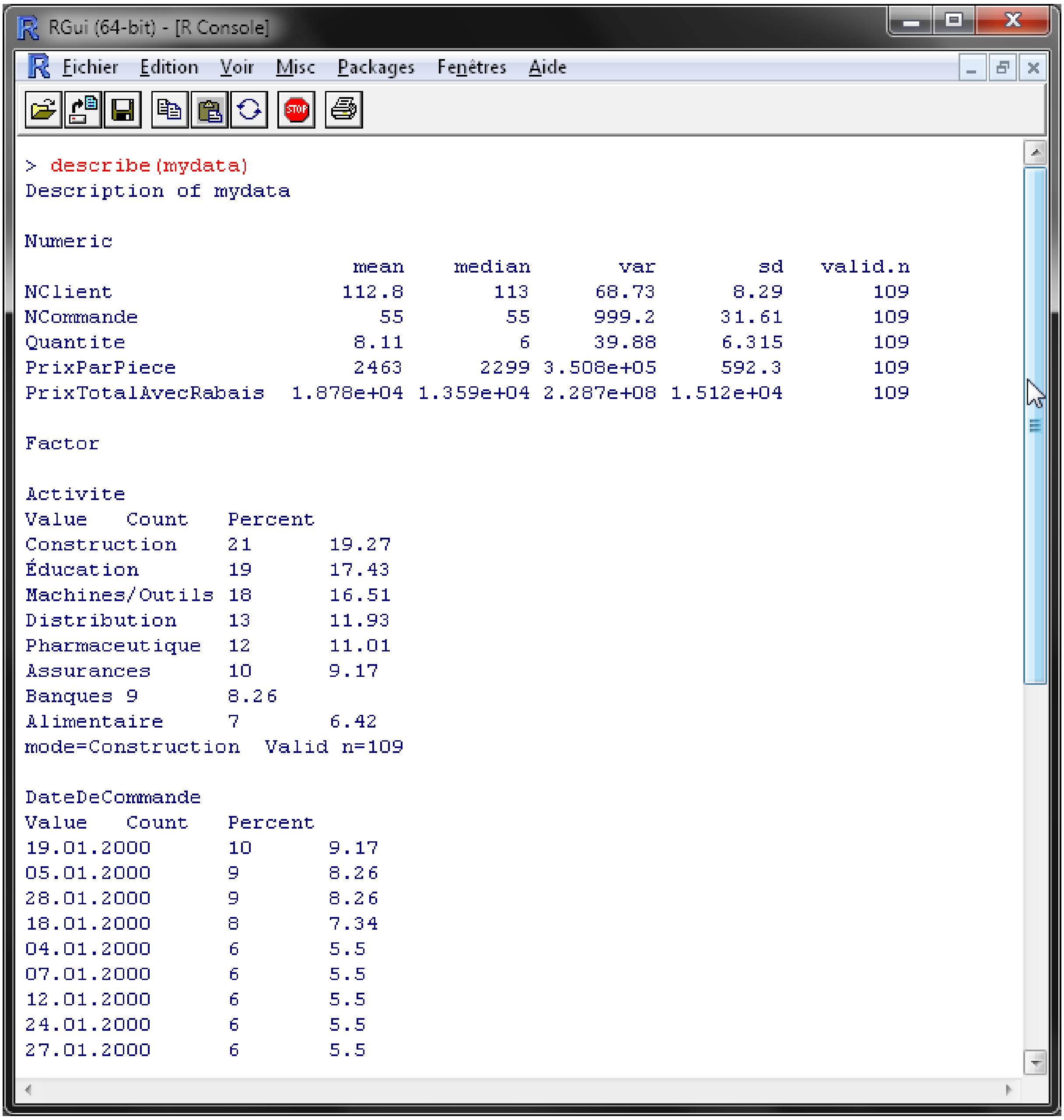
et pour des valeurs numériques l'idée est la même:



Ou en passant par la fonction **mlv( )** du package **modeest**:



Il y a aussi la commande **describe( )** du package **psych** qui permet de se faire une idée du contenu d'un data frame:




```
> describe(mydata)
Description of mydata

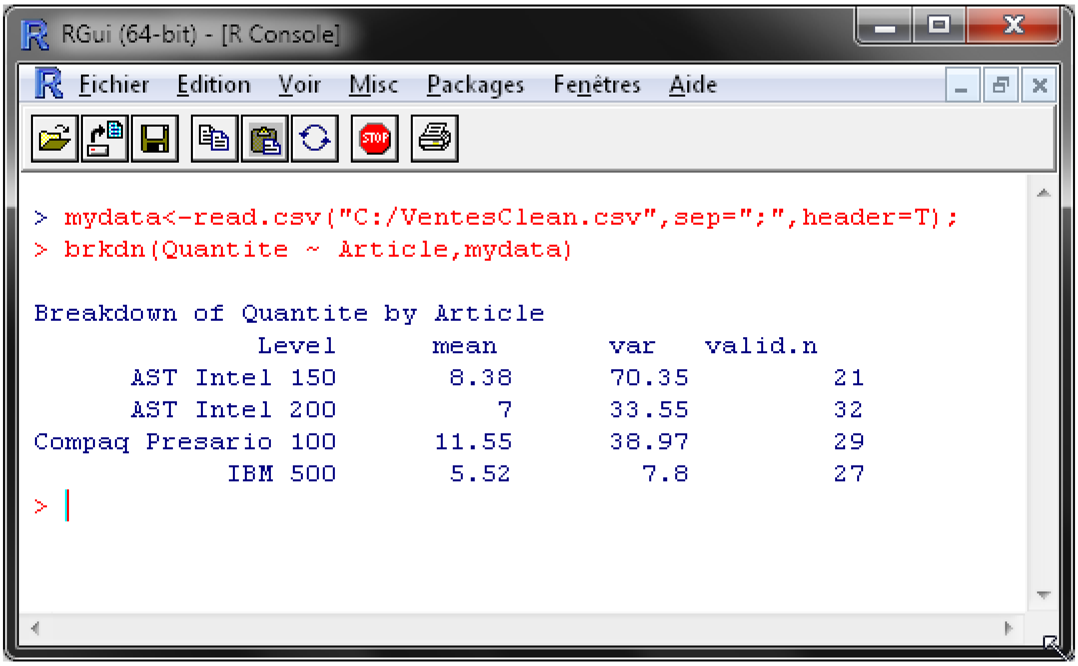
Numeric
                          mean    median       var        sd   valid.n
NClient                  112.8       113     68.73      8.29       109
NCommande                   55        55     999.2     31.61       109
Quantite                  8.11         6     39.88     6.315       109
PrixParPiece              2463      2299 3.508e+05     592.3       109
PrixTotalAvecRabais  1.878e+04 1.359e+04 2.287e+08 1.512e+04       109

Factor

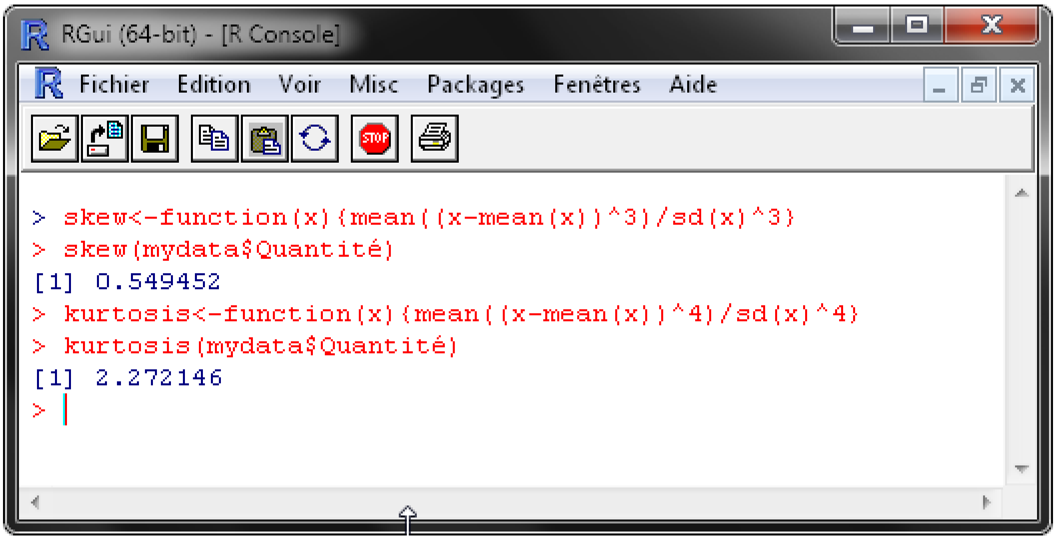
Activite
Value     Count    Percent
Construction    21       19.27
Éducation       19       17.43
Machines/Outils 18       16.51
Distribution    13       11.93
Pharmaceutique  12       11.01
Assurances      10       9.17
Banques 9       8.26
Alimentaire     7        6.42
mode=Construction  Valid n=109

DateDeCommande
Value     Count    Percent
19.01.2000      10       9.17
05.01.2000      9        8.26
28.01.2000      9        8.26
18.01.2000      8        7.34
04.01.2000      6        5.5
07.01.2000      6        5.5
12.01.2000      6        5.5
24.01.2000      6        5.5
27.01.2000      6        5.5
```

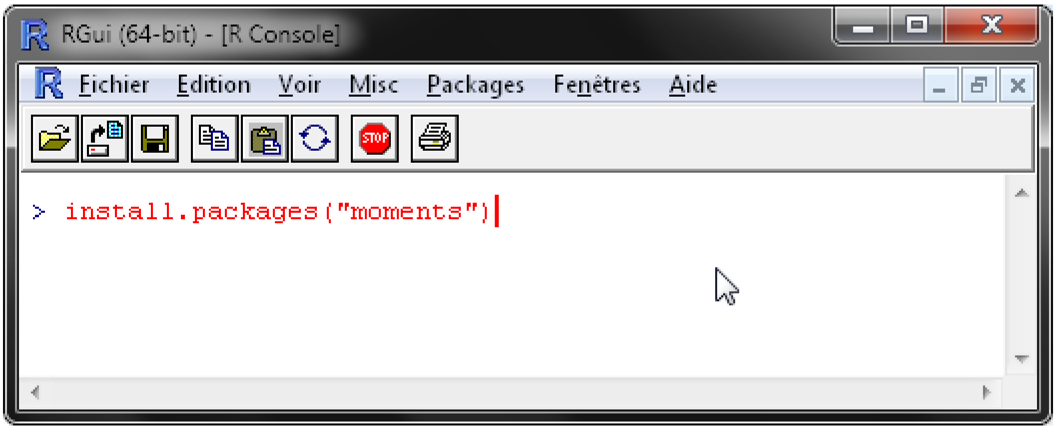
Il y a aussi la commande native **brkdn( )** qui signifie "breakdown" qui donne quelques informations triviales:



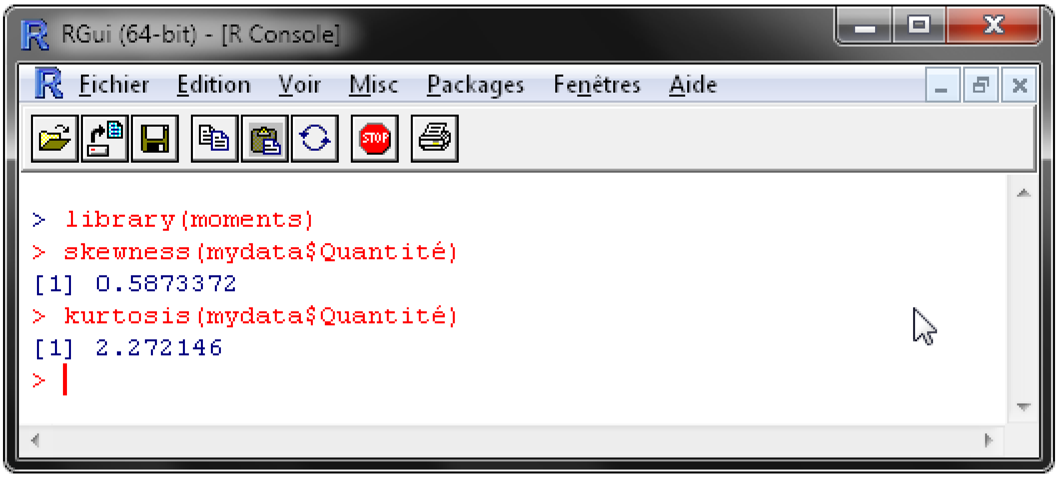
Comme la skewness (coefficient d'asymétrie) et le kurtosis (coefficient d'aplatissement) ne sont pas intégrés dans R, voyons comment les calculer en évitant d'installer des packages pour ce genre de détails:



Attention, il y a plusieurs manières de calculer la skewness et le kurtosis (en prenant la variance biaisée par exemple au lieu de la non biaisée ou encore simplement en se basant sur d'autres définitions). Par exemple, en installant le package **moments**:

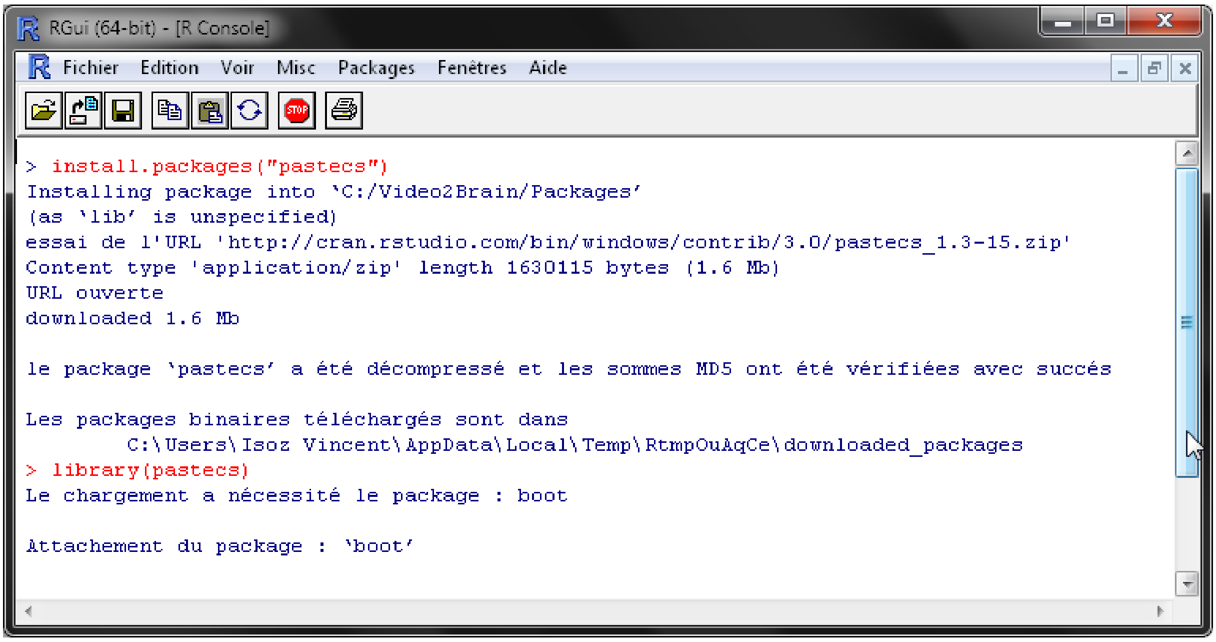


Nous avons alors:

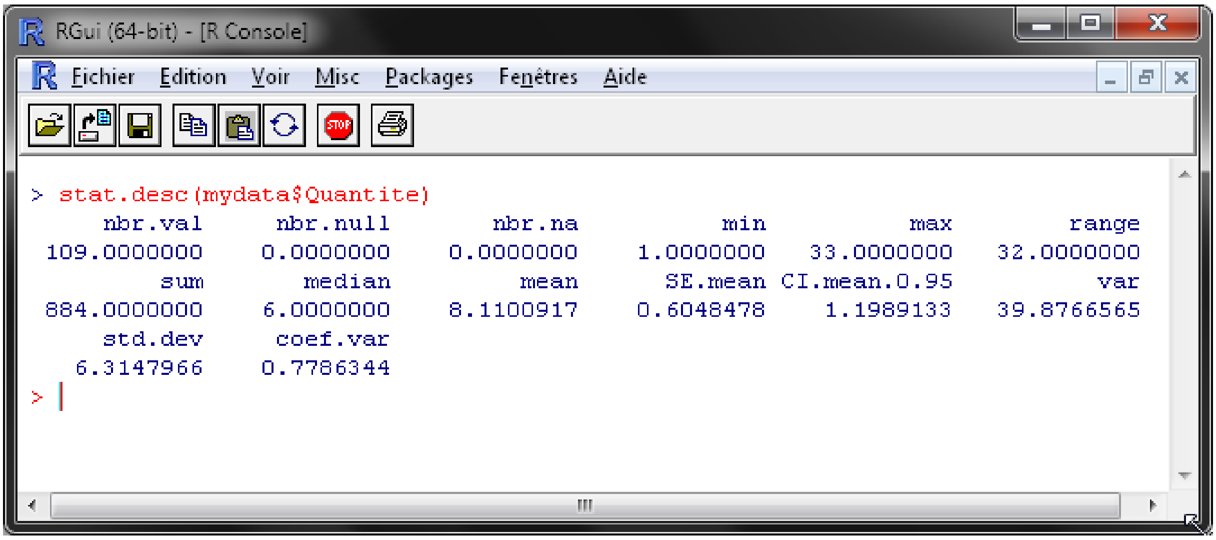


De même avec la package **PerformanceAnalytics**, etc.

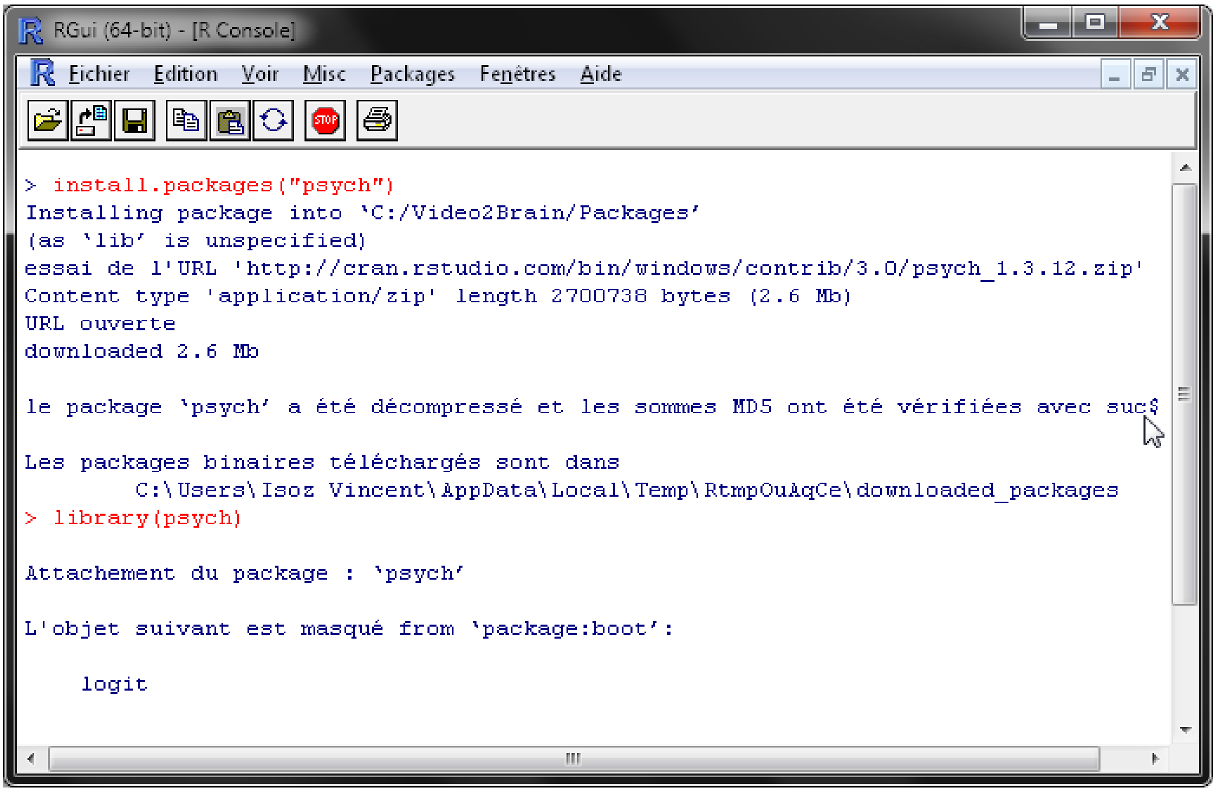
Sinon, toujours en restant dans le domaine des packages, nous avons en utilisant le package **pastecs**:



nous avons alors en utilisant la commande **stat.desc( )** :



ou encore avec le package **psych**:



nous avons en utilisant la commande **describeBy( )** :

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> describeBy(mydata$Quantite,mydata$Article)
group: AST Intel 150
  var  n mean   sd median trimmed  mad min max range skew kurtosis   se
1   1 21 8.38 8.39      5    6.82 4.45   1  33    32 1.53     1.48 1.83
------------------------------------------------------------
group: AST Intel 200
  var  n mean   sd median trimmed  mad min max range skew kurtosis   se
1   1 32    7 5.79    5.5    5.85 3.71   1  23    22  1.6     1.83 1.02
------------------------------------------------------------
group: Compaq Presario 100
  var  n  mean   sd median trimmed  mad min max range skew kurtosis   se
1   1 29 11.55 6.24     11   10.88 5.93   4  31    27 1.14     1.23 1.16
------------------------------------------------------------
group: IBM 500
  var  n mean   sd median trimmed  mad min max range skew kurtosis   se
1   1 27 5.52 2.79      5     5.3 2.97   2  12    10 0.59    -0.52 0.54
>
```

ou encore avec la fonction **summary_stats( )** du package **descriptr**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("descriptr")
> summary_stats(mydata$Quantité)
                        Univariate Analysis

 N                        109.00      Variance              39.88
 Missing                    0.00      Std Deviation          6.31
 Mean                       8.11      Range                 32.00
 Median                     6.00      Interquartile Range    7.00
 Mode                       2.00      Uncorrected SS     11476.00
 Trimmed Mean               7.49      Corrected SS        4306.68
 Skewness                   1.70      Coeff Variation       77.86
 Kurtosis                   3.14      Std Error Mean         0.60

                              Quantiles

              Quantile                           Value

             Max                                 33.00
             99%                                 30.44
             95%                                 15.20
             90%                                 22.00
             Q3                                  11.00
             Median                               6.00
             Q1                                   4.00
             10%                                  2.00
             5%                                   2.00
             1%                                   1.08
             Min                                  1.00

                            Extreme Values

                Low                               High

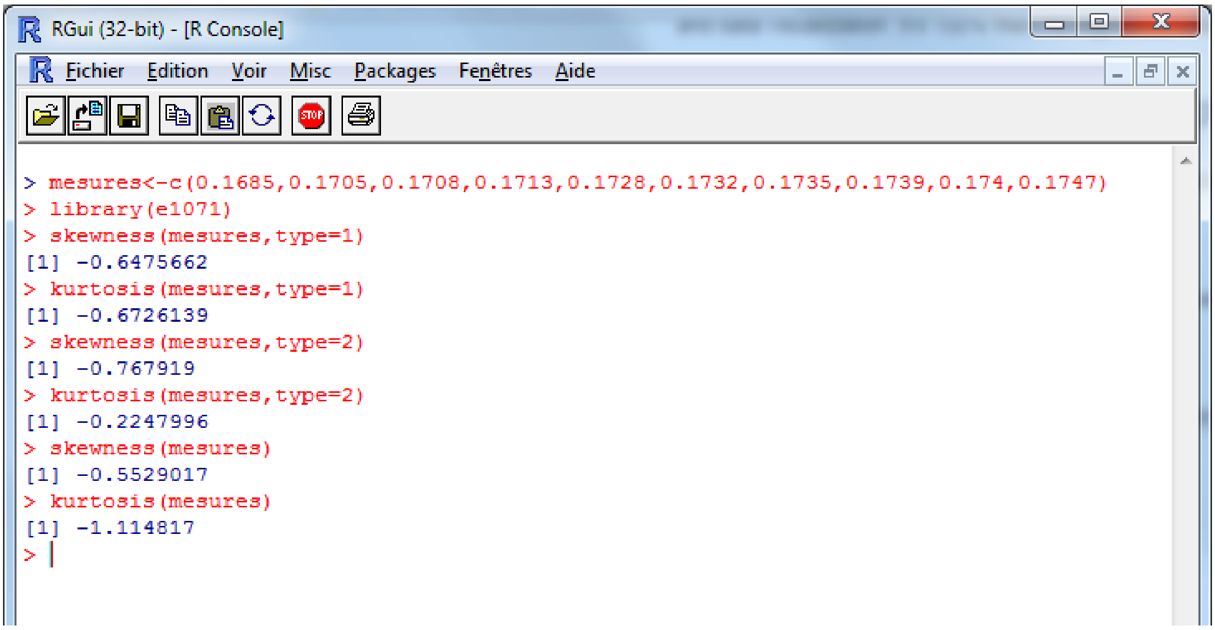
  Obs                        Value      Obs                        Value
  41                           1        75                           33
  46                           1        99                           31
   2                           2        26                           24
   6                           2        22                           23
   7                           2        37                           23
>
```

Exemple 13.: Kurtosis (aplatissement) et Skewness (asymétrie)

R 3.1.1

Comme il y a un grand nombre de méthodes pour calculer le Kurtosis (aplatissement) et le Skewness[4] (asymétrie), je tiens toujours à les traiter à part (même si nous les avons abordés plus haut très naïvement!).

Considérons l'exemple suivant utilisant les fonctions **kurtosis( )** et **skewness( )** du package **e1071** (il y a aussi le package **moment** qui contient les mêmes fonctions):



Nous conseillons vivement au lecteur de se concentrer sur le **type 2**. Effectivement, à ce moment là, nous retrouvons les mêmes valeurs que Minitab, Excel et SPSS.

Dans R, nous aurons alors les règles suivantes:

- Si le **skewness** est nul, la distribution est symétrique (cela ne signifiant pas que la symétrie a lieu sur un pic de la distribution car dans le cas bimodal l'axe de symétrie peut-être entre les deux valeurs modales symétriques). Si le skewness est positif, la distribution (la valeur modale/médiane) est penchée à droite (ou il y a des valeurs extrêmes à droite). Si le skewness est négatif, la distribution (la valeur modale/médiane) est penchée à gauche (ou il y a des valeurs extrêmes à gauche)

- Si le **kurtosis** est nul (mesokurtique) alors l'aplatissement est similaire à celui d'une distribution Normale. Si la valeur d'excès est supérieure à 0 alors la distribution d'intérêt (leptokurtique) est alors plus haute que celle d'une distribution Normale à moyenne égale (inversement - platikurtique - si l'excès de kurtosis est négatif bien évidemment)

[4] Au moins 11 méthodes de calculer le Skewness. Voir l'étude de Tabor, J. (2010), "Investigating the Investigative Task: Testing for Skewness - An Investigation of Different Test Statistics and their Power to Detect Skewness," Journal of Statistics Education, 18, 1-13. www.amstat.org/publications/jse/v18n2/tabor.pdf

Sinon à titre de comparaison (le premier est appelé "coefficient d'asymétrie de Fisher-Pearson" et le seconde " coefficient d'asymétrie ajusté et standardisé de Fisher-Pearson"):

$$\gamma_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}\right)^3}$$

$$\gamma_2 = \gamma_1\sqrt{\frac{n}{(n-1)(n-2)}}$$

$$\gamma_3 = \gamma_1\left(\frac{n-1}{n}\right)^{3/2}$$

et:

$$\kappa_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^4}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}\right)^4}$$

$$\kappa_2 = \frac{(N-1)\left((N+1)\kappa_1 - 3(N+1)\right)}{(N-2)(N-3)} + 3$$

$$\kappa_3 = \frac{N(N+1)}{(N-1)(N-2)(N-3)}\kappa_1 - \frac{3(N-1)^2}{(N-2)(N-3)}$$

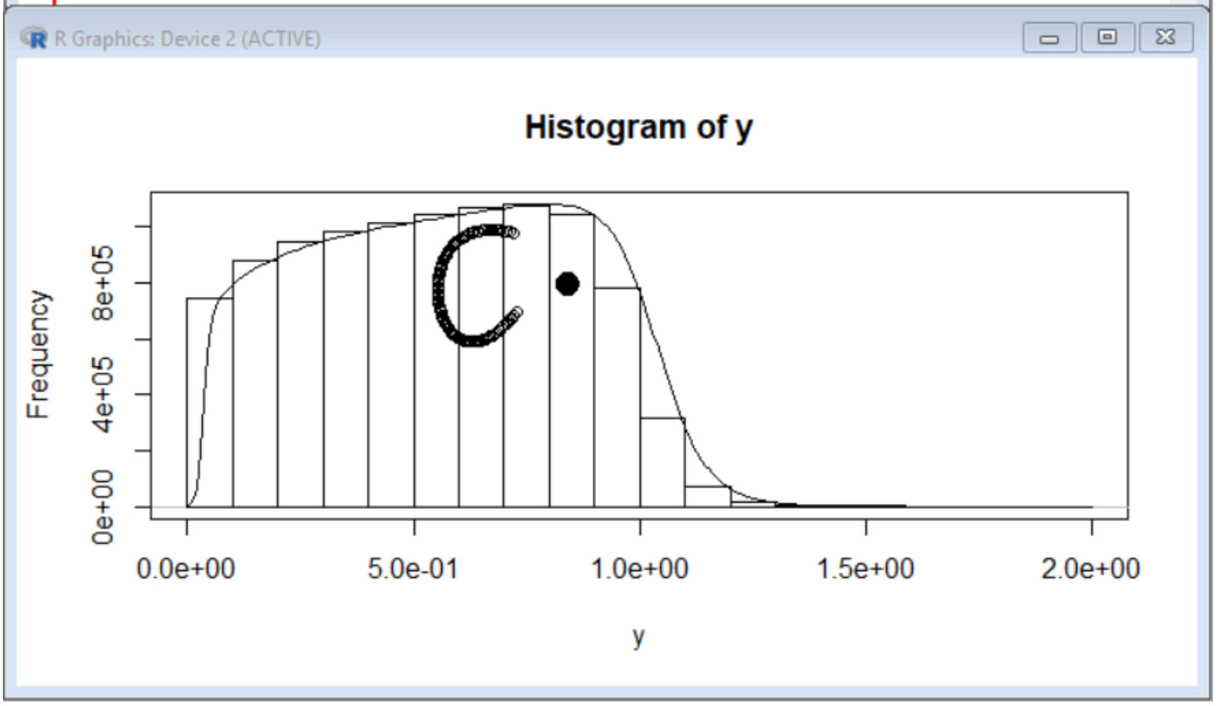
$$\kappa_4 = \kappa_1 - 3$$

$$\kappa_5 = \left(\kappa_4 + 3\right)\left(1 - \frac{1}{N}\right)^2 - 3$$

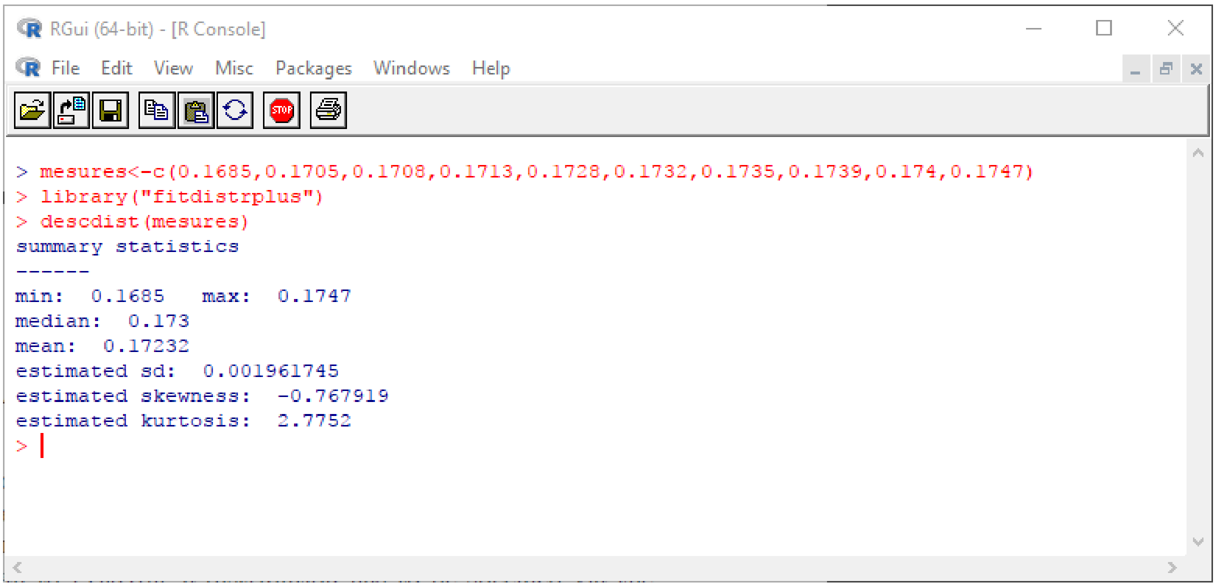
Skewness	R	$-0.647(\gamma_1)$,$-0.77(\gamma_2)$, $-0.552(\gamma_3)$
	MATLAB	$-0.647(\gamma_1)$,$-0.77(\gamma_2)$
	Mathematica	$-0.647(\gamma_1)$
	Minitab, SPSS, Excel	-0.77 (γ_3)
Kurtosis	R	$-1.114(\kappa_5)$,-0.22 (κ_3),-0.67 (κ_4)
	MATLAB	2.327 (κ_1),$2.775(\kappa_2)$
	Mathematica	$2.327(\kappa_1)$
	Minitab, SPSS, Excel	-0.22 (κ_3)

Voici sinon l'exemple utilisant la fonction de distribution de Dagum (appelée aussi "fonction éléphant") dont nous avons fait mention dans le cours théorique (fonction non symétrique mais avec Skewness nul):

```
> library("VGAM")
> library(knotR)
> y<-rdagum(n = 10000000, scale = 1, shape1 = 18.89306, shape2 = 0.05975542)
> plot(density(y),main="",xlab="",ylab="",yaxt="n",xaxt="n")
> par(new=TRUE)
> ear.x <- c(0.686, 0.501, 0.42, 0.68)
> ear.y <- c(0.698, 0.315, 1.095, 0.983)
> myseg(bezier(cbind(ear.x, ear.y)))
> par(new=TRUE)
> points(0.8,0.8, pch=19, cex=2)
> par(new=TRUE)
> hist(y)
> library("moments")
> skewness(y)
[1] 1.908e-04
```



Pour ceux qui veulent aller plus loin dans le diagnostic, toujous avec notre jeu de données initial, en utilisant la fonction **descidist( )** du package **fitdistrplus**, nous avons:



Ce qui donne comme graphique de diagnostique:

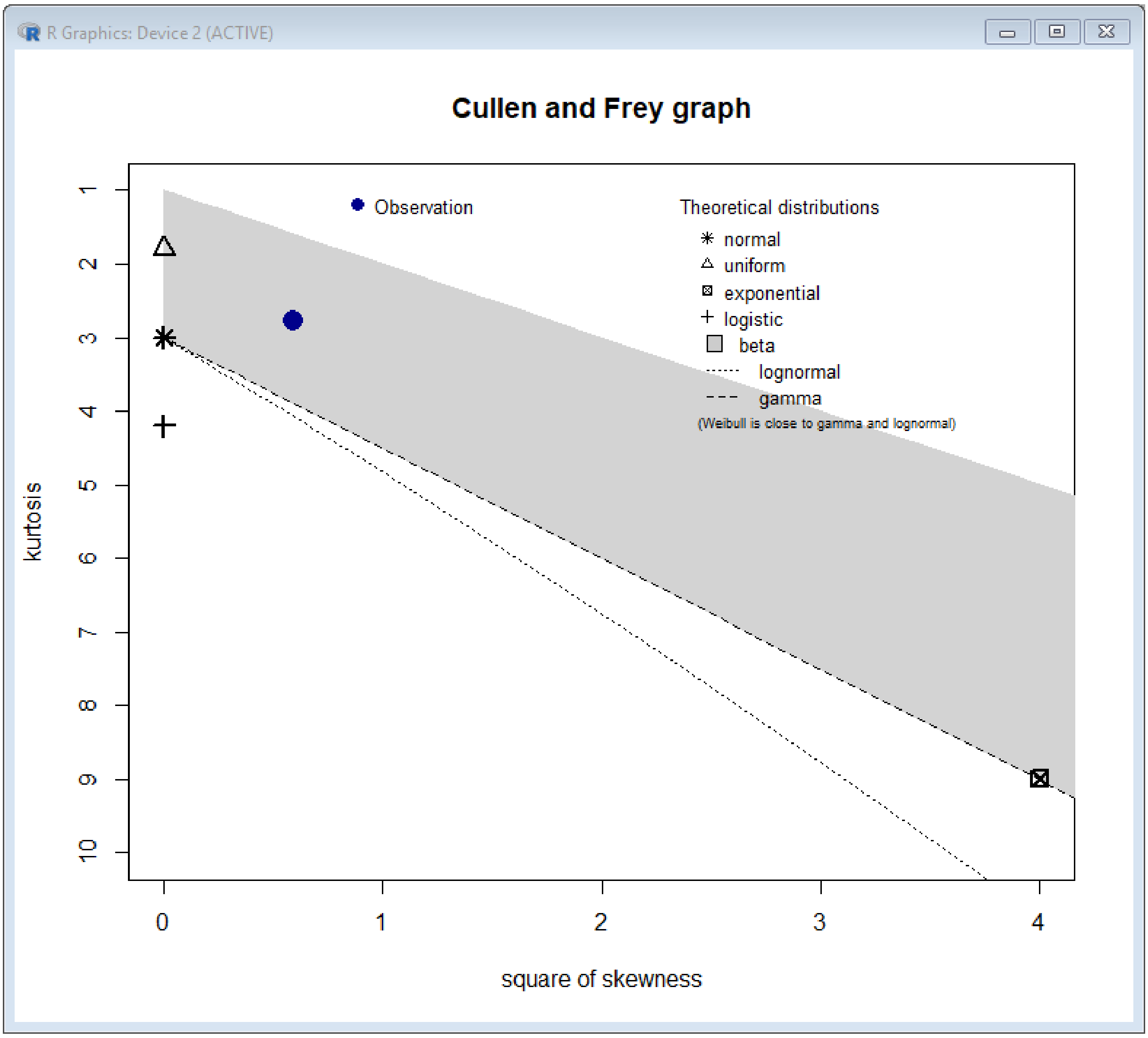


Figure 2 Graph de Cullen et Frey

Exemple 14.: Statistiques descriptives élémentaires multivariées de data frame

R 3.0.2

Commençons par deux grands classiques en utilisant les données suivantes utilisées dans le cours théorique:

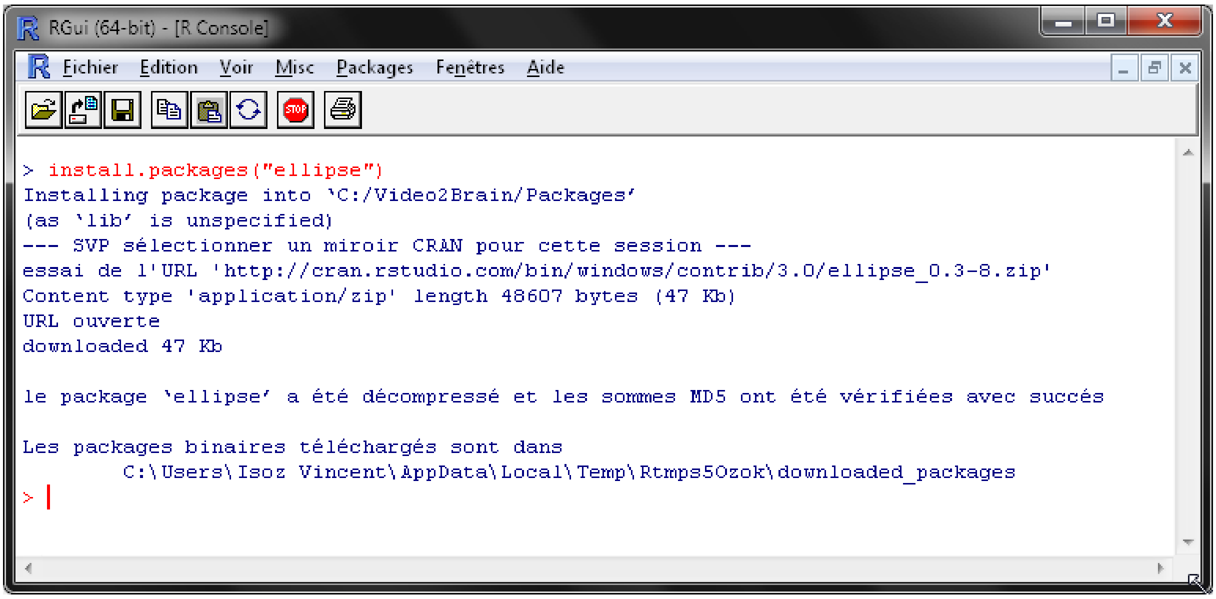
```
VariationsTitres.csv - Bloc-notes
Fichier  Edition  Format  Affichage  ?
Crédit Lyonnais;France Télécom;Lafarge;Saint-Gobain;Total Fina Elf
 -0.017516 ; -0.328775 ; 0.023053 ; 0.003086 ; 0.000615
 -0.198426 ; -0.020374 ; -0.104156 ; -0.106509 ; -0.093423
 0.122761 ; 0.197863 ; 0.103410 ; 0.023046 ; 0.057627
 -0.034988 ; 0.063419 ; 0.038501 ; 0.054376 ; -0.086538
 -0.000267 ; 0.018141 ; 0.023415 ; 0.040521 ; 0.125614
 -0.002667 ; -0.172160 ; -0.060057 ; -0.023602 ; 0.015586
 0.021390 ; -0.180791 ; 0.009128 ; 0.084602 ; 0.042971
 0.142932 ; 0.153366 ; 0.030151 ; 0.042345 ; 0.041789
 0.072148 ; -0.232346 ; 0.027317 ; 0.015608 ; -0.049718
 -0.034822 ; -0.229599 ; 0.037037 ; 0; -0.007729
 -0.039398 ; -0.545980 ; -0.075092 ; -0.043158 ; -0.014979
 -0.082719 ; 0.558855 ; -0.110891 ; -0.329373 ; -0.102798
```

La matrice des variances-covariances s'obtiendra à l'aide de la commande **cov( )** et la matrice des coefficients de corrélations à l'aide de la commande **cor( )** :

```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide

> setwd("C:/")
> mydata<-read.csv("VariationsTitres.csv",header=T,sep=";")
> #matrice des variances-covariances non biaisée (donc avec le n-1 au dénominateur
> cov(mydata)
                Crédit.Lyonnais France.Télécom       Lafarge Saint.Gobain Total.Fina.Elf
Crédit.Lyonnais     0.008314481   0.0012552396  0.0044392901  0.005360491    0.003719678
France.Télécom      0.001255240   0.0829056374 -0.0007385008 -0.015753161   -0.003122089
Lafarge             0.004439290  -0.0007385008  0.0044068038  0.005371749    0.002415016
Saint.Gobain        0.005360491  -0.0157531613  0.0053717490  0.011983778    0.004456106
Total.Fina.Elf      0.003719678  -0.0031220887  0.0024150157  0.004456106    0.004732372
> #matrice des coefficients de corrélations (estimateur non biaisé aussi)
> cor(mydata)
                Crédit.Lyonnais France.Télécom     Lafarge Saint.Gobain Total.Fina.Elf
Crédit.Lyonnais      1.00000000     0.04780981  0.73338852    0.5370196      0.5929914
France.Télécom       0.04780981     1.00000000 -0.03863644   -0.4997799     -0.1576209
Lafarge              0.73338852    -0.03863644  1.00000000    0.7391915      0.5288331
Saint.Gobain         0.53701959    -0.49977992  0.73919150    1.0000000      0.5917240
Total.Fina.Elf       0.59299138    -0.15762088  0.52883313    0.5917240      1.0000000
> 
```

Si on installe le package **ellipse**:



Nous pouvons ensuite représenter l'une ou l'autre des ces deux matrices sous la forme des ellipses de corrélation de la loi Normale bivariée avec la commande **plotcorr( )** du package **ellipse**:

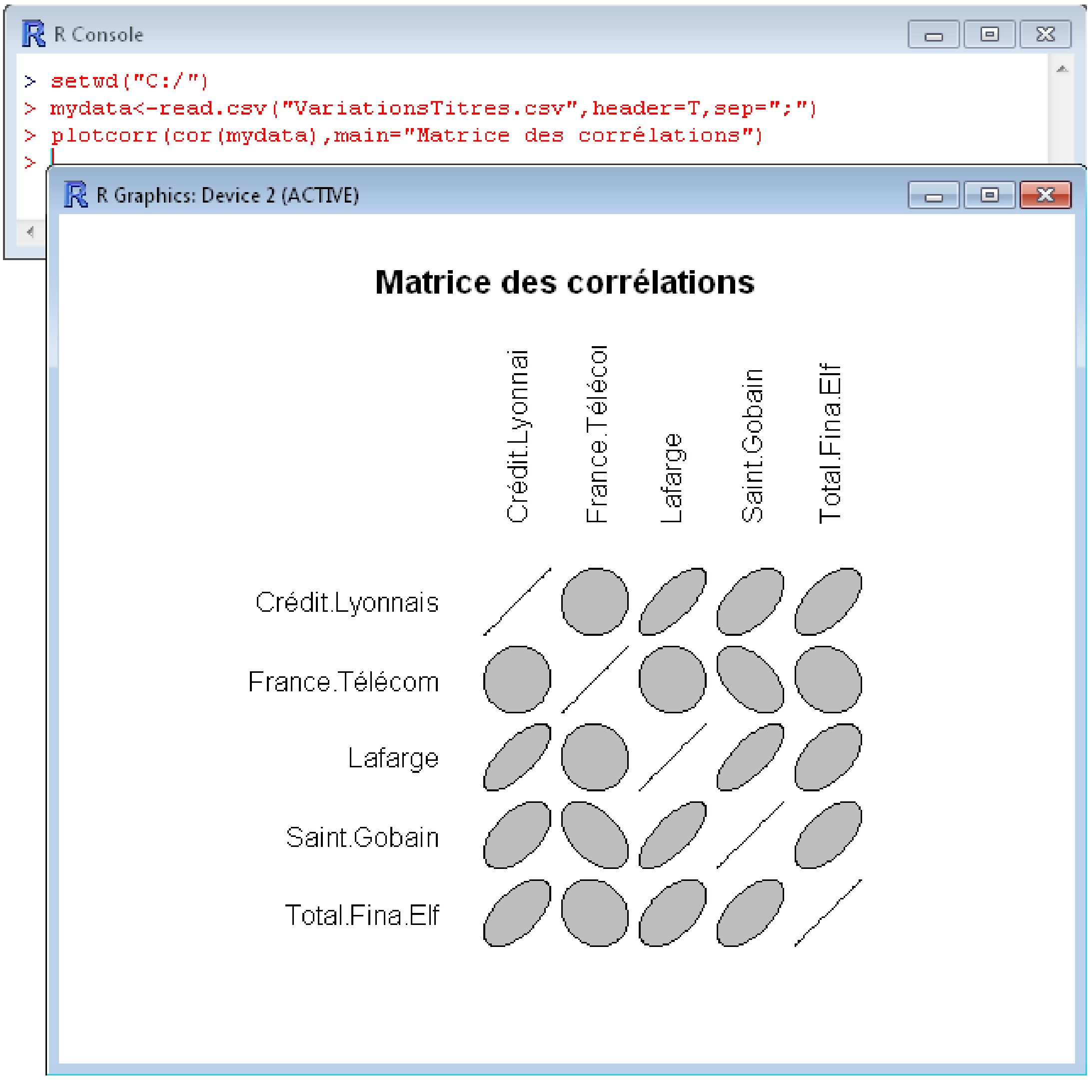


Figure 3 Graph de corrélation avec orientations

ou avec les packages **ggplot2** et **reshape2** nous pouvons construire une heatmap de corrélations avec les fonctions **qplot( )** et **melt( )** :

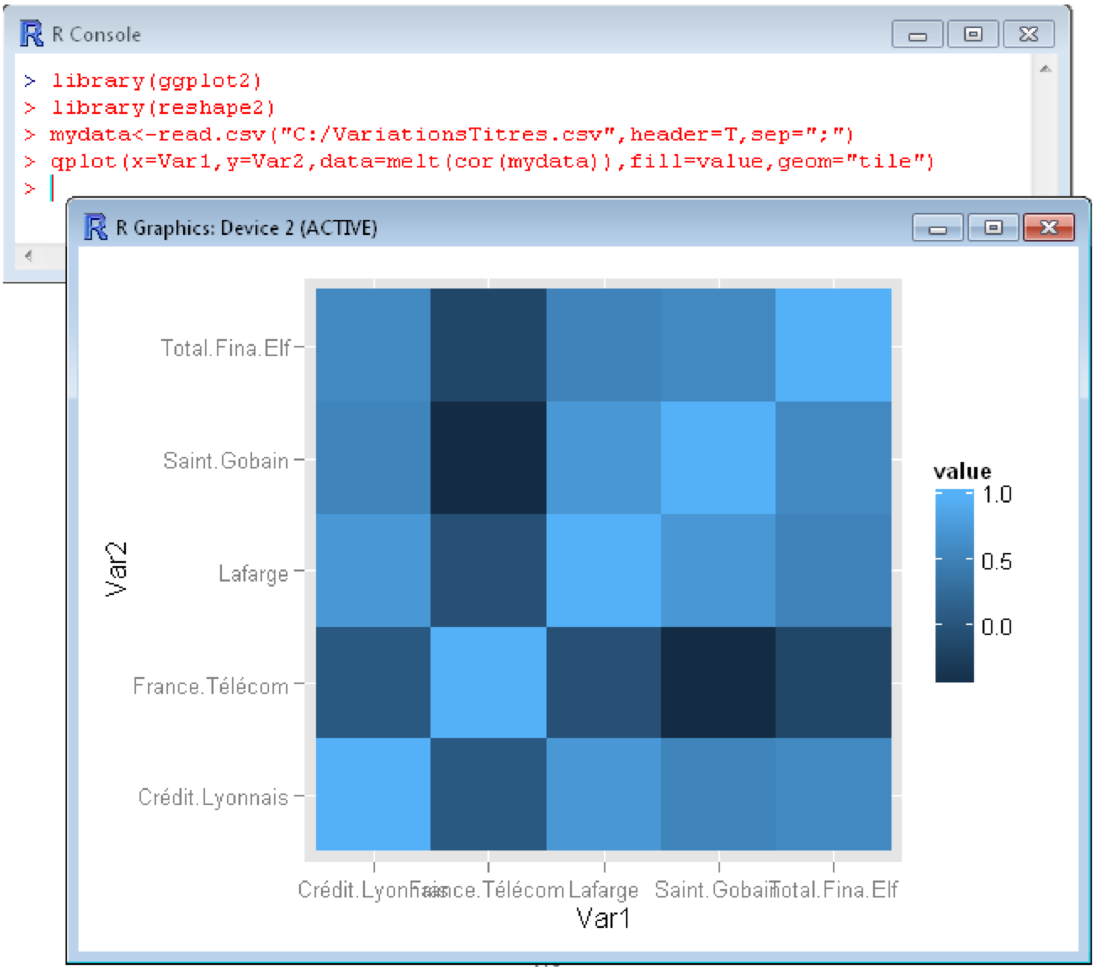
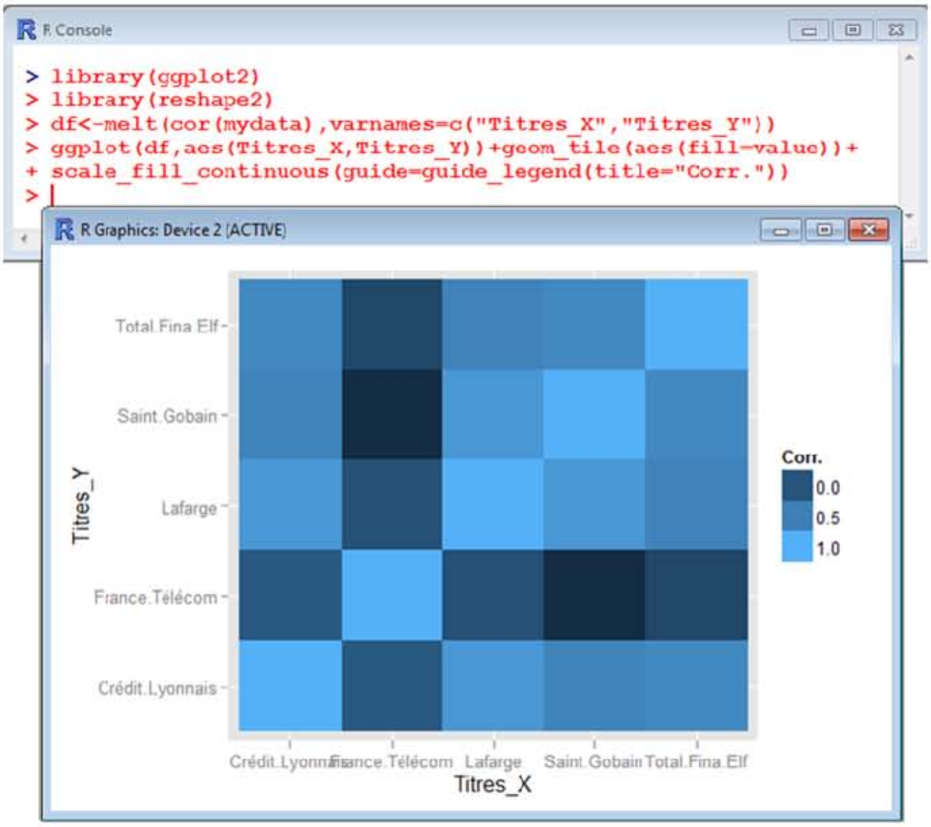


Figure 4 Graph matrice de corrélation sous forme de carte à chaleur

Au niveau des légendes ce n'est pas super. Nous pouvons faire mieux:


R Console
> library(ggplot2)
> library(reshape2)
> df<-melt(cor(mydata),varnames=c("Titres_X","Titres_Y"))
> ggplot(df,aes(Titres_X,Titres_Y))+geom_tile(aes(fill=value))+
+ scale_fill_continuous(guide=guide_legend(title="Corr."))
R Graphics: Device 2 (ACTIVE)
Total.Fina.Elf
Saint.Gobain
Lafarge
France.Télécom
Crédit.Lyonnais
Titres_Y
Crédit.Lyonnais
France.Télécom
Lafarge
Saint.Gobain
Total.Fina.Elf
Titres_X
Corr.
0.0
0.5
1.0

7. Manipulation de variables de type algébriques, vectorielles, matricielles, textes, dates temps

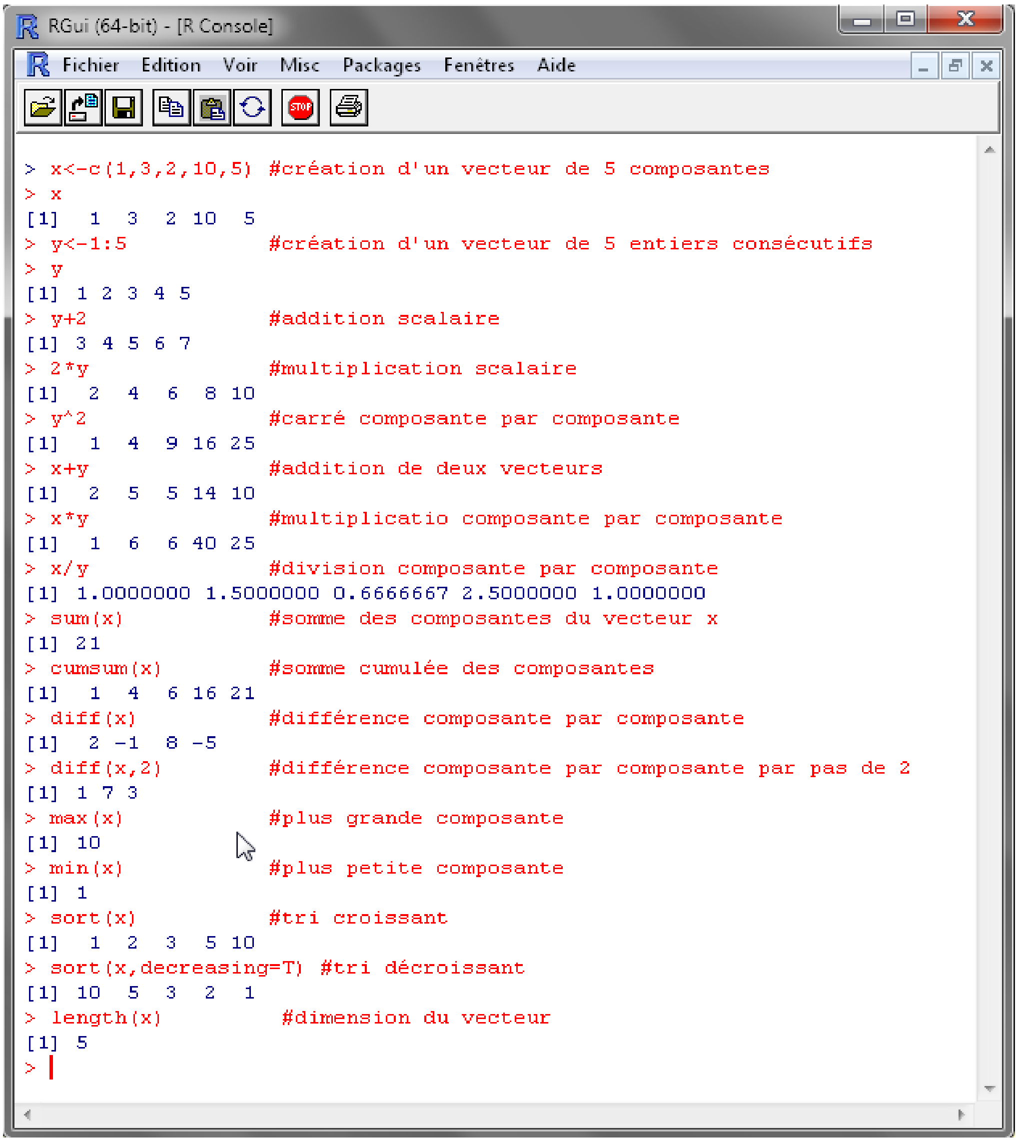
Contenu

Exemple 15.: Manipulations de variables vectorielles

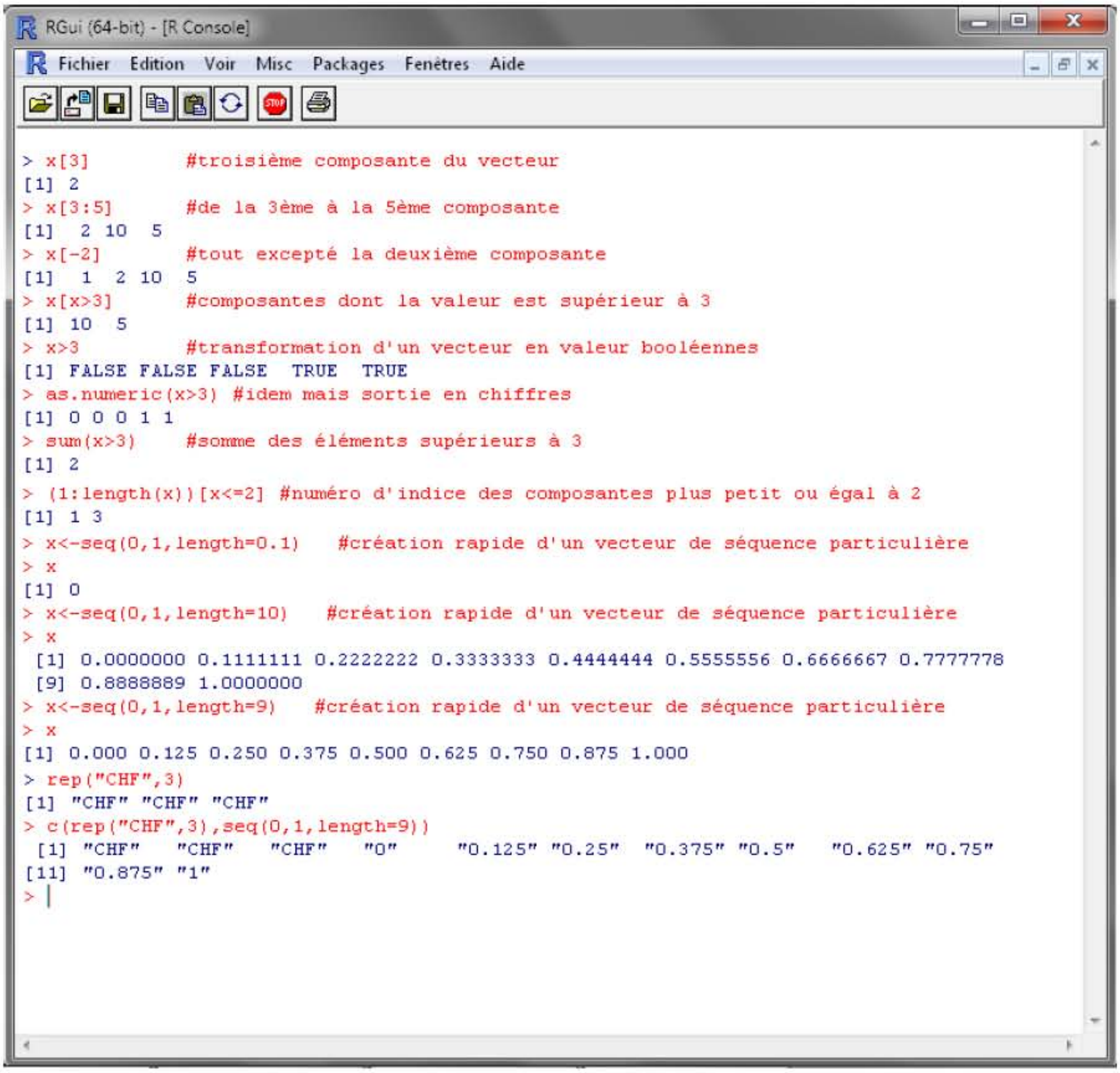
R 3.0.2

Le calcul vectoriel/matriciel a une importance majeure en statistiques et en finance. Il convient donc de voir si les opérations de base sont bien présentes avant de poursuivre et surtout (je le répète) de pouvoir faire plus tard du scripting.

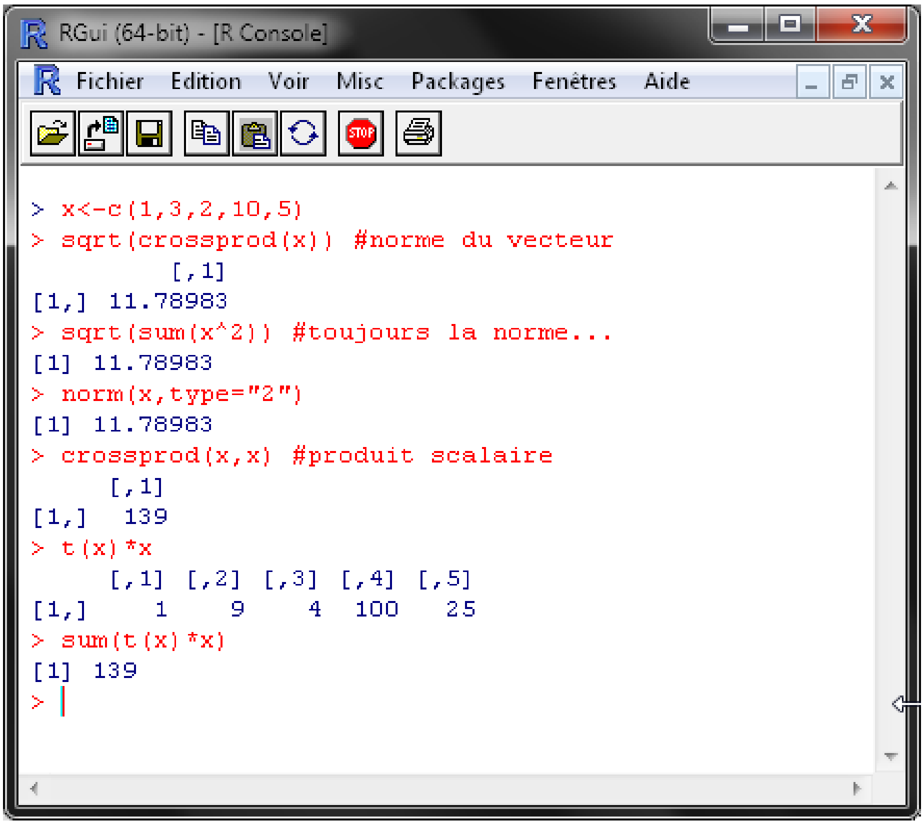
Voyons les cas classiques que l'on retrouvera énormément de fois même en dehors du scripting (gestion de data frame, génération de graphiques, etc.). Voici déjà une première série de commandes utiles utilisant les commandes **c( )** , **sum( )** , **cumsum( )** , **diff( )** , **max( )** , **min( )** , **sort( )** , **length( )** :



et une deuxième série utilisant les commandes **as.numeric( )** , **seq( )** et **rep( )** :

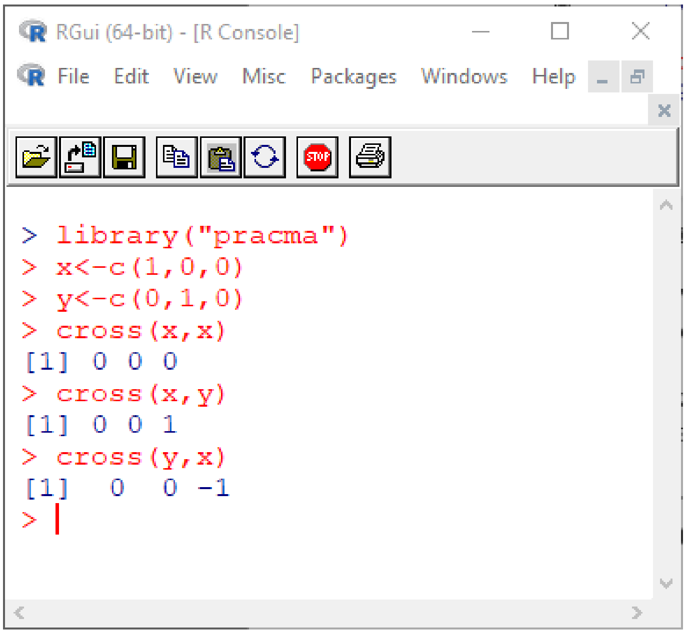

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> x[3]   #troisième composante du vecteur
[1] 2
> x[3:5]   #de la 3ème à la 5ème composante
[1] 2 10 5
> x[-2]   #tout excepté la deuxième composante
[1] 1 2 10 5
> x[x>3]   #composantes dont la valeur est supérieur à 3
[1] 10 5
> x>3   #transformation d'un vecteur en valeur booléennes
[1] FALSE FALSE FALSE TRUE TRUE
> as.numeric(x>3) #idem mais sortie en chiffres
[1] 0 0 0 1 1
> sum(x>3)   #somme des éléments supérieurs à 3
[1] 2
> (1:length(x))[x<=2] #numéro d'indice des composantes plus petit ou égal à 2
[1] 1 3
> x<-seq(0,1,length=0.1)   #création rapide d'un vecteur de séquence particulière
> x
[1] 0
> x<-seq(0,1,length=10)   #création rapide d'un vecteur de séquence particulière
> x
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667 0.7777778
 [9] 0.8888889 1.0000000
> x<-seq(0,1,length=9)   #création rapide d'un vecteur de séquence particulière
> x
[1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
> rep("CHF",3)
[1] "CHF" "CHF" "CHF"
> c(rep("CHF",3),seq(0,1,length=9))
 [1] "CHF"   "CHF"   "CHF"   "0"   "0.125" "0.25" "0.375" "0.5"   "0.625" "0.75"
[11] "0.875" "1"
>

et enfin le calcul de norme et du produit scalaire avec les commandes **crossprod( )** (oui c'est trompeur… **crossprod( )** est dans R, le produit d'un vecteur et de la transposée d'un autre vecteur et non pas le produit vectoriel que nous allons voir plus bas) **, norm( )** et avec la transposition **t( )** :

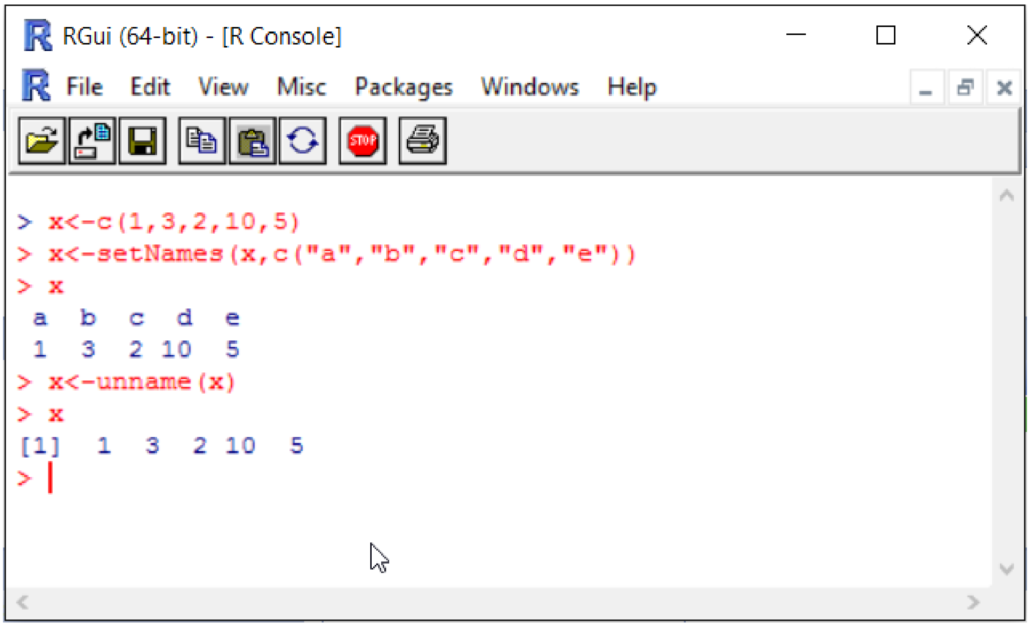


et donc nous ne pouvons curieusement pas obtenir directement le produit scalaire en multipliant la transposée d'un vecteur par un autre (dommage même si c'est un détail...).

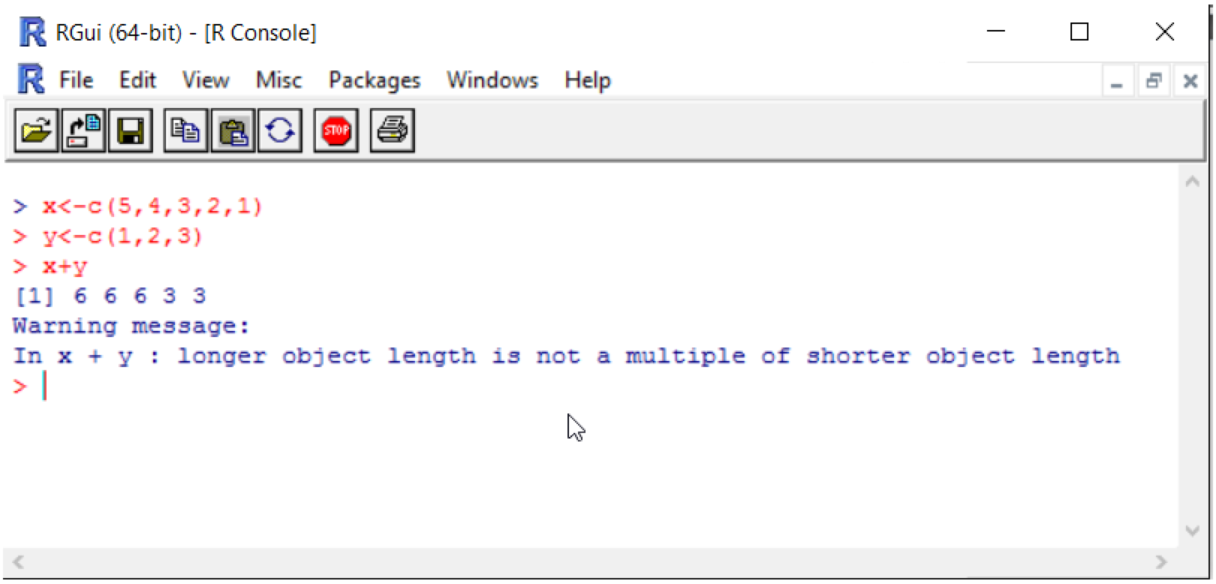
Maintenant voyons comme faire le produit vectoriel. Pour cela nous allons charger le package **pracma** et utiliser sa fonction **cross( )**:



Nous pouvons aussi nommer ou dénommer les composantes d'un vecteur si besoin avec les fonctions **setNames( )** ou **unname( )** :

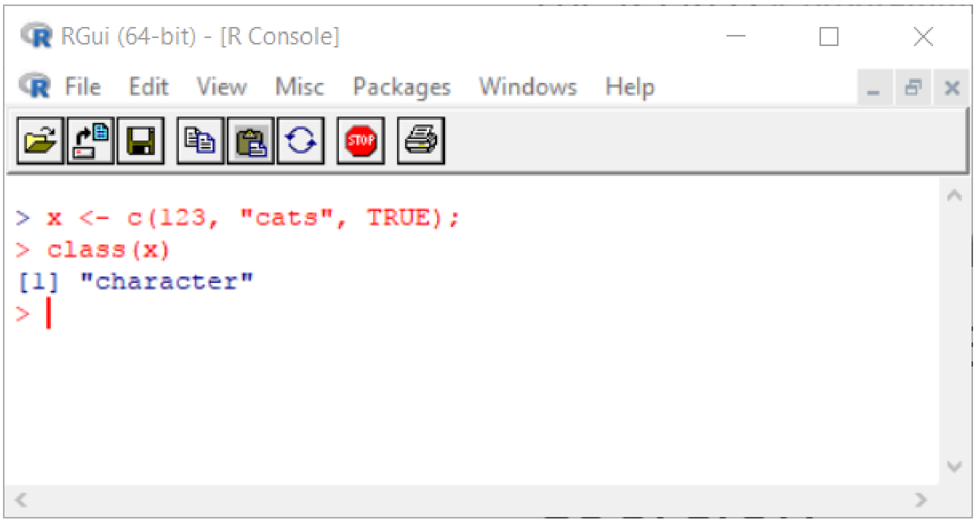


Attention!!! Lors de la somme de deux vecteurs de dimensions différentes, les développeurs de R ont fait un choix que nous trouvons dangereux! Voir l'exemple ci-dessous:



En d'autres termes, le vecteur y est "recyclé" autant de fois que nécessaire pour la somme se fasse!! Donc attention quand vous écrivez des programmes (scripts) de vérifier que les vecteurs aient les mêmes dimensions car R passe outre cette vérification contrairement à la majorité des logiciels de calculs scientifiques.

Relativement à une question que j'ai eue via Internet… considérons le vecteur suivant et sa classe:



Lorsqu'un vecteur contient un mélange de typage de données, R choisit le type qui semble le mien convenir. Quand cela arrive dans certains outils statistiques de R, un message indique souvent que le vecteur "*has been coerced into factors*" ce qui signifie qu'il a choisit de convertir la colonne en une colonne de type "facteurs". Dans la capture d'écran ci-dessus, R a choisit assez logiquement de convertir l'ensemble du vecteur en un vecteur de type caractères.

Voyons maintenant deux fonctions vectorielles particulières, appelées "maximum parallèle" et "minimum parallèle" et écrites **pmax( )** et **pmin( )**, utilisées surtout en finance quantitative (nous verrons une application dans la section concernant le pricing d'options vanille européennes via la méthode de Monte Carlo avec une réduction de la variance basée sur les variables antithétiques!):

Exemple 16.: Estimateurs robustes, M-estimateurs de localisation et d'échelle

R 3.0.2

Toujours dans les vecteurs regardons que la moyenne arithmétique **mean( )** et sa version élaguée soient bien présentes:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> mydata<-c(2,3,4,6,8,10,12,14,18,27)
> mean(mydata,trim=0.2)
[1] 9
> 
```

Ainsi que la classique et non moins importante moyenne pondérée[5] **weighted.mean( )** :


```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

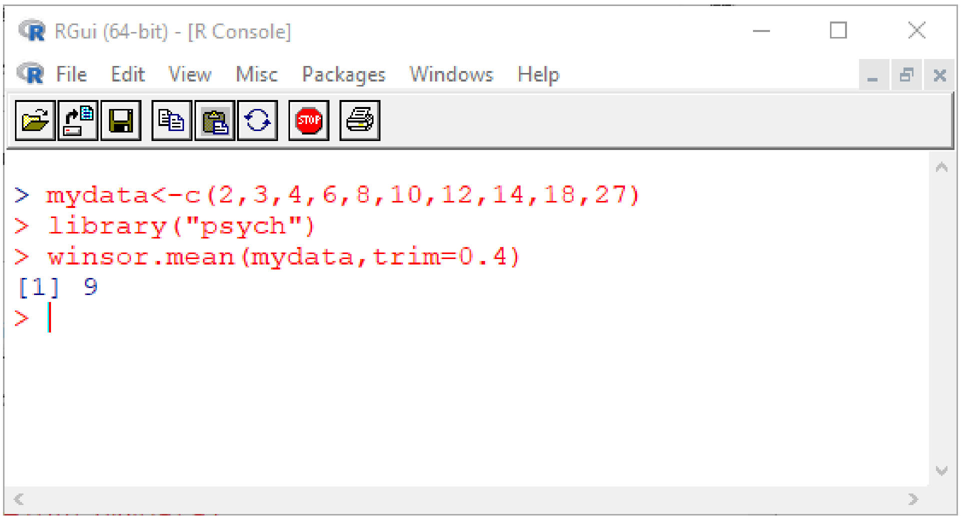
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> mydata<-aggregate(cbind(Quantité,Prix.total.avec.rabais)~Activité,data=mydata,sum,na.rm=TRUE)
> mydata
          Activité Quantité Prix.total.avec.rabais
1      Alimentaire       46               99929.55
2       Assurances       72              160881.35
3          Banques       74              151074.65
4     Construction      157              350426.74
5     Distribution      109              282693.40
6        Éducation      150              328367.95
7 Machines/Outils       138              335393.79
8   Pharmaceutique      138              338747.92
> #d'abord la moyenne arithmétique simple
> mean(mydata$Prix.total.avec.rabais)
[1] 255939.4
> #et donc la moyenne pondérée par les quantités
> weighted.mean(mydata$Prix.total.avec.rabais,mydata$Quantité)
[1] 289001.2
> 
```

Nous retrouvons bien la même valeur que dans le cours théorique et que dans SAS.

---

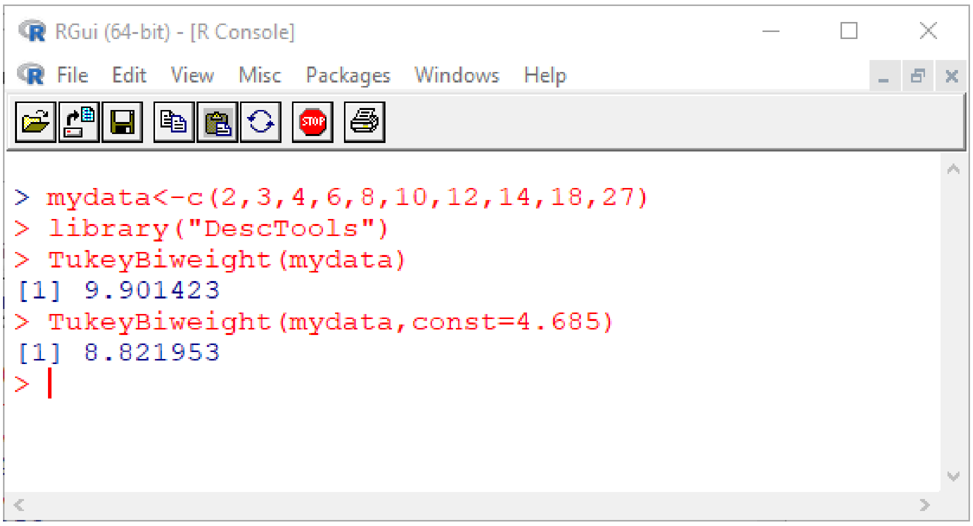
[5] Nous verrons la moyenne géométrique bien plus loin!

Voyons maintenant pour la moyenne Winsorisée en utilisant la fonction **winsor.mean( )** du package **psych**:



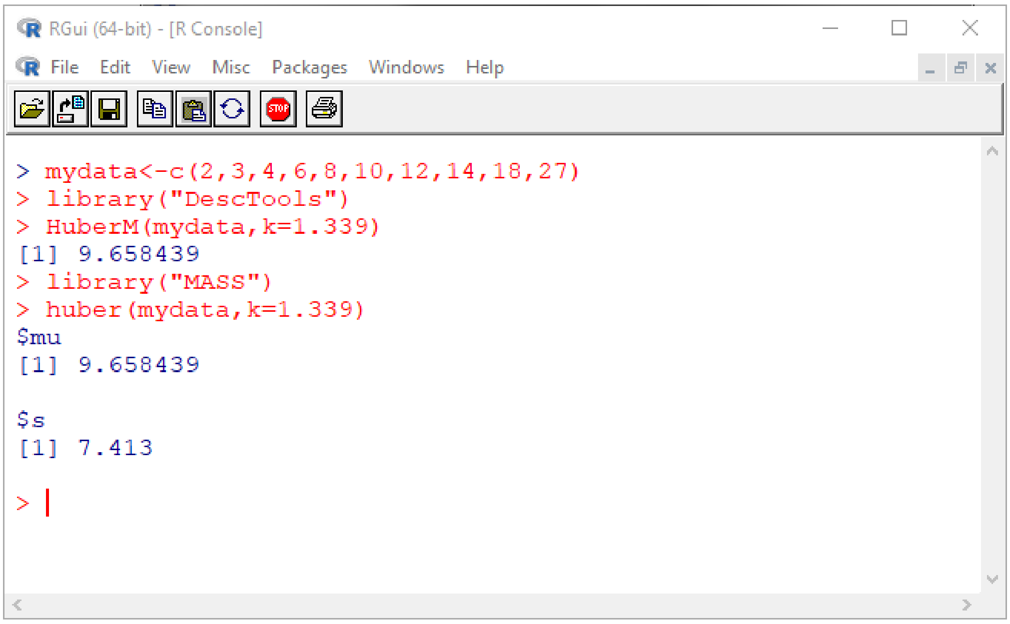
Nous retrouvons bien la même chose que dans le cours théorique mais à la différence de SAS nous avons dû indiquer 0.4 au lieu de 0.2...

Pour le M-estimateur biweight de Tukey nous avons plusieurs packages à choix mais nous allons nous reporter sur la fonction **TukeyBiweight( )** du package **DescTools**:



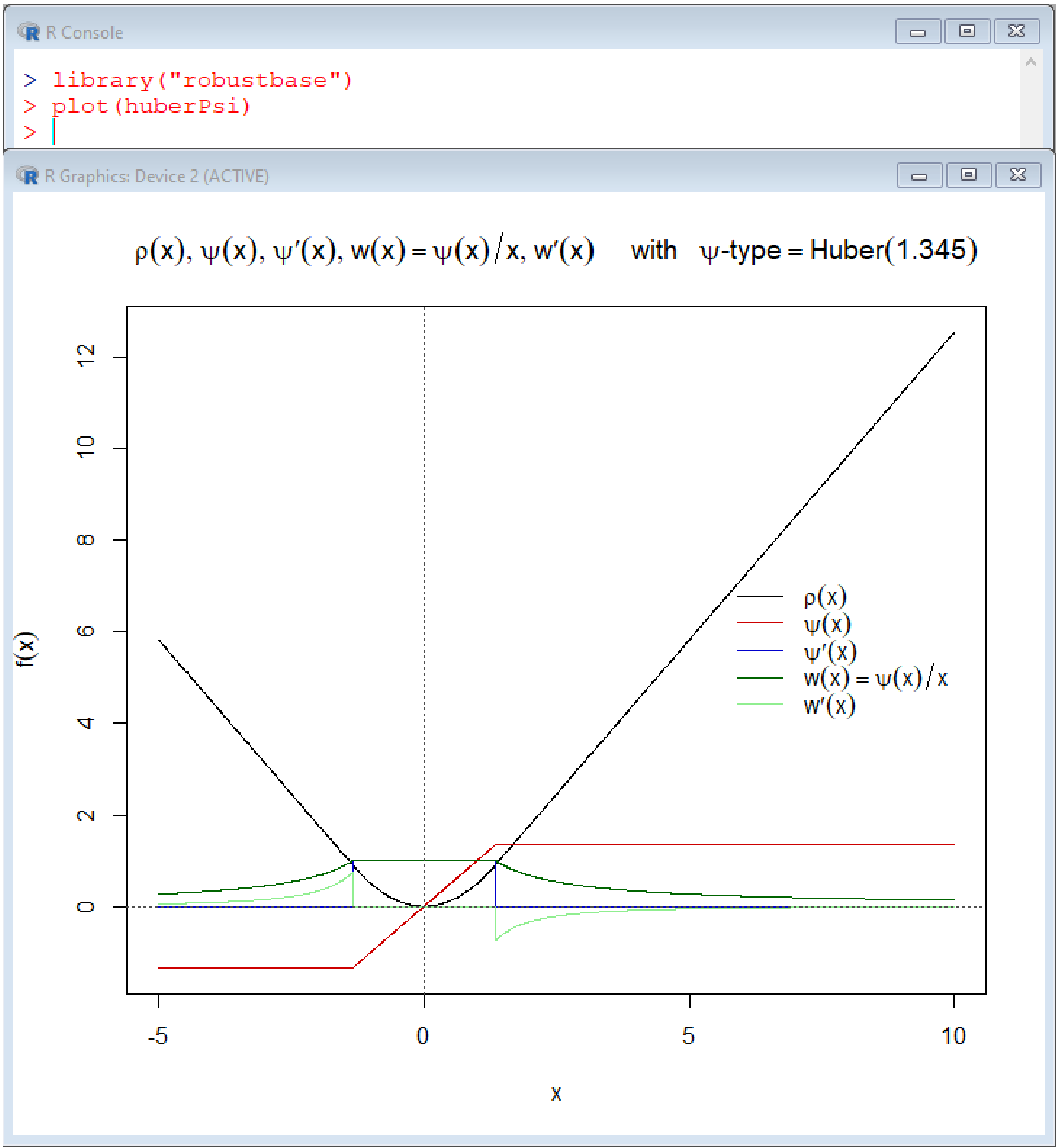
Nous n'obtenons donc pas la même chose qu'avec SPSS qui donne 8.75 (la joie de l'absence de normes... a priori).

Il y a aussi le M-estimateur de Huber dans le même package avec la fonction **HuberM( )** mais il y a aussi la fonction Huber( ) du package MASS qui renvoie le MAD (Mean Absolute Deviation) en plus:



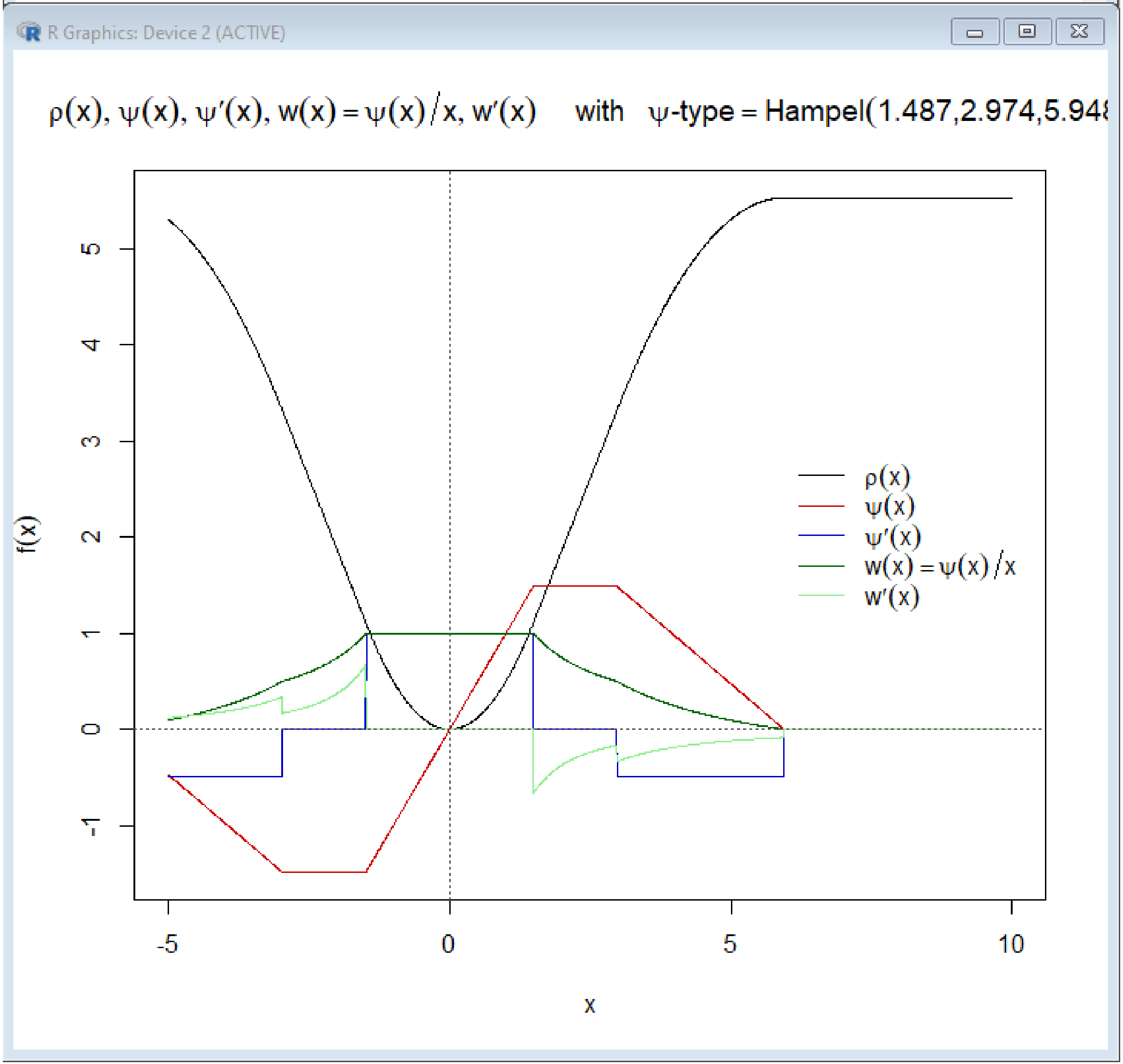
Mais qui là aussi ne donne pas la même chose que SPSS qui donne 9.09...

Avec le package **robustbase** nous pouvons afficher les différentes fonctions y relatives que nous avons introduit dans le cours théorique. Par exemple avec l'estimateur d'Huber:

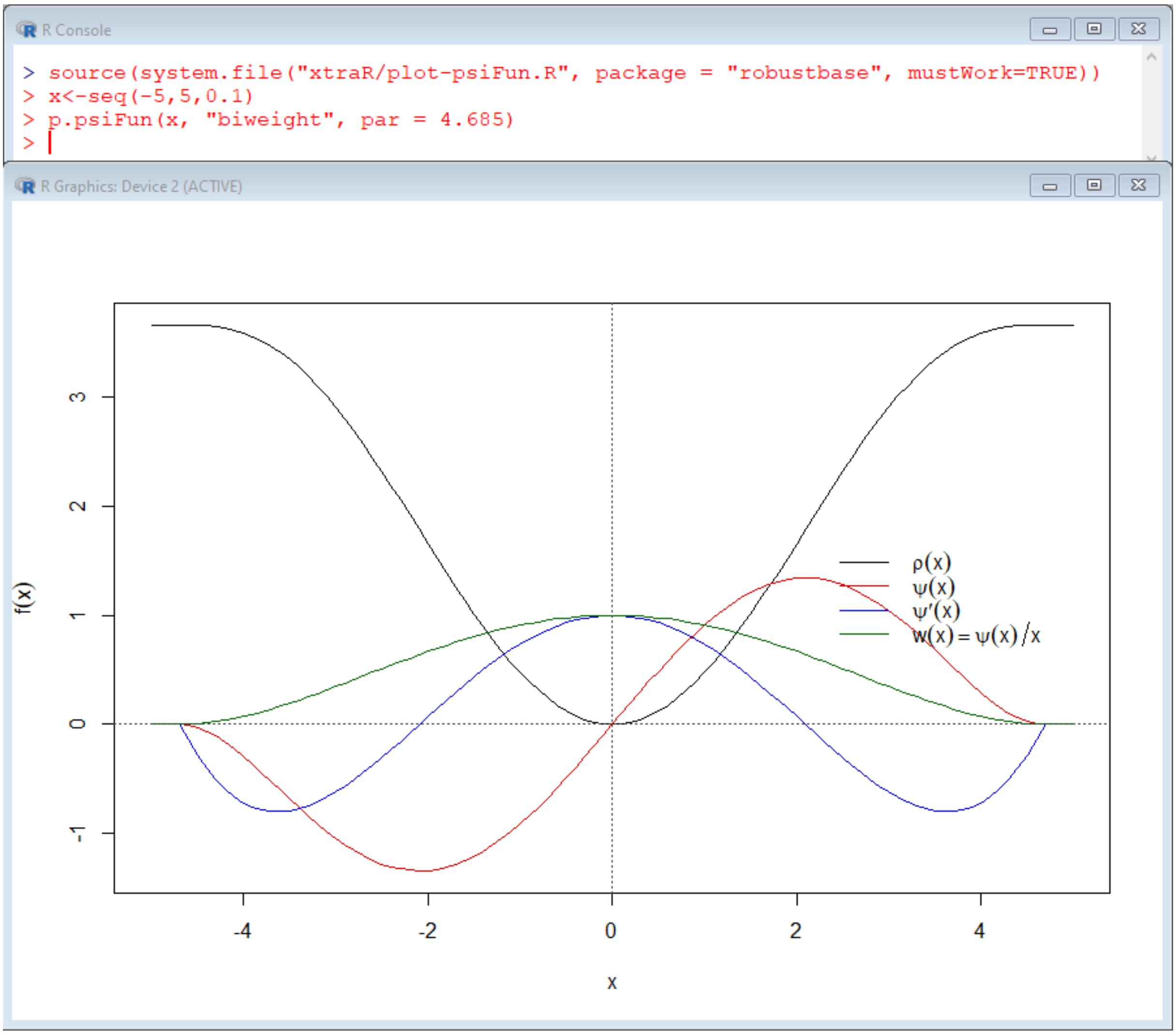


Ou celui de Hampel:

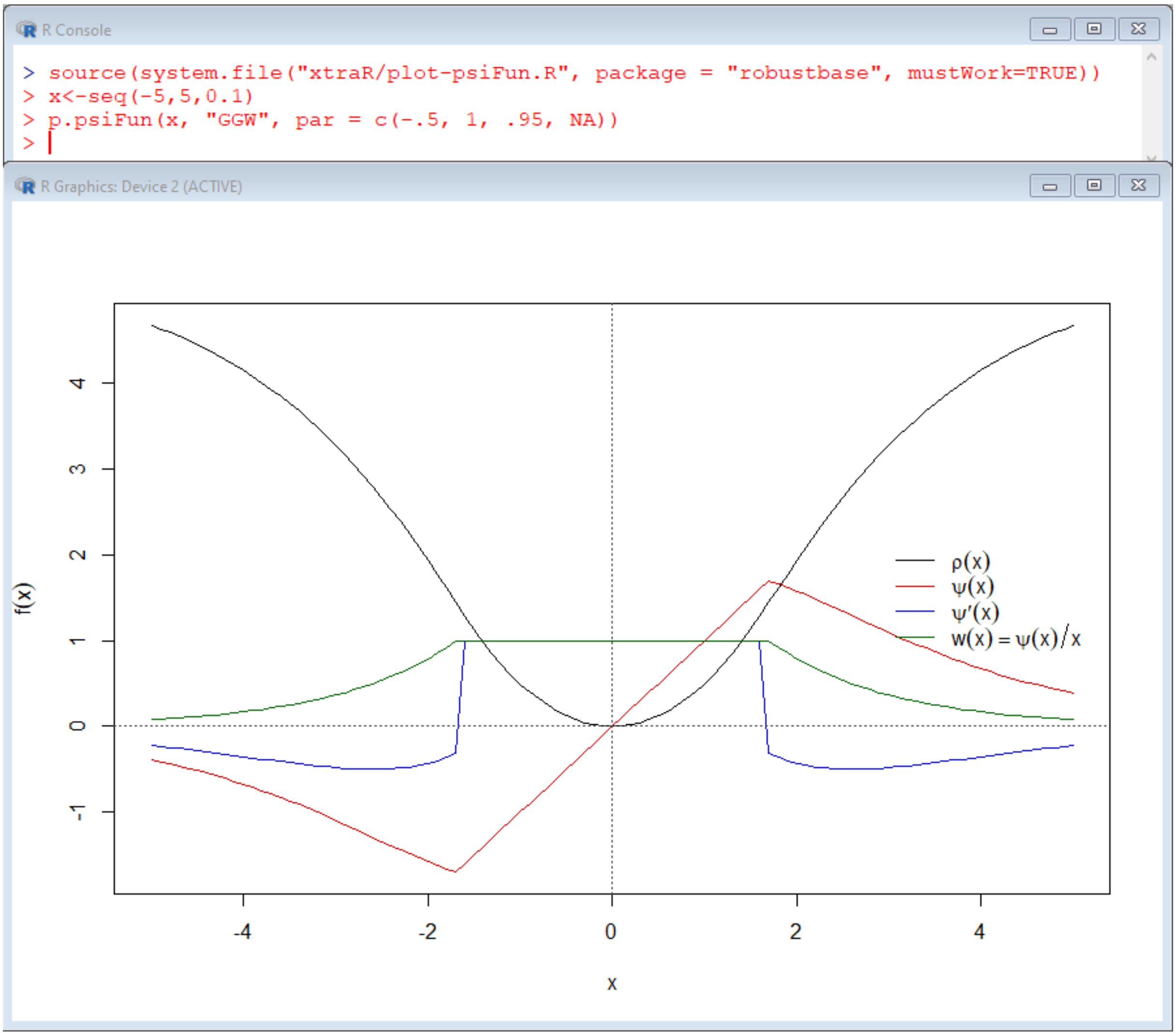
```
R Console
> library("robustbase")
> plot(hampelPsi)
>
```



Ou le Tukey biweight:

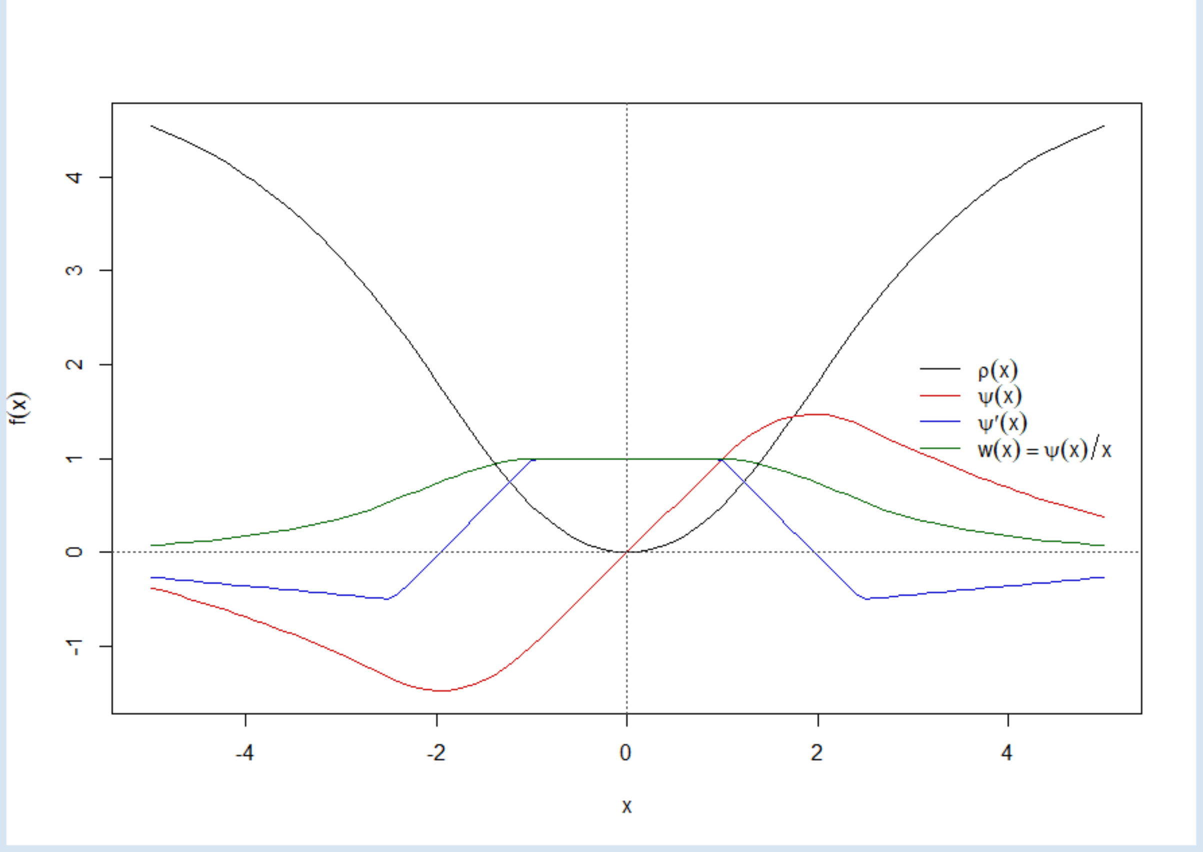


Ou la fonction Generalized Gauss-Weight:



Ou encore la linear quadratic:

```
> source(system.file("xtraR/plot-psiFun.R", package = "robustbase", mustWork=TRUE))
> x<-seq(-5,5,0.1)
> p.psiFun(x, "LQQ", par = c(-.5,1.5,.95,NA))
>
```



Voyons maintenant quelques estimateurs robustes et M-estimateurs de la dispersion avec les fonctions natives **sd( )** et **mad( )** mais aussi avec les fonctions **scale2Tau2( )** du package **robustbase** et **hubers( )** du package **MASS**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> mydata<-c(2,3,4,6,8,10,12,14,18,27)
> sd(mydata)
[1] 7.748835
> mad(mydata)
[1] 7.413
> library("robustbase")
> scaleTau2(mydata,mu.too=TRUE)
[1] 8.732576 7.049214
> library("MASS")
> hubers(mydata)
$mu
[1] 9.757147

$s
[1] 7.209564

>
```

Voyons maintenant comment calculer l'erreur-standard classique (de Pearson) et l'erreur standard aussi de certaines estimateurs robustes et M-estimateurs avec le package **WRS2** et ses fonctions **msmedse( )**, **trimse( )**, **winse( )**, **mestse( )**:

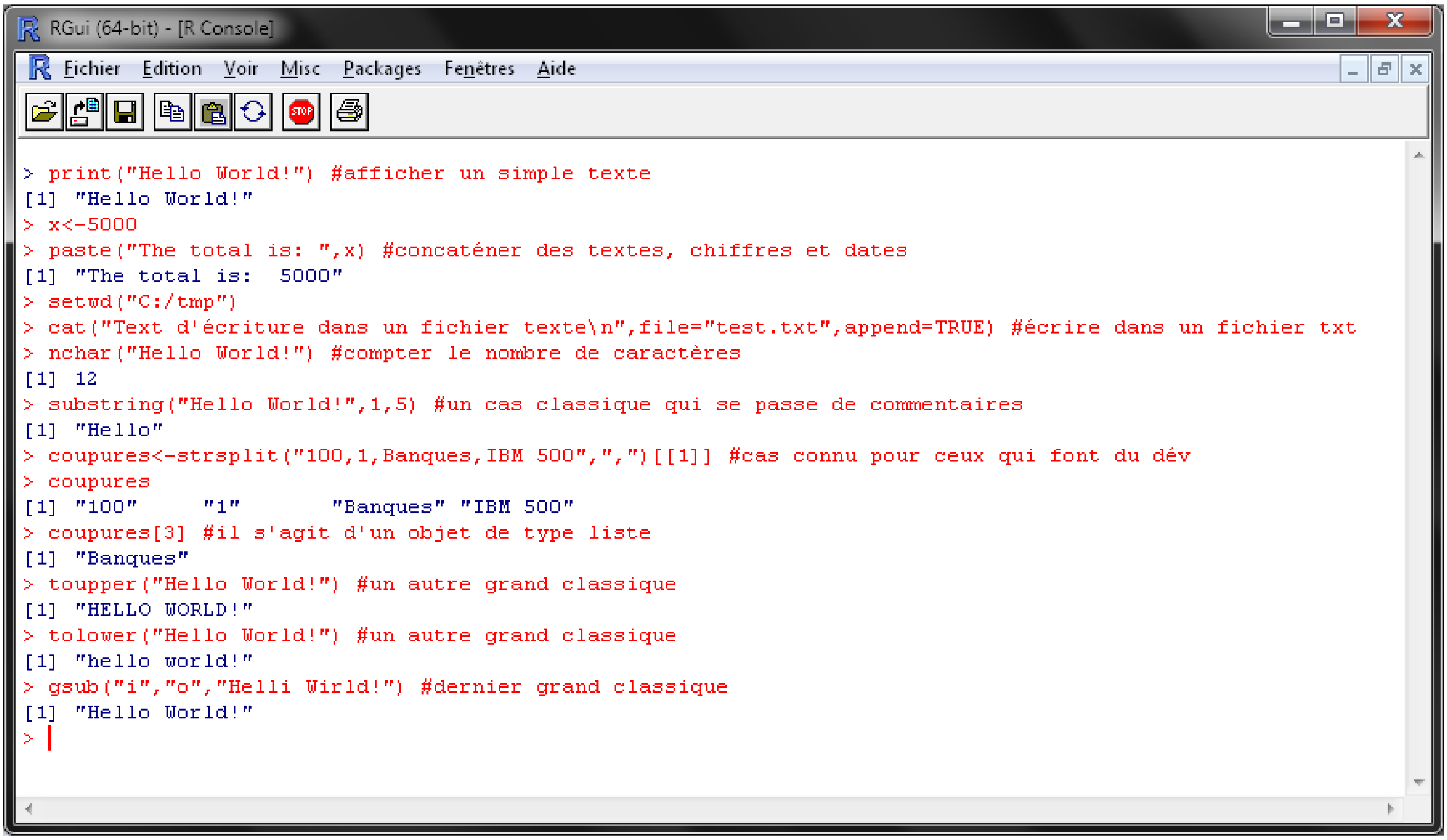
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> mydata<-c(2,3,4,6,8,10,12,18,27)
> #création de la fonction pour l'erreur standard classique
> se <- function(norm) sd(norm)/sqrt(length(norm))
> #on calcule l'erreur standard de Pearson
> se(mydata)
[1] 2.702879
> library("WRS2")
> #l'erreur standard de la médiane
> msmedse(mydata)
[1] 4.852806
> #l'erreur standard de la moyenne tronquée à 20%
> trimse(mydata,tr=0.20)
[1] 3.280183
> #l'erreur standard de la moyenne Winzorisée tronquée à 20%
> winse(mydata,tr=0.2)
[1] 2.624147
> #l'erreur standard d'un M-estimateur de Huber
> mestse(mydata,bend = 1.28)
[1] 2.760297
>
```

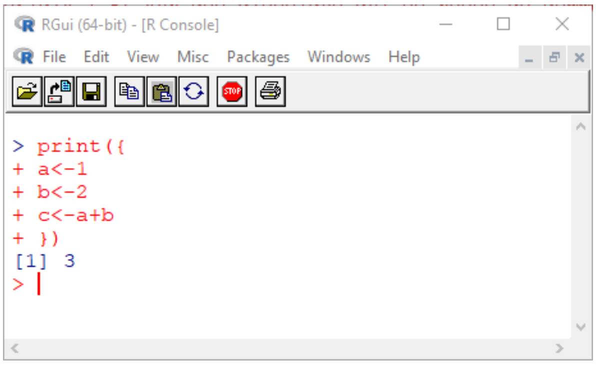
Exemple 17.: Manipulations de textes

R 3.0.2

Encore une fois, lorsque l'on fera du scripting outre manipuler des scalaires, des entités algébriques (vecteurs ou matrices) nous aurons aussi à manipuler des textes. Voici quelques alors quelques commandes classiques de manipulations des textes avec les commandes pour afficher, concaténer, etc.: **print( )** , **paste( )** , **cat( )** , **nchar( )** , **substring( )** , **strsplit( )** , **toupper( )** , **tolower( )** , **gsub( )** :



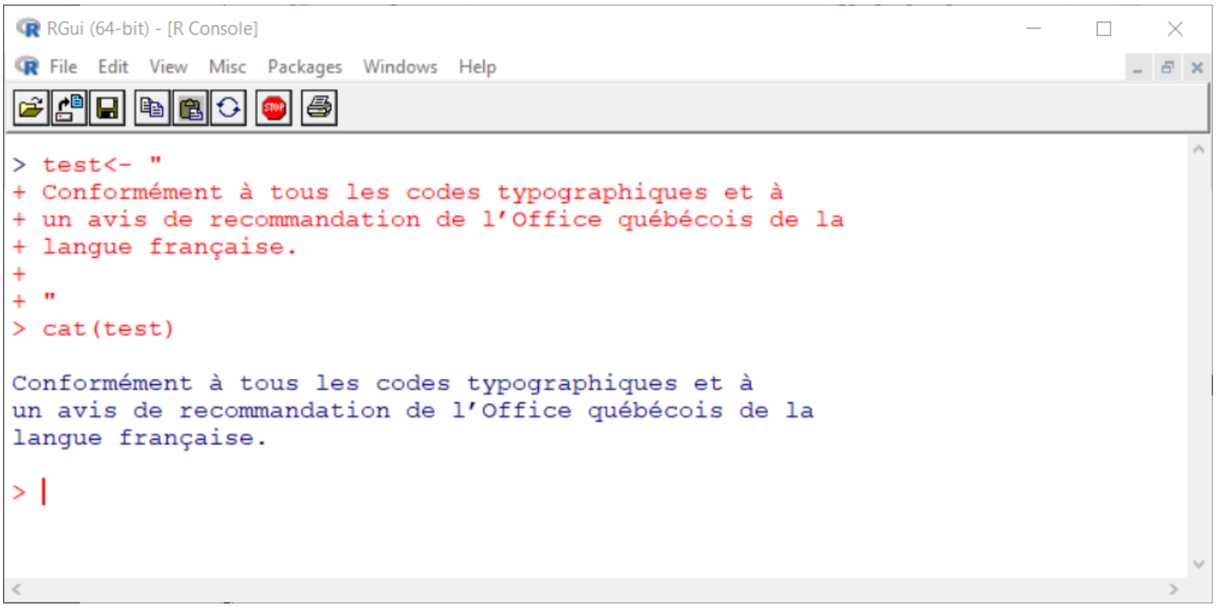
Notez qu'avec **print( )** nous pouvons exécuter plusieurs commandes sans les afficher:



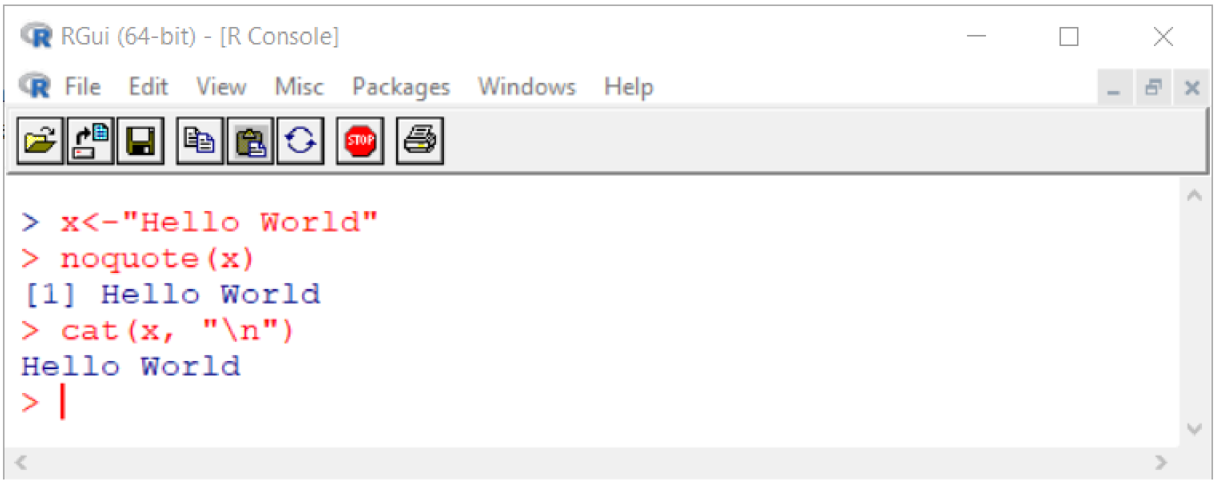
Ou encore les fameux nettoyages d'espaces en trop:



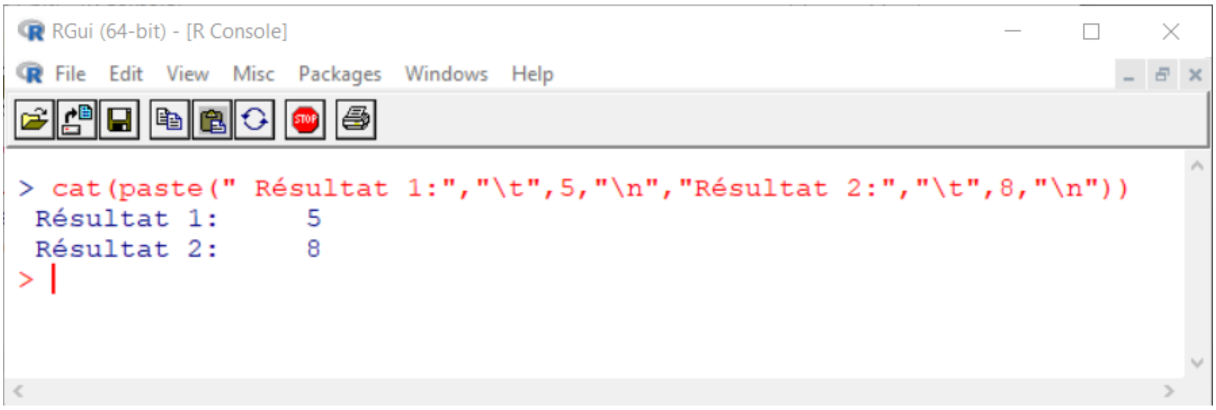
La fonction **cat( )** va aussi être utile pour écrire de longs textes sur plusieurs lignes dans les scripts ou dans la largeur disponible de la fenêtre de console:



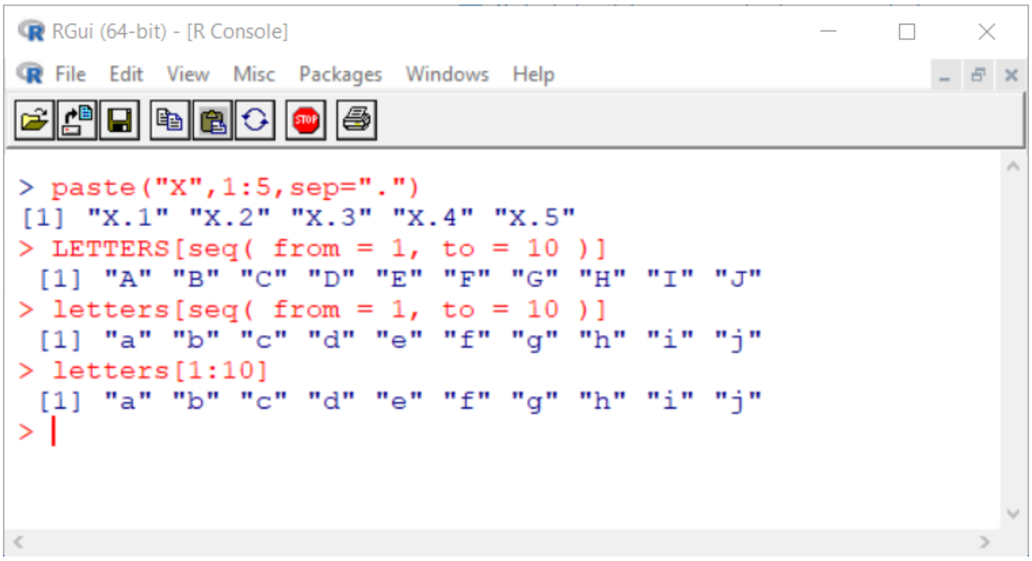
Ou pour afficher plus proprement on peut utiliser les fonctions natives **noquote( )** ou **cat( )** :



Ce qui permet aussi d'utiliser le symbole de tabulation \t très utilisé dans les sorties statistiques de R pour générer des tableaux (comme ceux de l'ANOVA par exemple!):

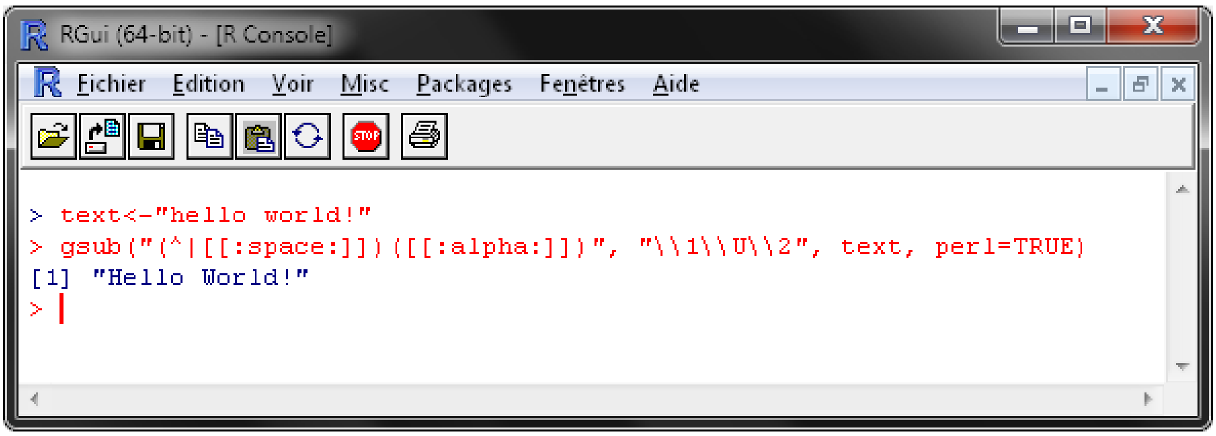


Avec la commande **paste( )** nous pouvons aussi générer de façon astucieuse des noms de variables incrémentées ou avec **letters( )** pour des lettres:

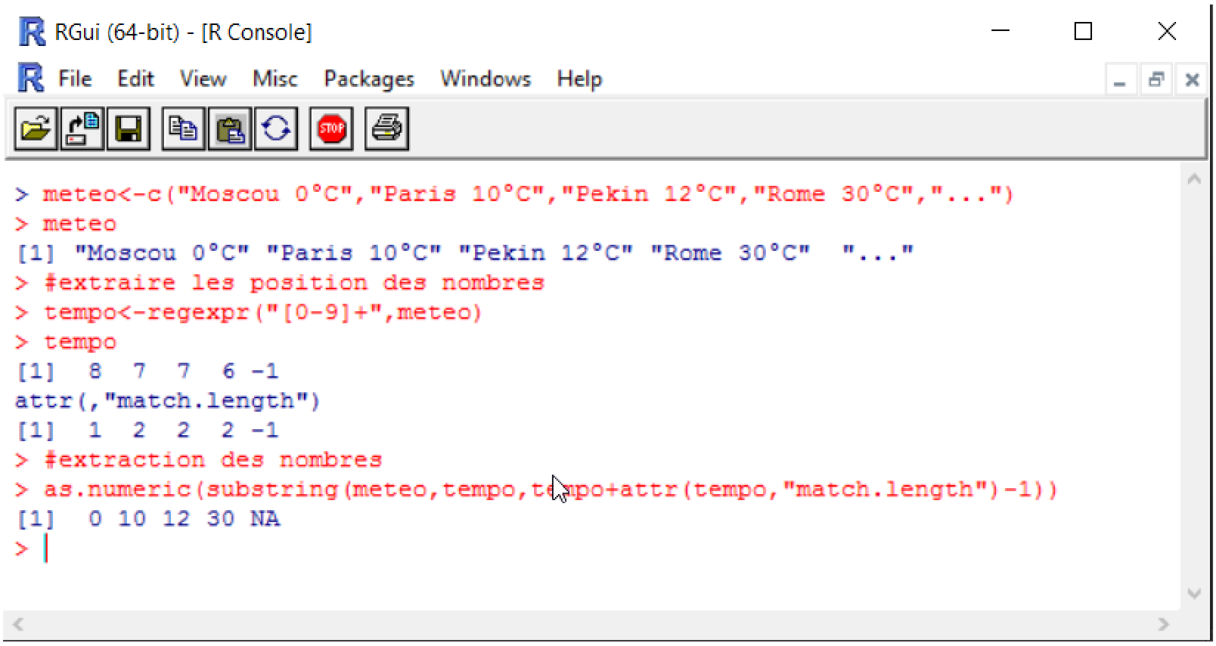


Utilisation classique des RegEx (Regular Expressions)

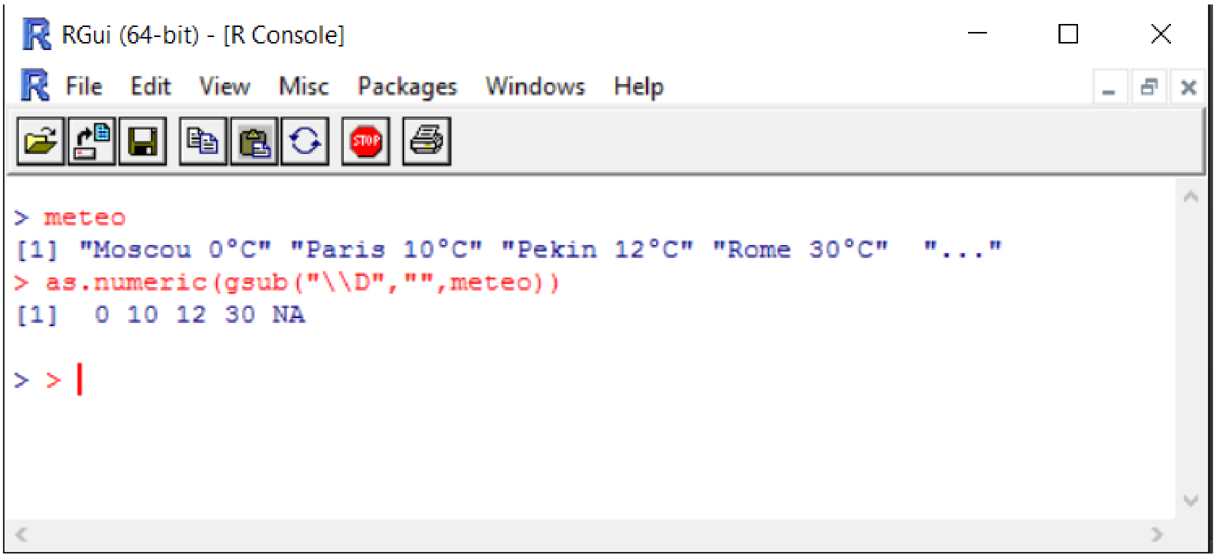
Pour mettre la première lettre d'un mot en majuscule ou de plusieurs mots il faudra passer par une regular expression (regex):



Toujours avec les RegEx faisons maintenant un cas un peu plus compliqué mais typique dans certaines techniques de Text Mining: l'extraction de nombres d'un vecteur de textes. Voyons d'abord cela avec la fonction **regexpr( )** :



Sinon **gsub( )** fait cela directement:



Un grand classique du RegEx est de créer une fonction qui nettoie automatiquement les titres de colonnes de data frames (peut-être qu'il existe un package pour cela?) et en plus c'est très instructif (!!!):

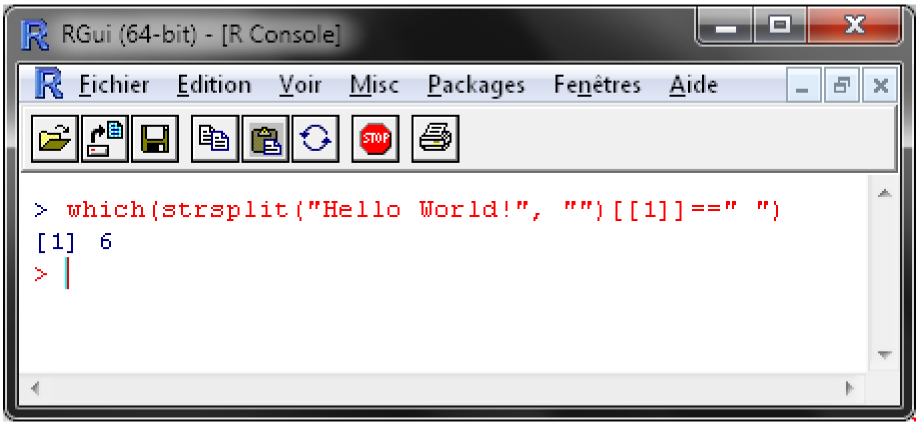
```
> clean_names <- function(headers_to_clean, unique = FALSE) {
+   n <- if (is.data.frame(headers_to_clean)) colnames(headers_to_clean) else headers_to_clean
+
+   n <- gsub("%+", "_pct_", n)
+   n <- gsub("\\$+", "_dollars_", n)
+   n <- gsub("\\++", "_plus_", n)
+   n <- gsub("-+", "_minus_", n)
+   n <- gsub("\\*+", "_star_", n)
+   n <- gsub("#+", "_cnt_", n)
+   n <- gsub("&+", "_and_", n)
+   n <- gsub("@+", "_at_", n)
+   n <- gsub("[^a-zA-Z0-9_]+", "_", n)
+   n <- gsub("([A-Z][a-z])", "_\\1", n)
+   n <- tolower(trimws(n))
+   n <- gsub("(^_+|_+$)", "", n)
+   n <- gsub("_+", "_", n)
+
+   #pour gérer les colonnes qui ont des noms identiques
+   if (unique) n <- make.unique(n, sep = "_")
+
+   if (is.data.frame(headers_to_clean)) {
+     colnames(headers_to_clean) <- n
+     headers_to_clean
+   } else {
+     n
+   }
+ }
>
```

On peut tester avec une colonne fictive (cela marche aussi avec un data frame vu comment la fonction est écrite):

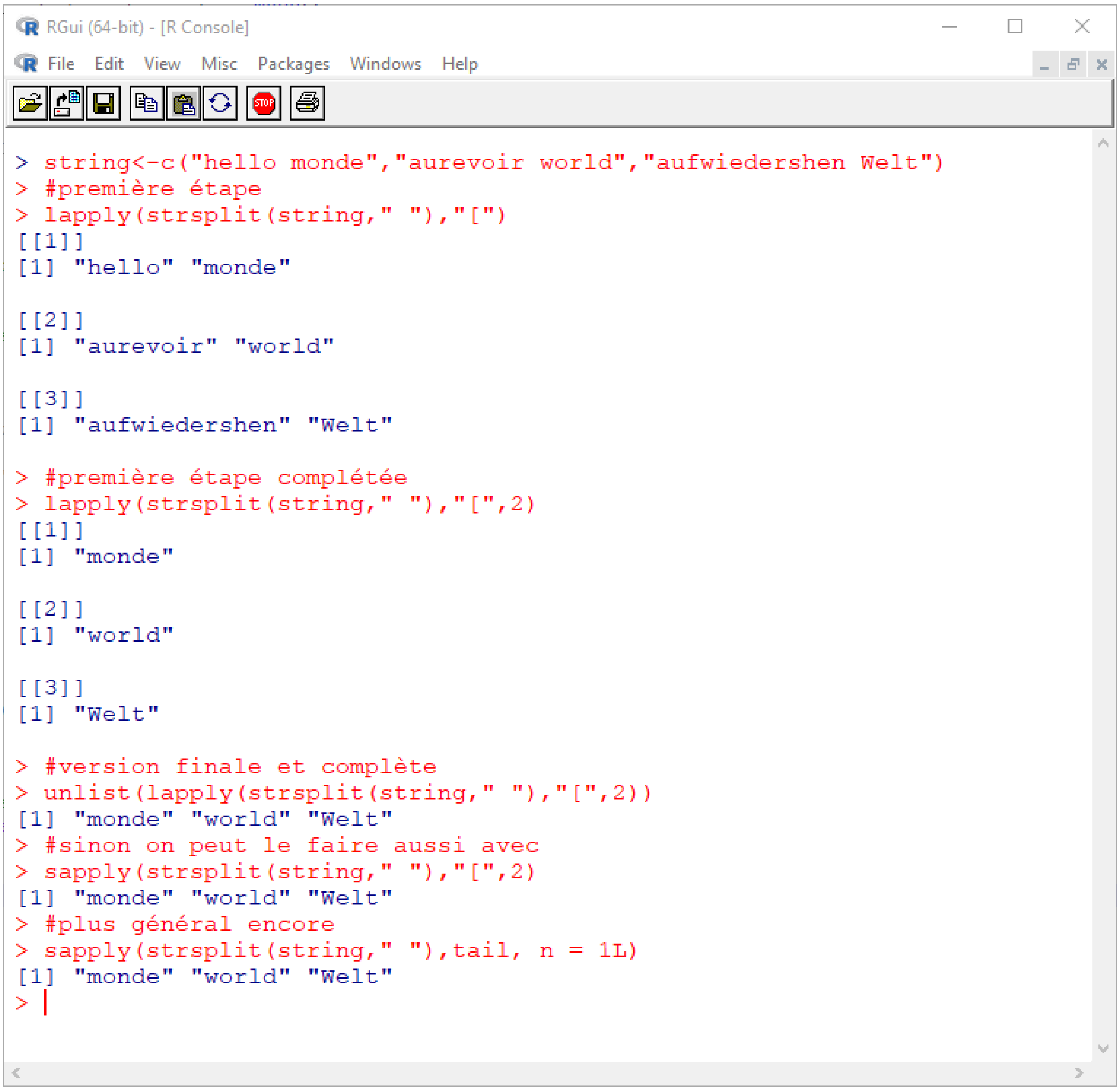
```
> #exemple noms:
> noms<-c(
+      "   a", "a  ",
+      "a %", "a", "$a", "$$$a", "GDP ($)", "GDP (us$)",
+      "a (#)", "a & b", "#", "$",
+      "a_cnt",
+      "Aa&Bb", "camelCasePhrases",
+      "AlphaBetaGamma", "Alpha       Beta", "Beta  !!! Gamma",
+      "a + b", "a - b", "a * b"
+    )
> clean_names(noms,unique=T)
 [1] "a"               "a_1"              "a_pct"            "a_2"
 [5] "dollars_a"       "dollars_a_1"      "gdp_dollars"      "gdp_us_dollars"
 [9] "a_cnt"           "a_and_b"          "cnt"              "dollars"
[13] "a_cnt_1"         "aa_and_bb"        "camel_case_phrases" "alpha_beta_gamma"
[17] "alpha_beta"      "beta_gamma"       "a_plus_b"         "a_minus_b"
[21] "a_star_b"
>
```

Trouver la position de chaînes de caractères

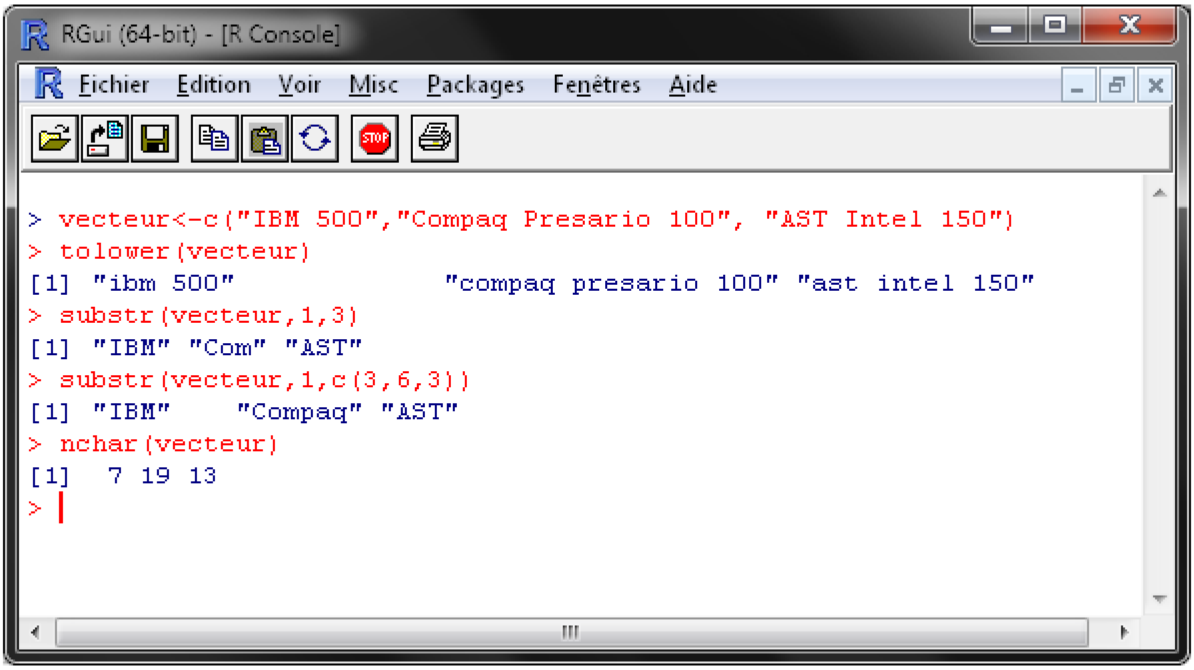
Indiquons aussi le code pour trouver la position d'une certaine chaîne de caractère:



Remarquons l'usage suptil de **strsplit( )**, avec les fonctions **lapply( )**, **sapply( )** et **unlist( )** pour extraire par exemple le dernier mot des textes d'un vecteur de textes:

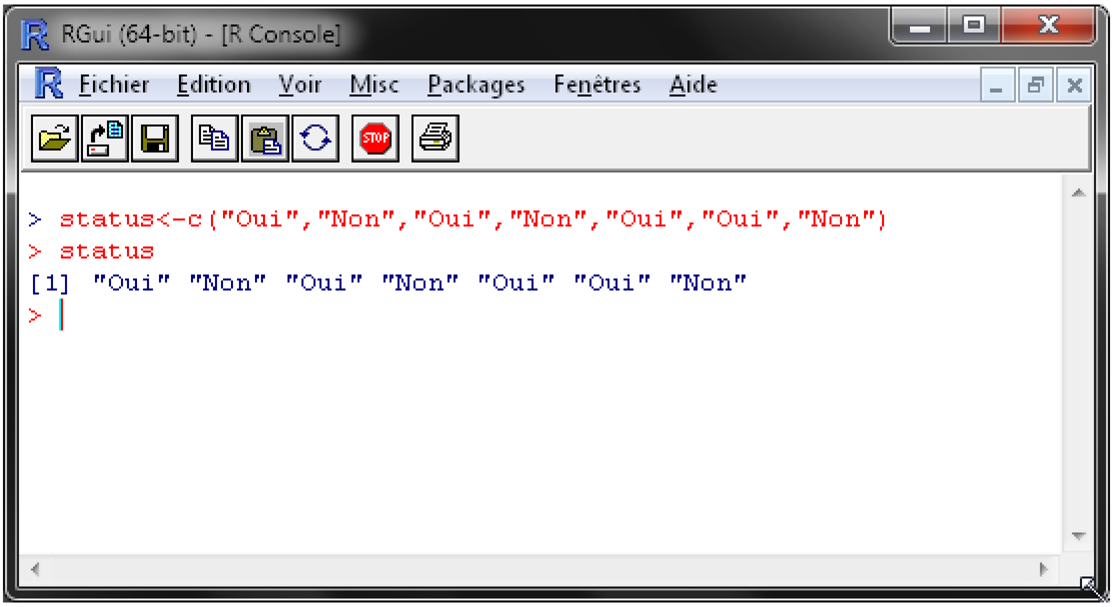


Évidemment certaines commandes que nous venons de voir s'appliquent aussi à des vecteurs (cas très important puisque les data frames sont composés de vecteurs!):

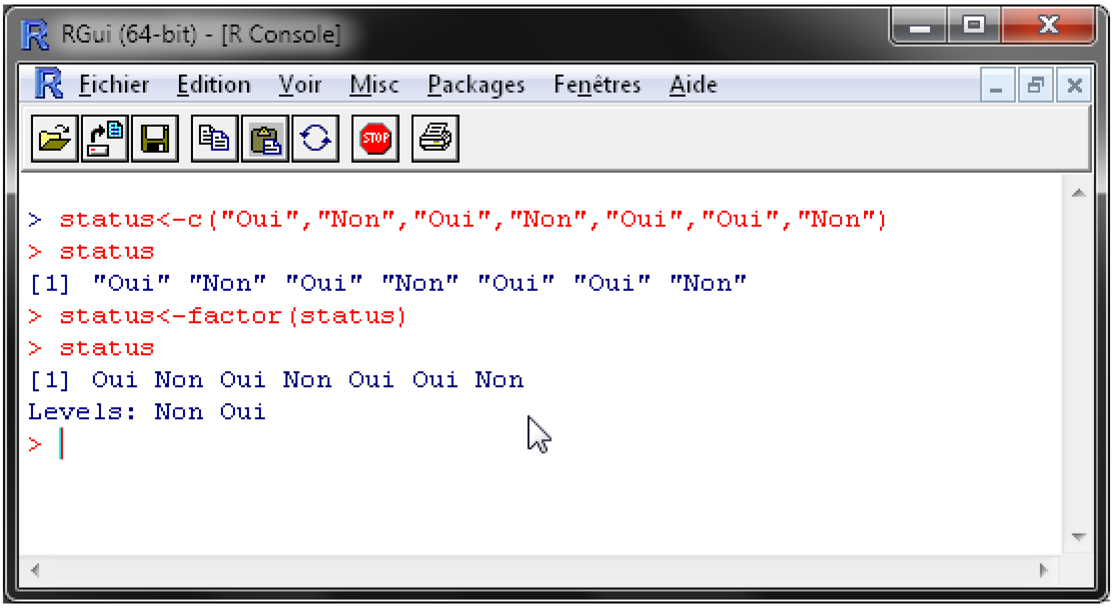


Facteurs

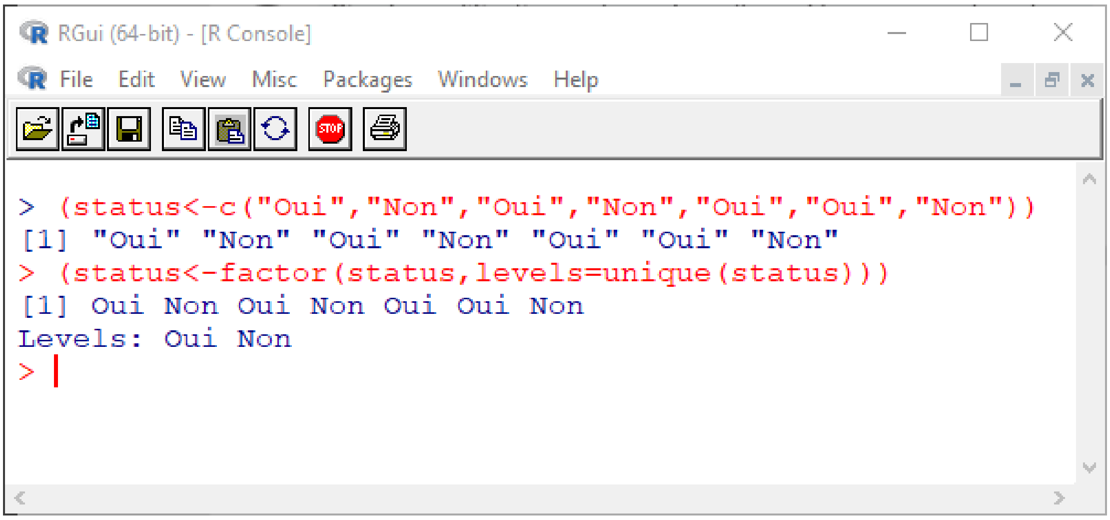
Si nous créons un vecteur avec des valeurs textuelles:



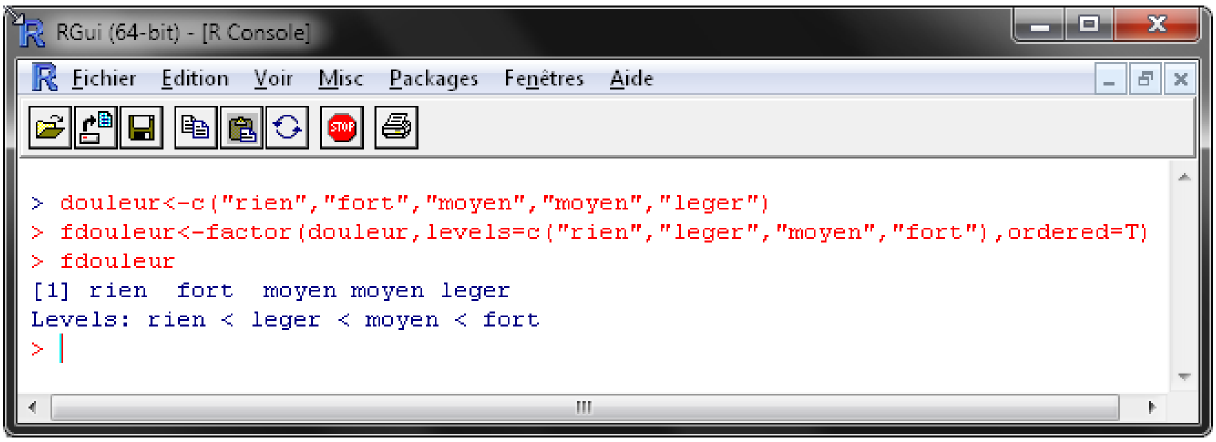
celui-ci va prendre de la mémoire pour rien car dans le cas présent nous voyons que nous avons affaire à des facteurs. Dès lors pour optimiser la mémoire de R, nous pouvons lui dire qu'il s'agit de facteurs et il associera simplement à chaque texte une valeur 1, 2 (ou plus si nous avons plus de deux facteurs). Pour faire cela, nous utilisons la commande **factor( )** tel que ci-dessous:



On peut aussi garder l'ordre d'origine des facteurs:



Cela donne alors des variables qualitatives non ordonnées. Si nous avons besoin de définir des variables qualitatives ordonnées, nous utiliserons la syntaxe suivant (attention! si l'argument *levels* n'est pas précisé, c'est l'ordre alphabétique qui sera utilisé):



Attention!!!! Observez bien le piège subtil suivant:

```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
> #maintenant par un filtre éliminons tous les "setosa"
> iris2 = subset(iris, Species != "setosa")
> #vérifions qu'il n'y en ait effectivement plus du tout
> table(iris2$Species)

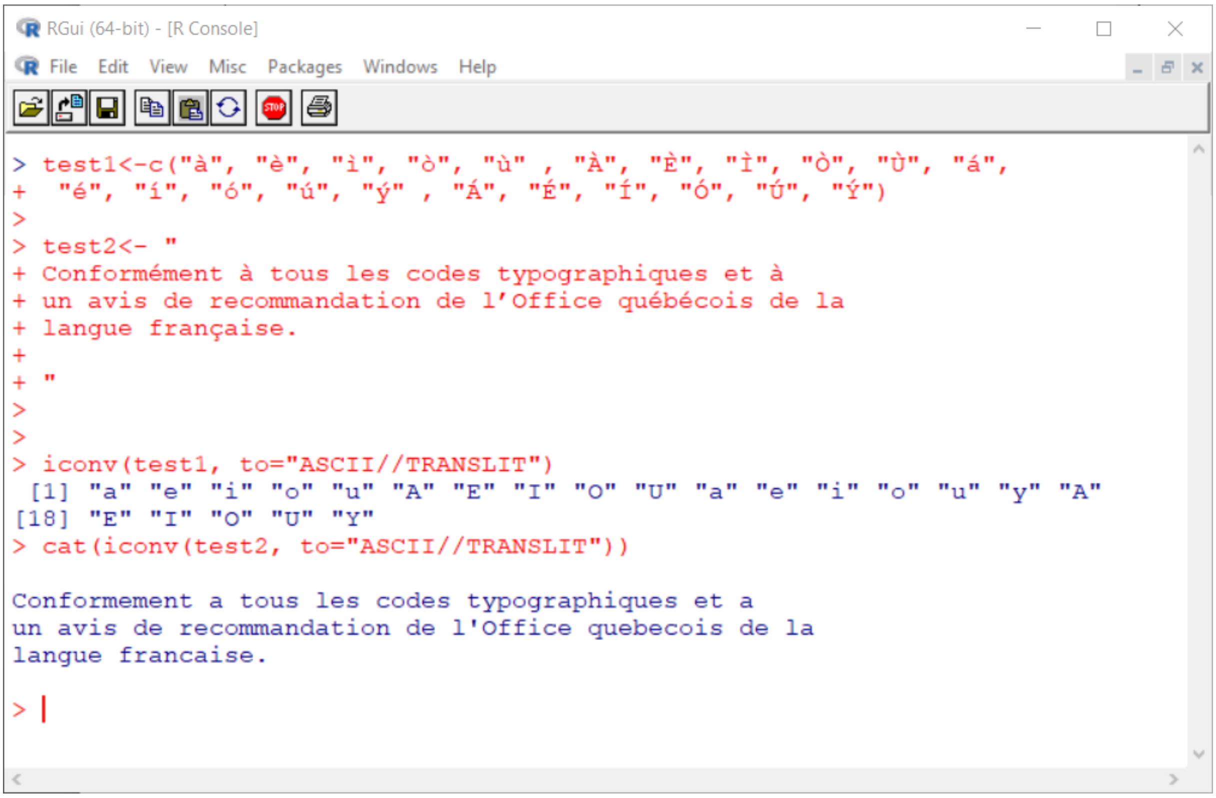
    setosa versicolor  virginica
         0         50         50
> #maintenant redamdons les levels:
> levels(iris2$Species)
[1] "setosa"     "versicolor" "virginica"
> #Les "setosa" sont toujours là! Pou extraire les valeurs uniques
> #d'une colonne à facteurs, privilégiez toujours la commande unique()
> unique(iris2$Species)
[1] versicolor virginica
Levels: setosa versicolor virginica
>
```

Text Mining de base

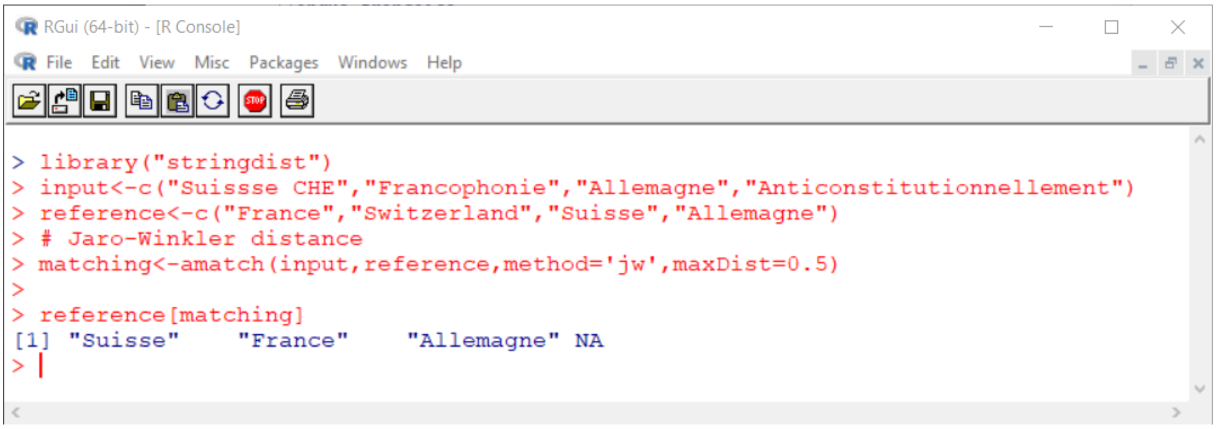
Pour ceux qui feraient du texte mining, voici quelques fonctions utiles après avoir classé tous les mots dans des composantes de vecteurs utilisant **grep( )** (signifiant Global Regular Expression Print):

```
> #chercher les composantes ayant les lettres "a" ou "u"
> grep("[au]",c("Equateur","Pôle Nord","Pôle Sud"))
[1] 1 3
> grep("[au]",c("Equateur","Pôle Nord","Pôle Sud"),value=TRUE)
[1] "Equateur" "Pôle Sud"
> #ou simplement un mot
> grep("Pôle Nord",c("Equateur","Pôle Nord","Pôle Sud"))
[1] 2
> grep("Pôle Sud",c("Equateur","Pôle Nord","Pôle Sud"))
[1] 3
>
```

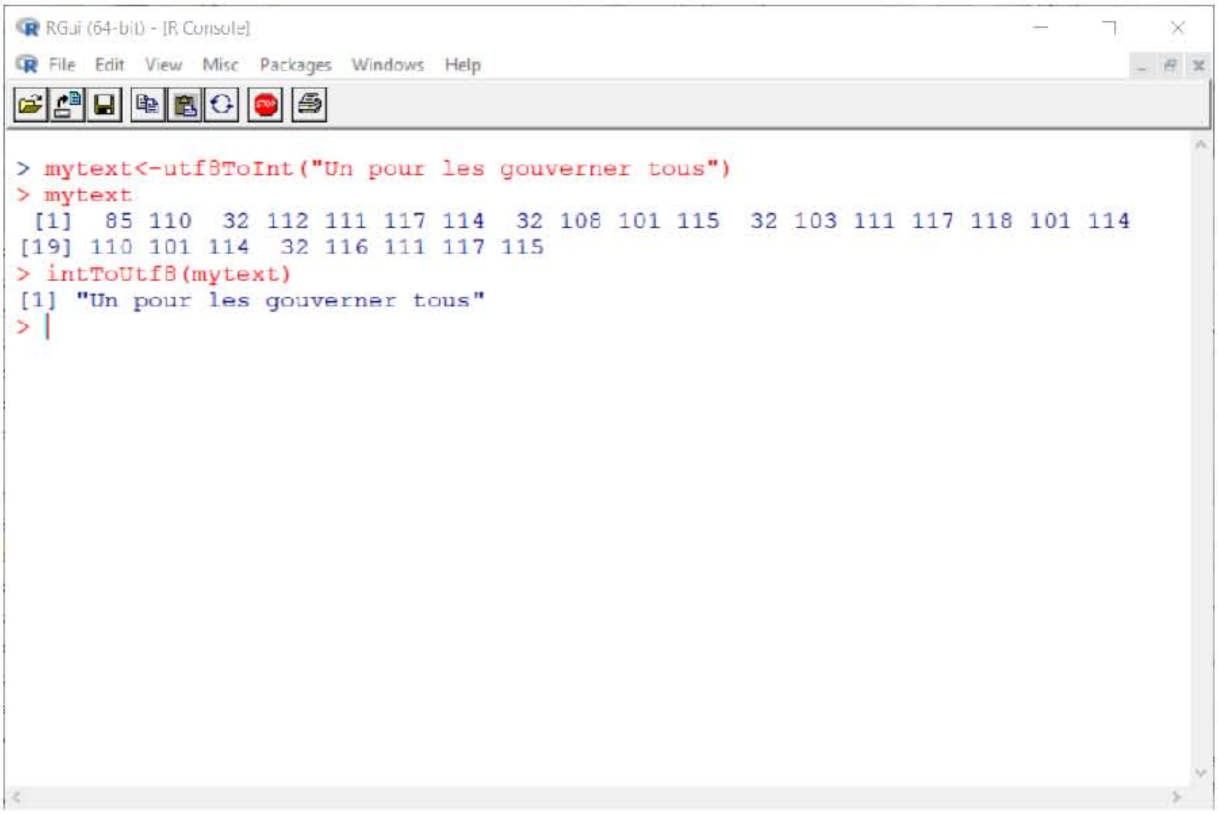
Enfin on a le cas classique dans le Text Mining qui consiste à supprimer les accents avec la fonction native **iconv( )** :



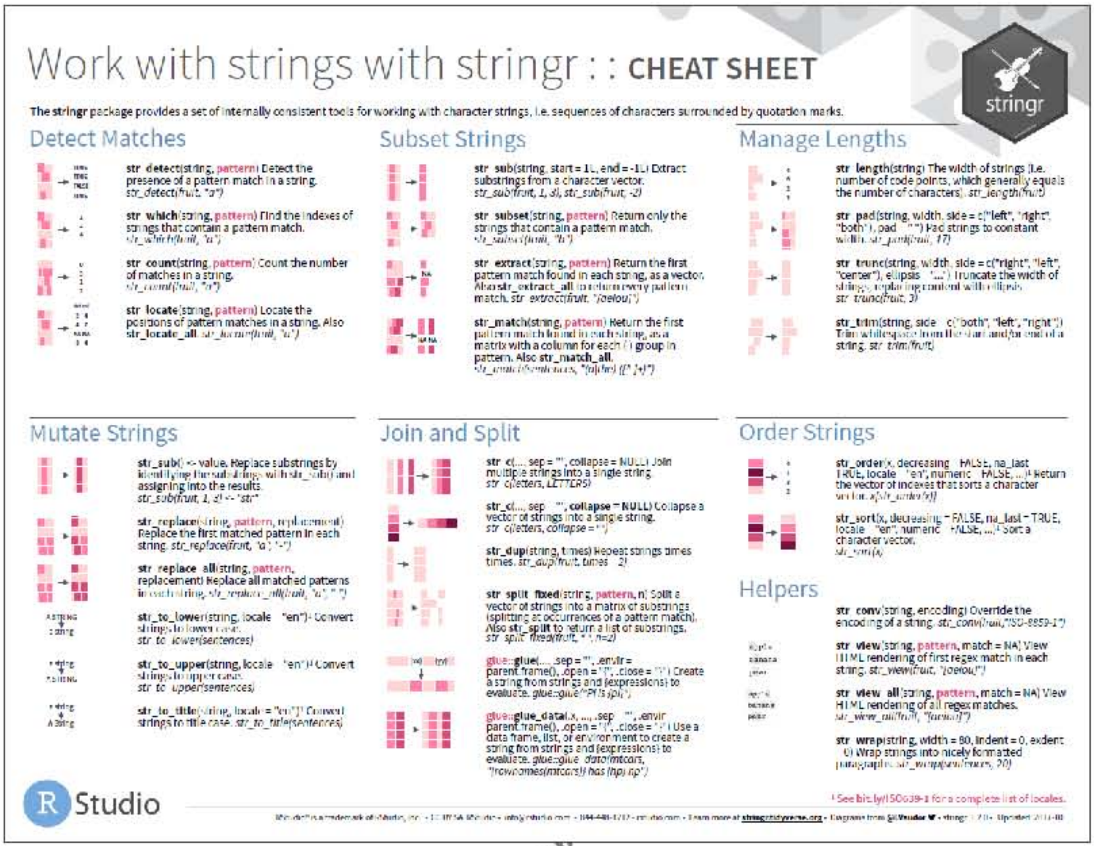
Signalons aussi dans le domaine du nettoyage de données "l'approximate text matching" qui permet souvent de remplacer des données textuelles partiellement erronées par des humains par des standards prédéfinis. Pour cela, nous allons utiliser le package **stringdist** et sa fonction **amatch( )** en particulier avec la distance de Jaro-Winkler qui est très performante à ce jeu là…:



Quand on fait de l'approximate text matching ou du Text Mining il peut être nécessaire avec certaines méthodes de convertir les lettres en leurs valeurs numériques et inversement! Voici comment effectuer ceci avec les fonctions **utf8ToInt( )** et **intToUtf8( )** :


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mytext<-utf8ToInt("Un pour les gouverner tous")
> mytext
[1] 85 110 32 112 111 117 114 32 108 101 115 32 103 111 117 118 101 114
[19] 110 101 114 32 116 111 117 115
> intToUtf8(mytext)
[1] "Un pour les gouverner tous"
>

Pour ceux qui veulent aller plus loin sur la manipulation de chaînes de caractères, il y a le package **stringr** dont voici la cheatsheet qui en résume assez bien les concepts majeurs:



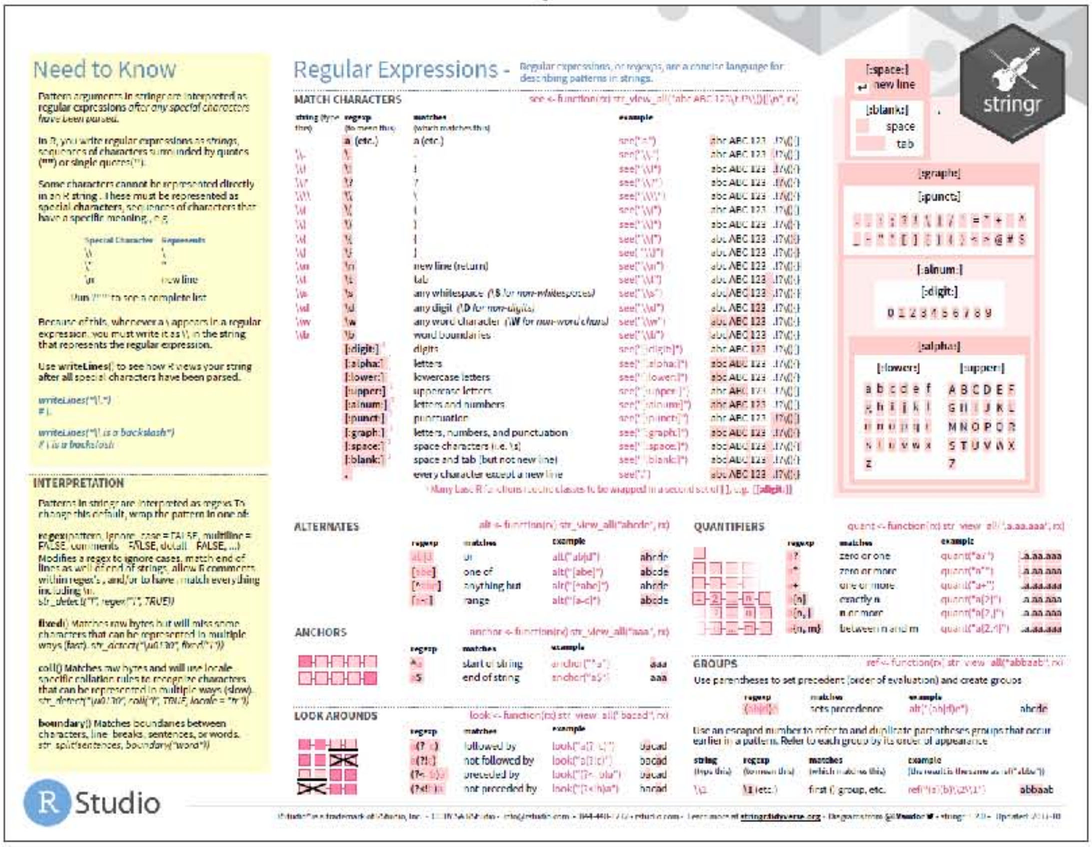


Figure 5 Cheatsheet stringr

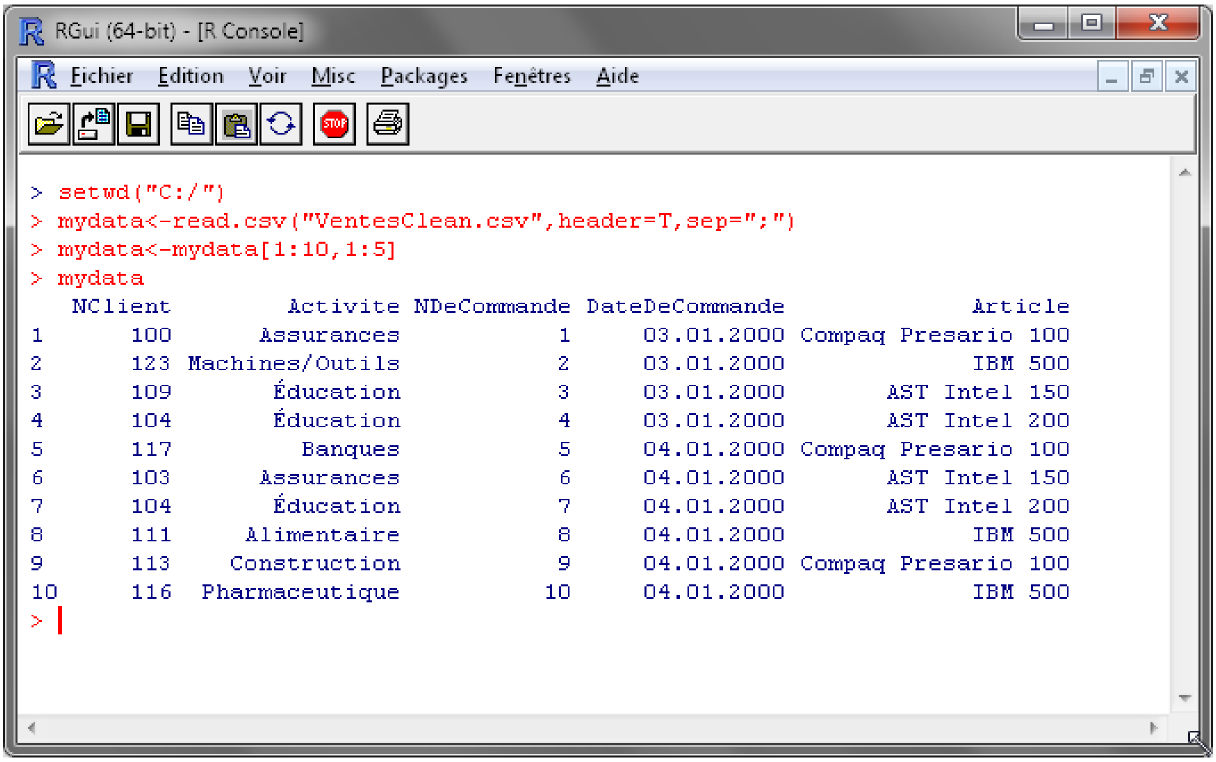
Exemple 18.: Manipulations de dates

R 3.0.2

Pour travailler sur les dates nous allons faire une entorse en supposant que nous avons déjà étudié l'import de fichiers *.csv (voir plus loin pour les détails relativement à ce sujet).

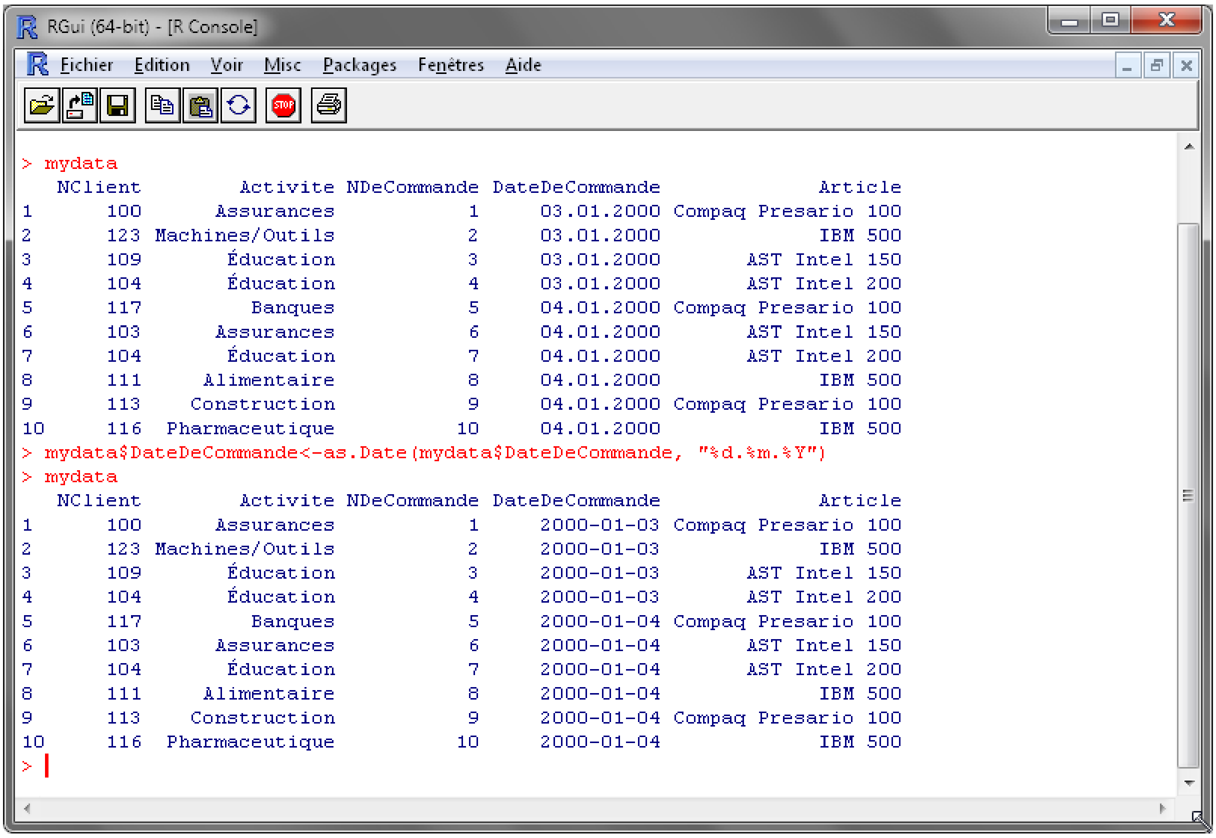
Pourquoi? Simplement parce que c'est la manipulation de nettoyage la plus fréquente (analyses temporelles oblige...) et donc la plus importante puisque la majorité des pays ne respectent pas la norme ISO 8601. Bon par la même occasion cela vous montrera comment (dans l'idée) on modifie toute une colonne d'un data frame aussi pour des textes (l'idée étant la même).

Considérons donc l'import du sous-ensemble du fichier *.csv suivant:




```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata<-mydata[1:10,1:5]
> mydata
   NClient         Activite NDeCommande DateDeCommande             Article
1      100       Assurances           1     03.01.2000 Compaq Presario 100
2      123 Machines/Outils            2     03.01.2000             IBM 500
3      109        Éducation           3     03.01.2000       AST Intel 150
4      104        Éducation           4     03.01.2000       AST Intel 200
5      117          Banques           5     04.01.2000 Compaq Presario 100
6      103       Assurances           6     04.01.2000       AST Intel 150
7      104        Éducation           7     04.01.2000       AST Intel 200
8      111      Alimentaire           8     04.01.2000             IBM 500
9      113     Construction           9     04.01.2000 Compaq Presario 100
10     116  Pharmaceutique           10     04.01.2000             IBM 500
> 
```

et nous allons mettre le format de date aux normes pour commencer:



```
> mydata
   NClient        Activite NDeCommande DateDeCommande            Article
1      100      Assurances           1     03.01.2000 Compaq Presario 100
2      123 Machines/Outils           2     03.01.2000            IBM 500
3      109       Éducation           3     03.01.2000      AST Intel 150
4      104       Éducation           4     03.01.2000      AST Intel 200
5      117         Banques           5     04.01.2000 Compaq Presario 100
6      103      Assurances           6     04.01.2000      AST Intel 150
7      104       Éducation           7     04.01.2000      AST Intel 200
8      111     Alimentaire           8     04.01.2000            IBM 500
9      113    Construction           9     04.01.2000 Compaq Presario 100
10     116  Pharmaceutique          10     04.01.2000            IBM 500
> mydata$DateDeCommande<-as.Date(mydata$DateDeCommande, "%d.%m.%Y")
> mydata
   NClient        Activite NDeCommande DateDeCommande            Article
1      100      Assurances           1     2000-01-03 Compaq Presario 100
2      123 Machines/Outils           2     2000-01-03            IBM 500
3      109       Éducation           3     2000-01-03      AST Intel 150
4      104       Éducation           4     2000-01-03      AST Intel 200
5      117         Banques           5     2000-01-04 Compaq Presario 100
6      103      Assurances           6     2000-01-04      AST Intel 150
7      104       Éducation           7     2000-01-04      AST Intel 200
8      111     Alimentaire           8     2000-01-04            IBM 500
9      113    Construction           9     2000-01-04 Compaq Presario 100
10     116  Pharmaceutique          10     2000-01-04            IBM 500
>
```

comme vous pouvez le voir avec la commande **as.Date( )** , les dates sont converties par défaut au format normalisé! L'argument de **as.Date( )** est donc le format de date tel qu'il se trouve à l'origine dans le data frame pour que R puisse en comprendre la structure.

La syntaxe des formats de dates est la suivante dans R:

- %d: Jour en tant que numéro (01-31)
- %a: Jour abrégé en 3 lettres
- %A: Jour non abrégé
- %m: Mois en chiffres (01-12)
- %b: Mois abrégé en 3 lettres
- %B: Mois non abrégé
- %y: Année sur 2 chiffres
- %Y: Année sur 4 chiffres
- %H: Heure sur 0-24
- %h: Heure sur 1-12
- %j: Numéro du jour dans l'année (0-366)
- %p: Indicateur AM/PM pour les heures
- %S: Secondes
- %c: Date et Heure locale
- %x: Date locale
- %X: Heure locale
- %z: Fuseau Horaire
- %U: Numéro de la semaine dans l'année (0-53) utilisant le premier dimanche comme
- %W: Numéro de la semaine dans l'année (0-53) utilisant le premier lundi comme 1er jour de la première semaine

- %w: Numéro de la semaine (0-6) où dimanche est le numéro 0

Sinon voici un cas très instructif:

```
> #Quand on travaille avec des dates et heures, pour éviter les problèmes
> #de paramètres régionaux, toujours enregistrer ces derniers pour les remplacer
> #temporairement par un standard plus commun:
> local_time<-Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME","English")
[1] "English_United States.1252"
>
> #on affiche la date système par exemple pour voir si elle apparaît
> #bien en anglais pour le coup
> format(Sys.Date(),format="%a %b %d %H:%M:%S %Y")
[1] "Tue Mar 17 00:00:00 2020"
>
> #ensuite pour le fun on prend un date générée en tant que texte comme exercice
> (d<-date())
[1] "Tue Mar 17 17:52:00 2020"
> class(d)
[1] "character"
> #et on la restransforme en date (mais en perdant les heures)
> (d1<-as.Date(d,"%a %b %d %H:%M:%S %Y"))
[1] "2020-03-17"
> class(d1)
[1] "Date"
>
> #mais si on veut pas perdre les heures:
> (d2<-base::strptime(d,format="%a %b %d %H:%M:%S %Y"))
[1] "2020-03-17 17:52:00 CET"
> class(d2)
[1] "POSIXlt" "POSIXt"
>
> #on revient sur les paramètres régionaux francophones
> Sys.setlocale("LC_TIME",local_time)
[1] "French_Switzerland.1252"
>
> #on peut les reformater pour les réafficher si on veut
> format(d1,"%Y %b %d %H:%M")
[1] "2020 mars 17 00:00"
> format(d2,"%Y %b %d %H:%M")
[1] "2020 mars 17 17:52"
> 
```

Maintenant revenons sur des exemples simples avec les commandes **Sys.Date( )** , **Sys.Timezone( )** , **format( )** , **weekdays( )** et **months( )** , **ISOdate( )** , **as.numeric( )** , **difftime( )** :

```
> Sys.Date() #date du jour
[1] "2014-05-30"
> Sys.timezone() #fuseau horaire
[1] "CEST"
> Sys.Date()+21 #ajout de jours calendaires à une date
[1] "2014-06-20"
> format(Sys.Date(),format="%d%m%y") #format d'une date selon son choix
[1] "300514"
> format(Sys.Date(),format="%A, %d, %B %Y") #plus élégant
[1] "vendredi, 30, mai 2014"
> weekdays(Sys.Date())
[1] "vendredi"
> months(Sys.Date())
[1] "mai"
> as.numeric(format(Sys.Date(),"%Y")) #l'année seule
[1] 2014
> year(Sys.Date()) #...
Erreur : impossible de trouver la fonction "year"
> difftime(ISOdate(2013,12,20),Sys.Date(),units="days")
Time difference of -160.5 days
> quarters(Sys.Date())
[1] "Q2"
>
```

Notez que **year( )** n'existe curieusement pas par défaut à ce jour dans R! Il en va de même des fonctions **day( )** et **month( )**… Heureusement que le package **lubridate** comble ce manquement. Ce dernier contient aussi la fonction **week( )** qui retourne le numéro de la semaine, **isoweek( )** qui retourne le numéro de la semaine conforme aux normes ISO et **wday( )** qui retourne le numéro du jour dans la semaine:

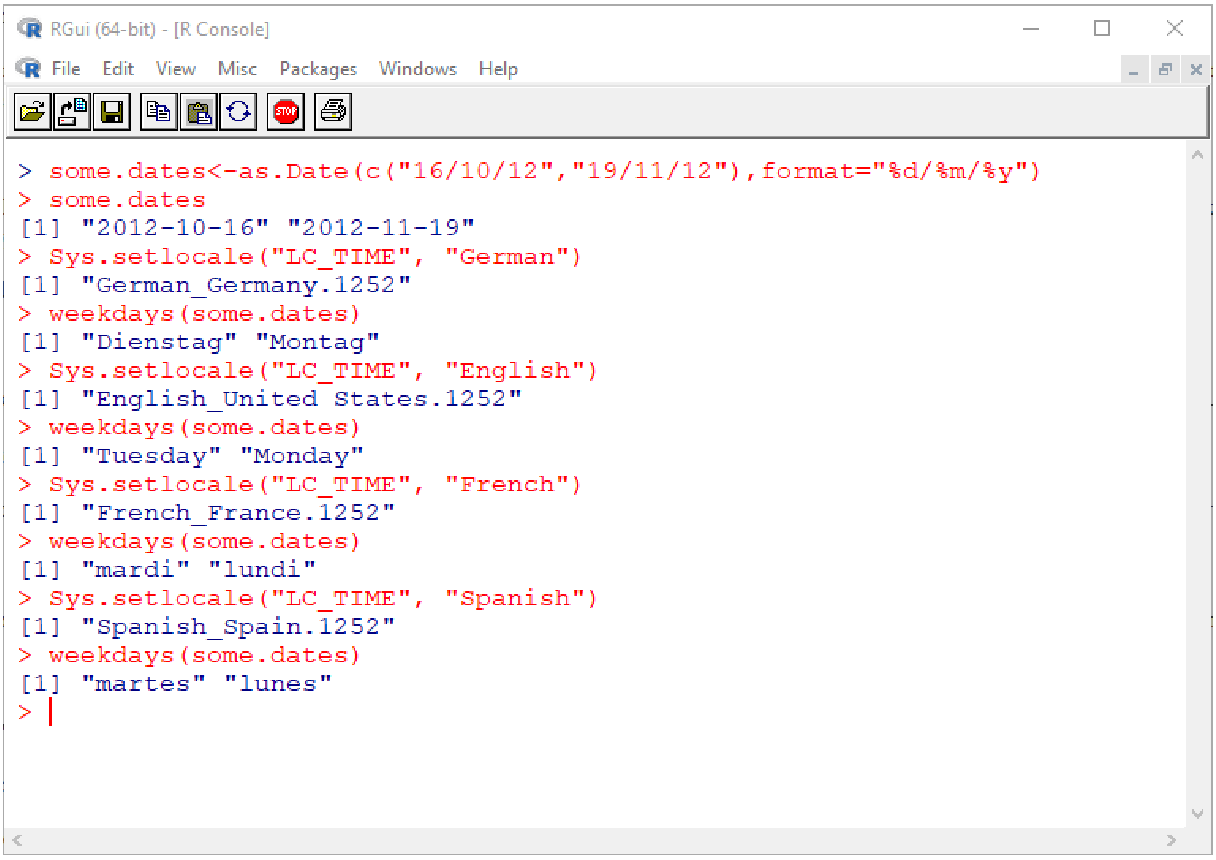
```
> as.integer(format(as.Date("2014-03-16"),"%W"))+1 #Dimanche attendu semaine 11
[1] 11
> as.integer(format(as.Date("2014-03-17"),"%W"))+1 #Dimanche, attendu semaine 12
[1] 12
>
> #Mieux encore
> library("lubridate")
>
> week(ymd("2014-03-16","2014-03-17","2014-03-18","2014-01-01"))
[1] 11 11 11  1
> isoweek(ymd("2014-03-16","2014-03-17","2014-03-18","2014-01-01"))
[1] 11 12 12  1
>
>
> #attention cependant aussi avec les jours!!!
> base::weekdays(as.Date("2014-03-16"))
[1] "Sunday"
>
> #avec lubridate on peut choisir le jour de départ
> options(lubridate.week.start = 1)
> wday(as.Date("2014-03-16"))
[1] 7
>
```

Comme nous pouvons le voir R prend les paramètres régionaux de la machine pour afficher le nom des jours. Nous verrons comment gérer cela avec un package un peu plus loin.

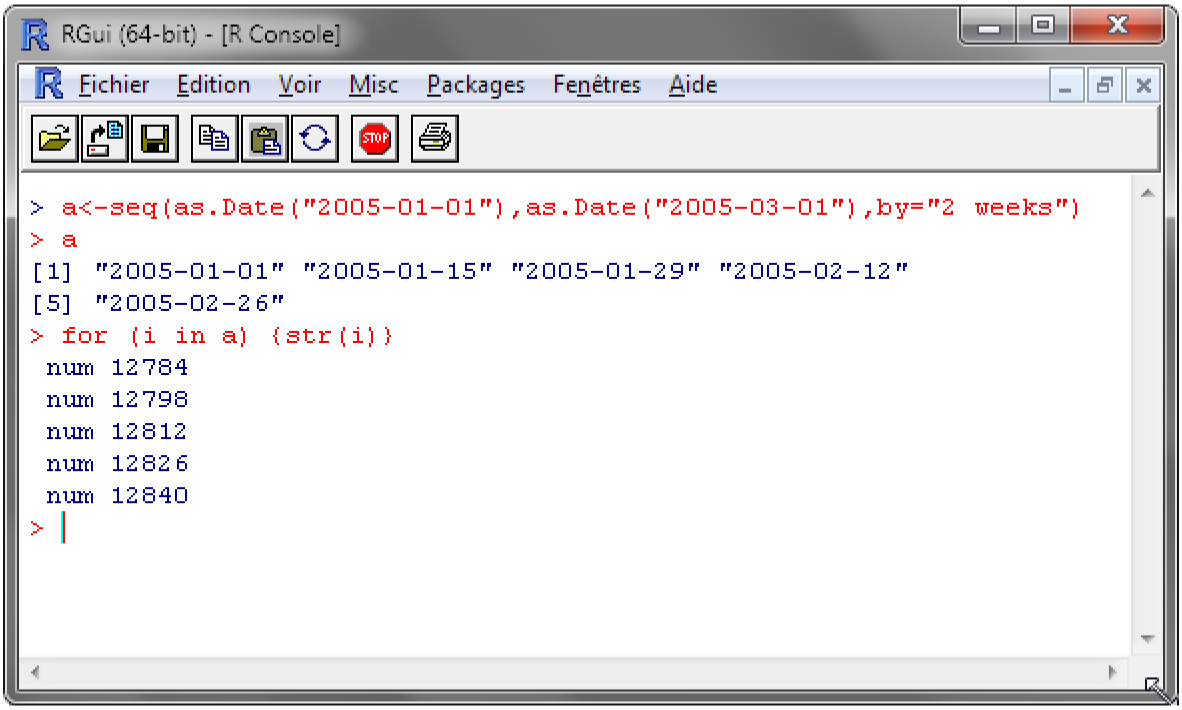
Créons maintenant quelques séquences classiques de dates en utilisant la commande **seq( )** déjà vue plus haut:


```
> seq(as.Date("2005-01-01"),as.Date("2005-01-31"),by="day")
 [1] "2005-01-01" "2005-01-02" "2005-01-03" "2005-01-04" "2005-01-05" "2005-01-06"
 [7] "2005-01-07" "2005-01-08" "2005-01-09" "2005-01-10" "2005-01-11" "2005-01-12"
[13] "2005-01-13" "2005-01-14" "2005-01-15" "2005-01-16" "2005-01-17" "2005-01-18"
[19] "2005-01-19" "2005-01-20" "2005-01-21" "2005-01-22" "2005-01-23" "2005-01-24"
[25] "2005-01-25" "2005-01-26" "2005-01-27" "2005-01-28" "2005-01-29" "2005-01-30"
[31] "2005-01-31"
> seq(as.Date("2005-01-01"),as.Date("2005-07-31"),by="month")
[1] "2005-01-01" "2005-02-01" "2005-03-01" "2005-04-01" "2005-05-01" "2005-06-01"
[7] "2005-07-01"
> seq(as.Date("2005-01-01"),as.Date("2005-03-30"),by=15)
[1] "2005-01-01" "2005-01-16" "2005-01-31" "2005-02-15" "2005-03-02" "2005-03-17"
> seq(as.Date("2005-01-01"),as.Date("2005-03-30"),by="2 weeks")
[1] "2005-01-01" "2005-01-15" "2005-01-29" "2005-02-12" "2005-02-26" "2005-03-12"
[7] "2005-03-26"
>
```

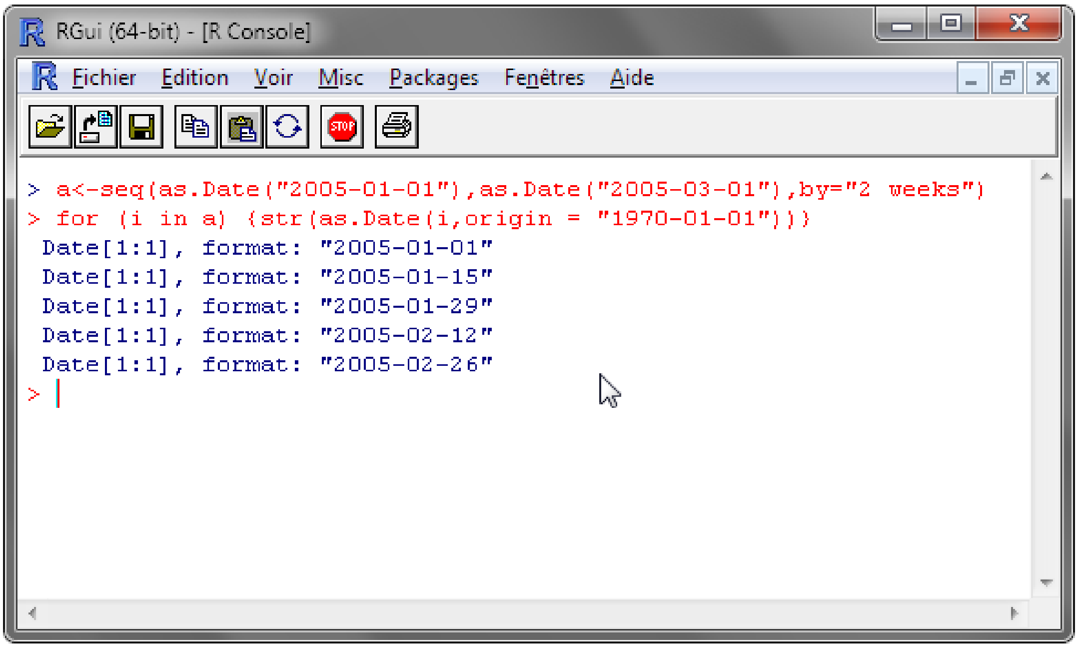
Maintenant voyons comment changer la langue d'affichage des noms des jours et des mois sous un environnement Microsoft Windows à l'aide de la fonction système **Sys.setlocale( )**:



Attention pour le moment où viendra le scripting car lorsque l'on boucle sur des vecteurs contenant des dates celles-ci sont à nouveau par défaut transformées en nombres:

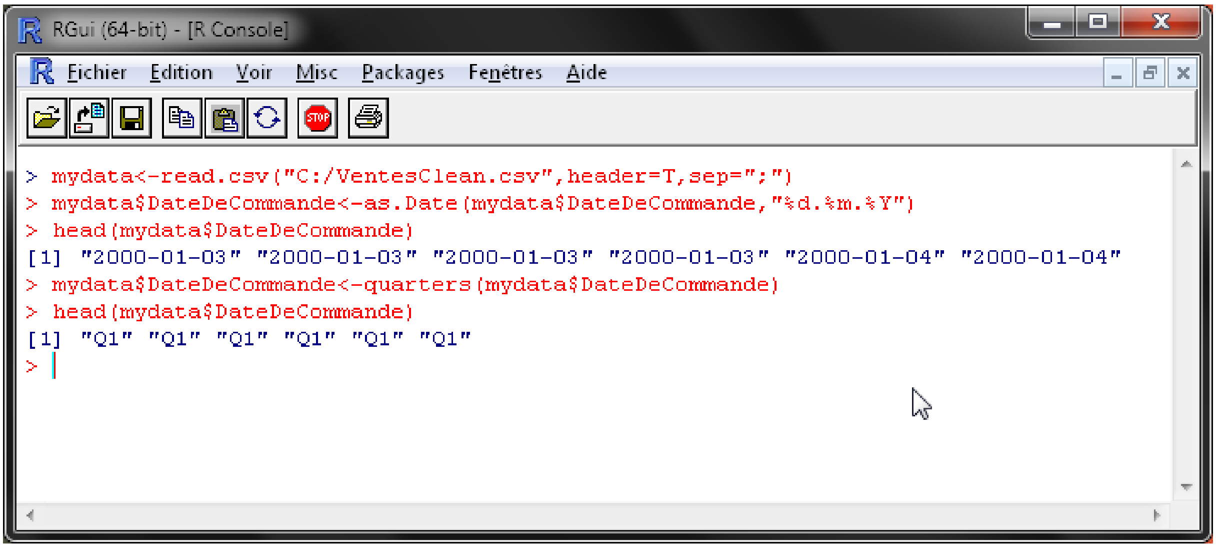


et pour corriger cela:



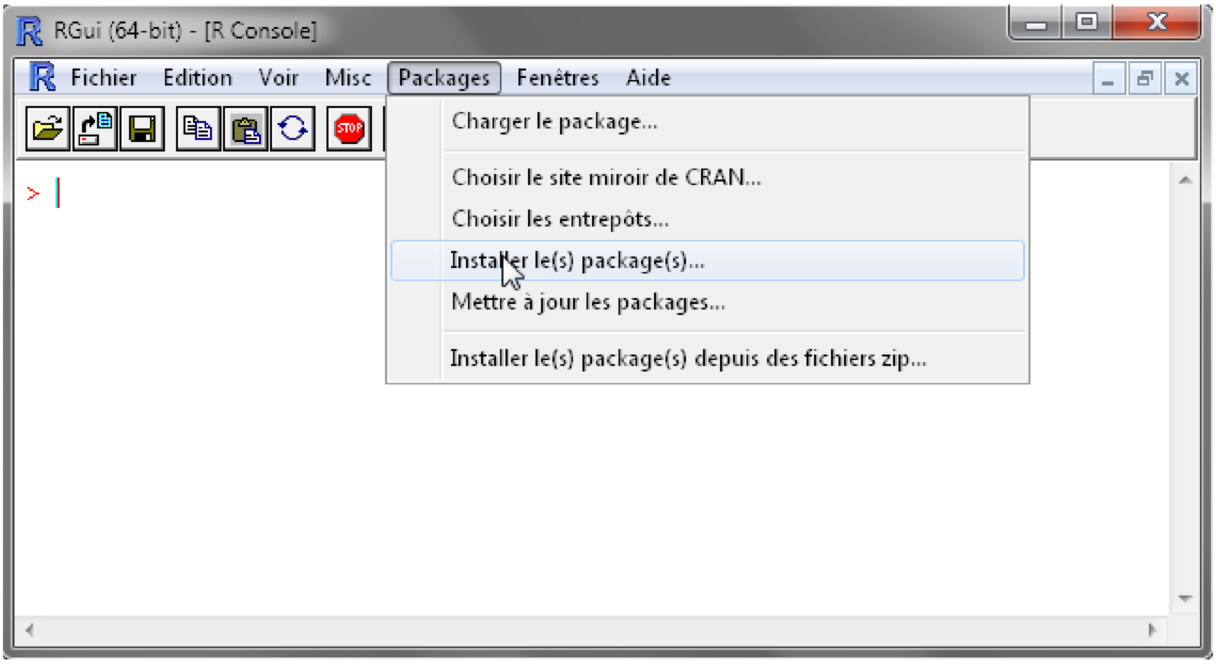
et comme vous pouvez le voir par la même occasion, l'origine de système de date par défaut de R est 1970-01-01 mais pouvons à l'aide du paramètre **origin** changer cela comme nous voulons.

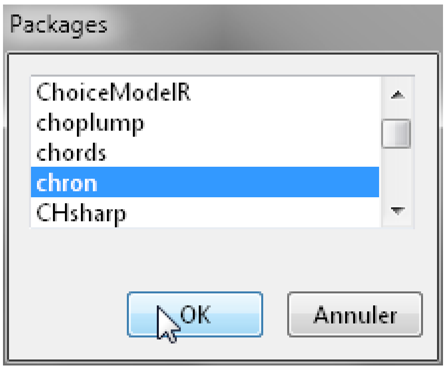
Nous pouvons aussi transformer une colonne de dates en trimestres avec la commande native **quarters( )** :



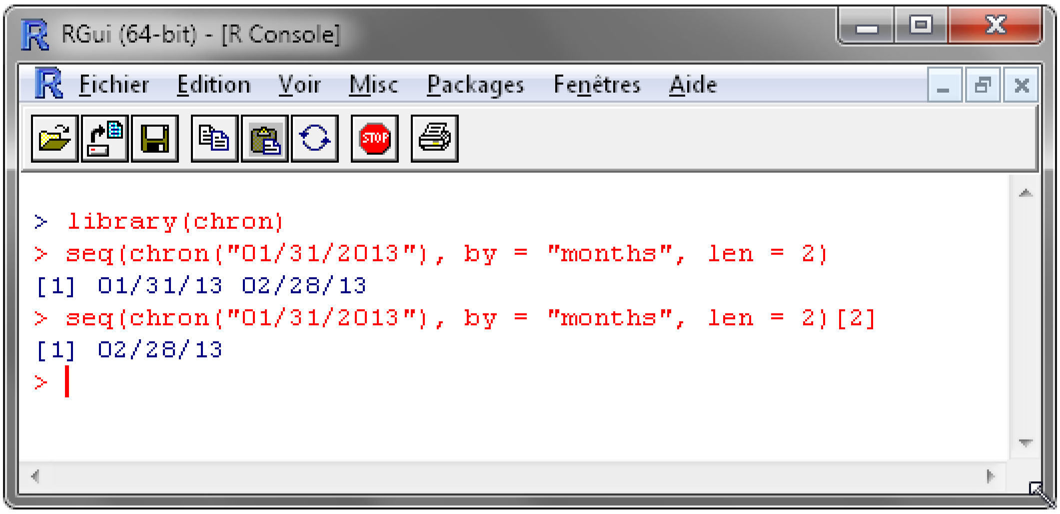
Maintenant voyons des manipulations triviales sur les dates mais qui semblent nécessiter des packages (je compléterai au fur et à mesure que je rencontrerai d'autres cas chez mes clients):

Nous allons installer le package **chron** pour faire de la somme de mois:



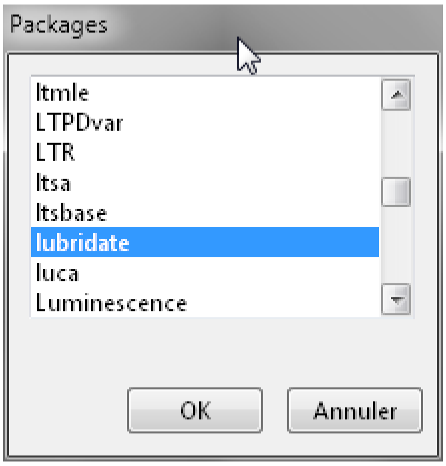


et ensuite en utilisant la commande **chron( )** avec un format d'entrée ridicule...

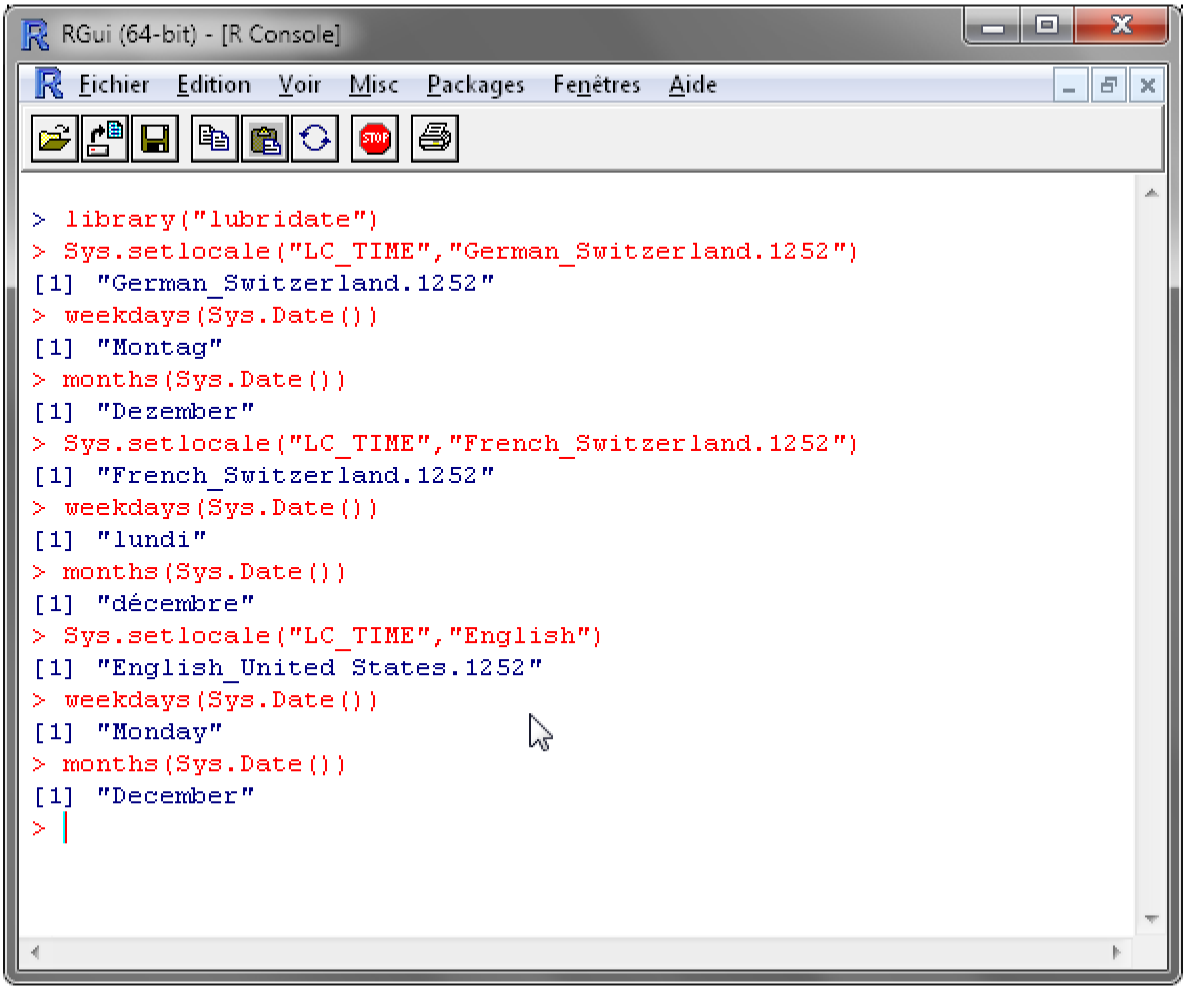


Il y a d'autres packages qui ont des options similaires pour ajouter des mois à une date.

Sinon pour gérer des dates dans d'autres langues, on installera le package **lubridate**:



Voyons l'exemple classique de la suisse (français, allemand, anglais):



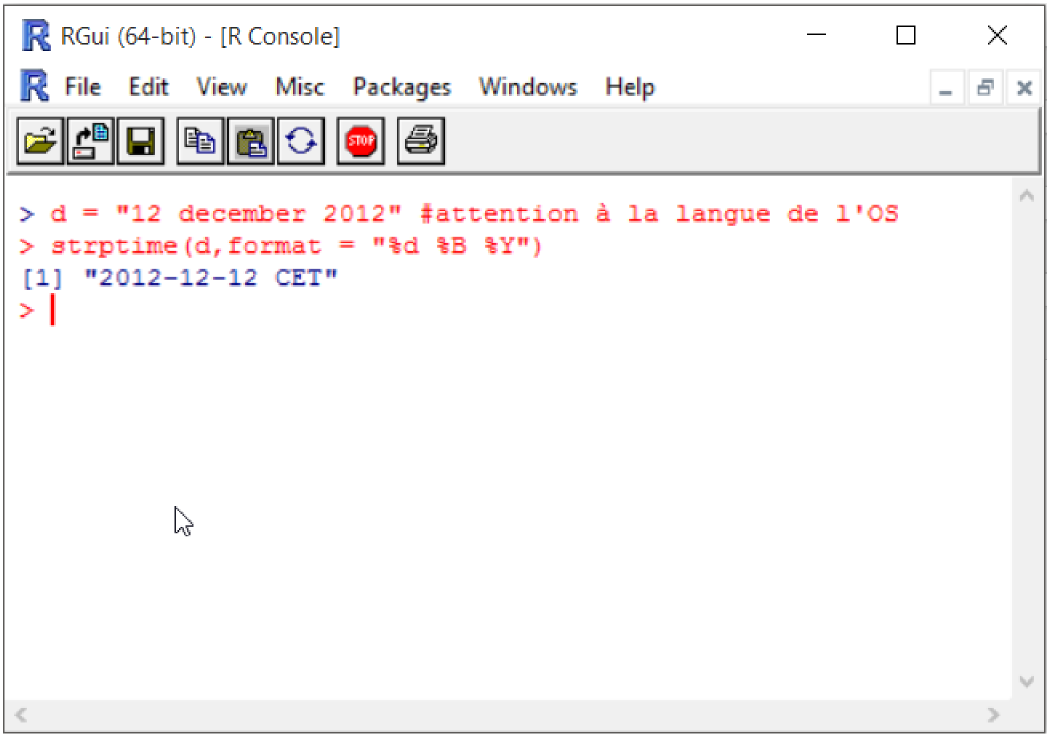
La libraire **lubridate** est aussi très utile pou convertir différentes formats de dates en un seul système de dates comme l'illustre l'exemple ci-dessous:

```
> library("lubridate")
>
> (data <- data.frame(initialDiagnose = c("14.01.2009", "9/22/2005",
+          "4/21/2010", "28.01.10", "09.01.2009", "3/28/2005",
+          "04.01.2005", "2005-01-04", "Created on 9/17/2010", "03 01 2010")))
        initialDiagnose
1            14.01.2009
2             9/22/2005
3             4/21/2010
4              28.01.10
5            09.01.2009
6             3/28/2005
7            04.01.2005
8            2005-01-04
9  Created on 9/17/2010
10           03 01 2010
>
> parse_date_time(data$initialDiagnose, orders = c('mdy', 'dmy','y-m-d'))
 [1] "2009-01-14 UTC" "2005-09-22 UTC" "2010-04-21 UTC" "2010-01-28 UTC" "2009-01-09 UTC"
 [6] "2005-03-28 UTC" "2005-01-04 UTC" "2005-01-04 UTC" "2010-09-17 UTC" "2010-03-01 UTC"
>
```

Pour obtenir les dates de chômées de quelques bourses nous avons la commande **holiday*( )** du package **timeDate** de Rmetrics:

```
> library(timeDate)
> holidayZURICH()
Zurich
 [1] [2014-01-01] [2014-01-02] [2014-04-18] [2014-04-21] [2014-04-21] [2014-05-01]
 [7] [2014-05-29] [2014-06-09] [2014-08-01] [2014-09-08] [2014-12-25] [2014-12-26]
>
```

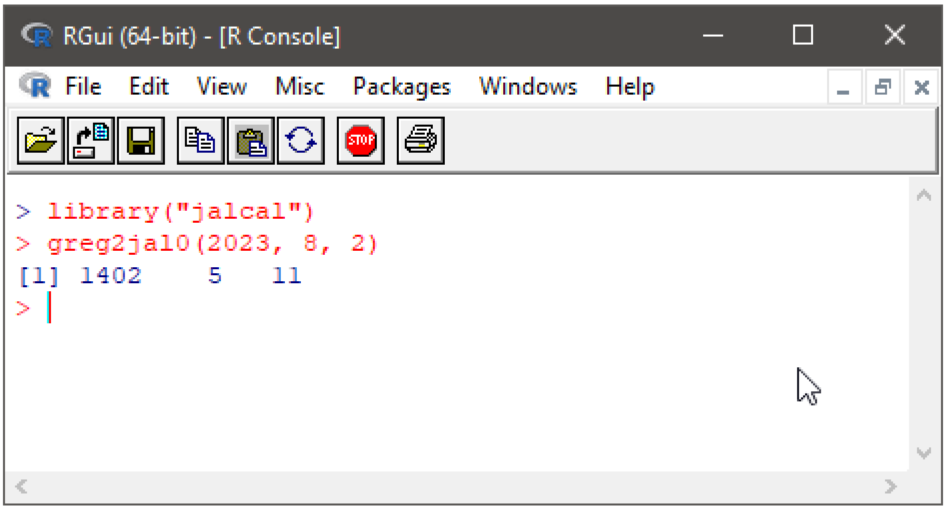
Enfin, sans package, voyons comme changer une date en toutes lettres en une date R avec la commande **strptime( )** :



Nous savons que plus de 180 calendriers religieux sont utilisés dans le Monde puisqu'il y a 45 religions monothéistes révélées et plus 28'000 polythéistes. La conversion de la majorité des religions est assez simple mais le calendrier Hijri (appelé aussi calendrier Perse, ou Solaire Hijri ou encore Hijri Shamsi) qui sont les calendriers officiels de l'Iran et de l'Afghanistan ont des règles un peu tordues…

Heureusement il existe un package nommé **jalcal** pour convertir les dates du calendrier grégorien ou calendrier Hijri.

Voyons un exemple:



Pour ceux qui souhaitent savoir comment l'année est calculée, voici le code source disponible dans le GitHub du package:

```r
do.greg2jal <- function(year, month, day)
{
  year <- as.integer(year)
  month <- as.integer(month)
  day <- as.integer(day)
  if (month < 1 | month > 12)
    stop("month is outside the range 1-12")
  if (day < 1 | day > 31)
    stop("day is outside the range 1-31")

  gdm <- c(0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334)
  year2 <- ifelse(month > 2, year + 1, year)
  days <- 355666 +
    365 * year +
    (3 + year2) %/% 4 -
    (year2 + 99) %/% 100 +
    (year2 + 399) %/% 400 +
    day +
    gdm[month]

  jyear <- -1595 + 33 * (days %/% 12053)
  days <- days %% 12053

  jyear <- jyear + 4 * (days %/% 1461)
  days <- days %% 1461
  if (days > 365)
  {
    jyear <- jyear + (days - 1) %/% 365
    days <- (days - 1) %% 365
  }

  if (days < 186)
  {
    jmonth <- 1 + (days %/% 31)
    jday <- 1 + days %% 31
  } else {
    jmonth <- 7 + (days - 186) %/% 30
    jday <- 1 + (days - 186) %% 30
  }

  return(c(jyear, jmonth, jday))
}
```

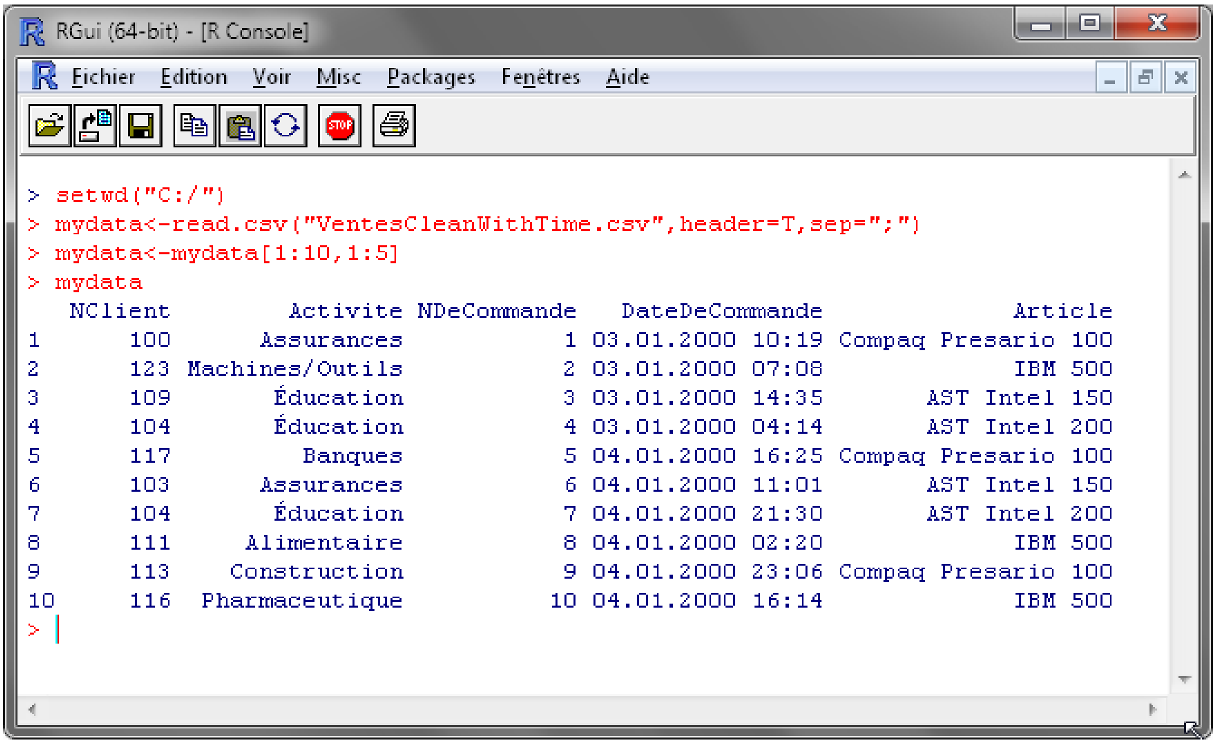
Comme on peut le voir c'est pas compliqué mais pas trivial non plus…

Exemple 19.: Manipulations de dates avec du temps

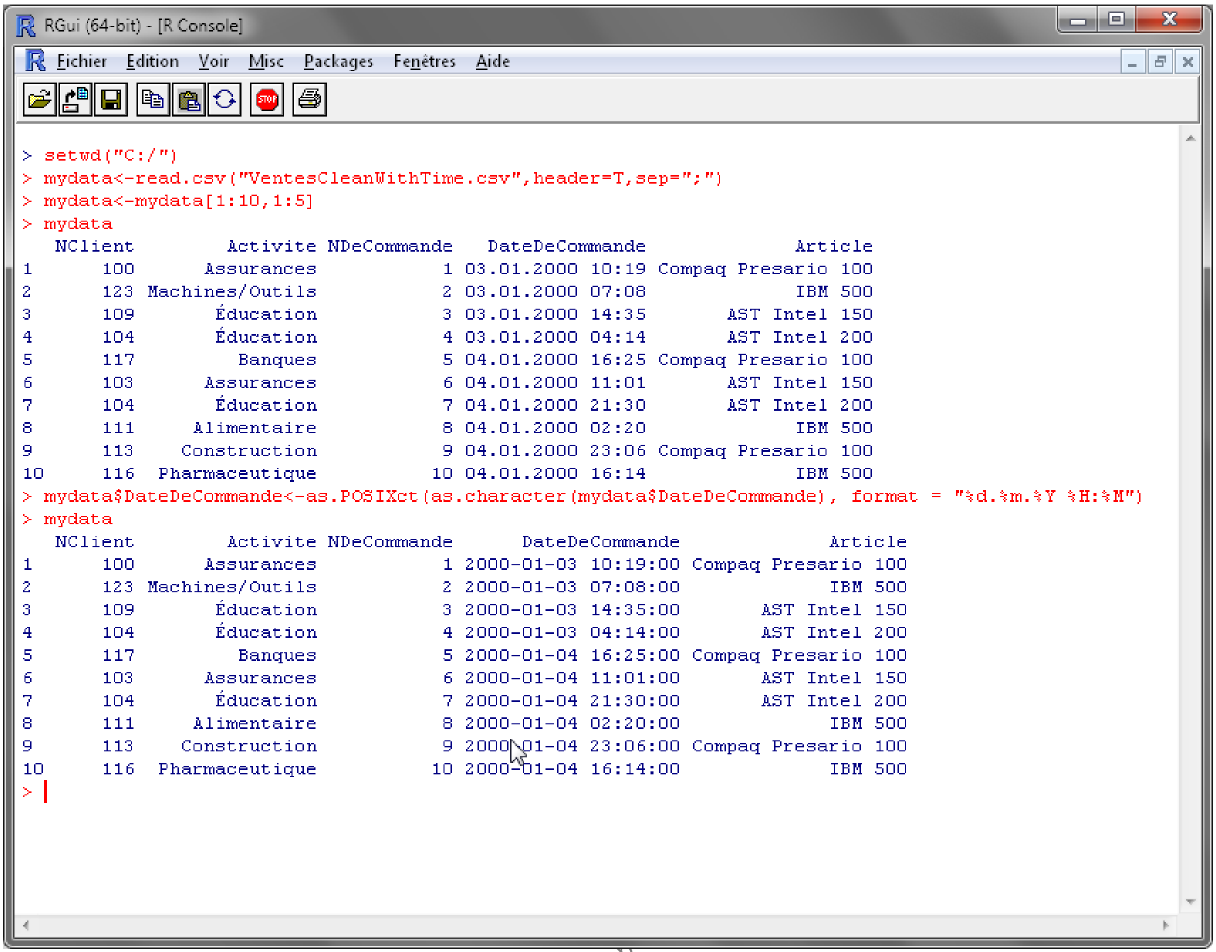
R 3.0.2

Pour travailler sur les heures et minutes nous allons faire encore une fois une entorse en supposant que nous avons déjà étudié l'import de fichiers *.csv (voir plus loin pour les détails relativement à ce sujet).

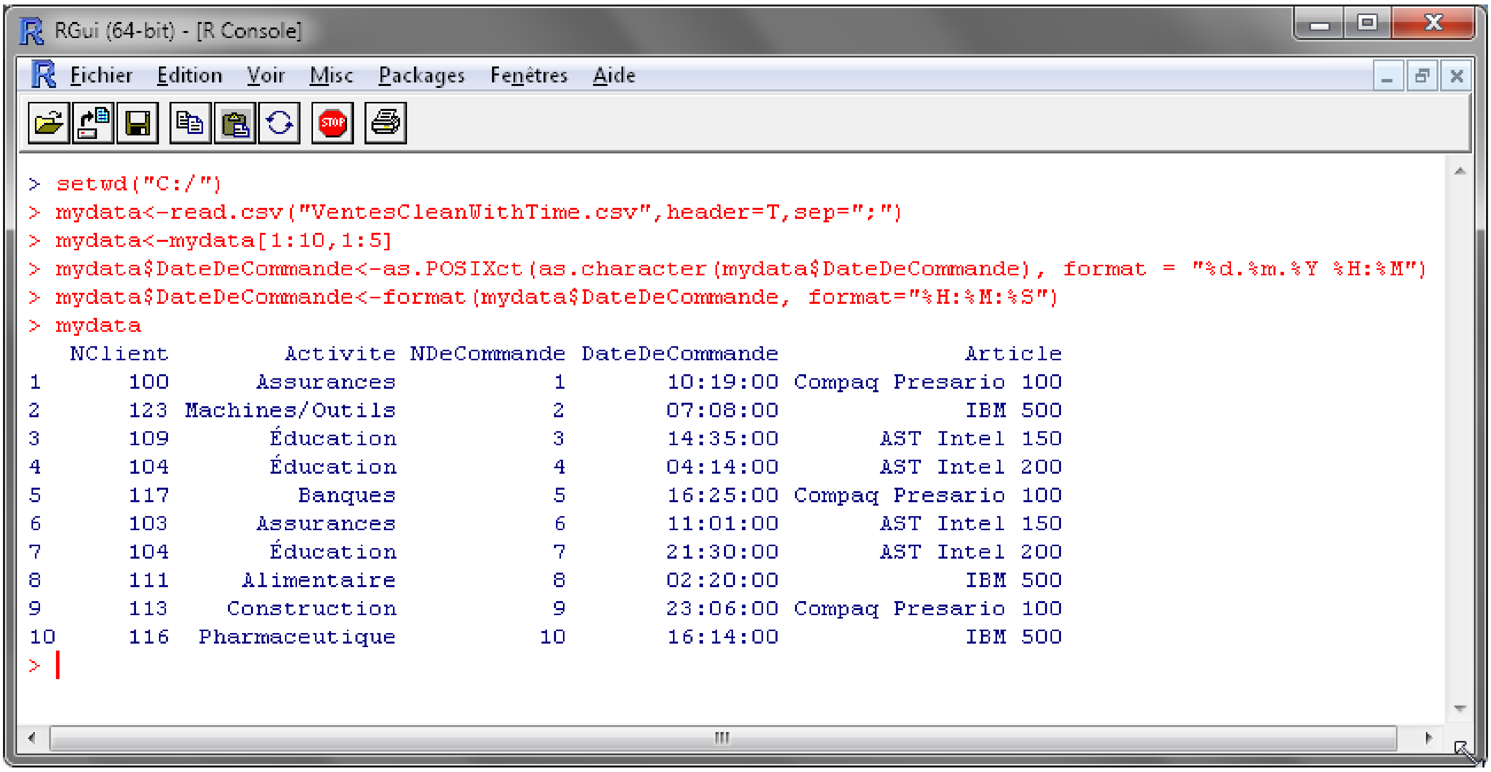
Considérons donc l'import du sous-ensemble du fichier *.csv suivant:



et nous allons nous assurer que R comprenne la colonne *DateDeCommande* comme étant de type date et heure sinon quoi nous aurons des problèmes avec l'analyse des séries chronologiques à l'aide de la commande **as.POSIXct( )** (qui encode les dates et heures en nombre de secondes écoulées depuis l'origine, le 1er janvier 1970):



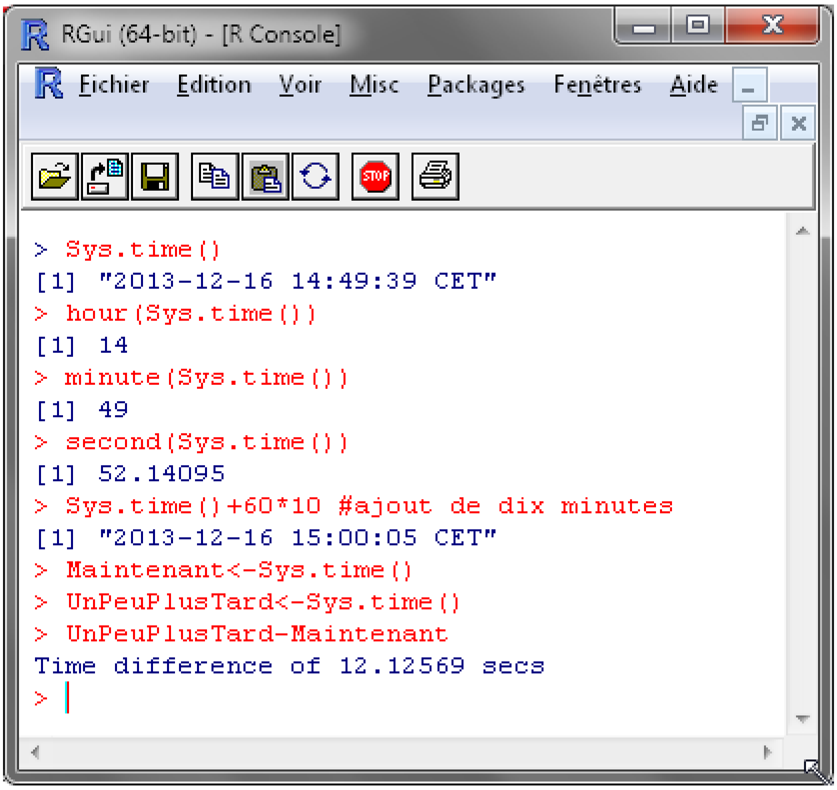
Sinon pour n'avoir que le temps en utilisant la commande **format( )** :



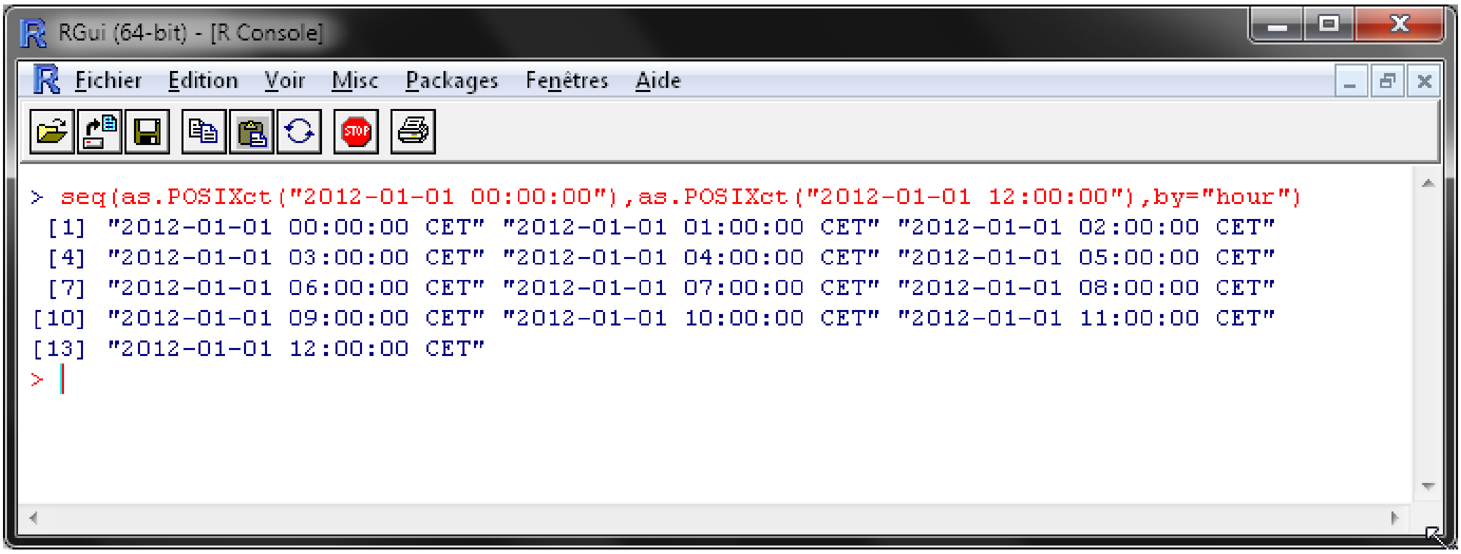
Voici la liste des formats:

Code	Meaning	Code	Meaning
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Decimal date
%H	Decimal hours (24 hour)	%I	Decimal hours (12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute	%p	Locale-specific AM/PM
%S	Decimal second	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT	%Z	Time zone (character)

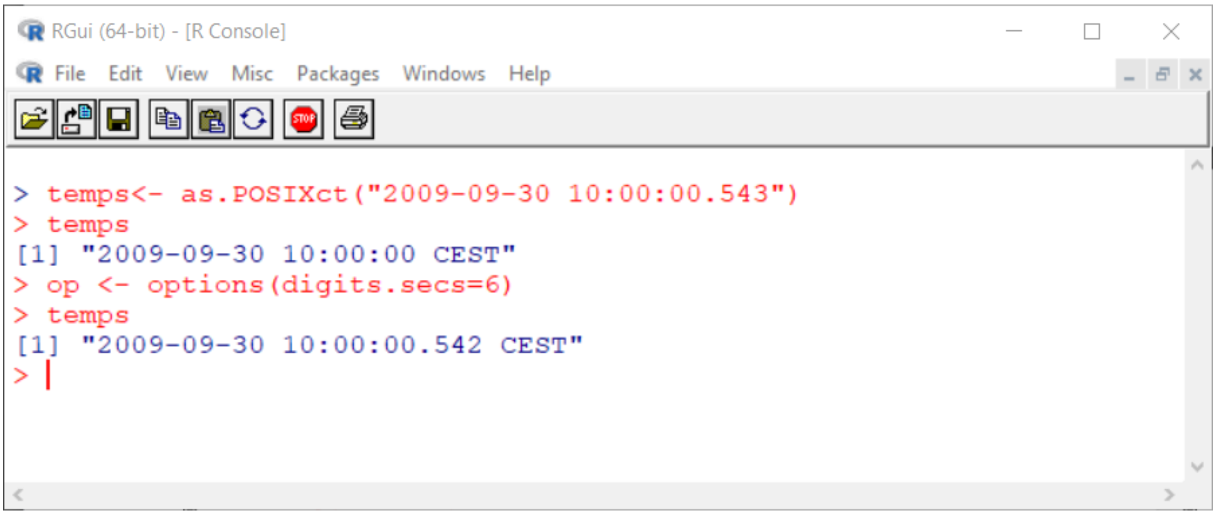
Et quelques autres petites commandes comme **Sys.time( )** , **hour( )** , **minute( )** , **second( )** :



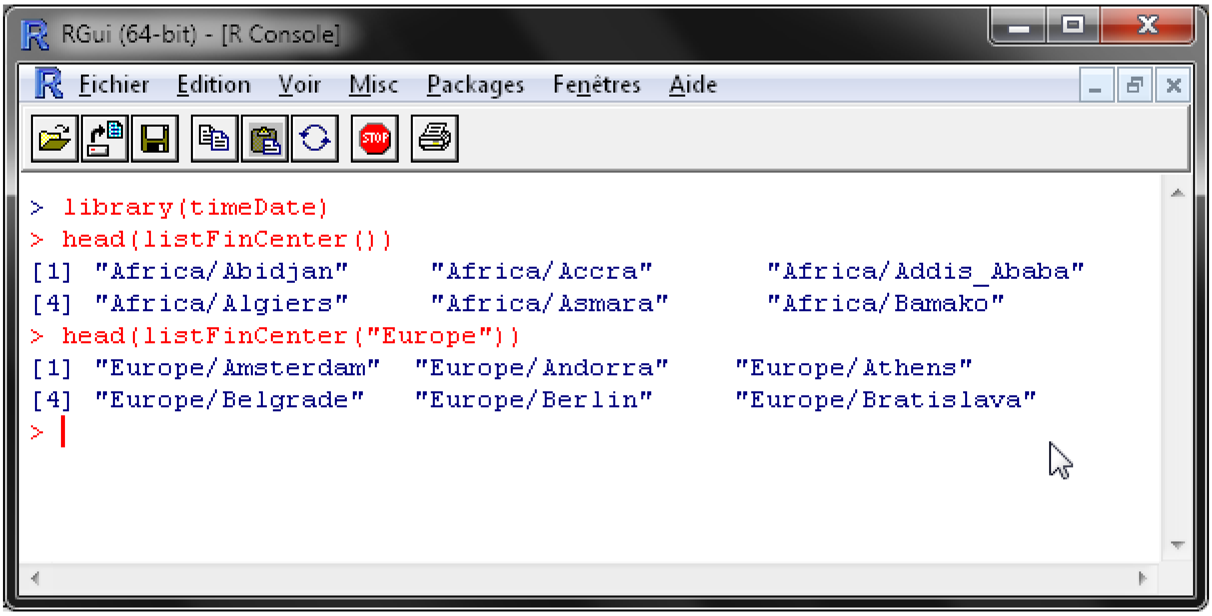
et pour créer des séquences de temps (utile pour les graphiques):



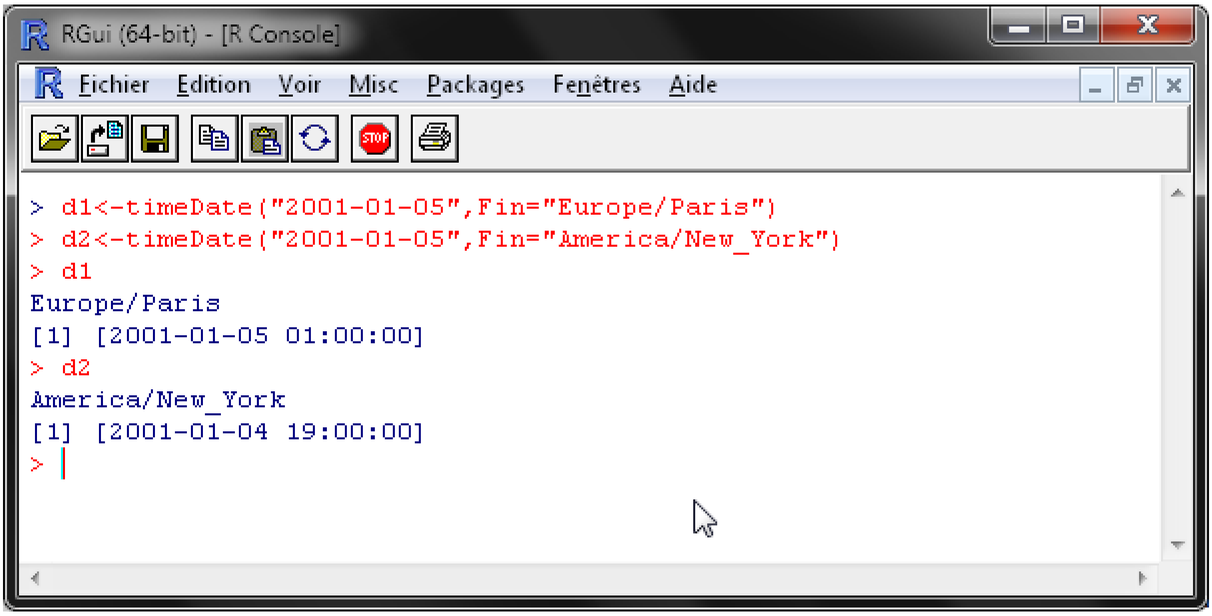
Notez que si vous travaillez dans certains domaines de la finance, les millisecondes vous seront utiles. Dès lors en faisant usage du paramètre **digits.secs** des options système:



Sinon toujours du package **timeDate( )** de RMetrics nous pouvons avoir les heures de différentes villes de référence pour les fuseaux horaires la finance:



Ensuite voici un exemple d'utilisation:



POSIXct et POSIXcl

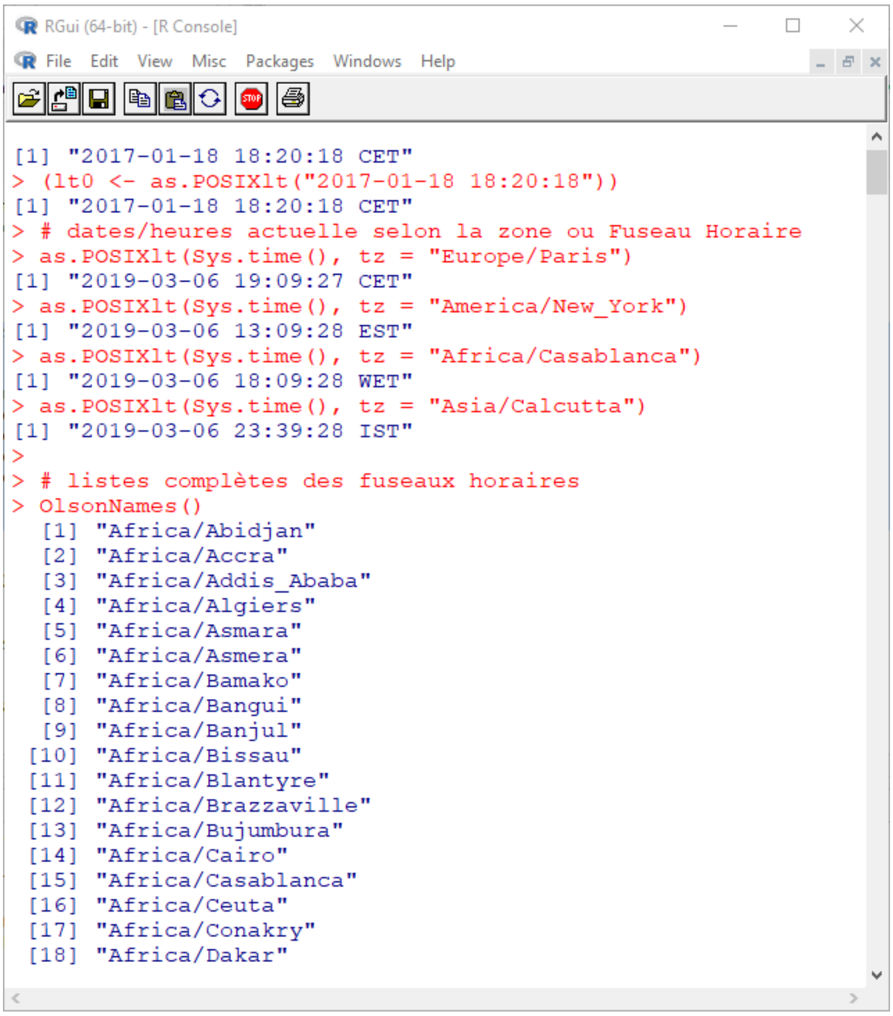
Il y a des livres entiers dédiés à la manipulation des dates avec R. Mais revenons cependant un peu sur les deux objets POSIXct et POSIXcl. Voyons déjà qu'ils sont effectivement stockés de manière différence (le premier en tant que numérique, secondes depuis le 1er Janvier 1970 et l'autre comme un objet de type liste):

```
> # la date et heure actuelle
> (now <- Sys.time())
[1] "2019-03-06 19:06:27 CET"
> # coercer en POSIXct()
> (dt.ct <- as.POSIXct(now))
[1] "2019-03-06 19:06:27 CET"
> class(dt.ct)
[1] "POSIXct" "POSIXt"
> mode(dt.ct)
[1] "numeric"
> unclass(dt.ct)
[1] 1551895587
> # coercer en POSIXlt
> (dt.lt <- as.POSIXlt(now))
[1] "2019-03-06 19:06:27 CET"
> class(dt.lt)
[1] "POSIXlt" "POSIXt"
> mode(dt.lt)
[1] "list"
>
```

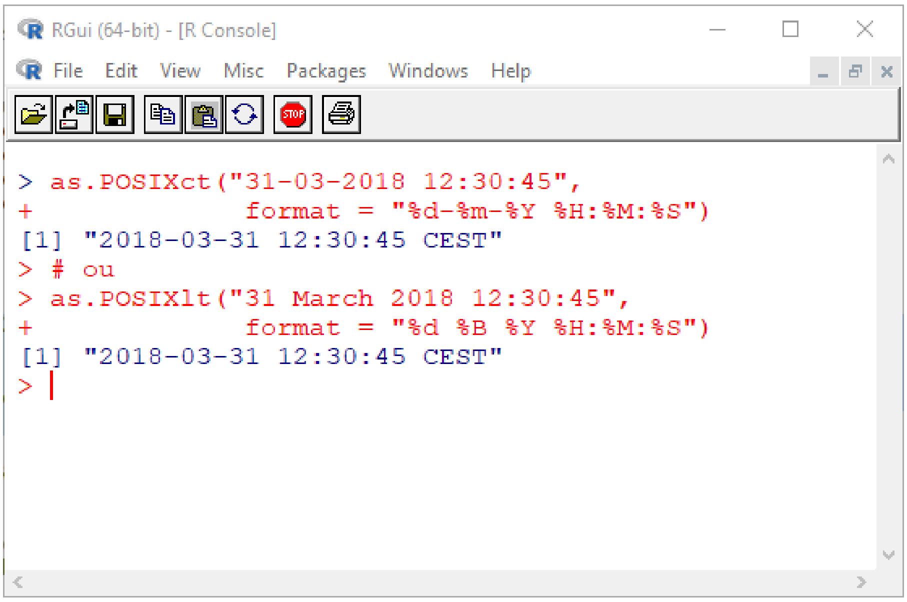
Regardons plus en détails la structure d'une date en POSIXlt:

```
> # obtenir les noms avec unlist()
> names(unclass(dt.lt))
 [1] "sec"    "min"    "hour"   "mday"   "mon"    "year"
 [7] "wday"   "yday"   "isdst"  "zone"   "gmtoff"
> # extraire les secondes
> dt.lt$sec
[1] 27.19556
> # extraire le jour du mois
> dt.lt$mday
[1] 6
>
```

Le type d'objet POSIXlt permet de spécifier le fuseau horaire sans passer par un package:



Que se passe-t-il si le format de date et heure fourni n'est pas standard (ISO 8601)? Là aussi, on utilise l'argument *format* qui sert à spécifier le format utilisé en utilisant **as.POSIXct( )** et **as.POSIXlt( )**:



Maintenant au niveau des opérations sur les objets POSIXT, signalons que nous disposons des fonction **strptime( )** et **strftime( )** qui permettent respectivement de passer d'une chaînes de caratères à un objet POSIXt et d'un objet POSIXt à une chaîne de caractères:

```
> strftime(as.POSIXlt("31 March 2018 12:30:45",
+                  format = "%d %B %Y %H:%M:%S"),"%d %b %Y %H:%M:%S")
[1] "31 Mar 2018 12:30:45"
> strptime("31 Mar 2018 12:30:45","%d %b %Y %H:%M:%S")
[1] "2018-03-31 12:30:45 CEST"
> 
```

Certaines opérations arithmétiques sont possibles comme l'addition et la soustraction:

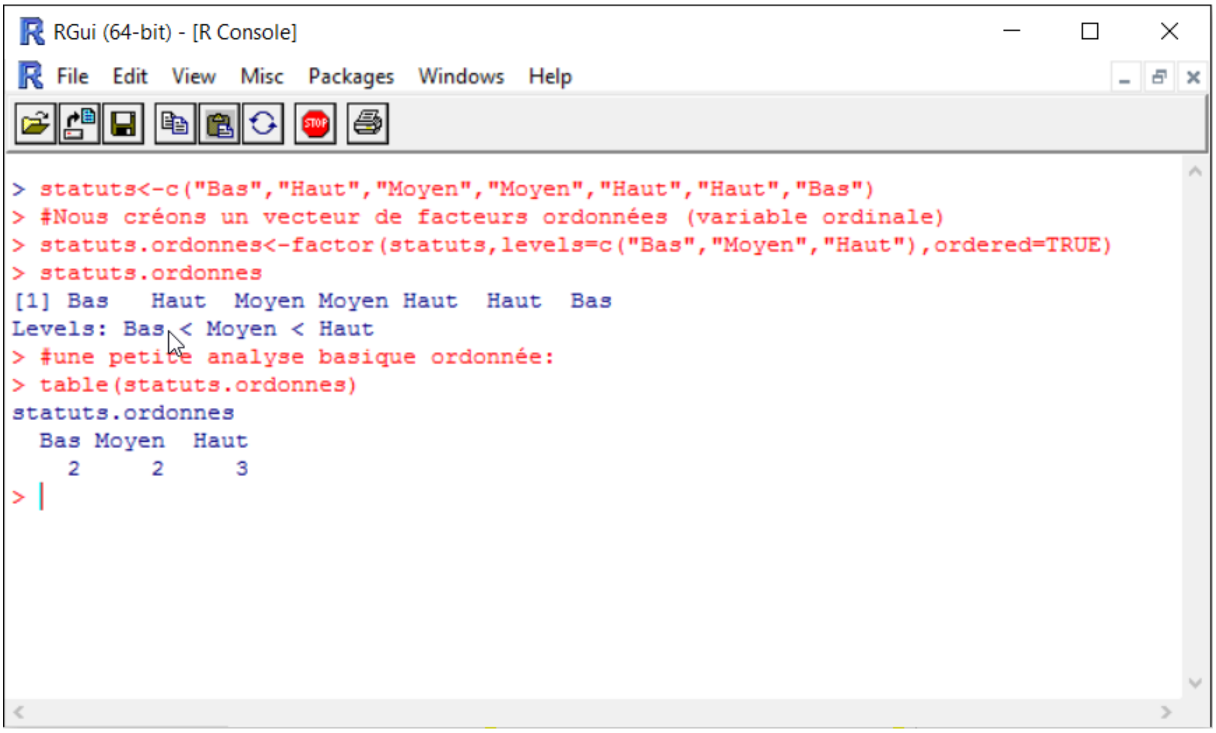
```
> (tmp0 <- as.POSIXlt("2017-01-18 18:20:18"))
[1] "2017-01-18 18:20:18 CET"
> (tmp1 <- as.POSIXlt("2017-02-19 20:20:18"))
[1] "2017-02-19 20:20:18 CET"
> ## différence de temps entre deux dates
> # différence en secondes
> difftime(tmp1, tmp0, units = "secs")
Time difference of 2772000 secs
> # différence en heures
> difftime(tmp1, tmp0, units = "mins")
Time difference of 46200 mins
> # différence en heures
> difftime(tmp1, tmp0, units = "hours")
Time difference of 770 hours
> # différence en heures
> difftime(tmp1, tmp0, units = "days")
Time difference of 32.08333 days
> # différence en heures
> difftime(tmp1, tmp0, units = "weeks")
Time difference of 4.583333 weeks
> ## ajout de temps à une date
> # ajouter 30 secondes
> tmp0 + 30
[1] "2017-01-18 18:20:48 CET"
> # ajouter une heure 30 min
> tmp0 + as.difftime("01:30:00", "%H:%M:%S")
[1] "2017-01-18 19:50:18 CET"
> 
```

Il y a aussi le package **lubridate** qui est toute une histoire à lui seul…

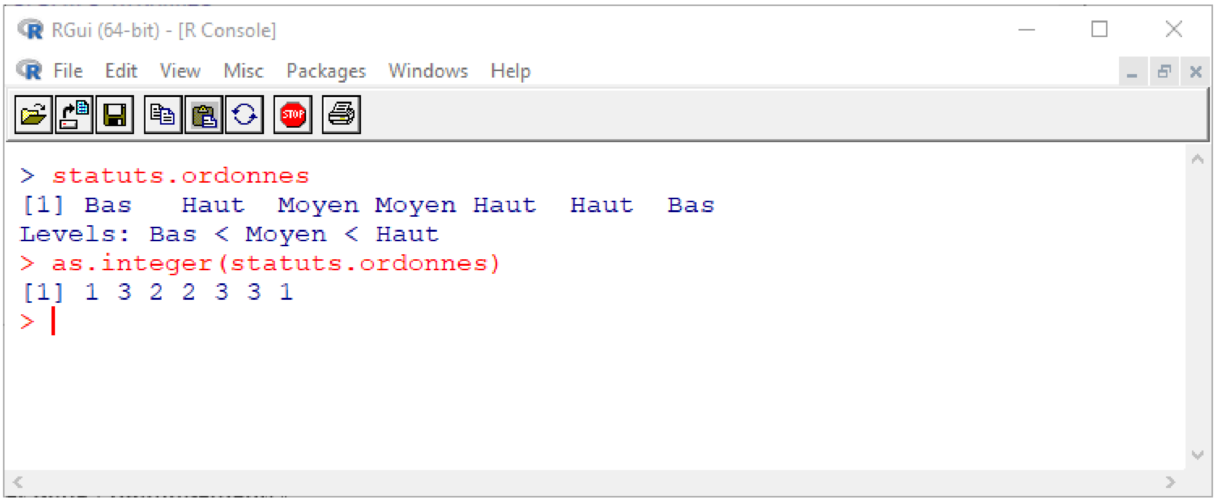
Exemple 20.: Manipulations de variables ordinales (facteurs)

R 3.2.1

Les variables ordinales permettant de traiter des variables triées dans un ordre particulier dans des tableaux statistiques ou des graphiques. Ceci se fait en utilisant la commande **factor( )** :



Nous pouvons reconvertir des facteurs en numériques (même si R en arrière-plan les traites de toute façon comme des valeurs numériques!):



et donc graphiquement:

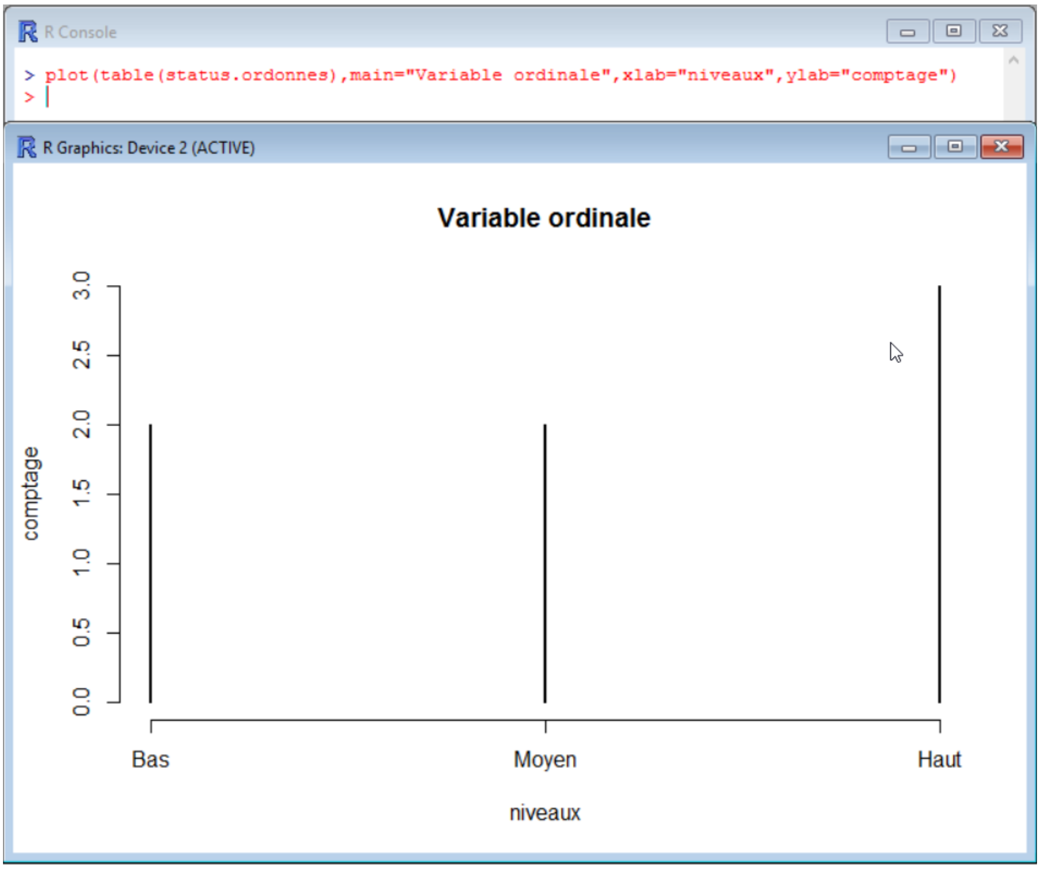


Figure 6 Graph de comptage de variables ordinales à tiges

ou:

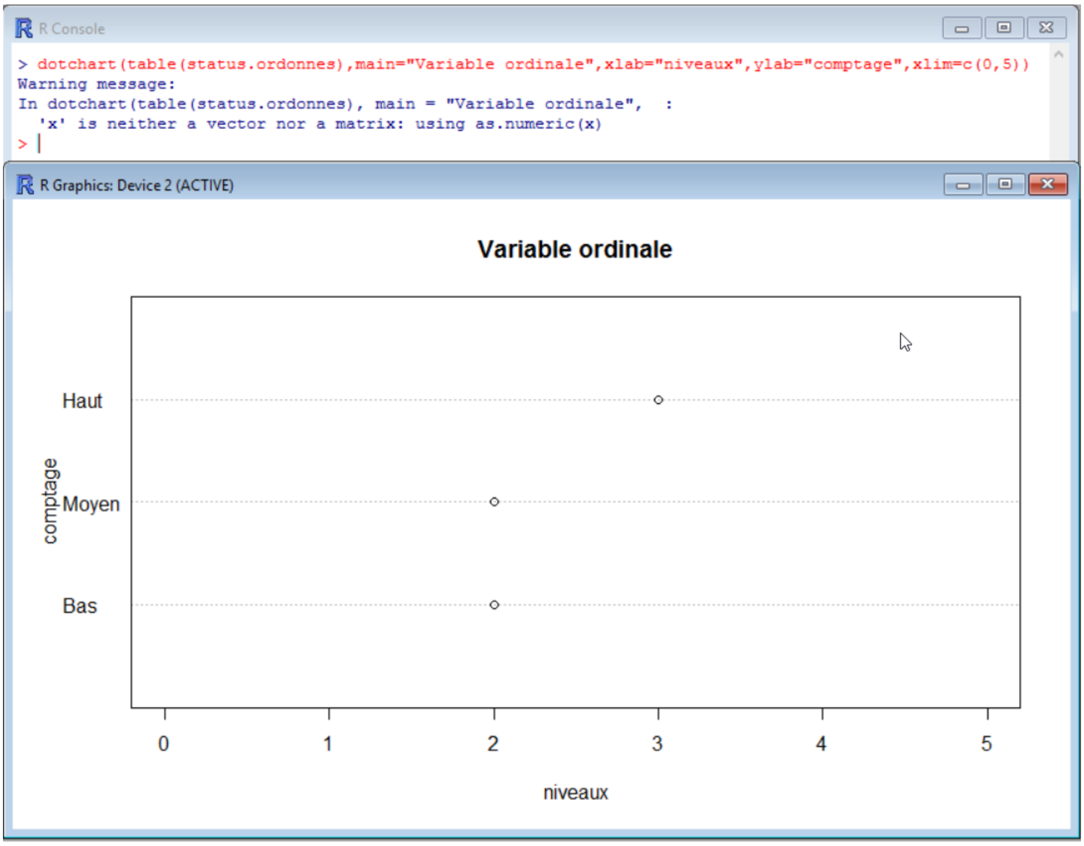


Figure 7 Graph de comptage de variables ordinales de type dotplot horizontal

Ou encore:

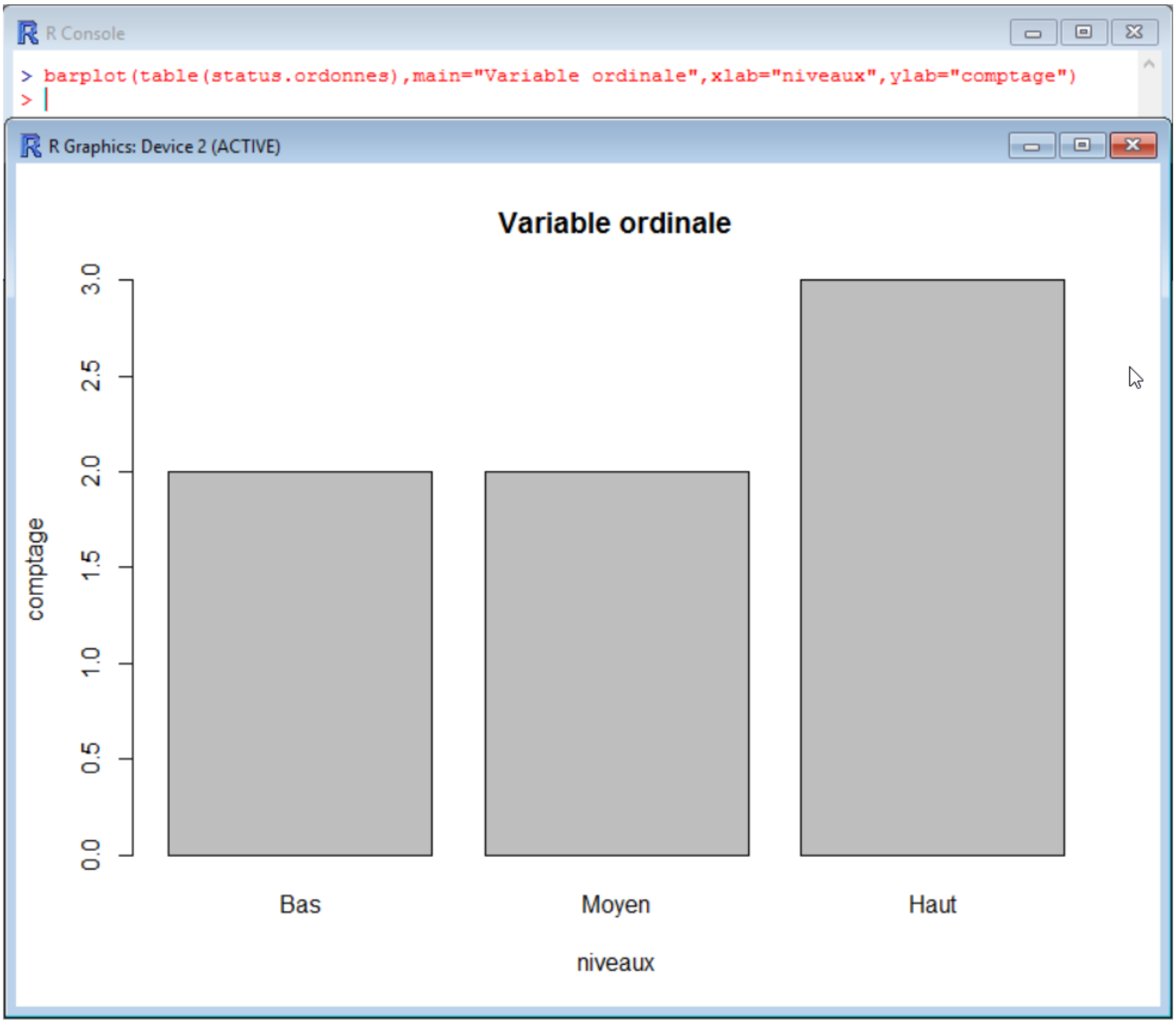


Figure 8 Graph de comptage de variables ordinales sous forme de diagamme à barres

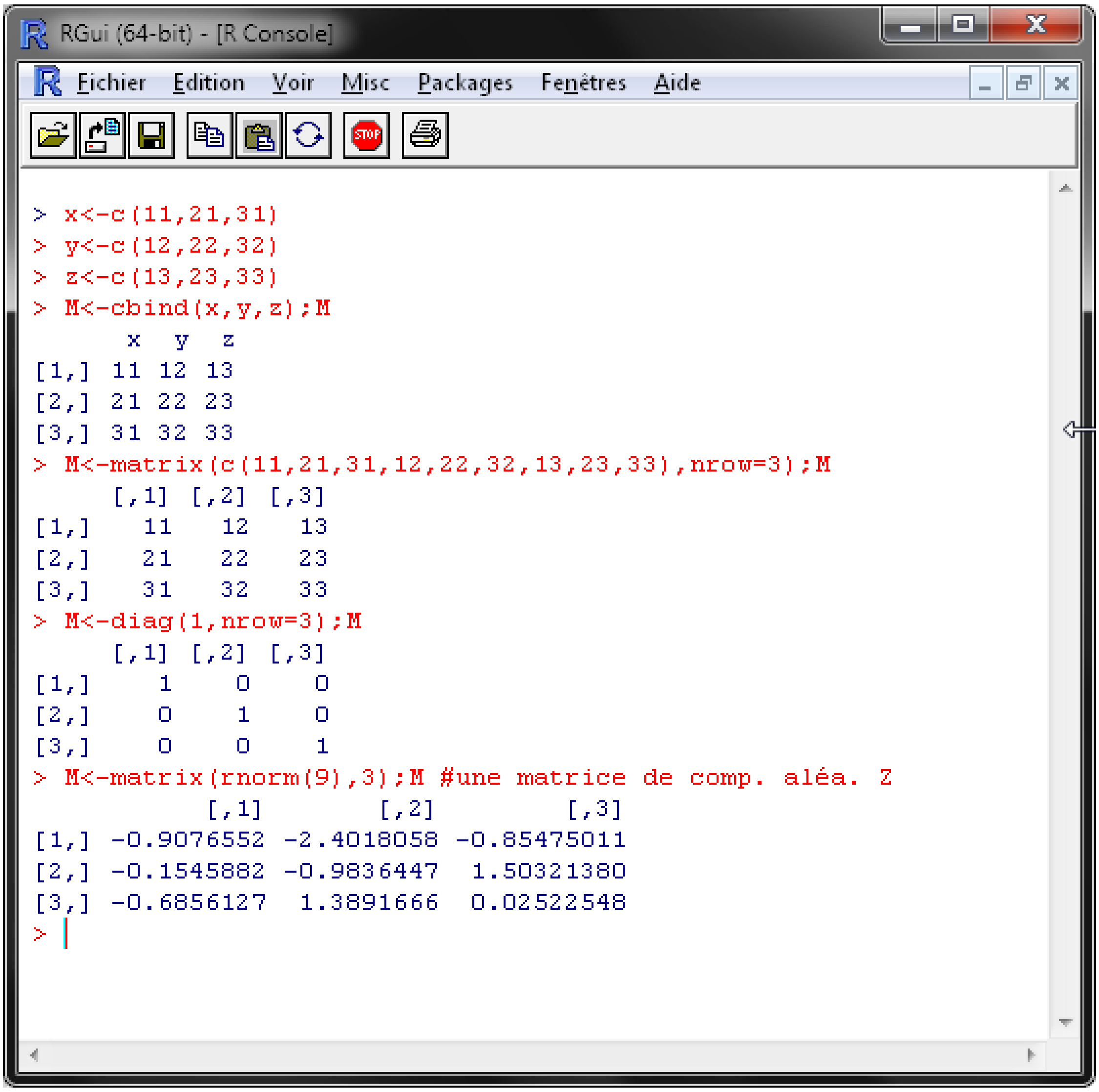
Exemple 21.: Manipulations de variables matricielles (matrices)

R 3.0.2

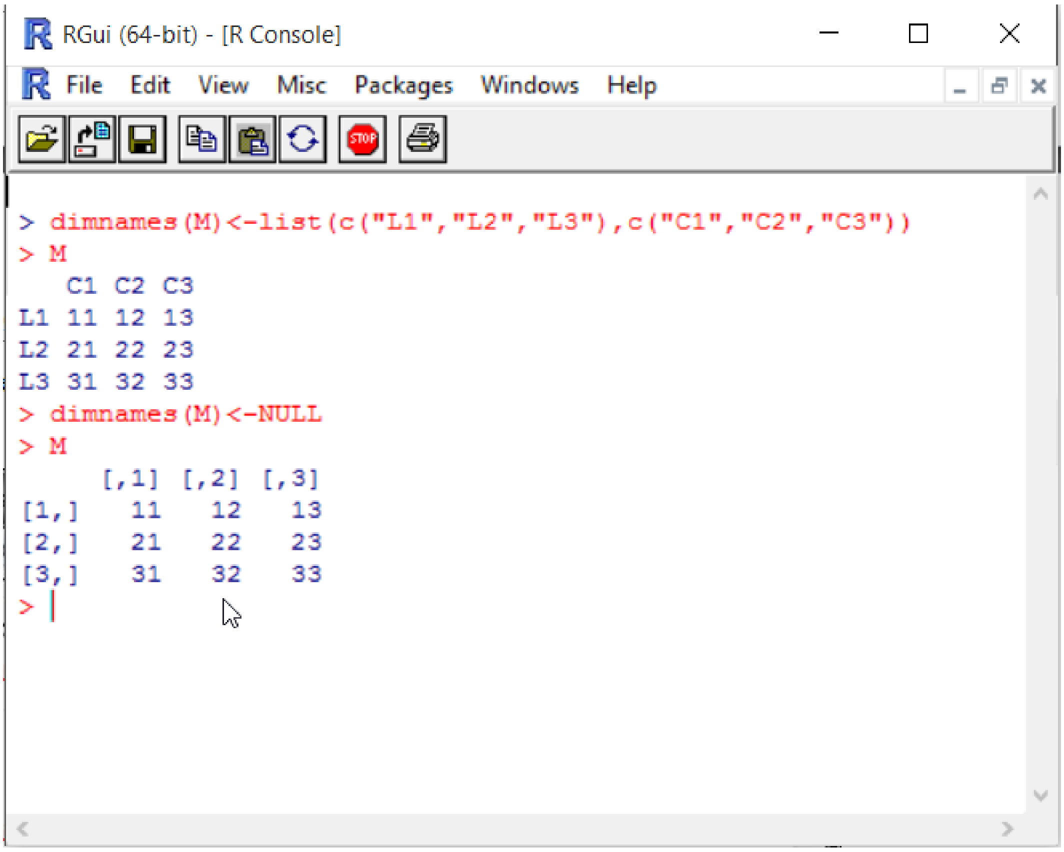
Encore une fois... inutile de préciser l'importance des matrices dans l'analyse financière. Il est donc important aussi de savoir les manipuler pour plus tard faire du scripting.

Création de matrices

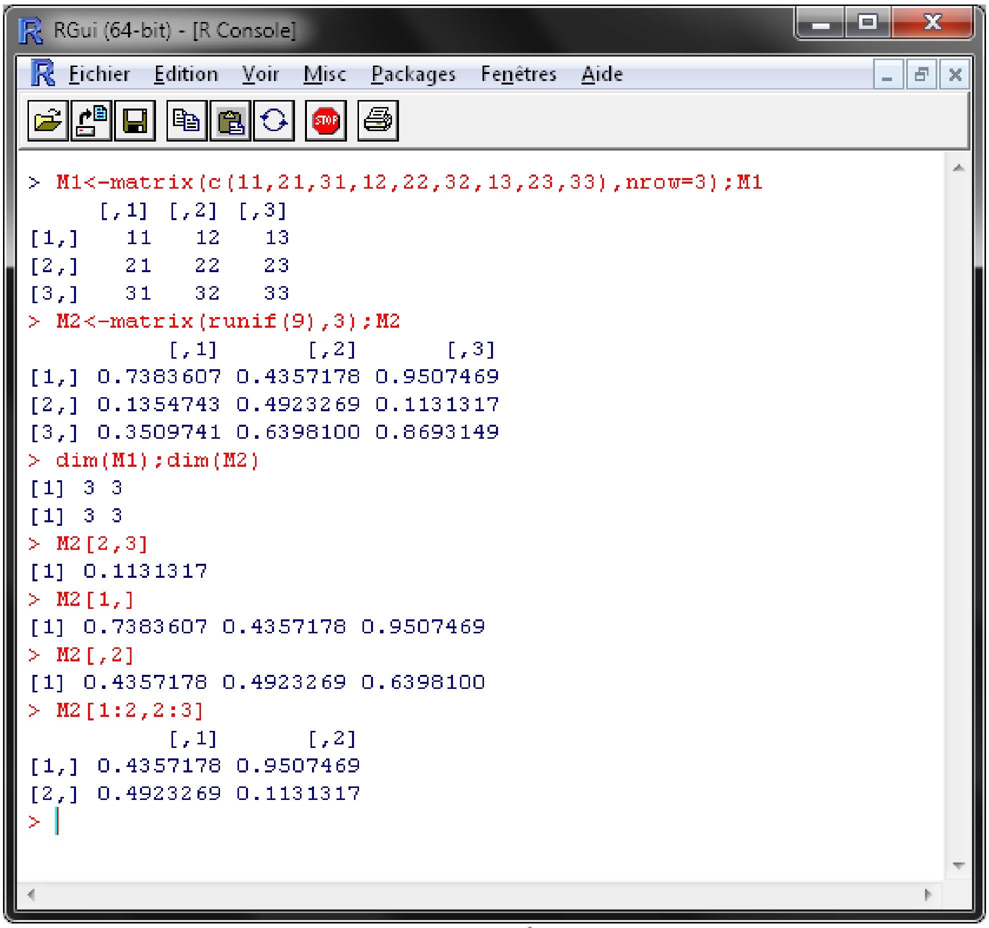
Voyons d'abord les trois cas classiques de création utilisant les commandes **cbind( )** et **matrix( )** et :



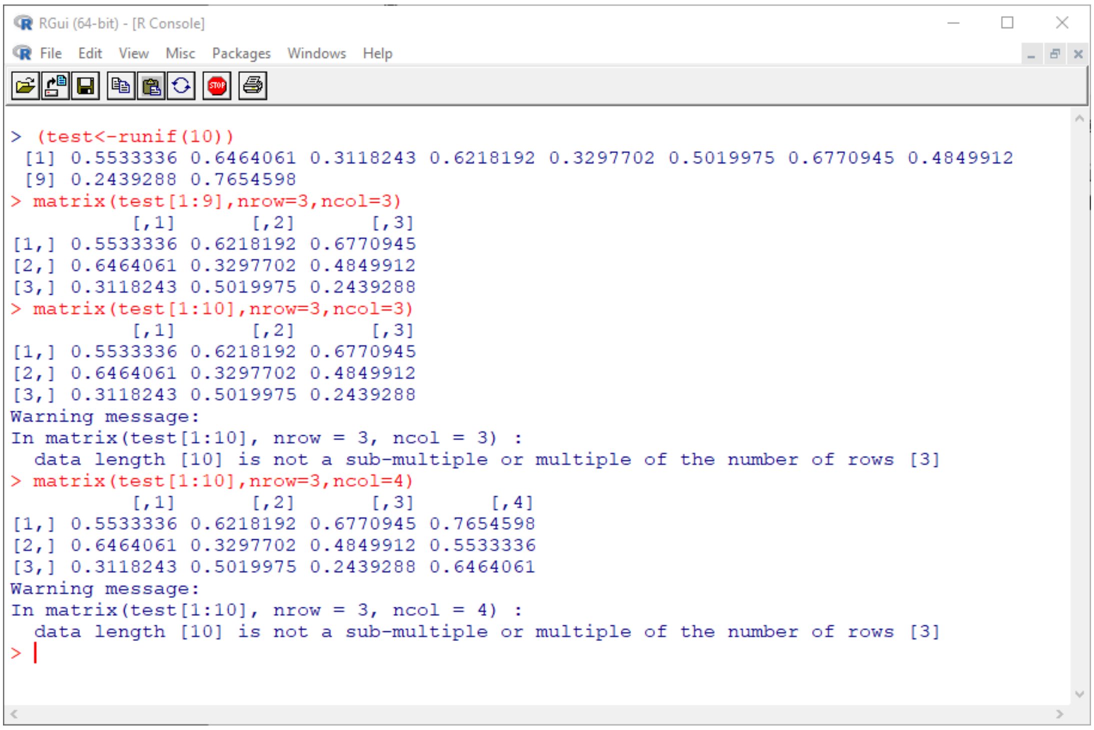
On peut aussi rapidement nommer ou "dénommer" les lignes et les colonnes avec la commande **dimnames( )**:



Ensuite quelques propriétés et accès au contenu des matrices utilisant entre autres la commande **dim( )** :

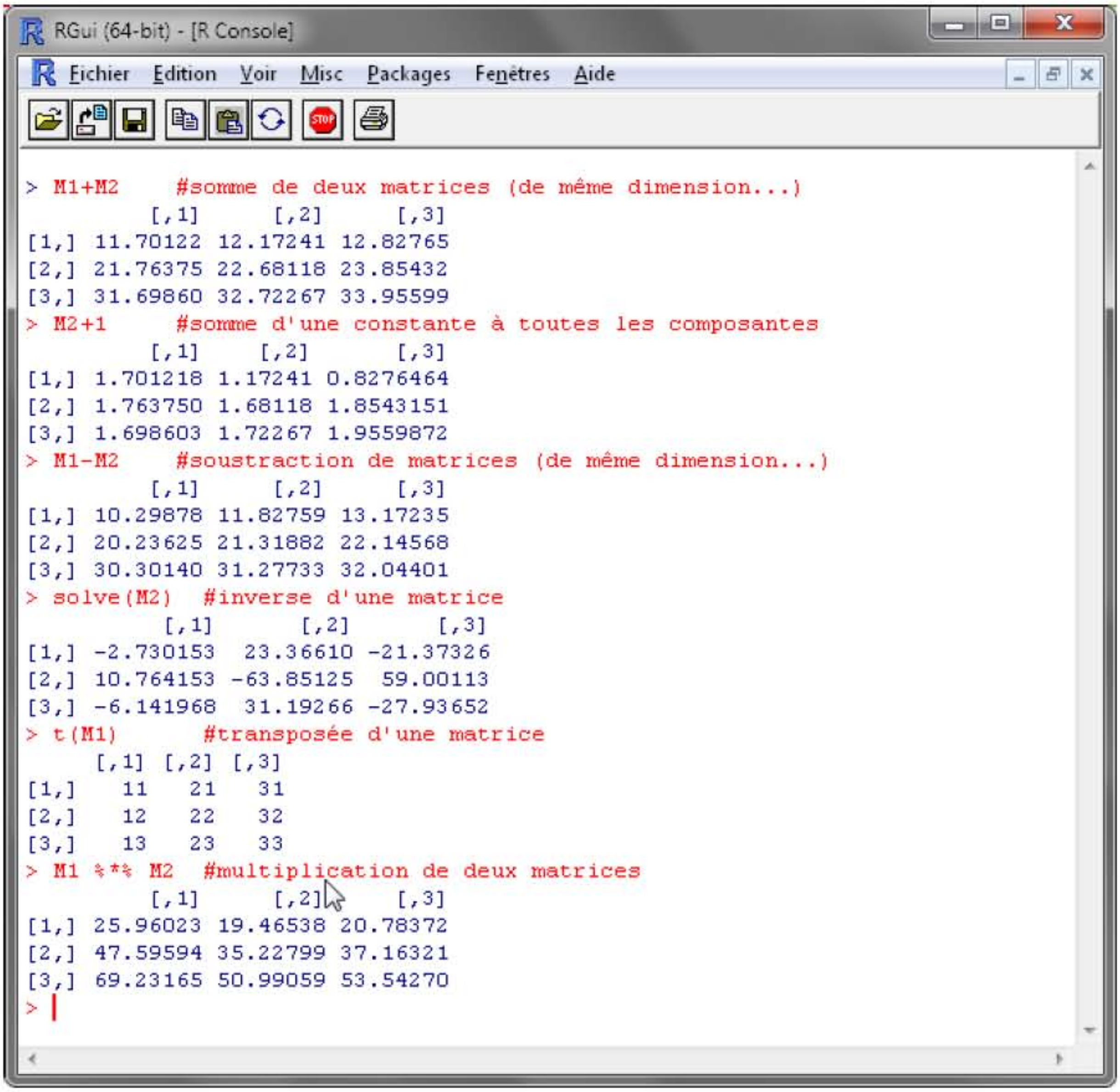


Maintenant il est important de considérer les scénarios de problèmes de dimensions suivants que l'on rencontre relativement fréquemment dans la pratique (observez et comprenez bien ce que fait R **dans chacun des cas**!!!!):

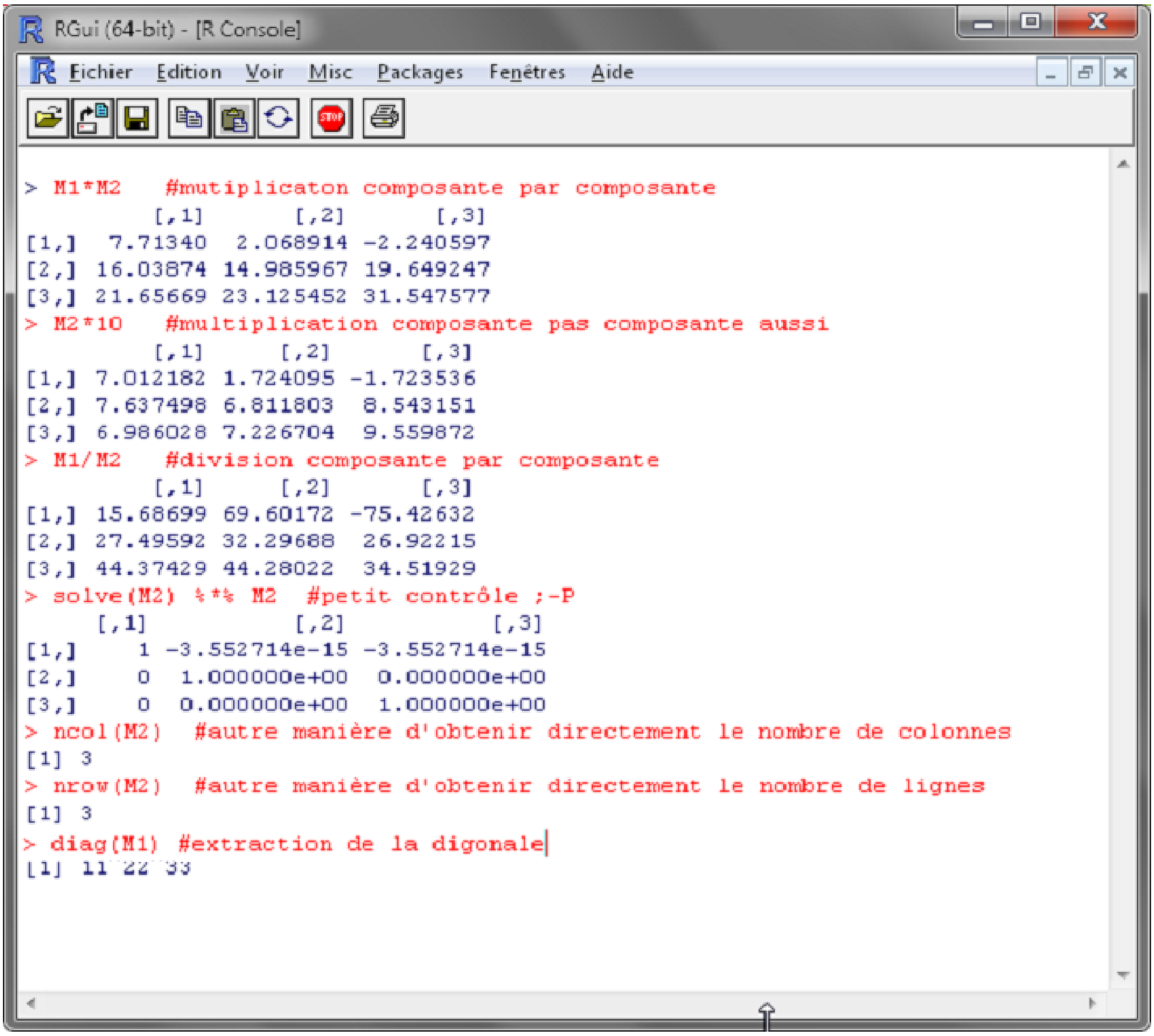


Opérations de base

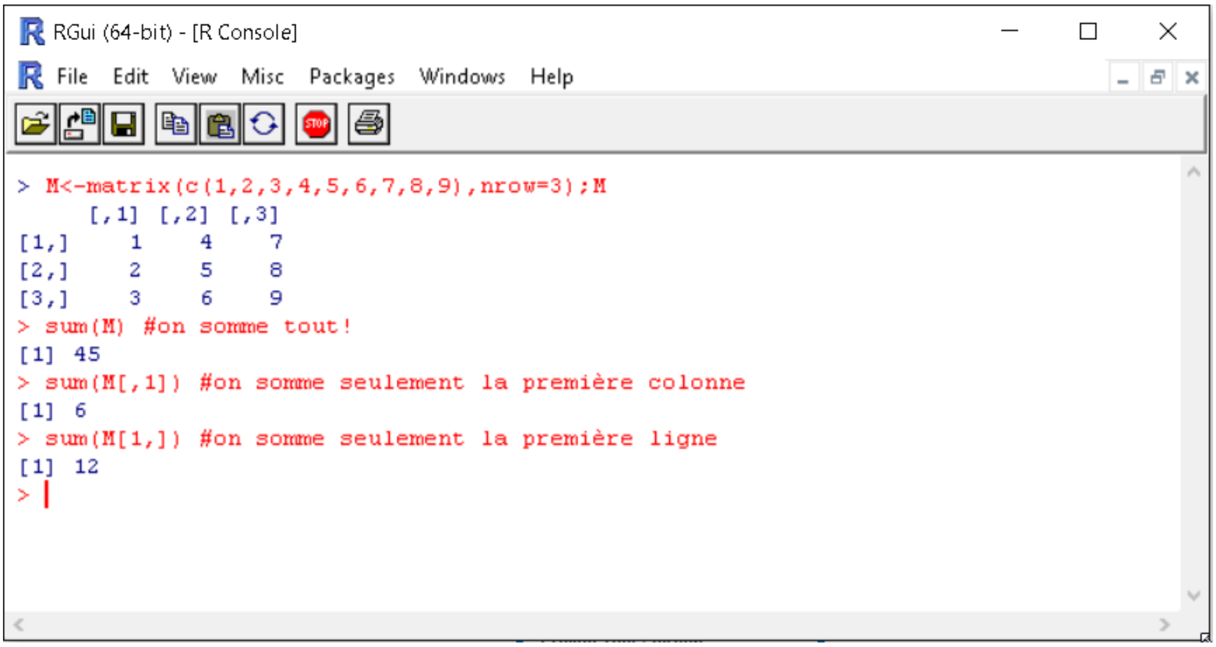
Maintenant voyons quelques opérations classiques utilisant entre autres les commandes **solve( )** , **t( )** :



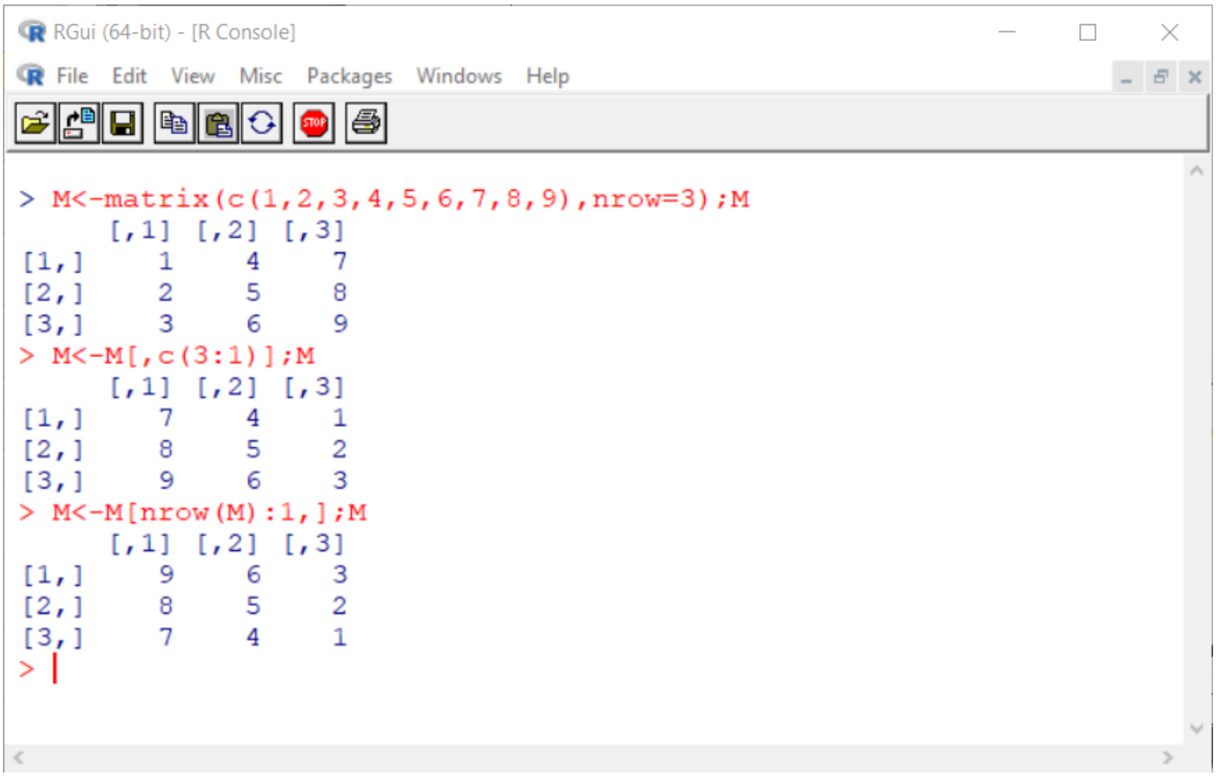
et encore quelques autres opérations et commande dont entre autres **solve( )** pour l'inverse d'une matrice, **ncol**( ) , **nrow**( ) et **diag( )** :



Voyons maintenant comment faire la somme de certaines lignes ou colonnes de la matrice:

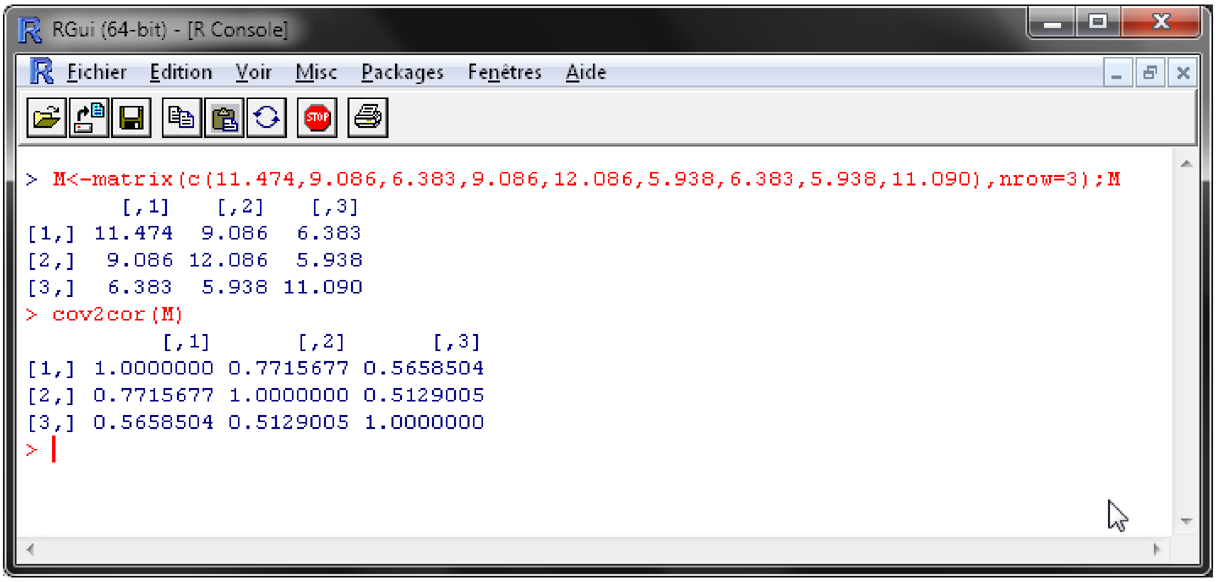


Enfin, pour le traitement d'image, le fait de savoir changer (inverser) l'ordre des colonnes ou des lignes:

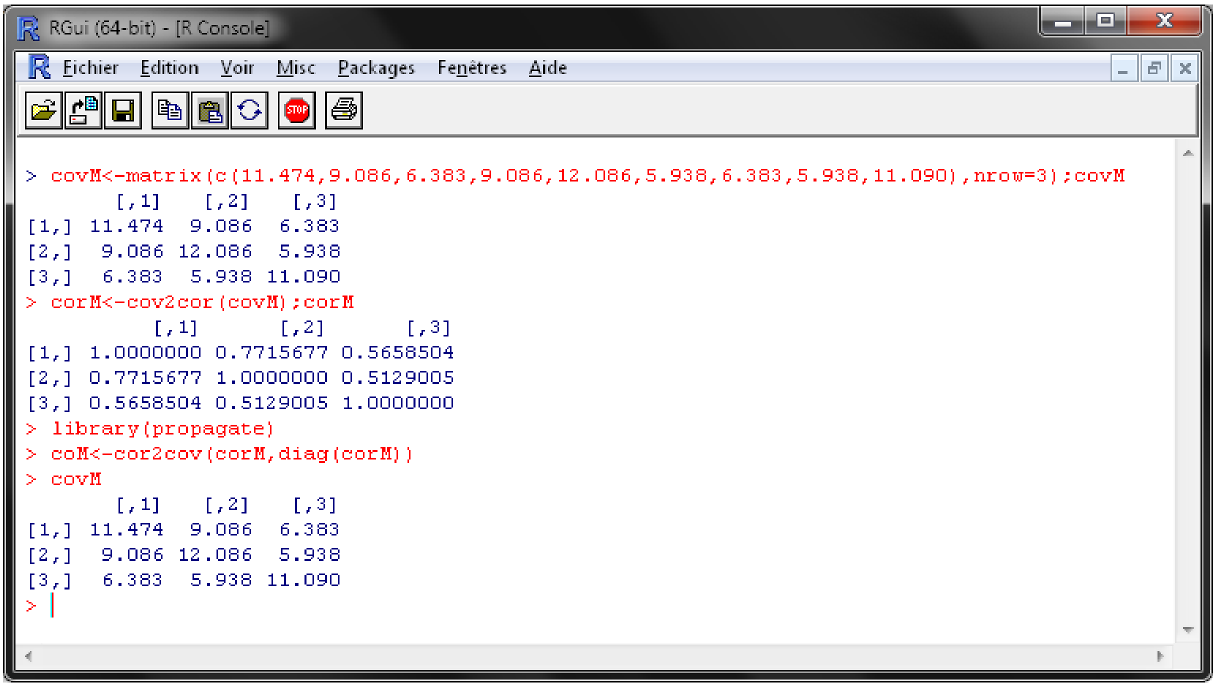


Matrice de variances-covariances

Toujours dans les manipulations simples, convertissons une matrice de variances-covariances en matrice de corrélation rapidement (parfois utile en finance quantitative):

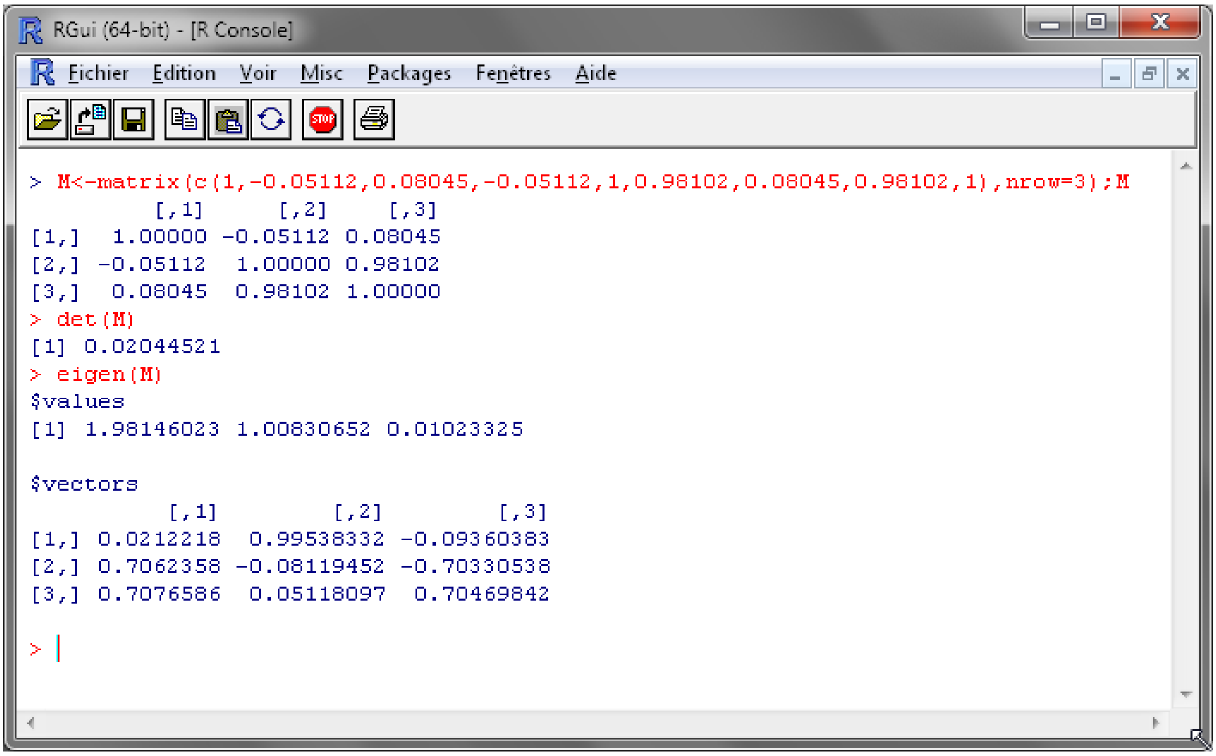


et pour l'inverse il faut utiliser à ce jour le package **propagate** (…) avec sa fonction **cor2cov( )** :



Déterminant et valeurs propres

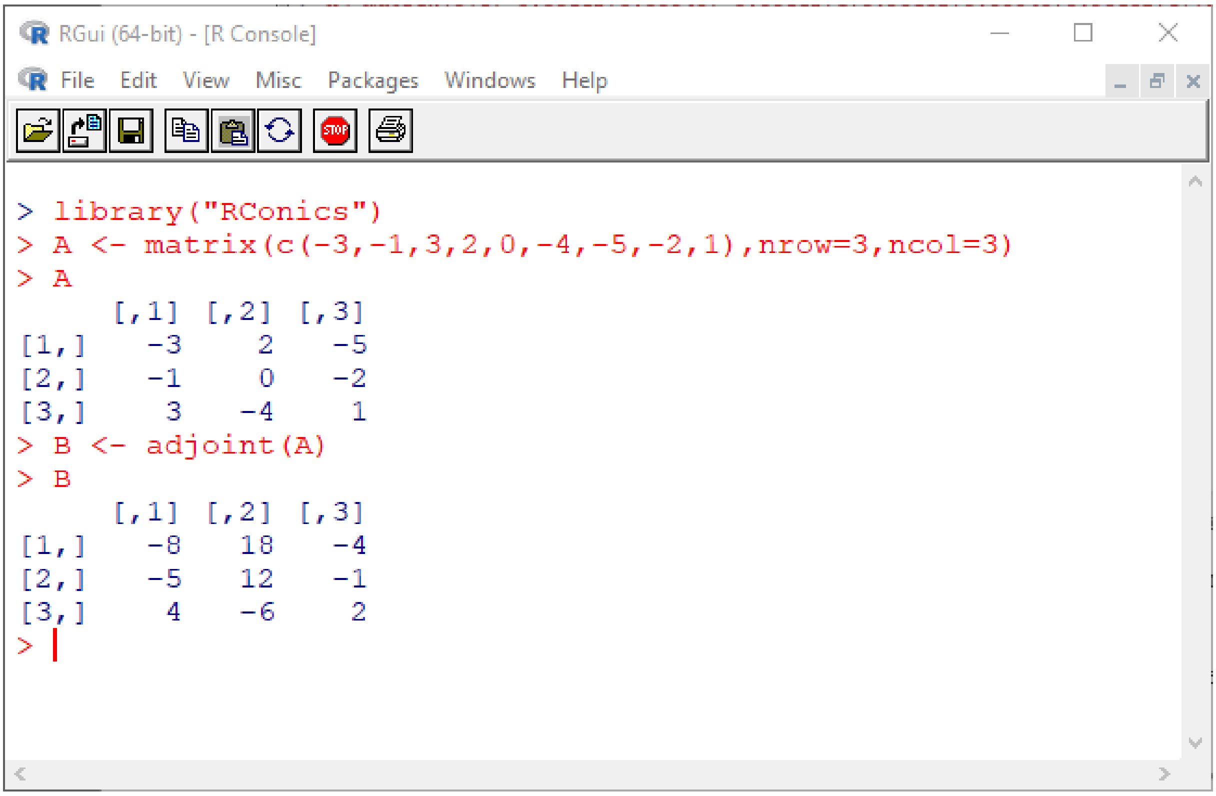
Maintenant nous allons faire des calculs légèrement plus complexes avec les matrices mais qui restent du niveau BAC (nous allons vérifier si nous retrouvons bien les calculs faits avec Excel, Minitab et Tanagra) en reprenant la matrice que nous retrouvons plus tard dans le cadre de l'analyse factorielle (et que nous avions calculé à la main dans le cours théorique):



Nous retrouvons donc le même déterminant, les mêmes valeurs propres et les mêmes vecteurs propres que dans les autres logiciels et que de ce qui a été calculé à la main.

Rappel: Une matrice carrée a toujours des valeurs propres. Par contre si elle n'est pas symétrique elle aura des valeurs propres complexes.

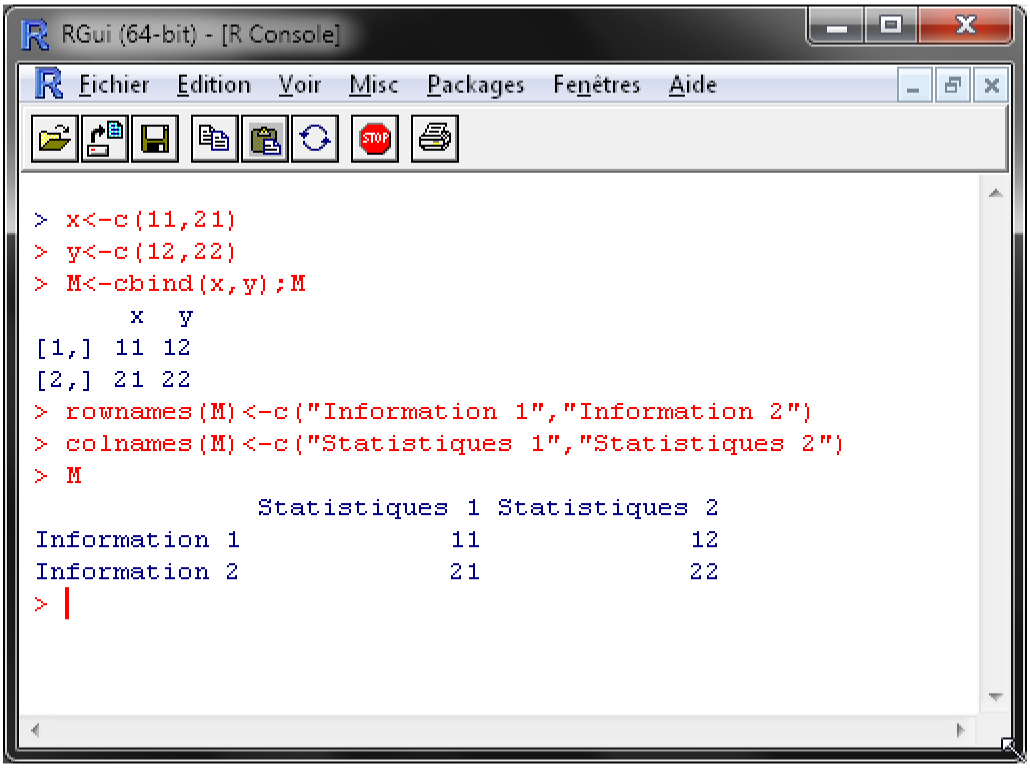
Voyons si nous pouvons aussi retrouver la comatrice (matrice des cofacteurs), donnée comme exemple sur le page Wikipédia qui décrit cette matrice. Pour cela, nous utiliserons la fonction **adjoint( )** du package **Rconics**:



C'est bien conforme au résultat indiqué sur la page Wikipédia.

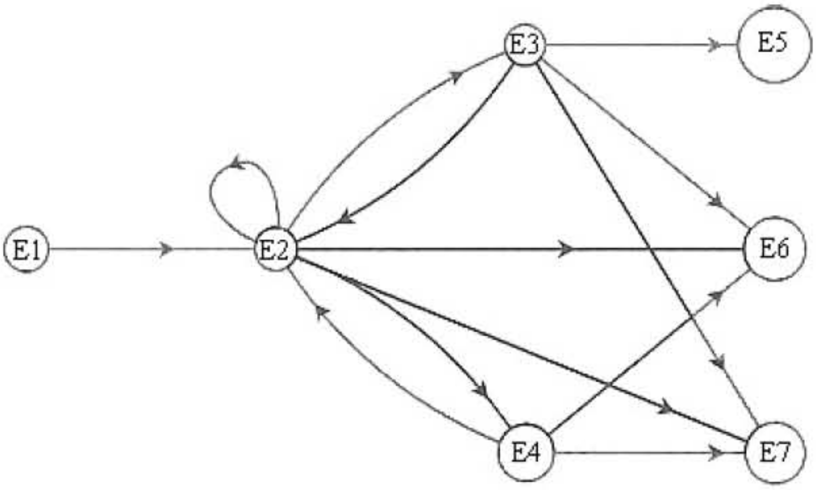
Lorsque l'on fait du scripting, il peut être utile de compiler des statistiques dans une matrice avec des noms aux colonnes et aux lignes (la matrice sera alors de squelette pour afficher des données statistiques diverses et variées).

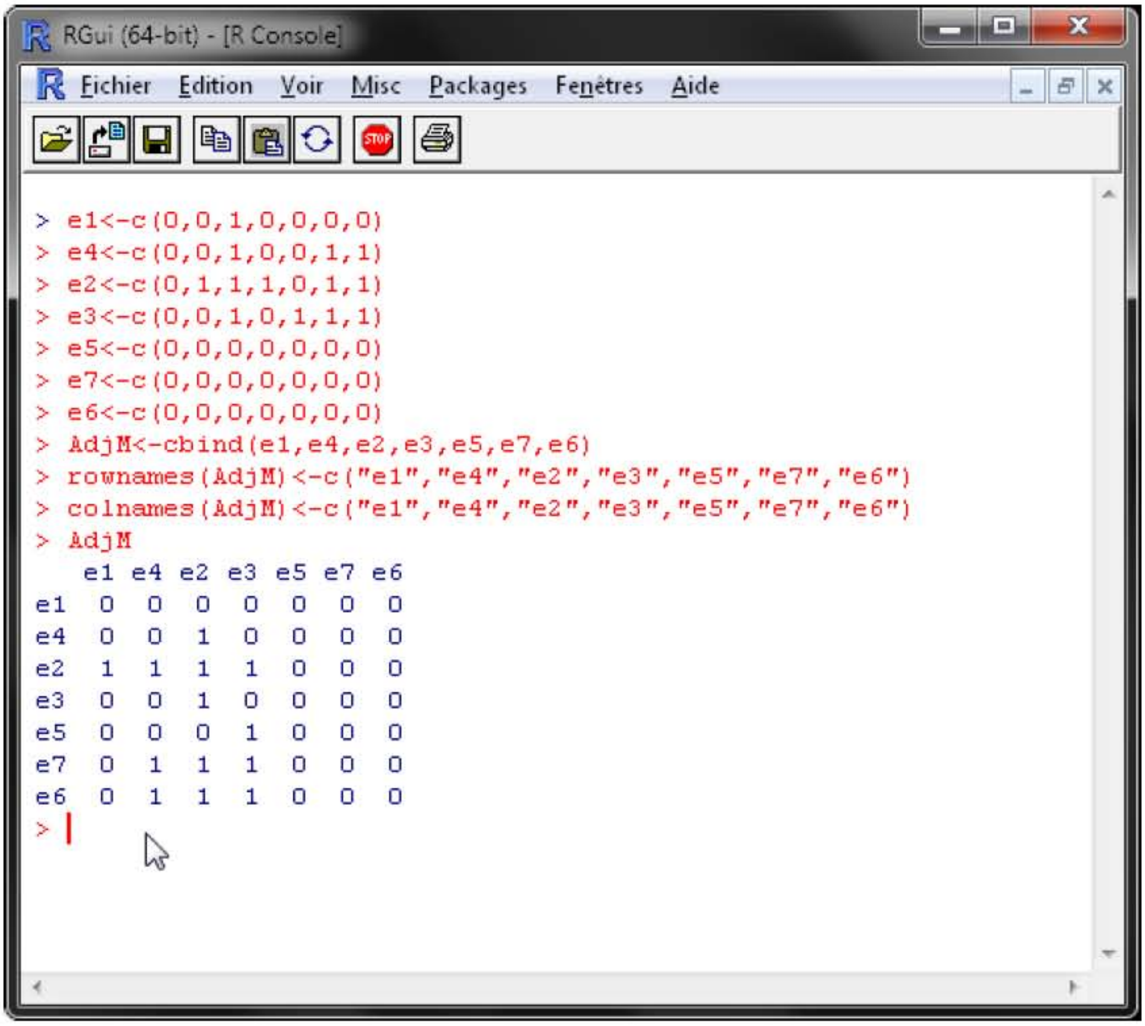
Voyons un exemple:



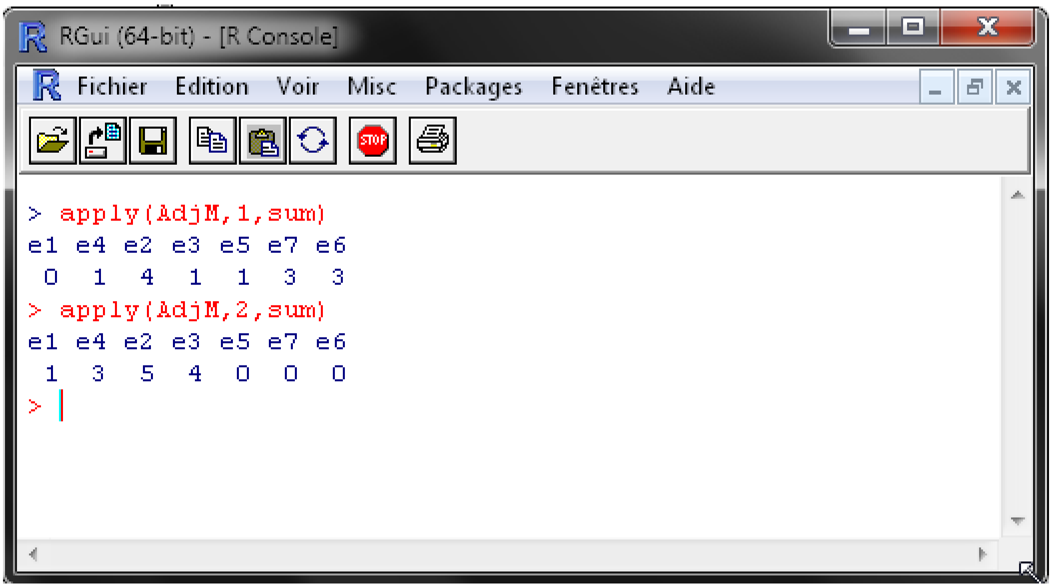
Matrice d'adjacence

Renommer des lignes et colonnes de matrices est aussi utile dans la théorie des graphes pour avoir une sortie écran plus esthétique. Cependant ce n'est pas tout. Considérons la matrice d'adjacence suivante vue dans le cours théorique de la théorie des graphes:



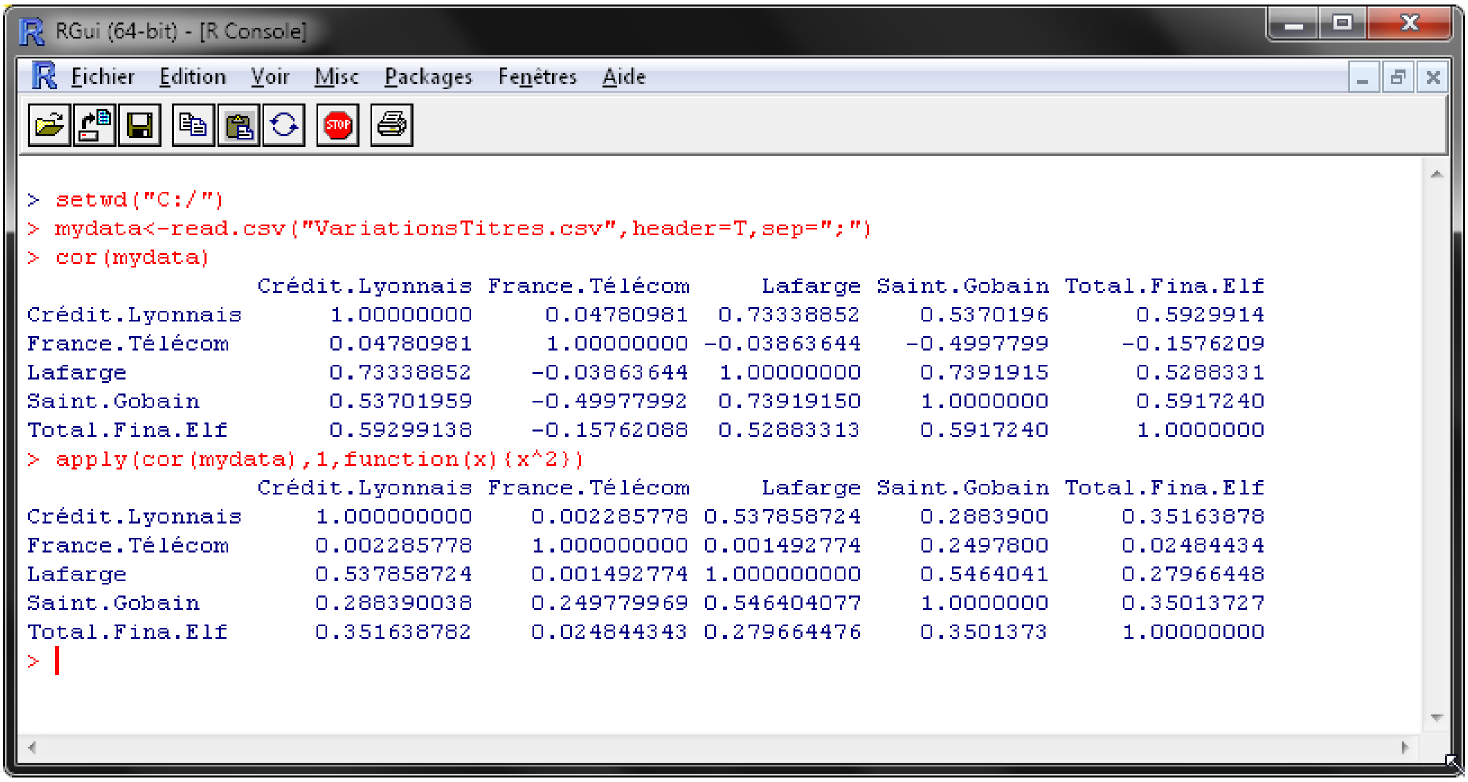


Ensuite, avec la commande **apply( )** , nous pouvons connaître le nombre de flèches entrantes ou sortantes:

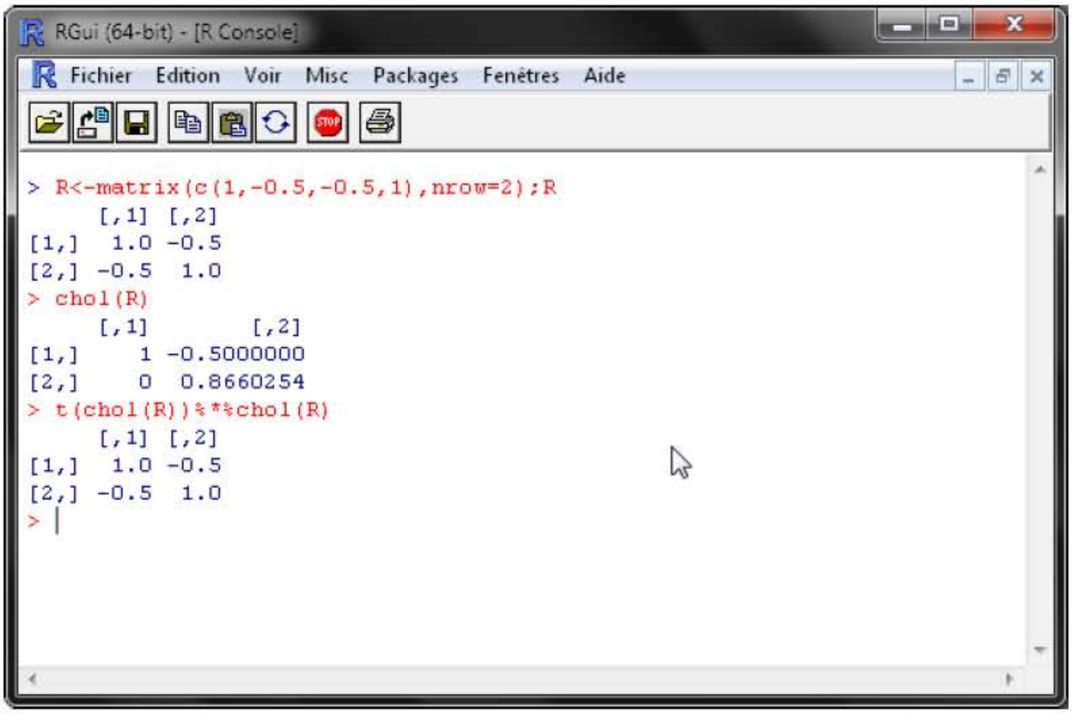


Le deuxième paramètre "1" ou "2" indiquant si nous souhaitons le calcul pour respectivement les lignes ou les colonnes.

Il y a un autre cas où **apply( )** peut être utilise lorsque la commande choisie n'est pas une fonction d'agrégation. Si nous souhaitons par exemple appliquer une fonction quelconque sur chaque composante (ici dans l'idée d'avoir le R^2), alors:

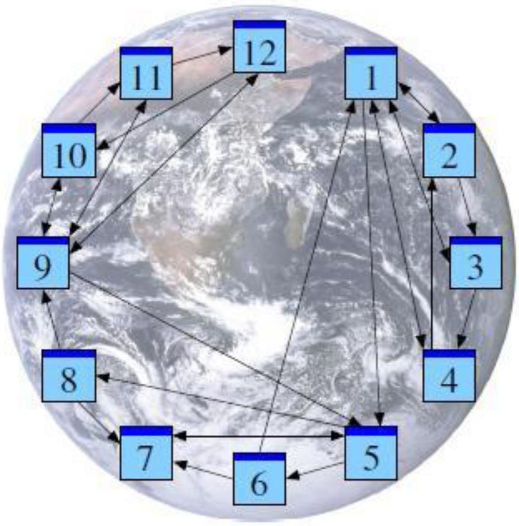


Enfin dans le cours théorique dans le cas des copulas Gaussien, nous avons introduit le concept de décomposition de Cholesky. Vérifions que nous obtenons bien la même chose que dans le cours théorique avec la fonction **chol( )** :


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> R<-matrix(c(1,-0.5,-0.5,1),nrow=2);R
     [,1] [,2]
[1,] 1.0 -0.5
[2,] -0.5 1.0
> chol(R)
     [,1]   [,2]
[1,]   1 -0.5000000
[2,]   0 0.8660254
> t(chol(R))%*%chol(R)
     [,1] [,2]
[1,] 1.0 -0.5
[2,] -0.5 1.0
>

Matrices stochastiques

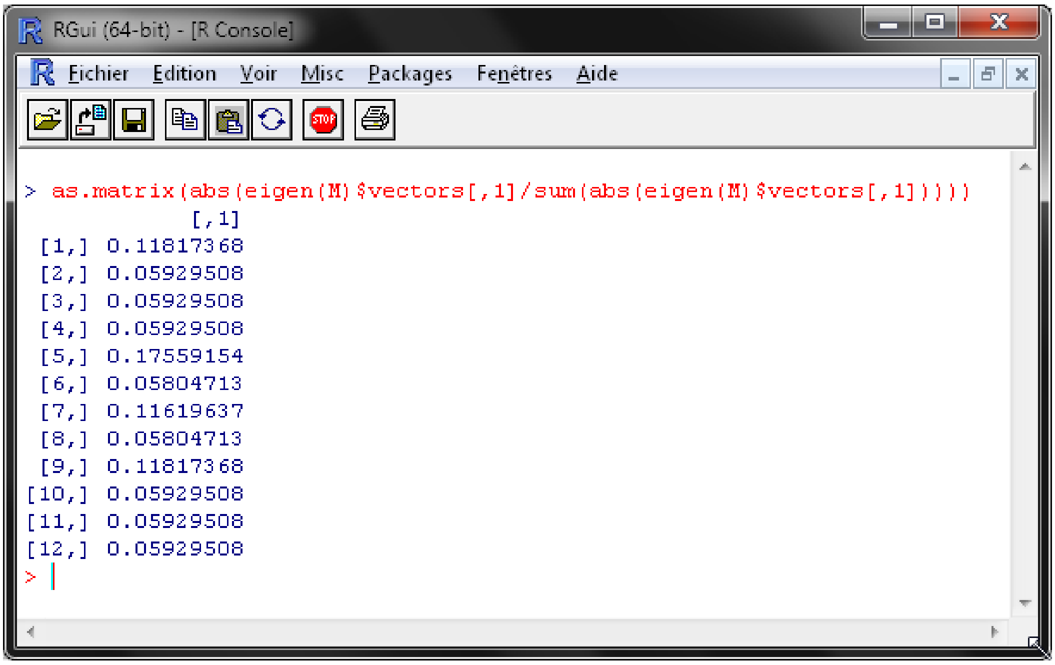
Voyons maintenant un encore une autre application avec les chaînes de Markov en se basant sur l'exemple utilisé dans le cours théorique d'ingénierie informatique du principe de base d'origine de fonctionnement de l'algorithme de Google. Nous avions donc le graphe suivant:



Nous construisons la matrice stochastique du graphe comme nous l'avons fait à la main dans le cours théorique:

```
> c1<-c(0,0.25,0.25,0.25,0.25,0,0,0,0,0,0,0)
> c2<-c(0.5,0,0.5,0,0,0,0,0,0,0,0,0)
> c3<-c(0.5,0,0,0.5,0,0,0,0,0,0,0,0)
> c4<-c(0.5,0.5,0,0,0,0,0,0,0,0,0,0)
> c5<-c(0,0,0,0,0,0.33,0.33,0.33,0,0,0,0)
> c6<-c(0.5,0,0,0,0,0,0.5,0,0,0,0,0)
> c7<-c(0,0,0,0,1,0,0,0,0,0,0,0)
> c8<-c(0,0,0,0,0,0,0.5,0,0.5,0,0,0)
> c9<-c(0,0,0,0,0.25,0,0,0,0,0.25,0.25,0.25)
> c10<-c(0,0,0,0,0,0,0,0,0.5,0,0.5,0)
> c11<-c(0,0,0,0,0,0,0,0,0.5,0,0,0.5)
> c12<-c(0,0,0,0,0,0,0,0,0.5,0.5,0,0)
> M<-cbind(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12);M
        c1  c2  c3  c4   c5  c6 c7  c8   c9 c10 c11 c12
 [1,] 0.00 0.5 0.5 0.5 0.00 0.5  0 0.0 0.00 0.0 0.0 0.0
 [2,] 0.25 0.0 0.0 0.5 0.00 0.0  0 0.0 0.00 0.0 0.0 0.0
 [3,] 0.25 0.5 0.0 0.0 0.00 0.0  0 0.0 0.00 0.0 0.0 0.0
 [4,] 0.25 0.0 0.5 0.0 0.00 0.0  0 0.0 0.00 0.0 0.0 0.0
 [5,] 0.25 0.0 0.0 0.0 0.00 0.0  1 0.0 0.25 0.0 0.0 0.0
 [6,] 0.00 0.0 0.0 0.0 0.33 0.0  0 0.0 0.00 0.0 0.0 0.0
 [7,] 0.00 0.0 0.0 0.0 0.33 0.5  0 0.5 0.00 0.0 0.0 0.0
 [8,] 0.00 0.0 0.0 0.0 0.33 0.0  0 0.0 0.00 0.0 0.0 0.0
 [9,] 0.00 0.0 0.0 0.0 0.00 0.0  0 0.5 0.00 0.5 0.5 0.5
[10,] 0.00 0.0 0.0 0.0 0.00 0.0  0 0.0 0.25 0.0 0.0 0.5
[11,] 0.00 0.0 0.0 0.0 0.00 0.0  0 0.0 0.25 0.5 0.0 0.0
[12,] 0.00 0.0 0.0 0.0 0.00 0.0  0 0.0 0.25 0.0 0.5 0.0
```

Maintenant voyons comment trouver la mesure invariante avec R en utilisant de l'algèbre linéaire des valeurs et vecteurs propres subtilement:



Nous retrouvons donc bien les valeurs calculées à la main dans le cours théorique.

Convolution de Matrices

La convolution de matrices est un sujet important dans la pratique du traitement d'images et particulièrement aussi dans certains domaines de l'intelligence artificielle. Donc voyons si nous arrivons à reproduire l'exemple du vu dans le cours théorique avec la fonction **convolution( )** du package **OpenImageR**:

```
> library(OpenImageR)
> data_Matrix<-cbind(c(0,0,0,0,0,0,0),c(0,1,1,1,0,0,0),c(0,1,2,2,1,1,1),$
+ c(0,1,1,2,2,1,1),c(0,0,1,1,1,1,1),c(0,0,1,1,1,1,1),c(0,0,0,0,0,0,0))
> data_Matrix
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    0    0    0    0    0    0    0
[2,]    0    1    1    1    1    0    0    0
[3,]    0    1    2    2    1    1    1    0
[4,]    0    1    2    2    2    1    1    0
[5,]    0    0    1    2    2    1    1    0
[6,]    0    0    1    1    1    1    1    0
[7,]    0    0    1    1    1    1    1    0
> kernel_Matrix<-cbind(c(-4,0,0),c(0,0,0),c(0,0,4))
> kernel_Matrix
     [,1] [,2] [,3]
[1,]   -4    0    0
[2,]    0    0    0
[3,]    0    0    4
>
> convolution(data_Matrix, kernel_Matrix, "same")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]   -4   -4   -4   -4    0    0    0    0
[2,]   -4   -8   -8   -4   -4   -4    0    0
[3,]   -4   -8   -4   -4    0    0    0    0
[4,]    0   -4   -4    0    4    0    4    4
[5,]    0   -4    0    4    4    4    4    4
[6,]    0   -4   -4    0    4    4    4    4
[7,]    0    0    0    4    4    4    4    4
>
```

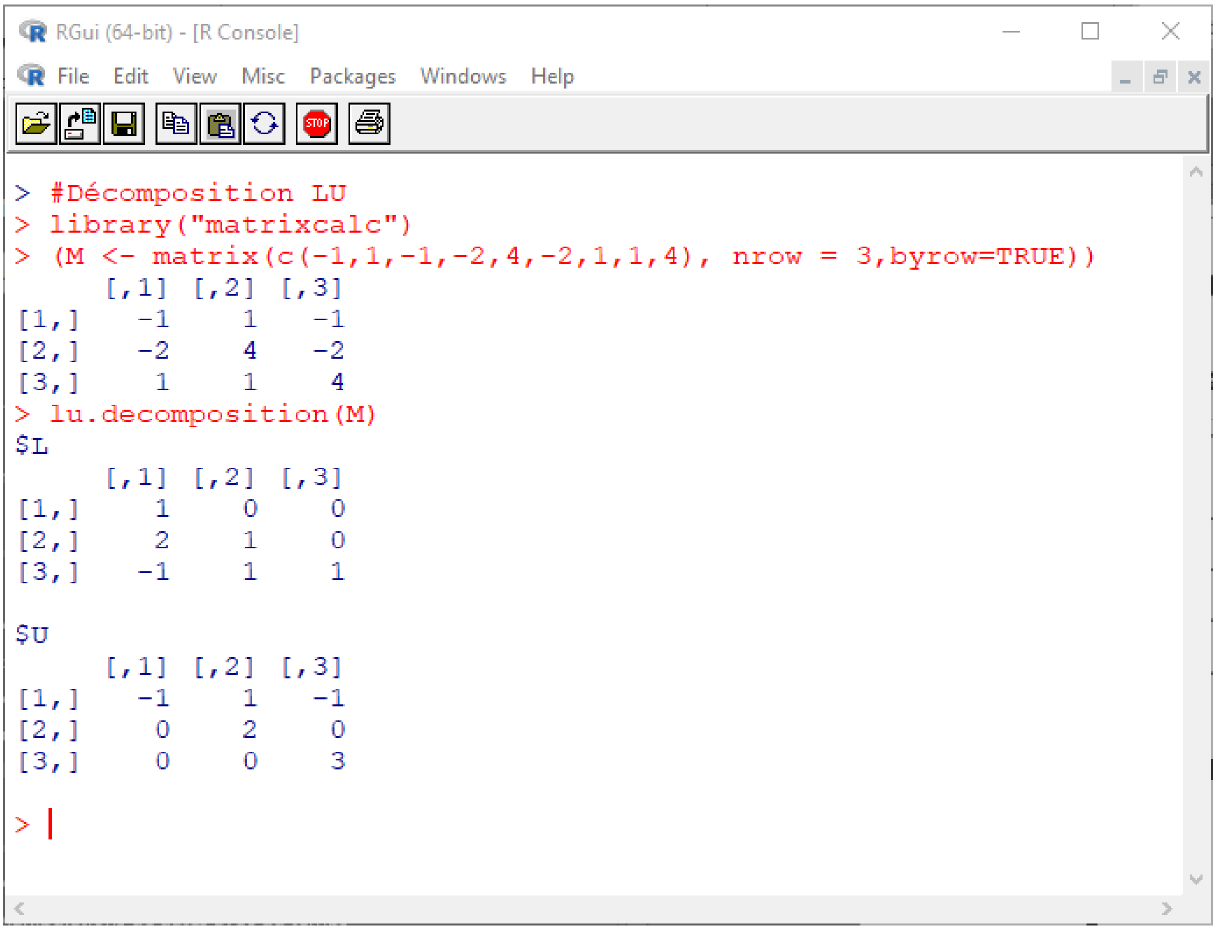
Décompositions de matrices

Nous allons mettre en pratique et vérifier si nous retombons sur les valeurs numériques obtenues lors du cours théorique relativement aux décompositions de matrices de type:

- LU
- QR
- Cholesky
- SVD (décomposition en valeurs singulières)

Décomposition LU

On va utiliser le package **matrixcalc** et sa fonction **lu.decomposition( )** pour vérifier si nous retombons bien sur les valeurs calculées à la main dans le cours théorique:



```
> #Décomposition LU
> library("matrixcalc")
> (M <- matrix(c(-1,1,-1,-2,4,-2,1,1,4), nrow = 3,byrow=TRUE))
     [,1] [,2] [,3]
[1,]   -1    1   -1
[2,]   -2    4   -2
[3,]    1    1    4
> lu.decomposition(M)
$L
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    2    1    0
[3,]   -1    1    1

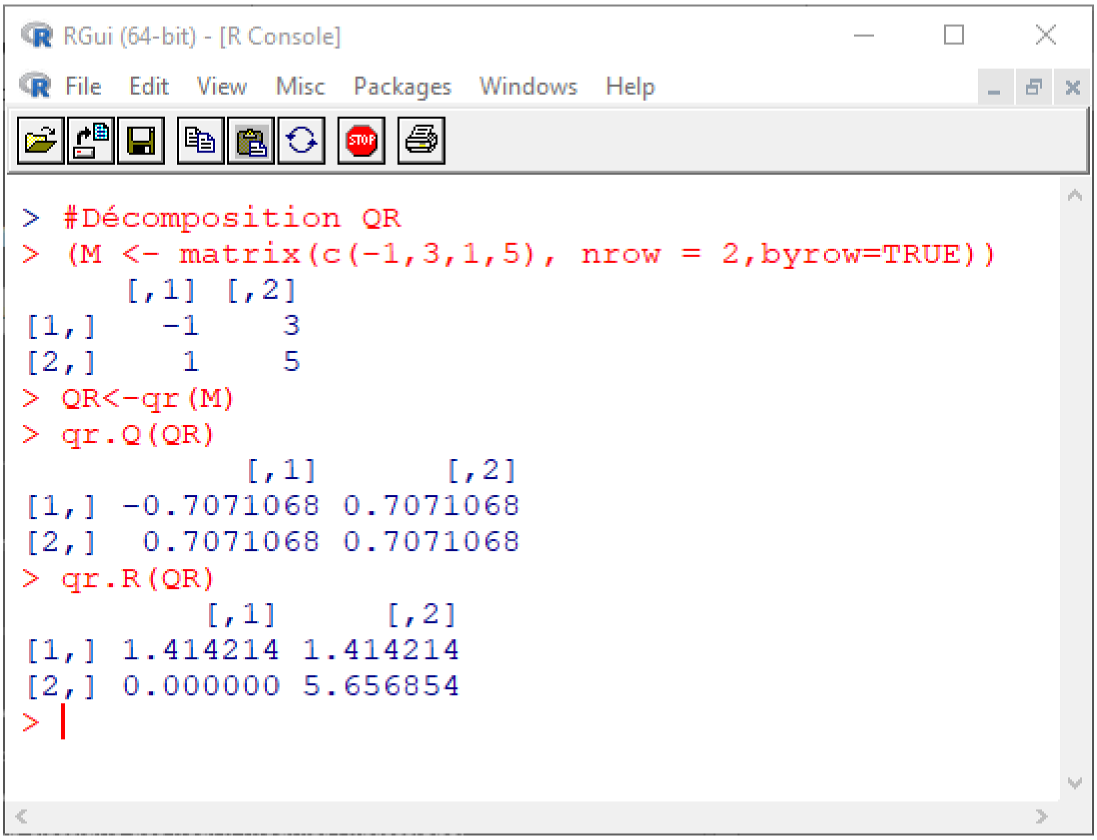
$U
     [,1] [,2] [,3]
[1,]   -1    1   -1
[2,]    0    2    0
[3,]    0    0    3

> 
```

Et nous remarquons que nous retombons bien sur les valeurs calculées à la main dans le cours théorique.

Décomposition QR

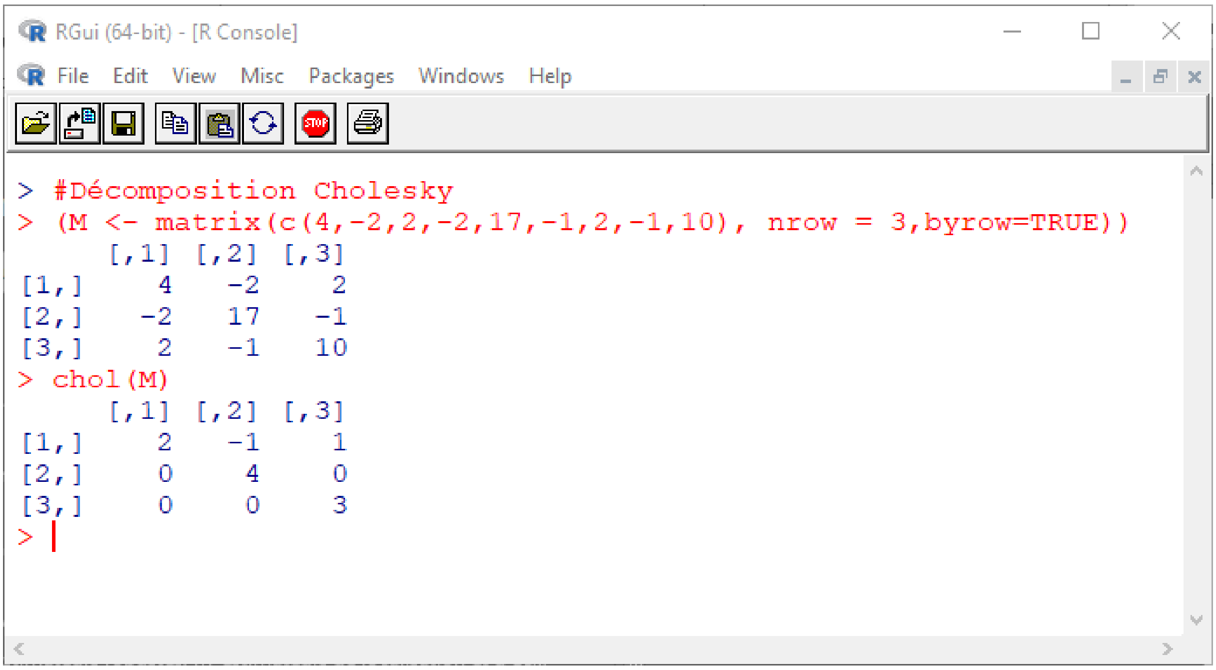
On va utiliser la fonction native **qr( )** pour vérifier si nous retombons bien sur les valeurs calculées à la main dans le cours théorique:



Et nous remarquons que nous retombons bien sur les valeurs calculées à la main dans le cours théorique.

Décomposition de Cholesky

On va utiliser la fonction native **chol( )** pour vérifier si nous retombons bien sur les valeurs calculées à la main dans le cours théorique:



Et nous remarquons que nous retombons bien sur les valeurs calculées à la main dans le cours théorique.

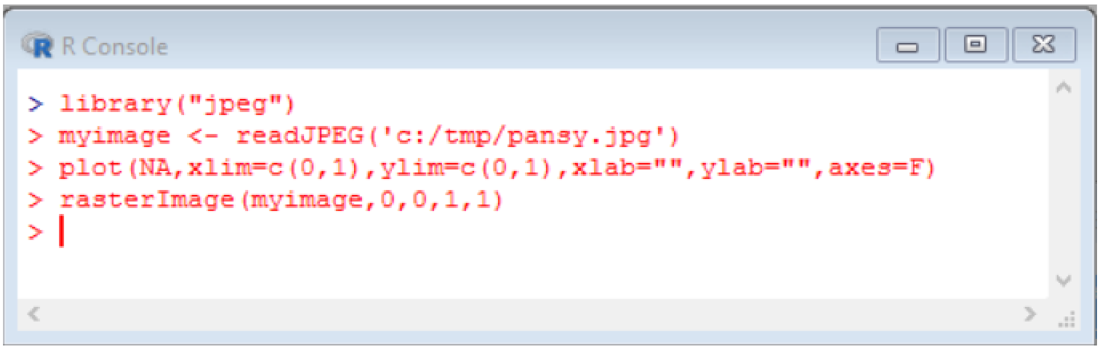
Décomposition en valeurs singulières

Let but ici est d'illustrer le théorème de décomposition en valeurs singulières que nous avons démontré dans le cours théorique.

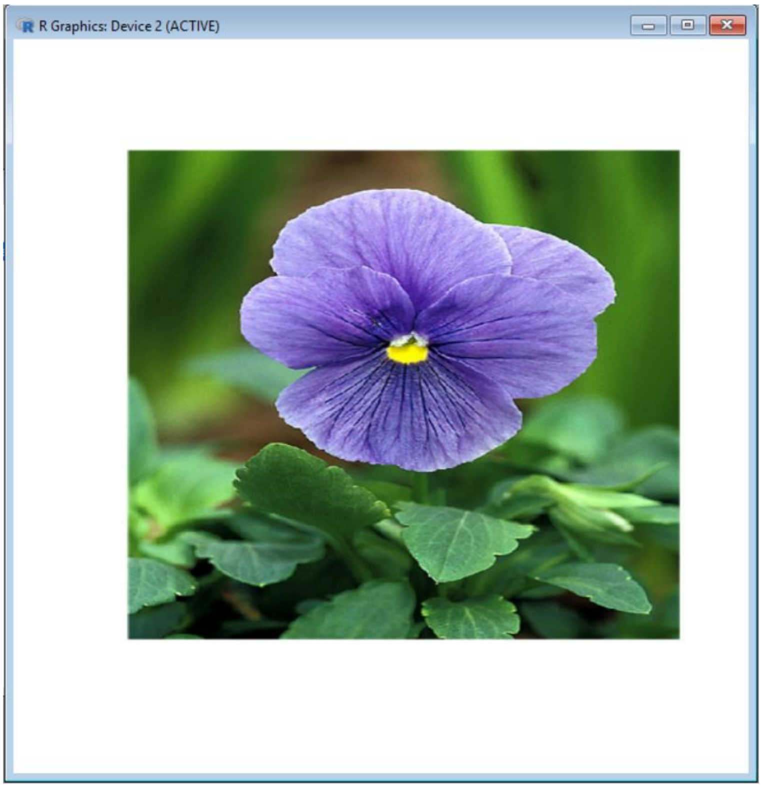
Nous partons de l'image suivante:



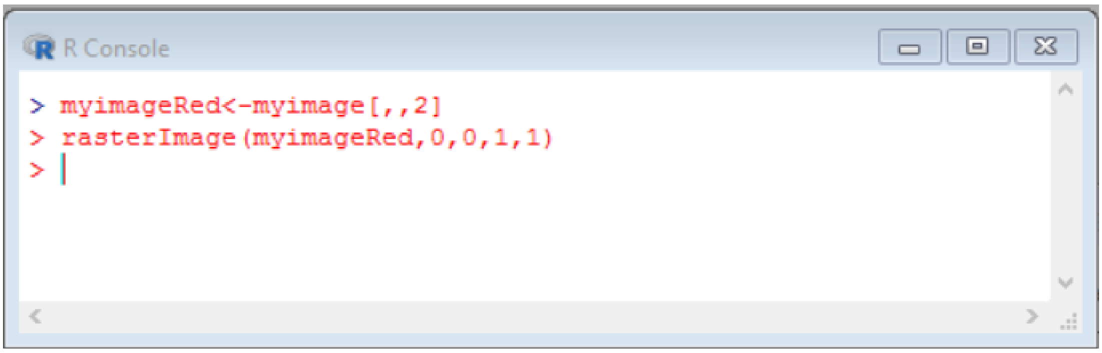
Ensuite, nous chargeons l'image et l'affichons:



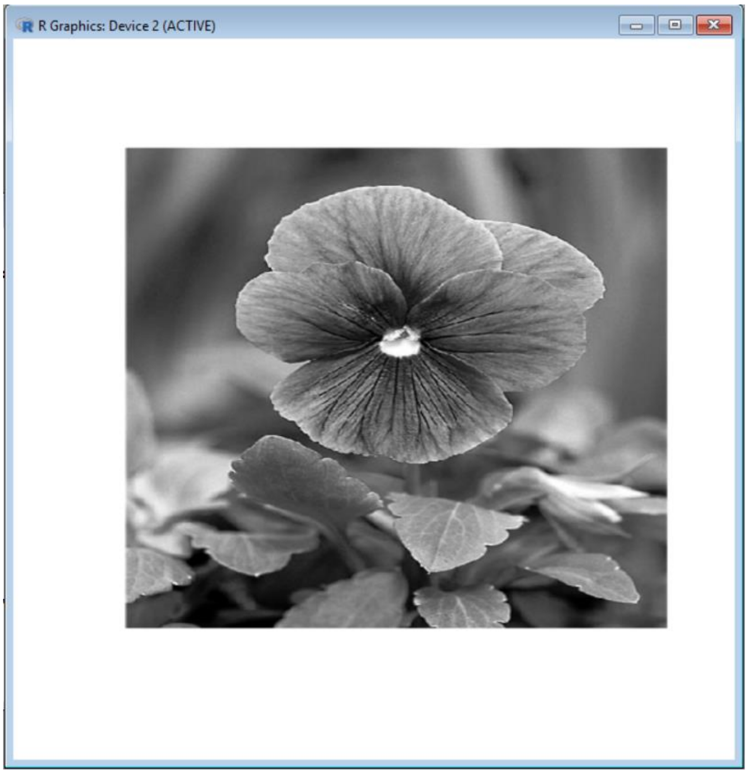
Nous obtenons:



Pour faciliter l'exemple nous allons travailler que sur la couche de rouge:



Ce qui donne:



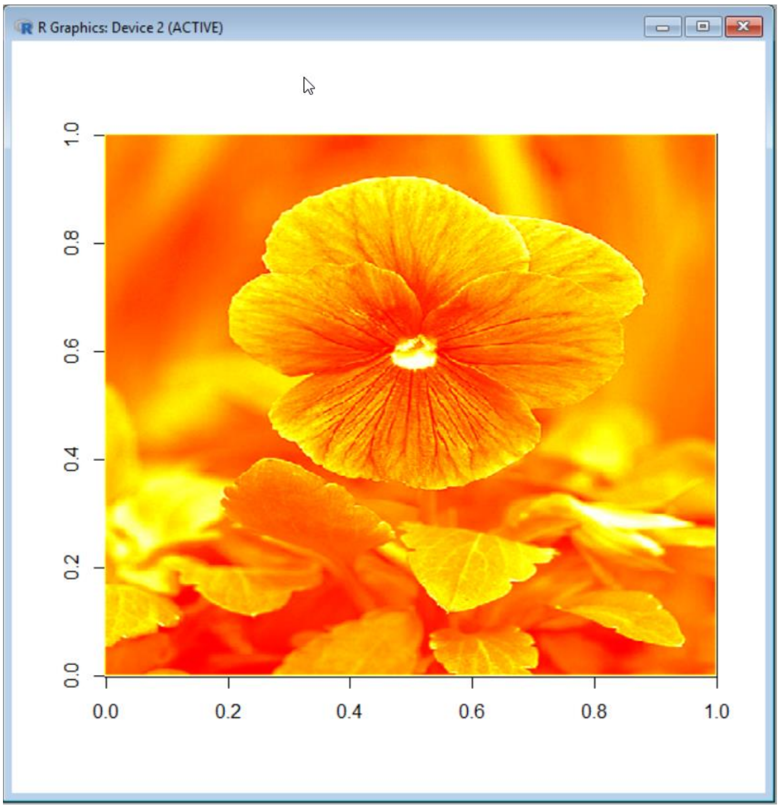
Nous effectuons la décomposition en valeurs singulières en utilisant la fonction **svd( )** :

```
R Console
> svd_image<-svd(myimageRed)
>
> d <- svd_image$d
> u <- svd_image$u
> v <- svd_image$v
>
```

Pour montrer que les trois matrices satisfont bien la relation démontrée dans le cours théorique, nous reconstruisons l'image d'origine avec les opérations mathématiques sur les matrices correspondantes:

```
R Console
> reconstruction <- u %*% diag(d) %*% t(v)
> #l'image va être affichée incorrectement donc on transforme la matrice
> reconstruction<-t(apply(reconstruction, 2, rev))
> #maintenant on peut afficher l'image reconstruite
> image(reconstruction, col = heat.colors(255))
>
```

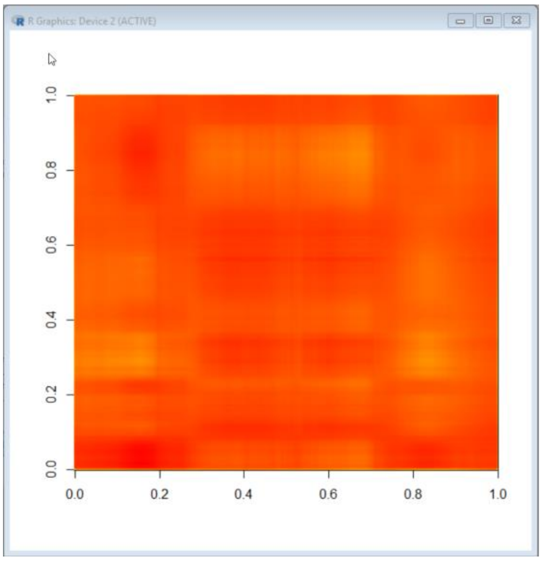
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **2** valeurs singulières:

```
> i<-2 #changer la vaelur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
> 
```

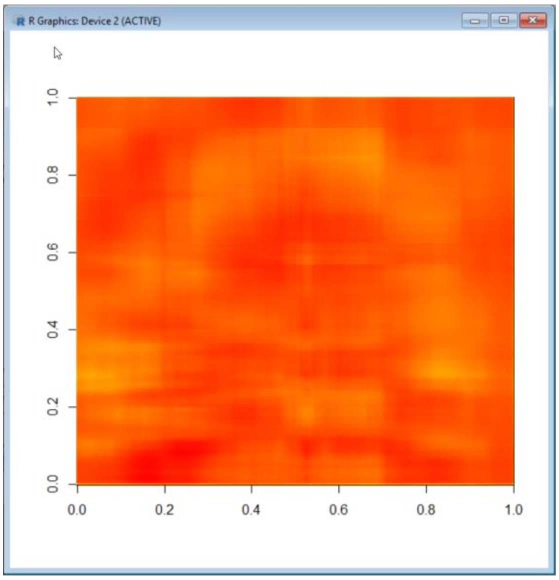
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **4** valeurs singulières:

```
R Console
> i<-4 #changer la vaelur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
> 
```

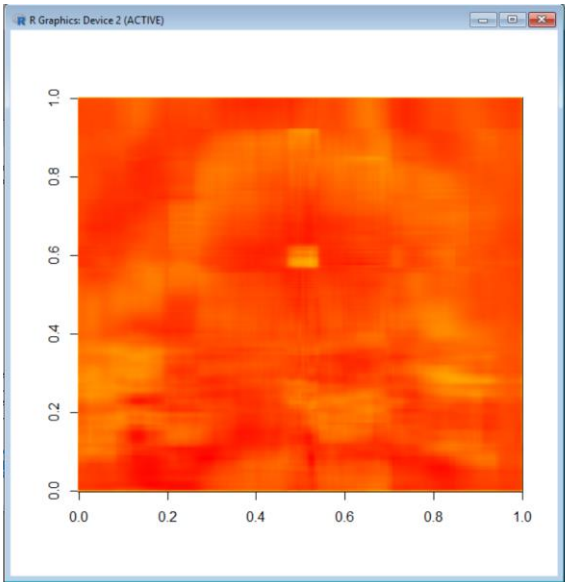
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **8** valeurs singulières:

```
R Console
> i<-8 #changer la valeur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
> 
```

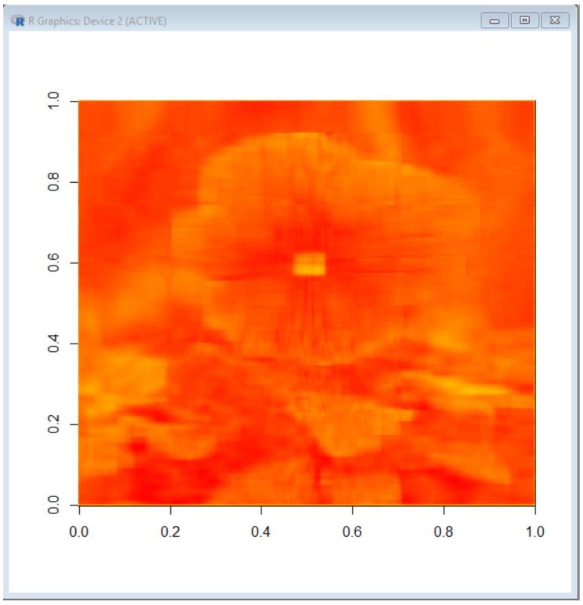
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **16** valeurs singulières:

```
> i<-16 #changer la valeur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
>
```

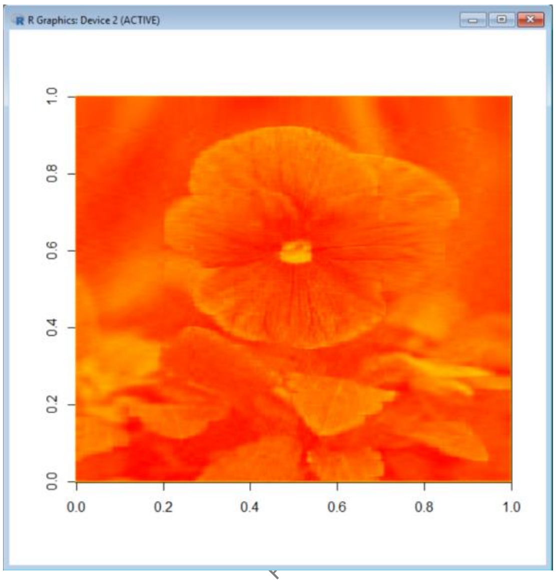
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **32** valeurs singulières:

```
> i<-32 #changer la valeur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
>
```

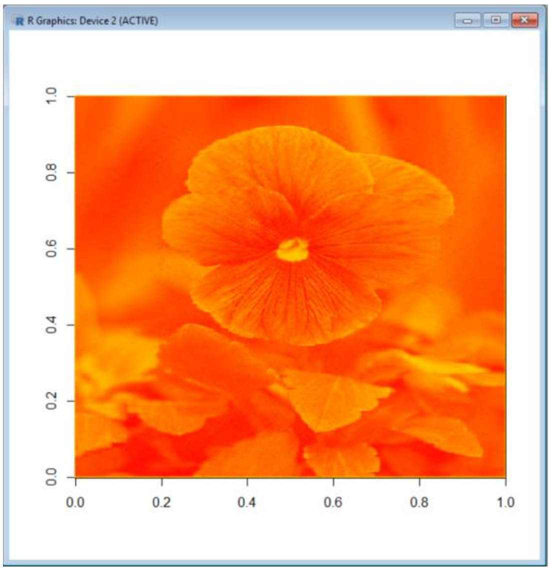
Ce qui donne:



Maintenant voyons ce que donne la reconstruction avec **64** valeurs singulières:

```
> i<-64 #changer la valeur
> imageCompressed<- u[,1:i] %*% diag(d[1:i]) %*% t(v[,1:i])
> image(t(apply(imageCompressed, 2, rev)), col = heat.colors(255))
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
0.0
0.2
0.4
0.6
0.8
1.0

Multiplication tensorielle (produit de Kronecker généralisé)

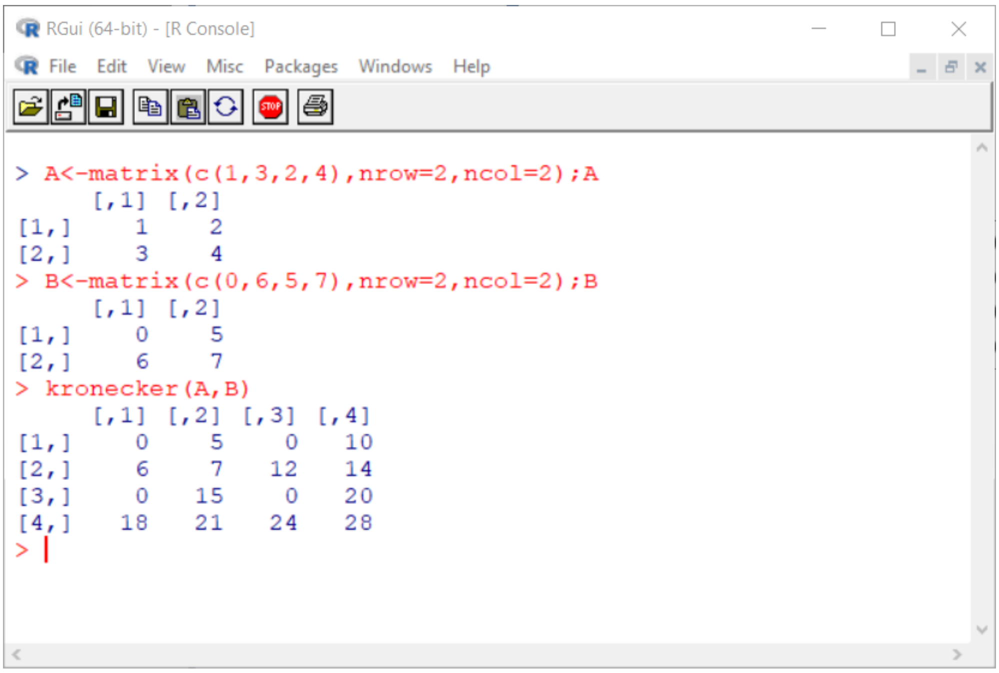
Nous avons vu dans le cours théorique (avec ces applications), que cette multiplication était définie par:

$$\mathbf{A}\otimes\mathbf{B}=\begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}$$

Avec l'exemple suivant:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\otimes\begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix}=\begin{bmatrix}1\cdot\begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix} & 2\cdot\begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix}\\ 3\cdot\begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix} & 4\cdot\begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix}\end{bmatrix}=\begin{bmatrix}1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5\\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7\\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5\\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7\end{bmatrix}=\begin{bmatrix}0 & 5 & 0 & 10\\ 6 & 7 & 12 & 14\\ 0 & 15 & 0 & 20\\ 18 & 21 & 24 & 28\end{bmatrix}$$

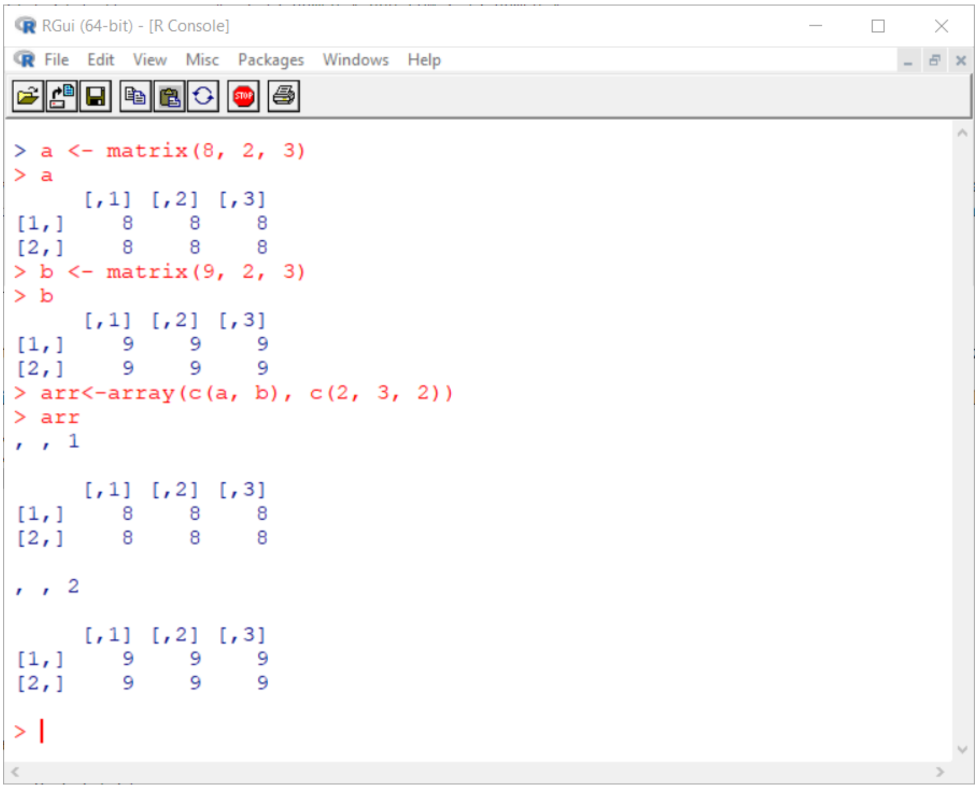
C'est très simple à obtenir dans R et il n'y a même pas besoin de packages particuliers, il suffit d'utiliser la fonction **kronecker( )** :



Exemple 22.: Utiliser des arrays

R 3.4.2

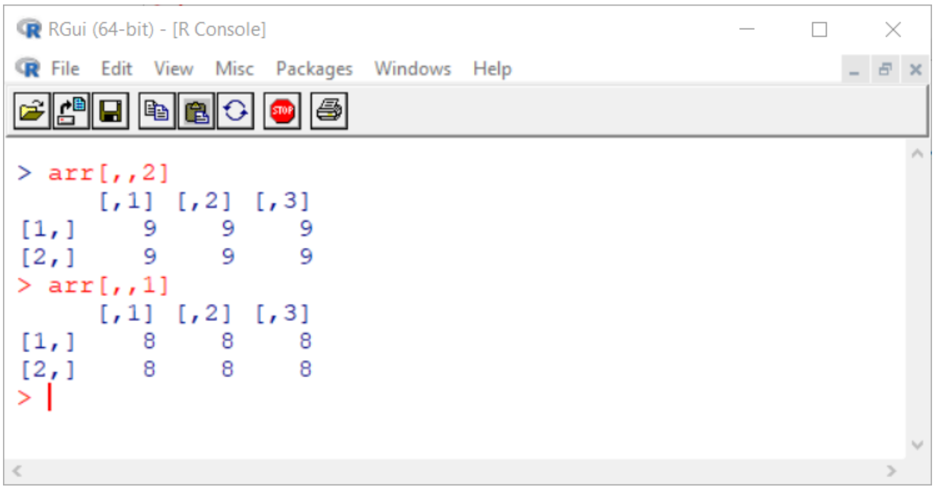
Les arrays (parfois assimilés à des "tenseurs") sont fréquents en traitement d'image (puisque les images RVB ne sont que des arrays avec 3 matrices, une pour chaque couche de couleur) mais aussi dans l'utilisation de certains réseaux de neurones. Voyons comment en créer un:



Où c(2,3,2) signifie:

c(*nombre de lignes des matrices*, *nombre de colonnes des matrices*, *nombre de matrices*)

Pour accéder aux sous-matrices:



Etc…

Exemple 23.: Utiliser des listes

R 3.0.2

Les listes, créées via la fonction **list( )** , sont considérées souvent comme le type de données le plus complexe de R car on peut y mélanger des vecteurs, des matrices et des textes en une seule entité (très pratique plutôt que de renvoyer lors de scripts des variables à n'en plus finir à l'utilisateur!). Voici une figure qui résume très bien la situation vue jusqu'ici:

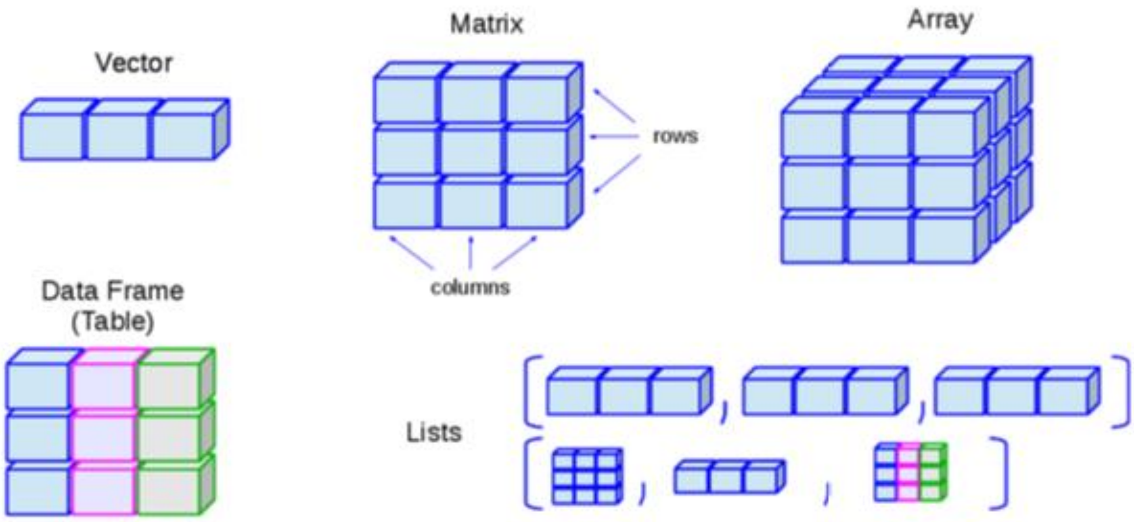
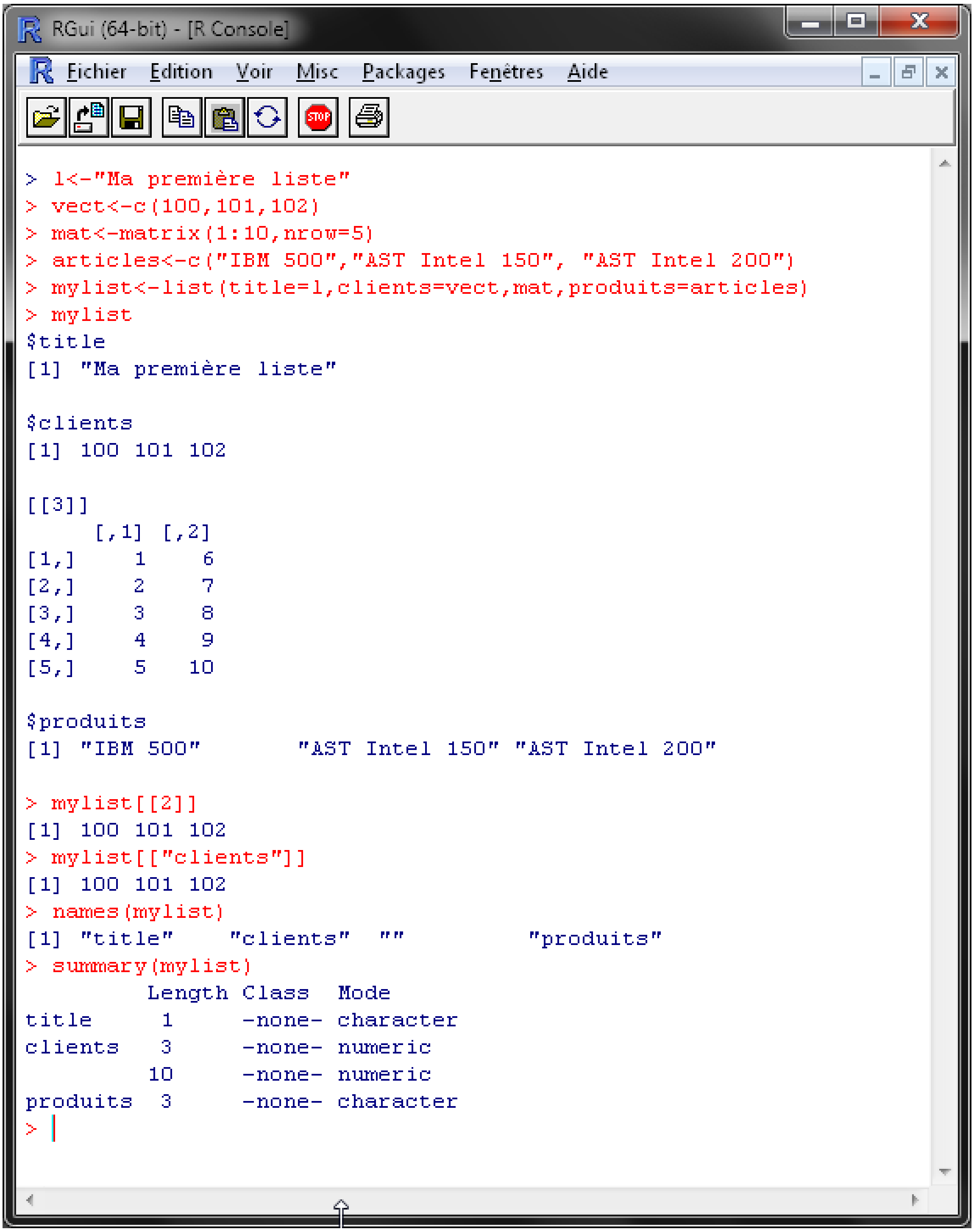
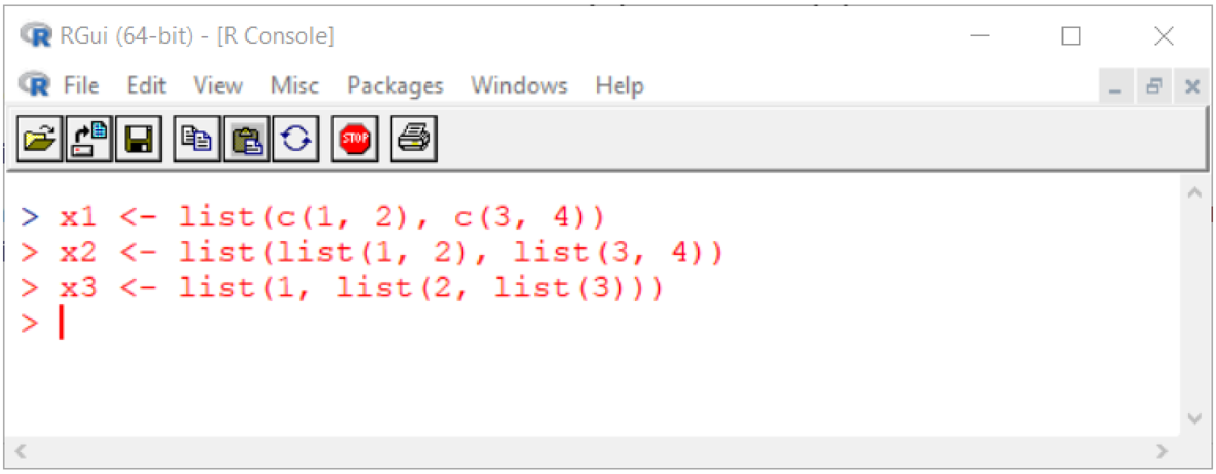


Figure 9 Vecteurs vs Matrice vs Array vs Dataframe vs Liste

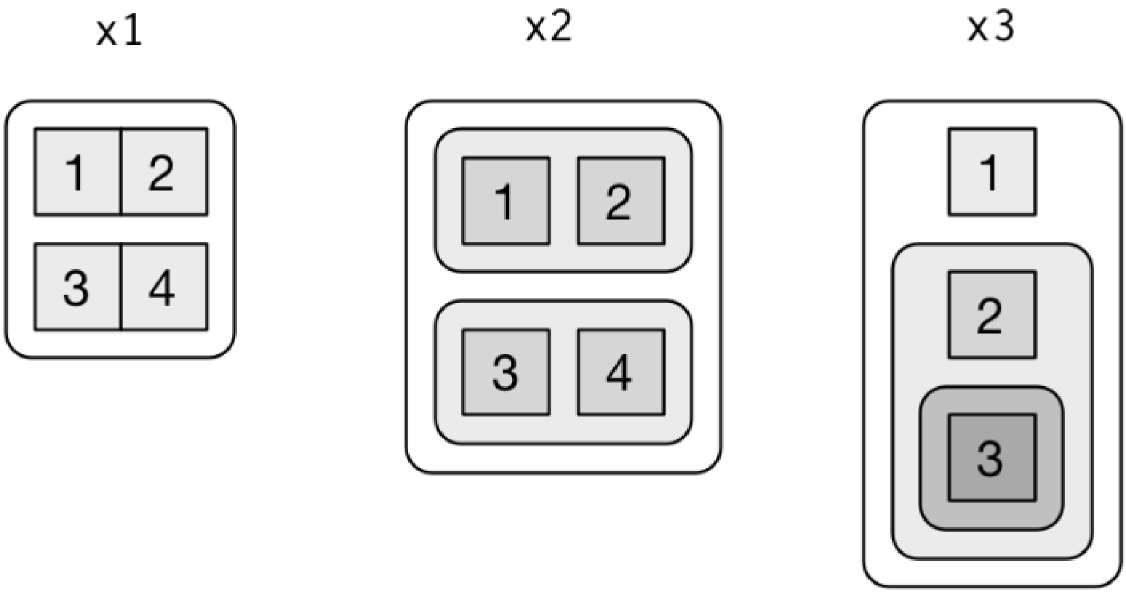
Voyons cela via un exemple:



Ou pour reprendre un exemple de Garrett Grolemund and Hadley Wickham qui est très bien illustré:



Ce qui donne de façon imagée:



Mais dans R (moins ludique…):

```
> x1
[[1]]
[1] 1 2

[[2]]
[1] 3 4

> x2
[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 2


[[2]]
[[2]][[1]]
[1] 3

[[2]][[2]]
[1] 4


> x3
[[1]]
[1] 1

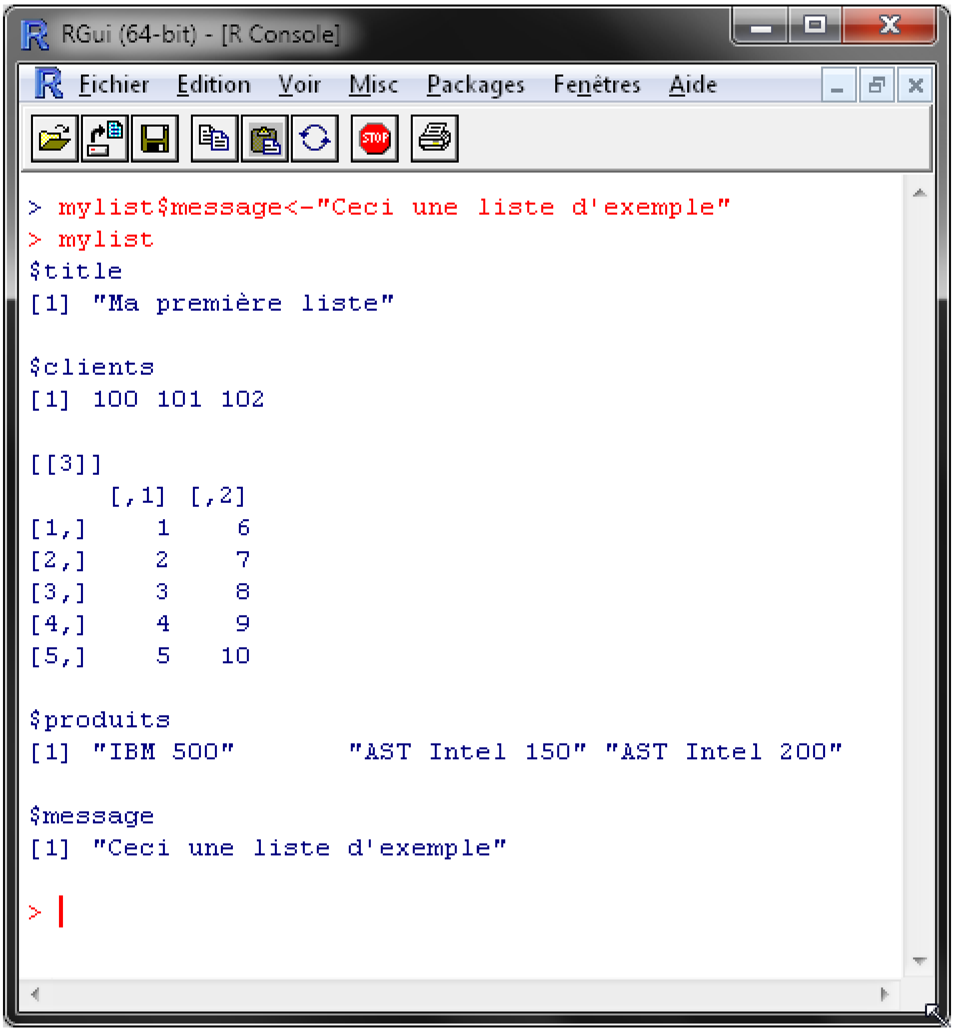
[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[[2]][[2]][[1]]
[1] 3

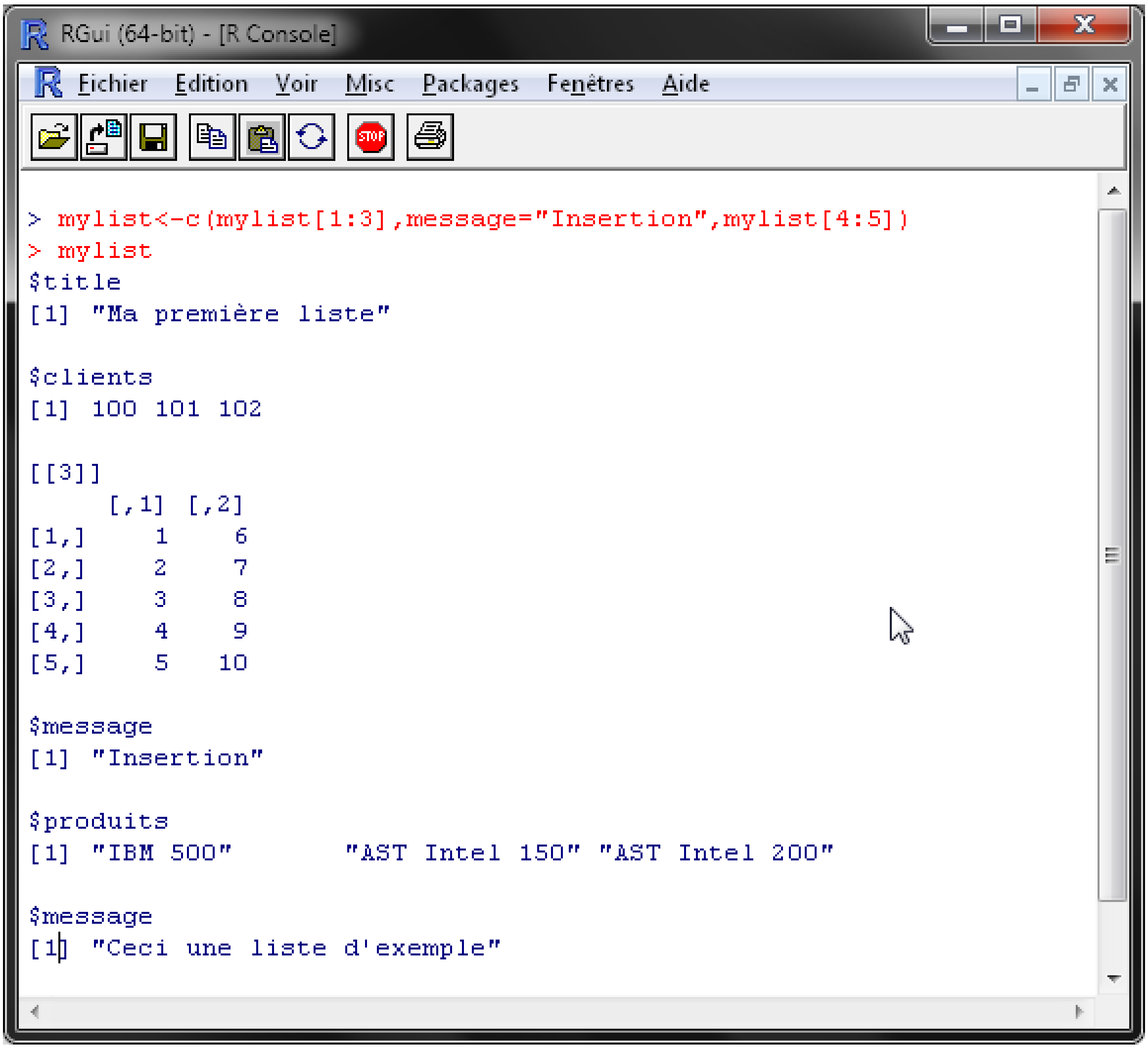


>
```

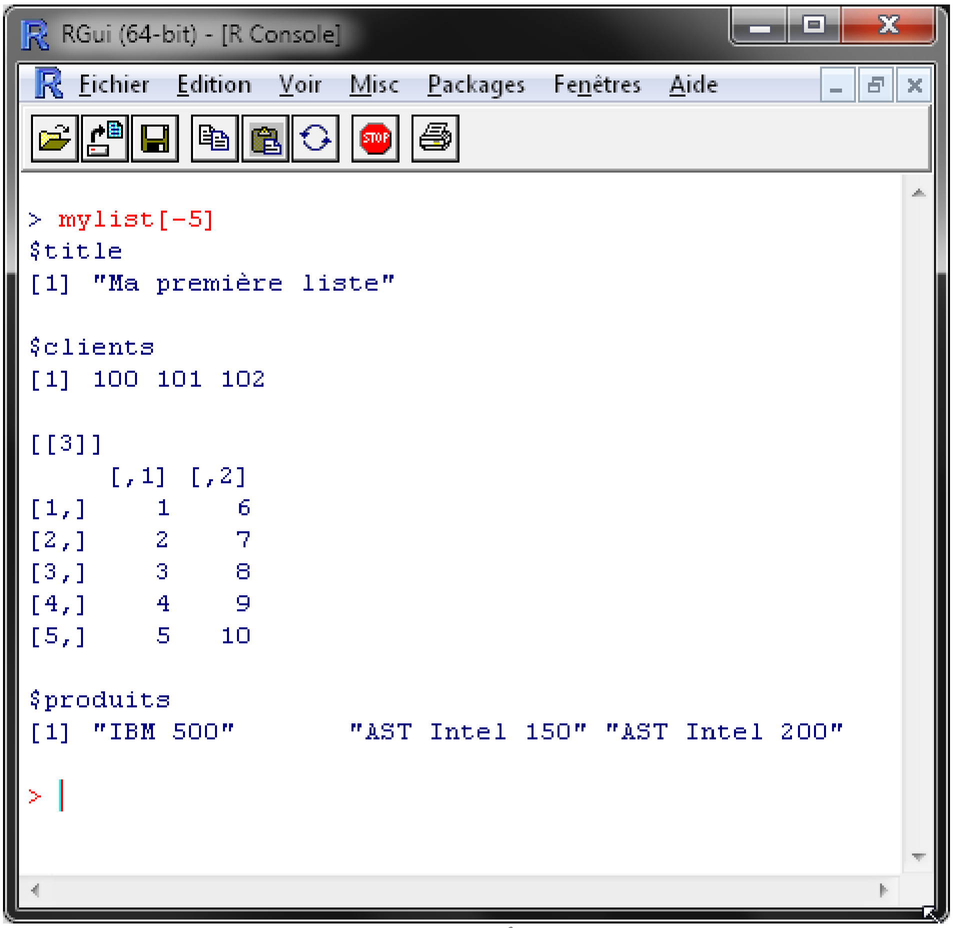
Il est très facile de rajouter des éléments à une liste existante:



Nous pouvons aussi choisir où exactement insérer la variable dans la liste:



Pour supprimer un élément d'une liste c'est tout aussi simple:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> mylist[-5]
$title
[1] "Ma première liste"

$clients
[1] 100 101 102

[[3]]
     [,1] [,2]
[1,]   1   6
[2,]   2   7
[3,]   3   8
[4,]   4   9
[5,]   5   10

$produits
[1] "IBM 500"   "AST Intel 150" "AST Intel 200"

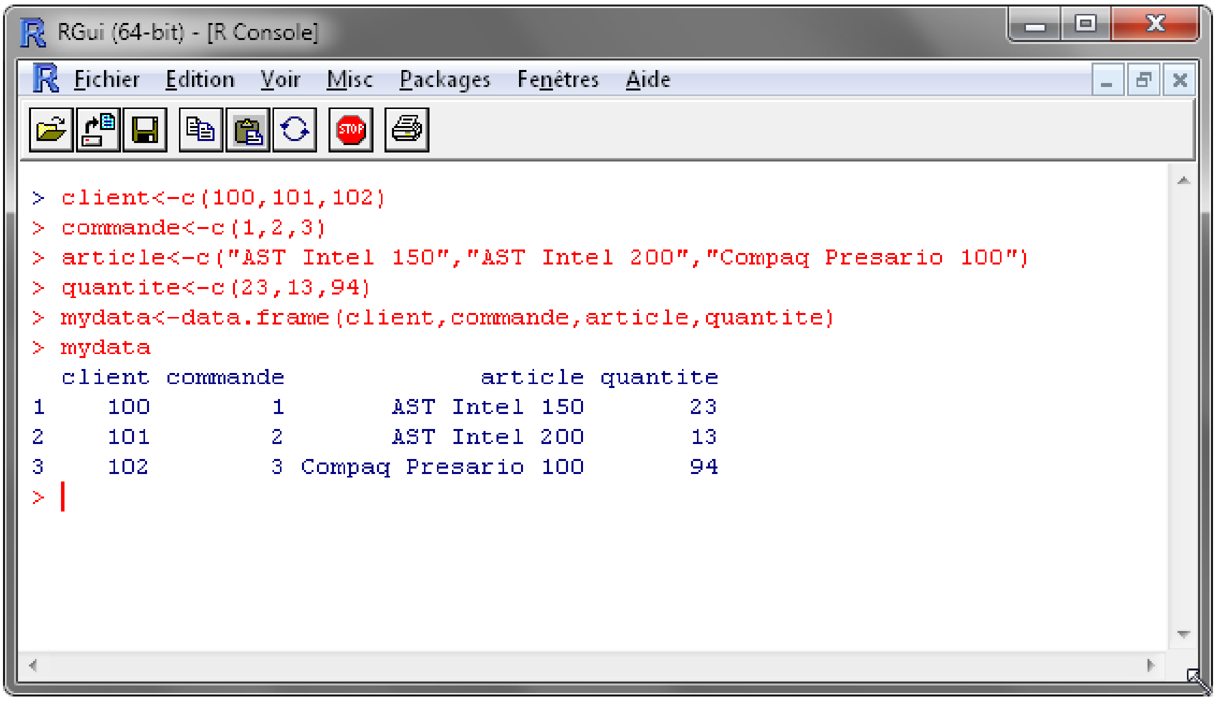
>

Exemple 24.: Création/Édition de données en ligne de commande

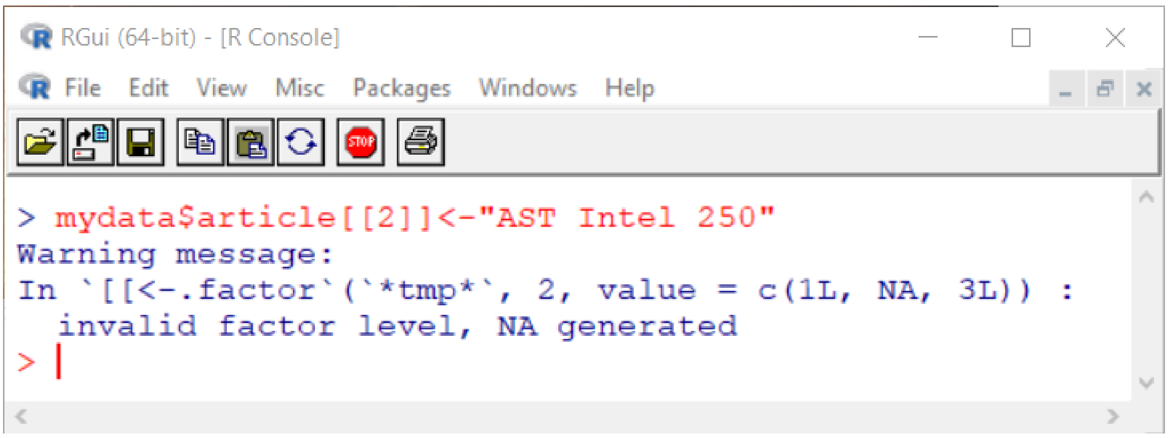
R 3.0.2

Bon voilà un sujet que nous n'avons traité ni dans le livre sur Minitab, ni dans celui sur SPSS puisque dans ces deux logiciels, la saisie se fait simplement comme dans un tableur.

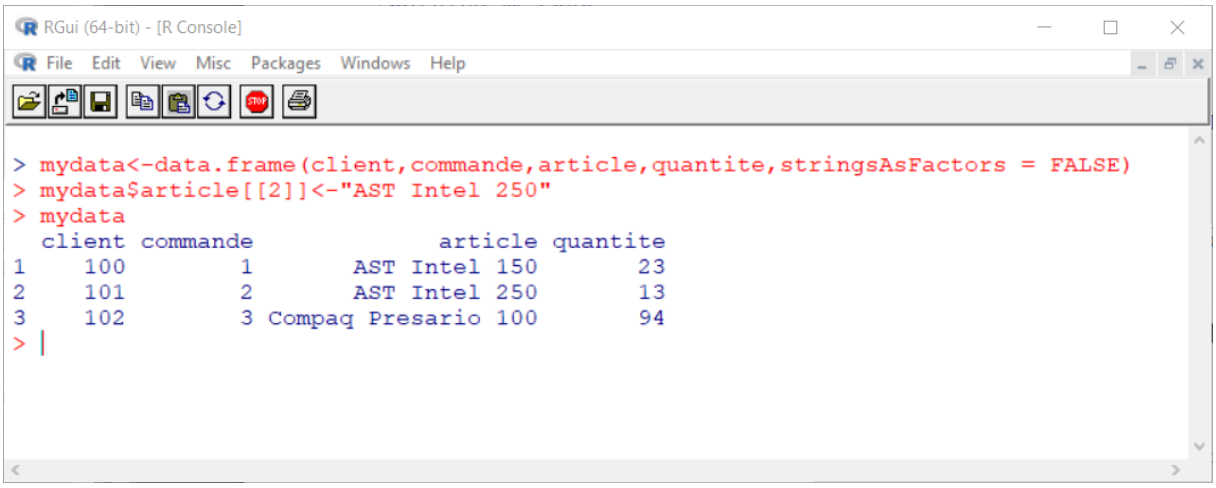
Dans R, il peut être utile cependant de savoir comment faire en utilisant la commande **c( )** pour générer des colonnes (vecteur) et la commande **data.frame( )** pour composer une matrice:



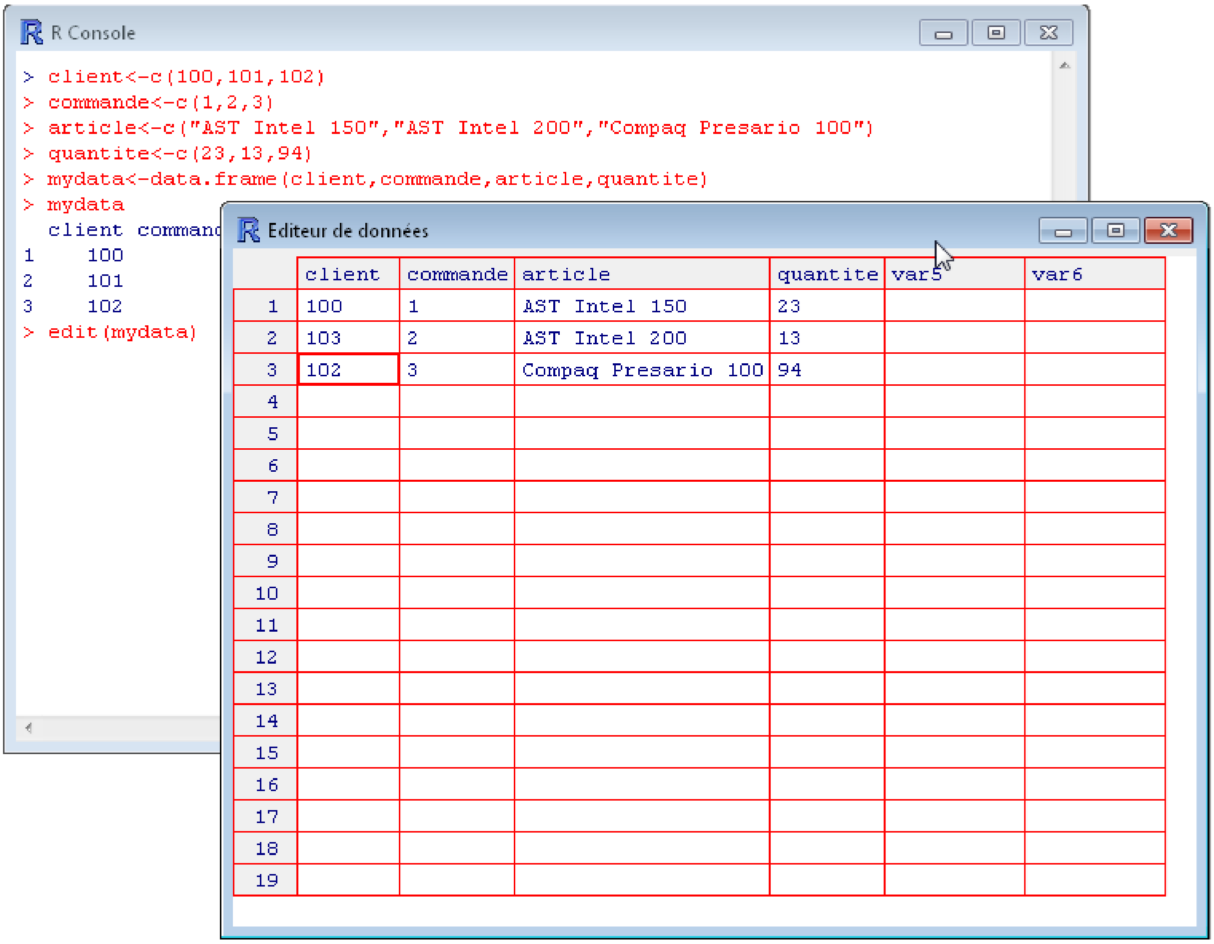
Attention!!! Si on construit un data frame de la manière ci-dessus on ne pourra plus changer facilement les valeurs des articles puisque la commande data.frame en fait automatiquement des niveaux (levels):



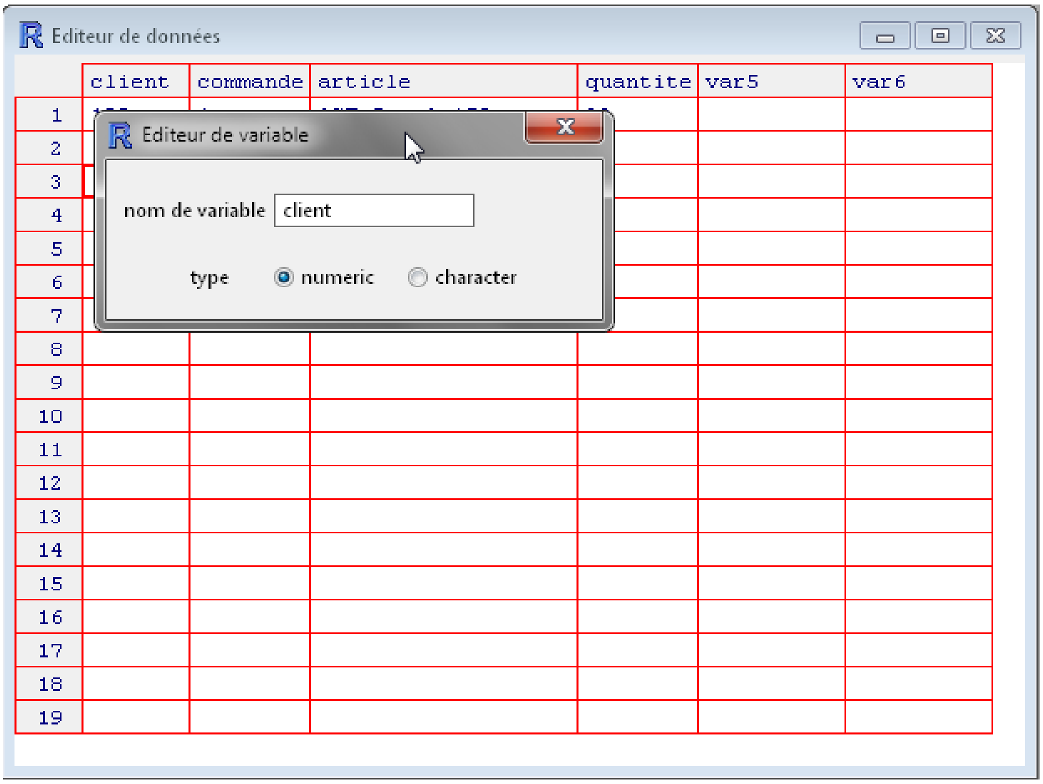
Il faut créer le data frame de la manière suivante en utilisant le paramètre **stringAsFactors** afin de pouvoir être libre de ses mouvements:



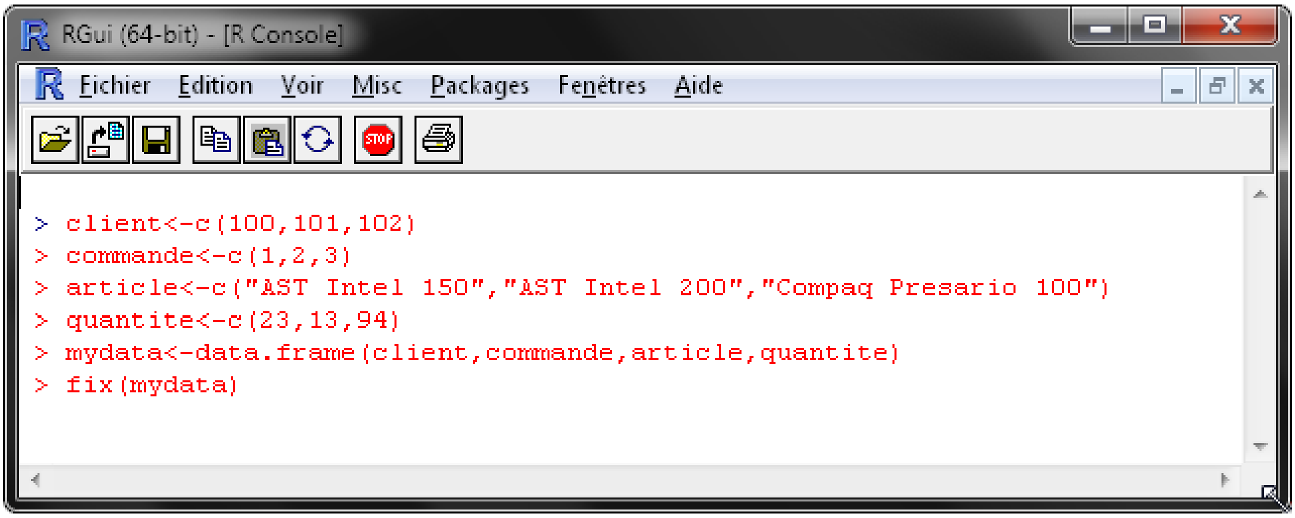
Ensuite pour changer les noms des variables on peut utiliser la commande **edit( )** dans une feuille similaire à celle d'un tableur:



et donc changer le nom des colonnes en cliquant dessus:

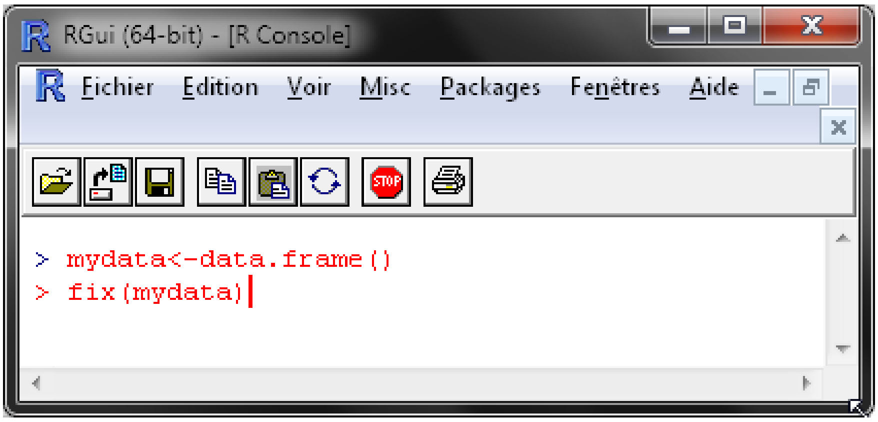


Par contre pour changer les valeurs, il vous faudra utiliser la commande **fix( )** :

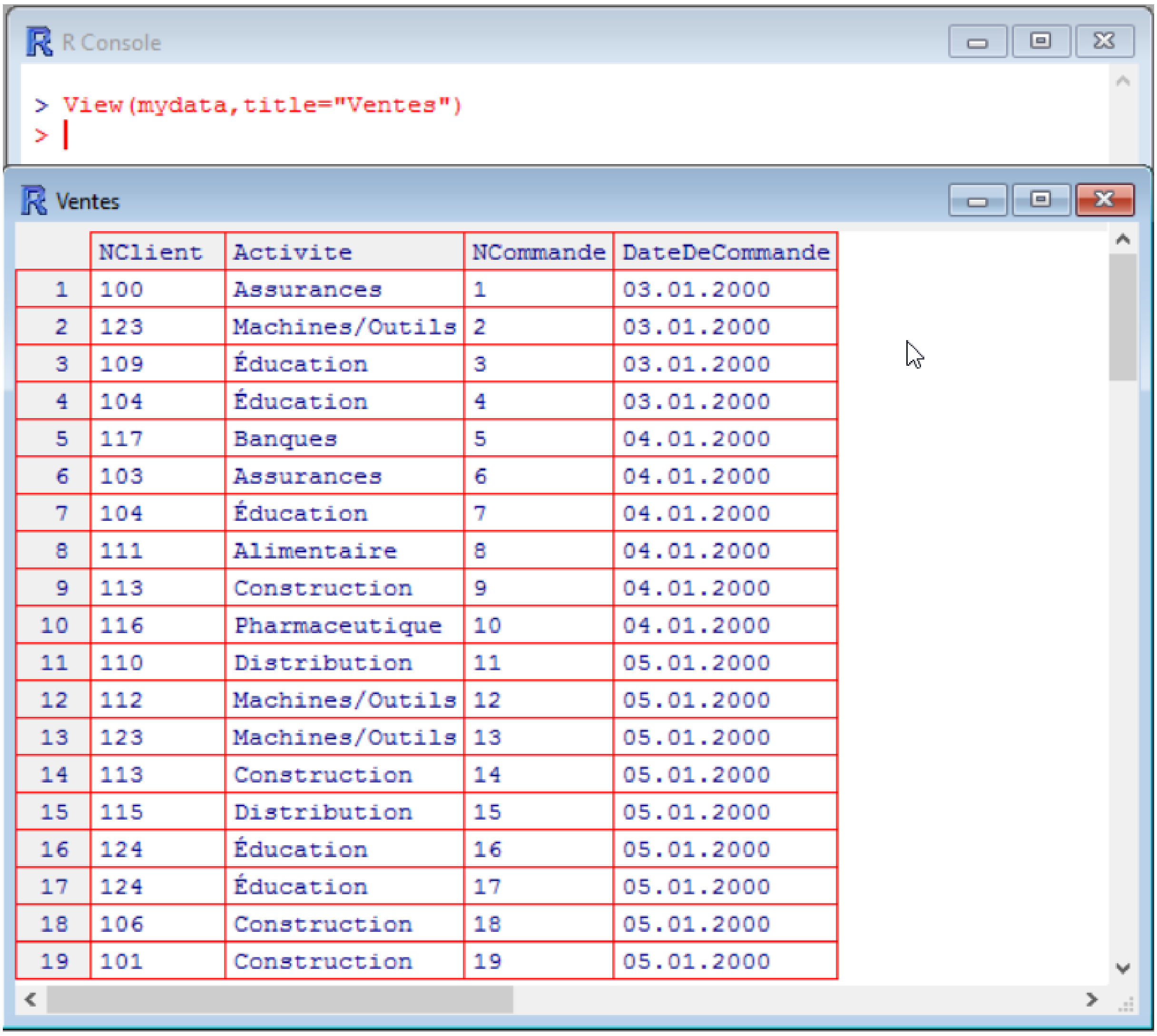


qui affiche exactement le même tableau mais cette fois avec les valeurs qui sont aussi modifiables directement.

Vous pouvez aussi utiliser **fix( )** pour créer un date frame en partant de zéro:



Enfin, indiquons que nous pouvons aussi afficher les données avec la commande **View** en ajoutant en plus un titre à la fenêtre:



	NClient	Activite	NCommande	DateDeCommande
1	100	Assurances	1	03.01.2000
2	123	Machines/Outils	2	03.01.2000
3	109	Éducation	3	03.01.2000
4	104	Éducation	4	03.01.2000
5	117	Banques	5	04.01.2000
6	103	Assurances	6	04.01.2000
7	104	Éducation	7	04.01.2000
8	111	Alimentaire	8	04.01.2000
9	113	Construction	9	04.01.2000
10	116	Pharmaceutique	10	04.01.2000
11	110	Distribution	11	05.01.2000
12	112	Machines/Outils	12	05.01.2000
13	123	Machines/Outils	13	05.01.2000
14	113	Construction	14	05.01.2000
15	115	Distribution	15	05.01.2000
16	124	Éducation	16	05.01.2000
17	124	Éducation	17	05.01.2000
18	106	Construction	18	05.01.2000
19	101	Construction	19	05.01.2000

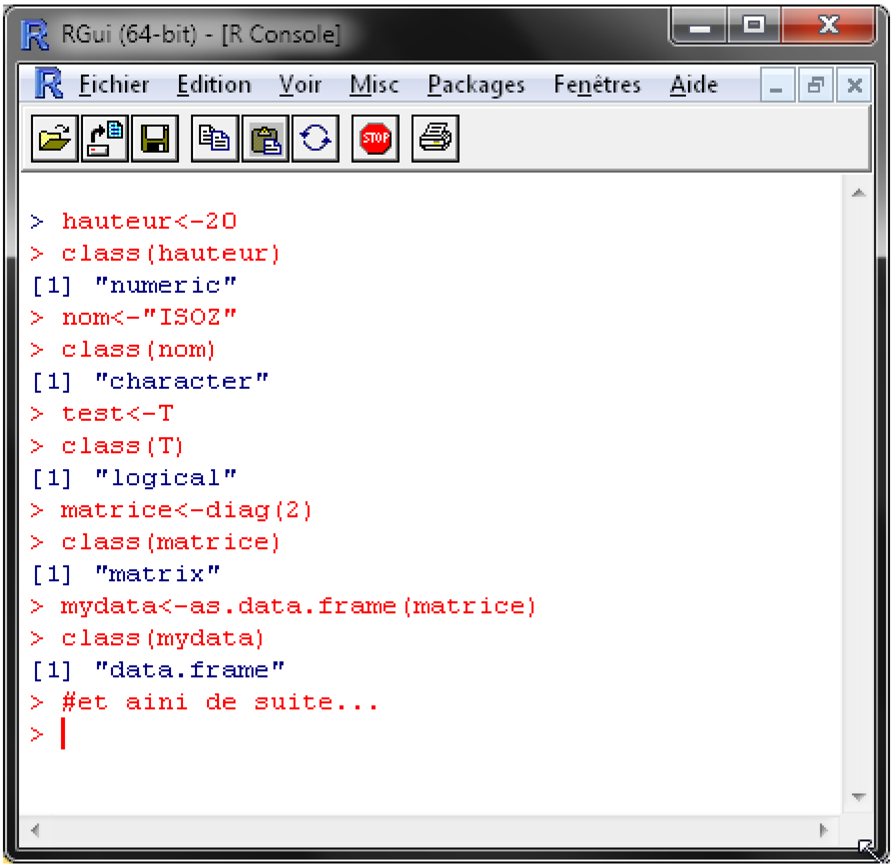
Il existe des méthodes beaucoup plus élaborées pour l'édition de données. Particulièrement le package **rhandsontable** dont une vignette est disponible ici:

https://jrowen.github.io/rhandsontable/

Exemple 25.: Identifier le typage

R 3.0.2

Lorsque vous utiliserez des scripts pris d'Internet et que vous en décortiquerez le fonctionnement il vous faudra parfois déterminer le typage d'une variable. Voyons quelques exemples utiles à l'aide de la commande **class( )** :



8. Algèbre scolaire

Contenu

Exemple 26.: Plotter (tracer) des fonctions algébriques

R 3.0.2

Bon rien de bien compliqué pour commencer avec les graphes. Nous allons tracer quelques polynômes avec des petites améliorations graphiques mineures (on complexifiera au fur à mesure de notre avancement dans le présent document avec de histogrammes, des doubles axes, etc.) avec les commandes **function( )** , **plot( )** , **grid( )** et **legend( )** :

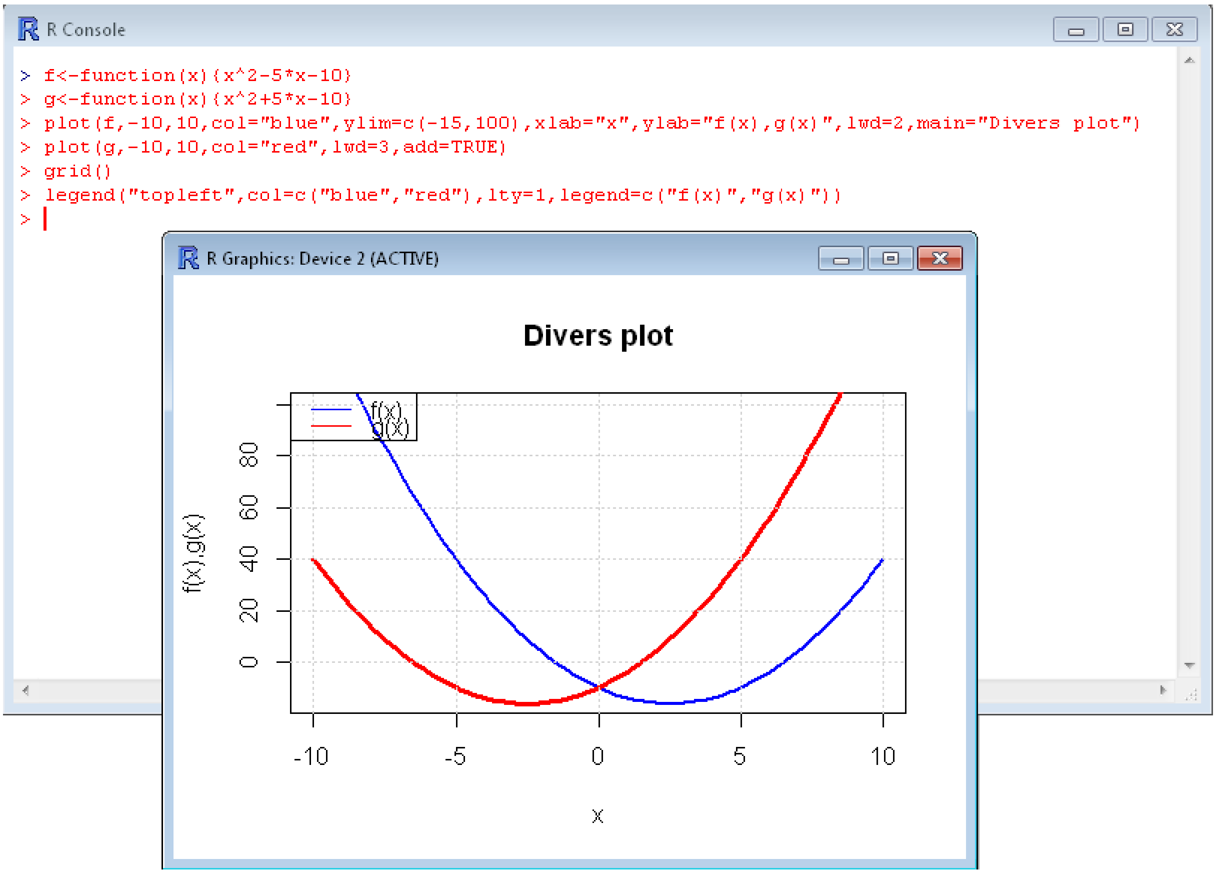
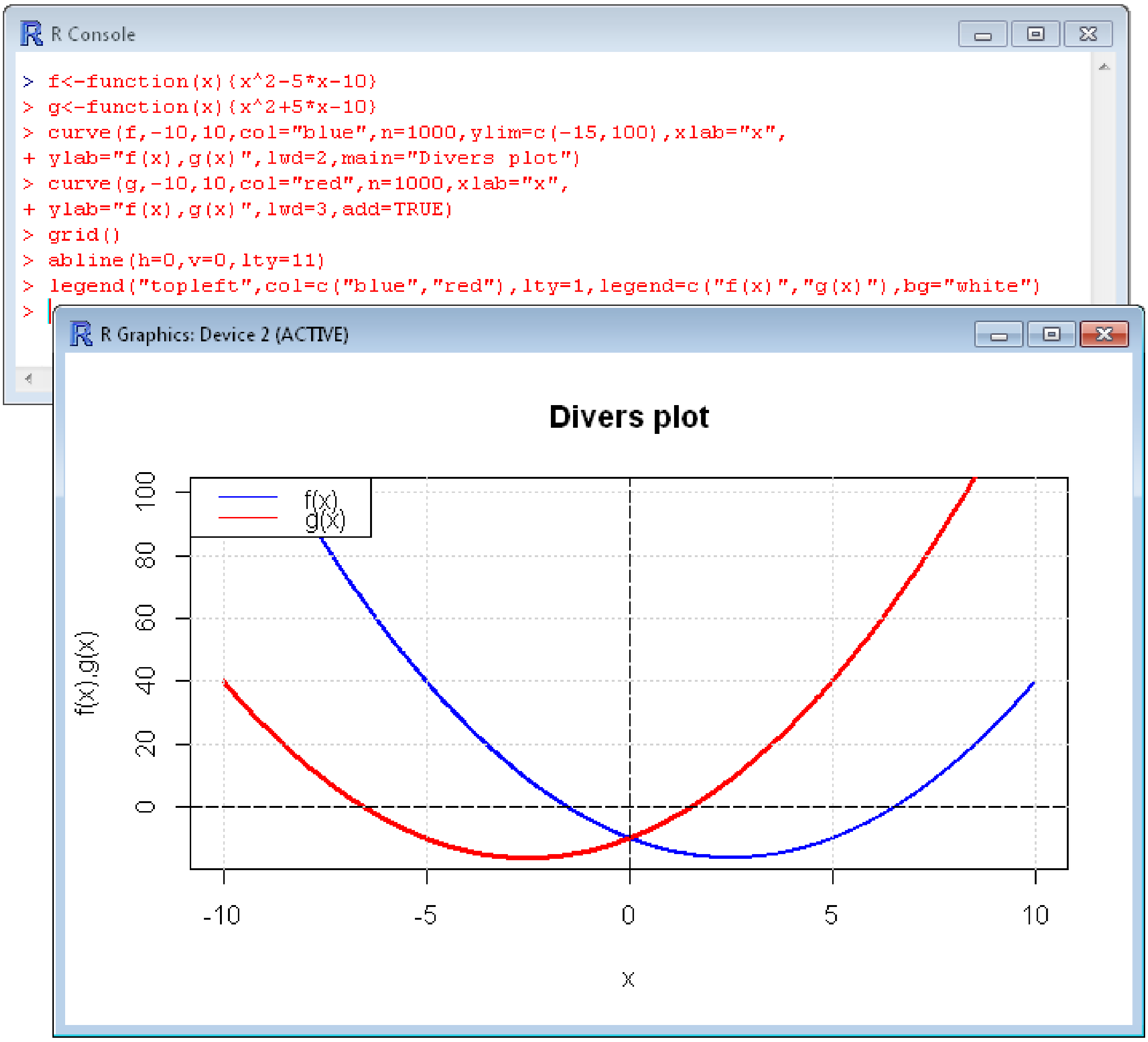
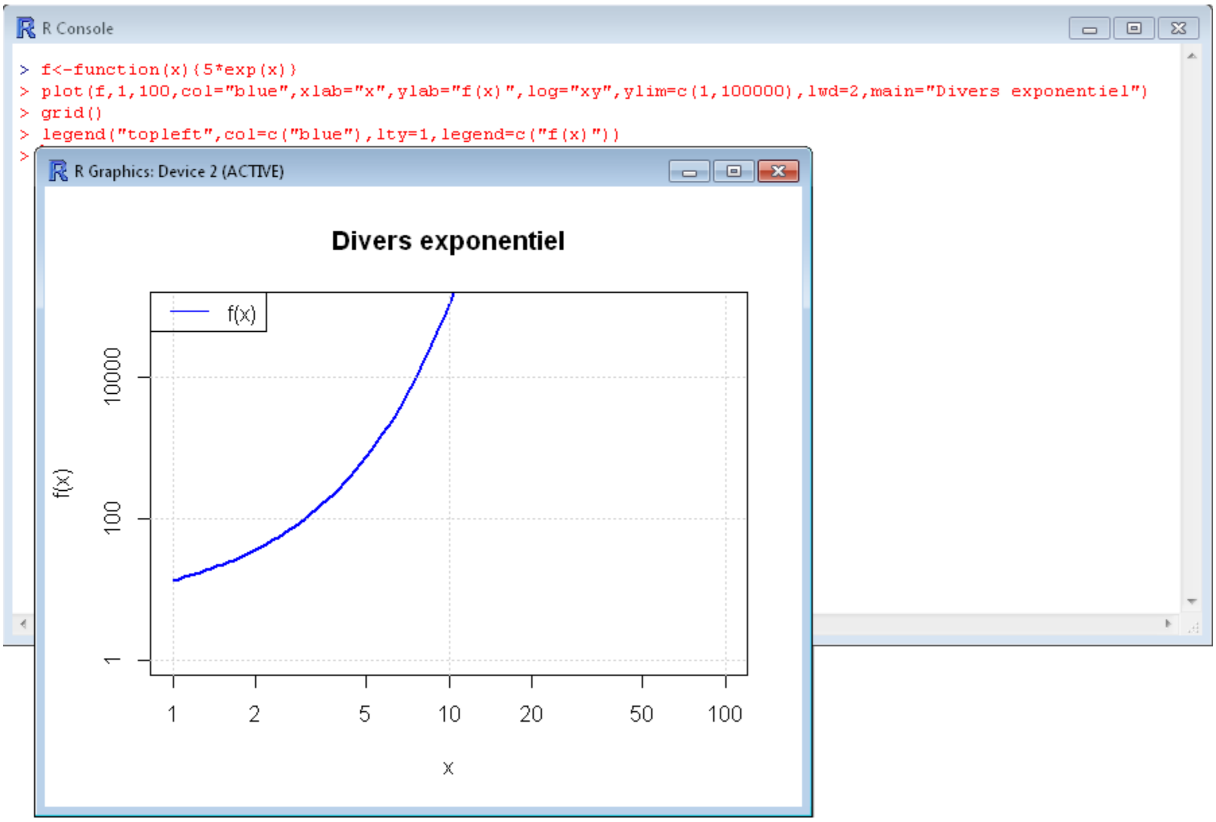


Figure 10 Graph de fonction f(x)

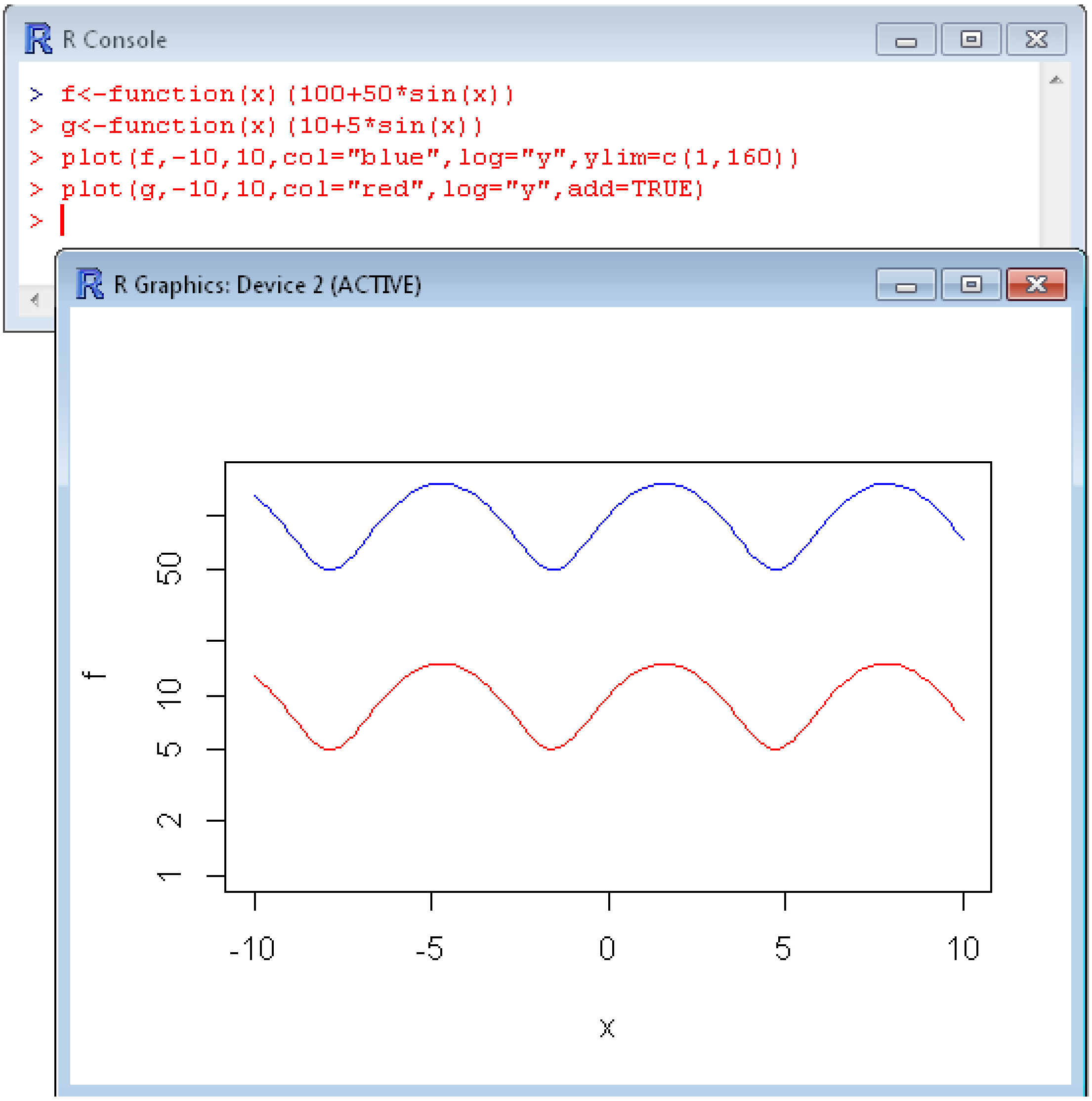
Nous pouvons obtenir la même chose avec la commande **curve( )** en améliorant au passage un peu le résultat (amélioration que l'on aurait pu appliquer aussi directement à l'exemple ci-dessus):



Ou une fonction exponentielle avec les deux axes en échelle logarithmique (de façon un peu basique):



Pour avoir que l'axe Y en échelle logarithmique il vous suffit de mettre de changer l'argument log (avec un petit exemple clin d'oeil scolaire de l'utilité de l'échelle logarithmique):



Avec certaines fonctions, R transforme directement l'expression algébrique de façon esthétique dans les légendes ou titres par exemple en utilisant la commande **expression( )** :

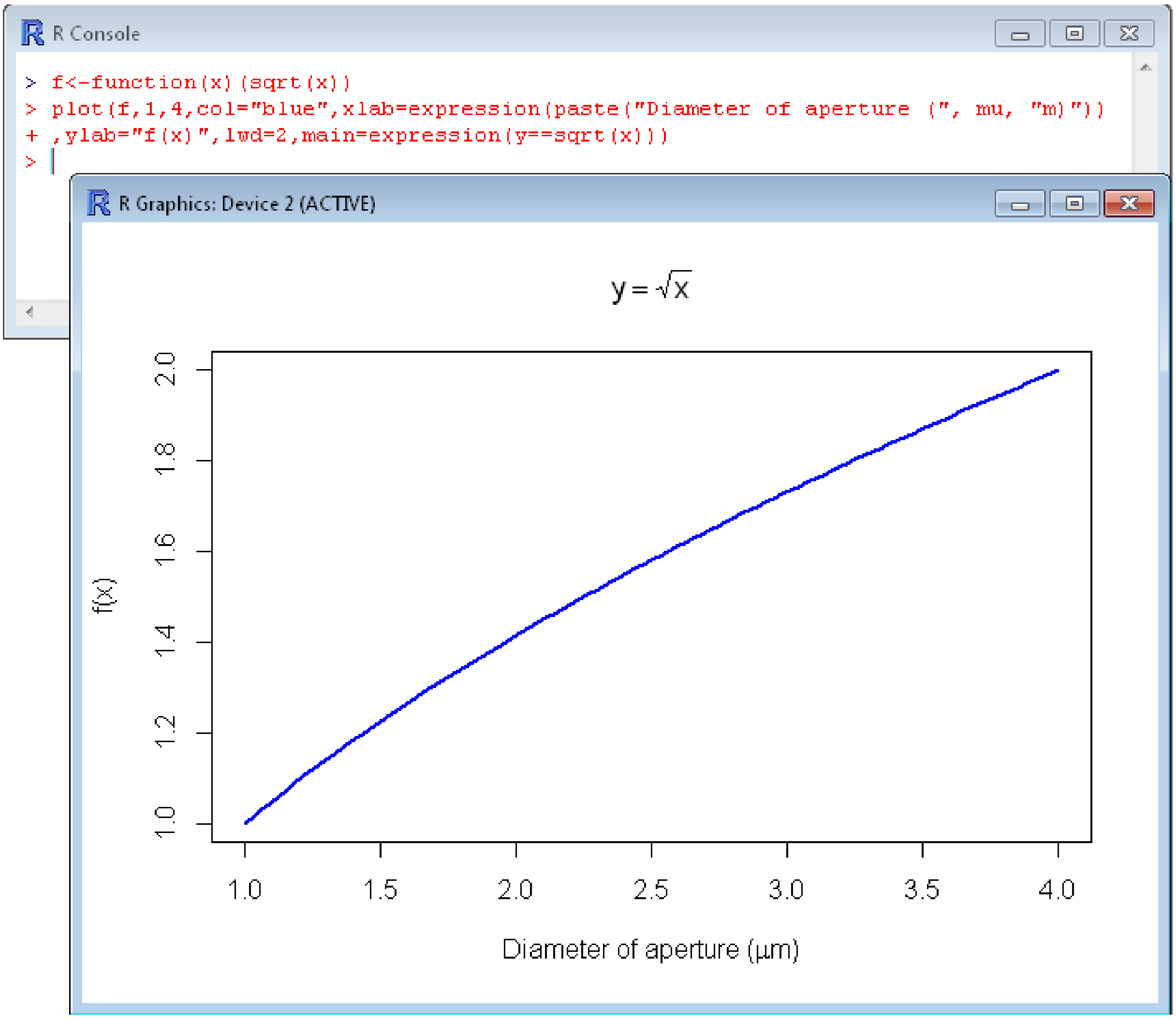


Figure 11 Graph avec équation dans le titre

ou encore:

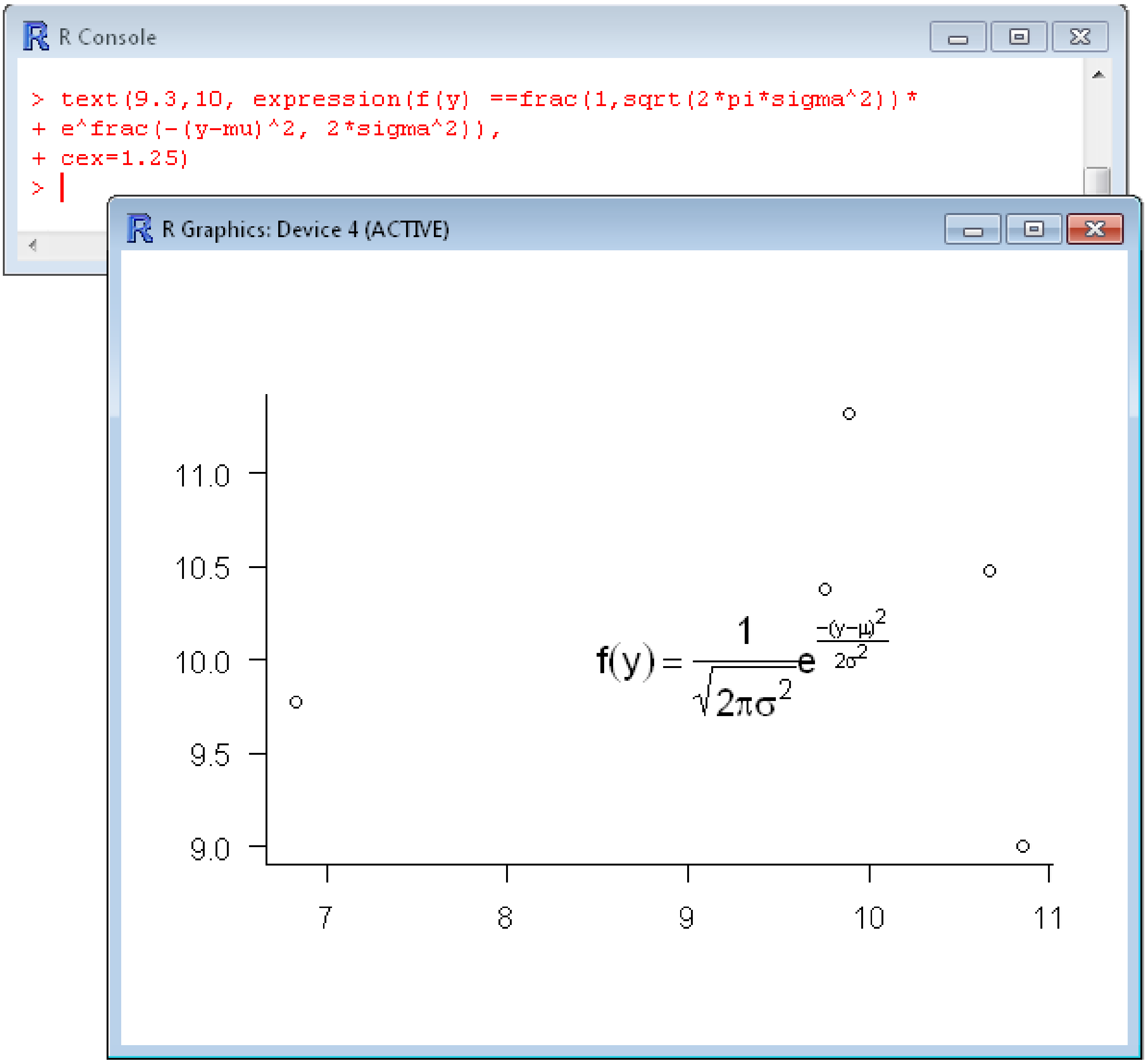
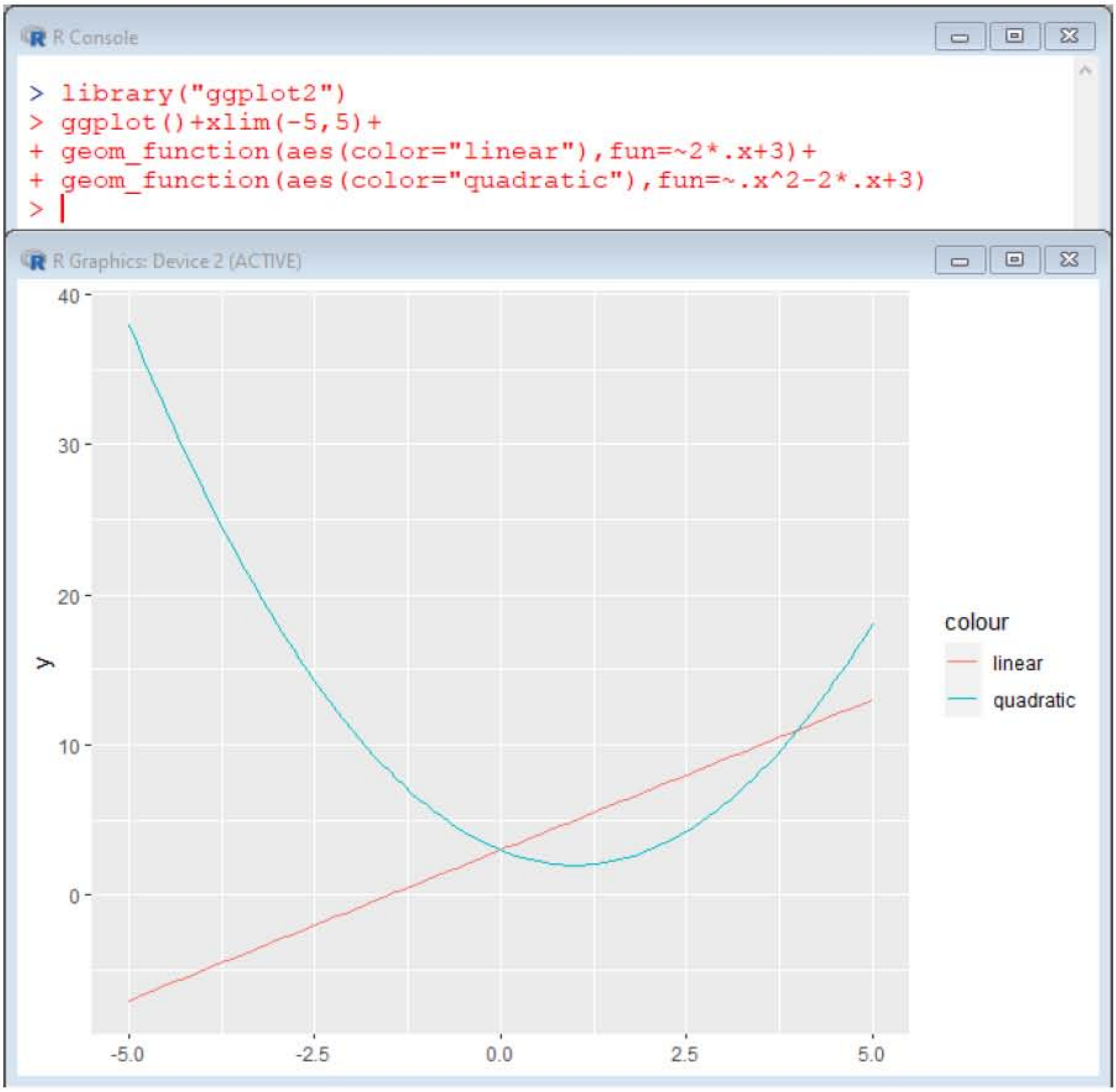


Figure 12 Graph avec équation dans la zone de traçage

Évidemment l'intérêt de savoir faire ce type de graphique c'est typiquement en finance lorsque l'on connaît explicitement le polynôme du deuxième degré de la frontière efficiente, de pouvoir aussi tracer la security market line dont l'équation de la droite est aussi explicitement connue!

Sinon, depuis mi 2020 on peut faire des tracer très simplement des fonctions analytiques avec le package ggplot2 comme le montre l'exemple ci-dessous à l'aide de **geom_function( )**:


R Console
> library("ggplot2")
> ggplot()+xlim(-5,5)+
+ geom_function(aes(color="linear"),fun=~2*.x+3)+
+ geom_function(aes(color="quadratic"),fun=~.x^2-2*.x+3)
>
R Graphics: Device 2 (ACTIVE)
40
30
20
10
0
y
-5.0
-2.5
0.0
2.5
5.0
colour
linear
quadratic

Exemple 27.: Animer (tracer) des fonctions algébriques

R 3.0.2

Avec la package **animation**, nous pouvons animer un plot dans un fichier *.gif à l'aide de la commande **savegif( )** .

Remarque: Faire des animations est très utile surtout pour les statistiques afin de voir la convergence des distributions ou dans les plots de cartes pour voir des chemins ou des densités évoluer.

Il faudra cependant au préalable télécharger et installer le logiciel gratuit:

http://www.imagemagick.org

Par exemple en animant simplement un sinus et un cosinus:

```
> library(animation)
>
> saveGIF({
+   for(i in 1:100){
+     curve(sin(x),from=-5+(i*0.05),to=5+(i*0.05),col="red",ylab="")
+     curve(cos(x),from=-5+(i*0.05),to=5+(i*0.05),add=TRUE,col="blue",ylab="")
+     legend("topright",legend=c("sin(x)","cos(x)"),fill=c("red","blue"),bty="n$
+   }
+ },interval=0.1,ani.width=550,ani.height=350)
Executing:
"C:\Program Files\ImageMagick-6.8.8-Q16\convert.exe" -loop 0 -delay 10
    C:/Users/ISOZVI~1/AppData/Local/Temp/RtmpIvEj2j/Rplot1.png
    C:/Users/ISOZVI~1/AppData/Local/Temp/RtmpIvEj2j/Rplot2.png
    C:/Users/ISOZVI~1/AppData/Local/Temp/RtmpIvEj2j/Rplot3.png
```

ce qui donne une image animée *.gif enregistrée dans un dossier temporaire de votre disque:

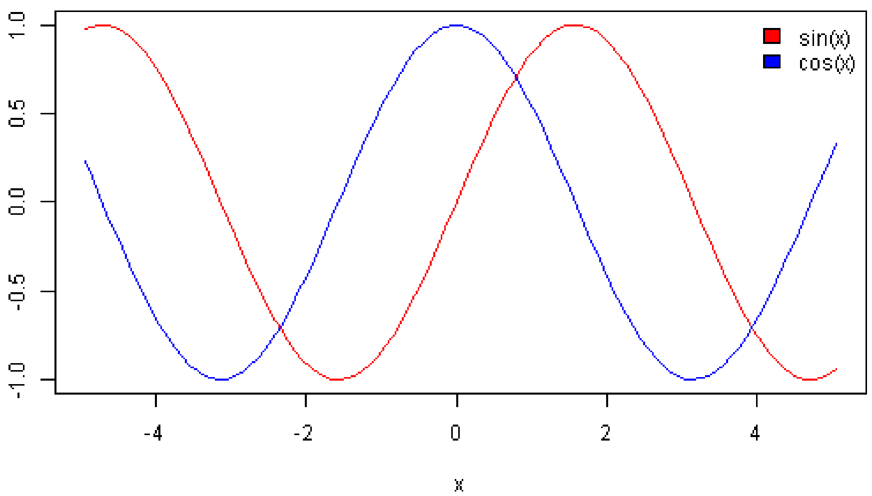


Figure 13 Graph animé au format GIF de fonctions f(x)

Sinon avec le package **av** et sa fonction **av_encode_video( )** on peut créer aussi une vidéo au format *.mp4. L'exemple reproduit ci-dessus mais avec une sortie vidéo donnera:

```
> library("av")
> png("input%03d.png", width = 1280, height = 720, res = 108)
> for(i in 1:100){
+ curve(sin(x),from=-5+(i*0.05),to=5+(i*0.05),col="red",ylab="")
+ curve(cos(x),from=-5+(i*0.05),to=5+(i*0.05),add=TRUE,col="blue",ylab="")
+ legend("topright",legend=c("sin(x)","cos(x)"),fill=c("red","blue"))
+ }
> dev.off()
null device
          1
> png_files <- sprintf("input%03d.png", 1:100)
> av_encode_video(png_files, 'output.mp4', framerate = 1)
using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
profile High, level 6.0, 4:2:0, 8-bit
264 - core 157 r2932M 303c484 - H.264/MPEG-4 AVC codec - Copyleft 2003-2018 - ht$
Output #0, mp4, to 'C:\tmp\output.mp4':
  Metadata:
    encoder         : Lavf58.12.100
    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuv420p, 1280x720, q$
Adding frame 101 at timestamp 100.00sec (100%) - video stream completed!
frame I:101   Avg QP: 2.90  size: 27614
mb I  I16..4: 89.5%  0.6%  9.9%
8x8 transform intra:0.6%
coded y,uvDC,uvAC intra: 5.8% 11.8% 11.2%
i16 v,h,dc,p: 91%  6%  3%  0%
i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 55% 17% 27%  0%  0%  0%  0%  0%  1%
i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 23% 17% 41%  3%  2%  6%  1%  5%  1%
i8c dc,h,v,p: 93%  3%  3%  0%
kb/s:220.91
[1] "C:\\tmp\\output.mp4"
> utils::browseURL('output.mp4')
>
```

Ce qui résulte bien en une vidéo dans le dossier indiqué à la fin du script:

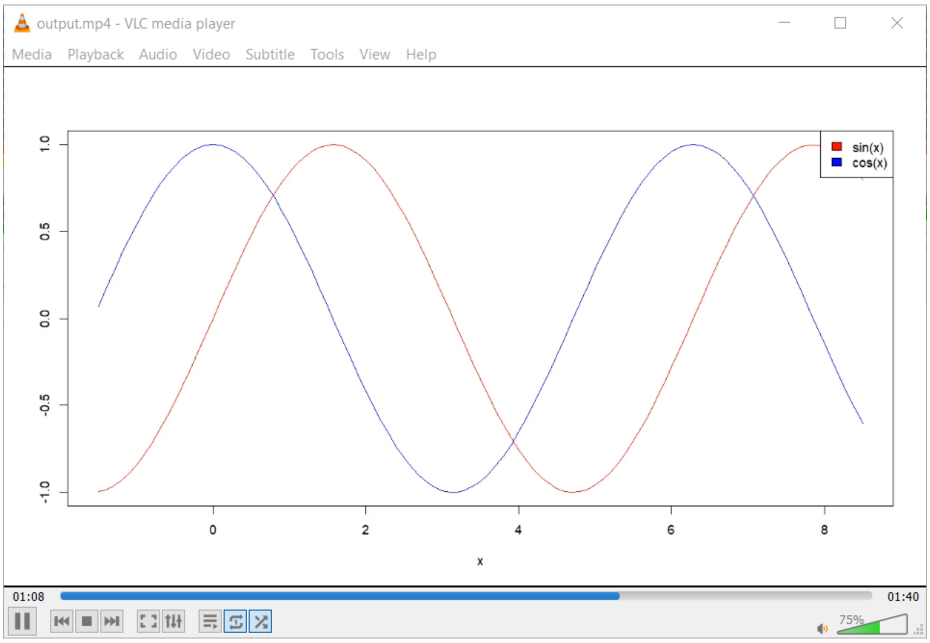
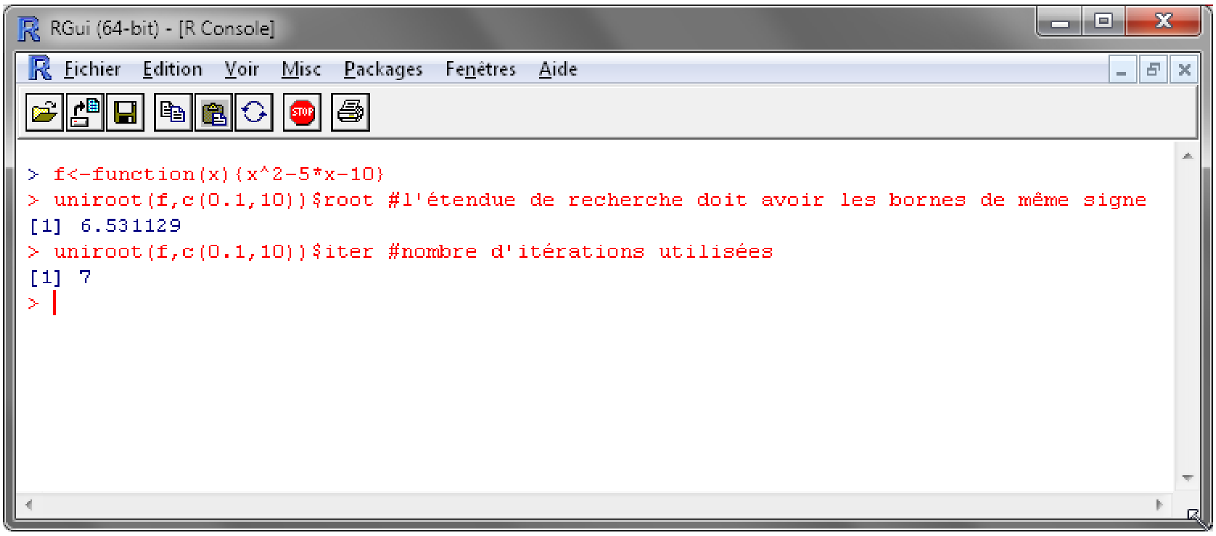


Figure 14 Graph animé au format AVI de fonctions f(x)

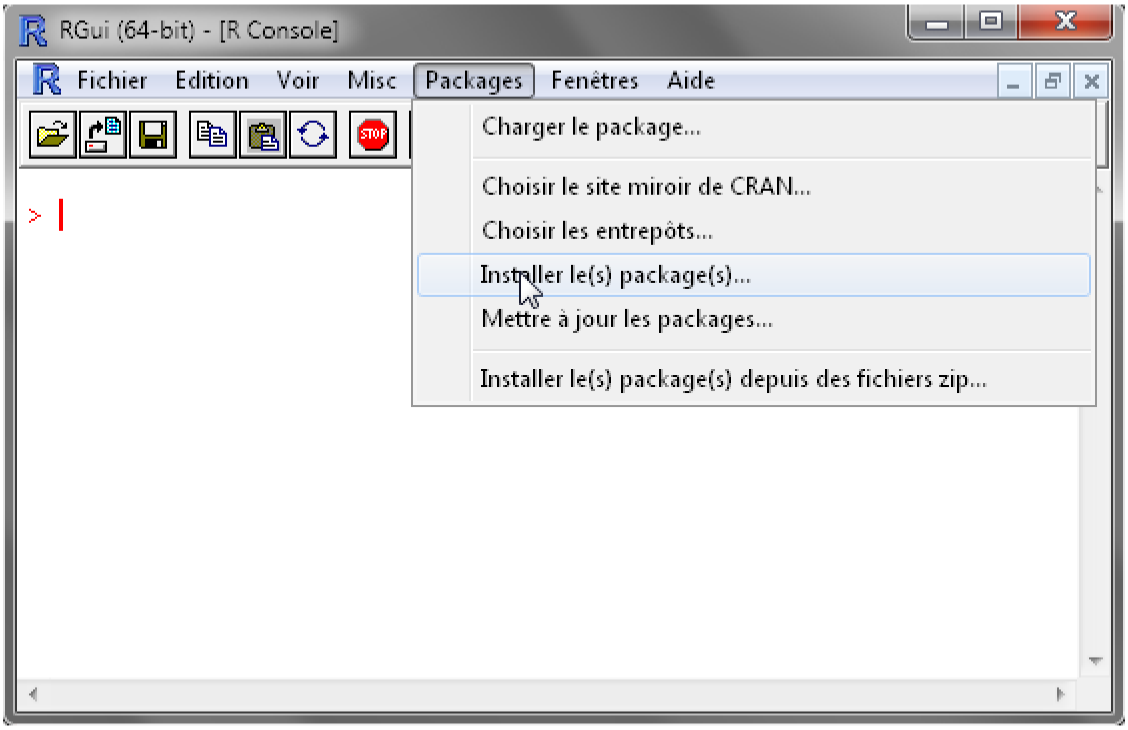
Exemple 28.: Racine(s) d'équations univariées

R 3.0.2

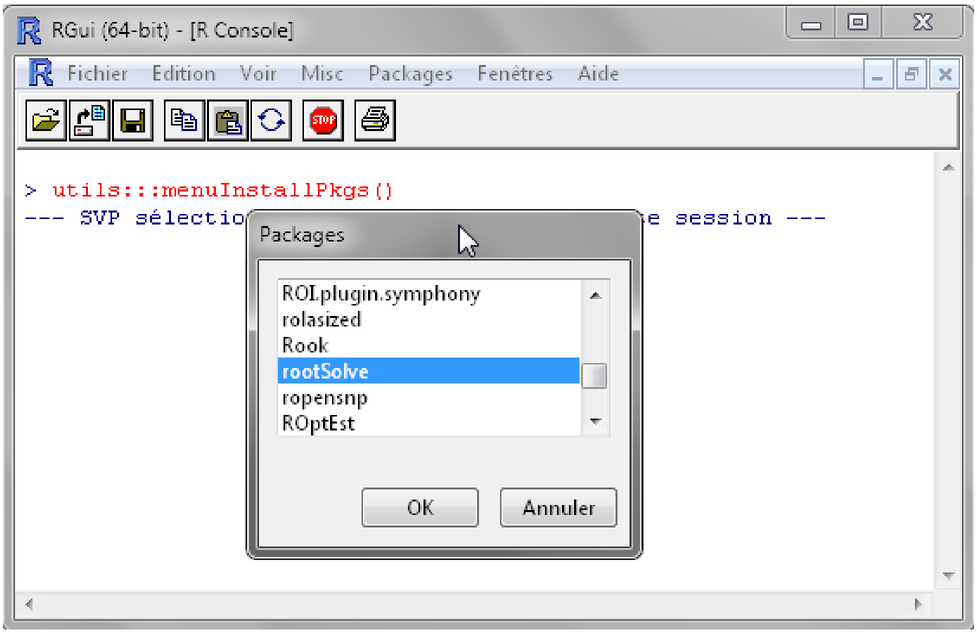
Pour rester dans les bases, voyons la méthode permet de trouver une racine unique à une équation supposée avoir des racines en utilisant la commande **uniroot( )** :



Pour trouver des racines multiples il faut à ma connaissance installer un package:



et choisir celui se nomme **rootSolve**:



Une fois ceci fait, nous pouvons retourner à notre exemple en chargeant la libraire avec la command **library( )** et en cherchant les racines avec **uniroot.all( )** et en les plottant avec la commande **points( )** :

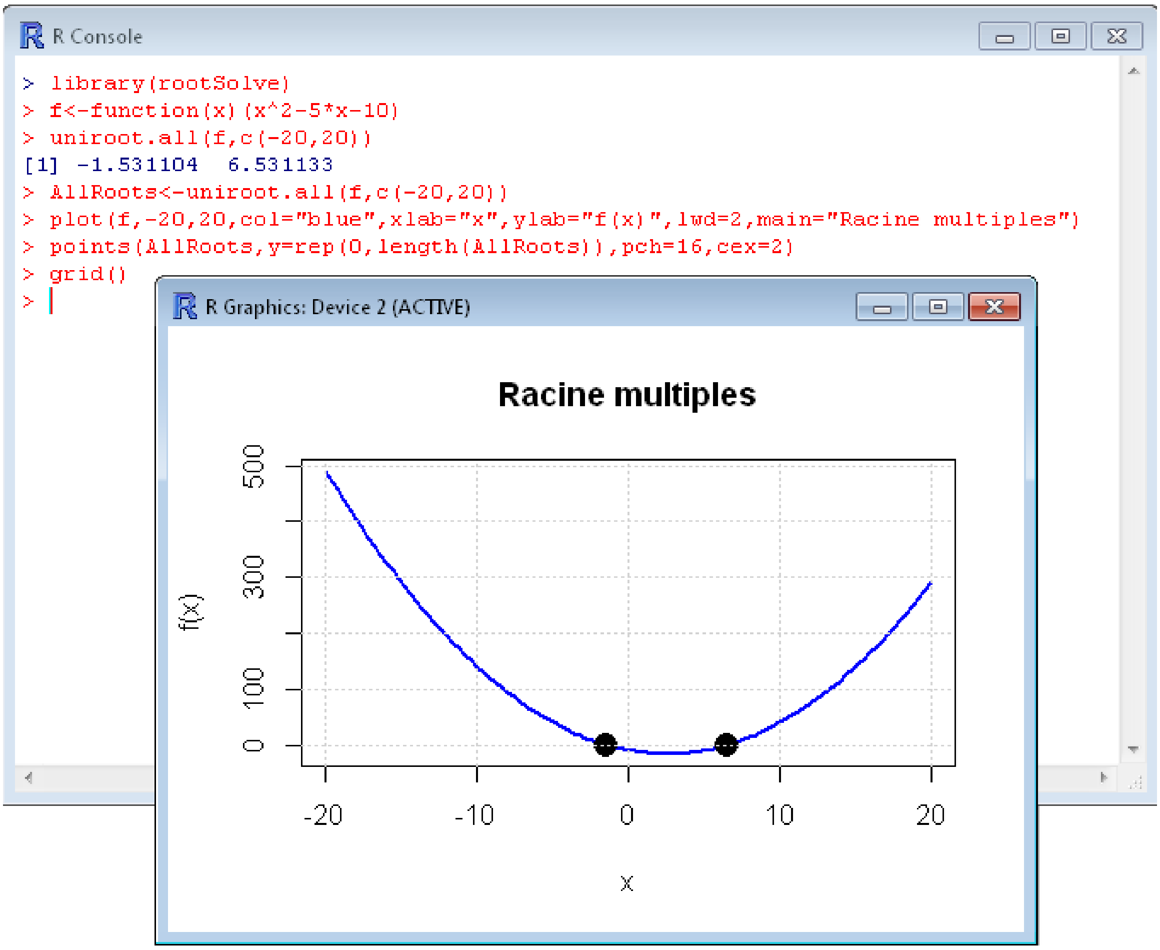
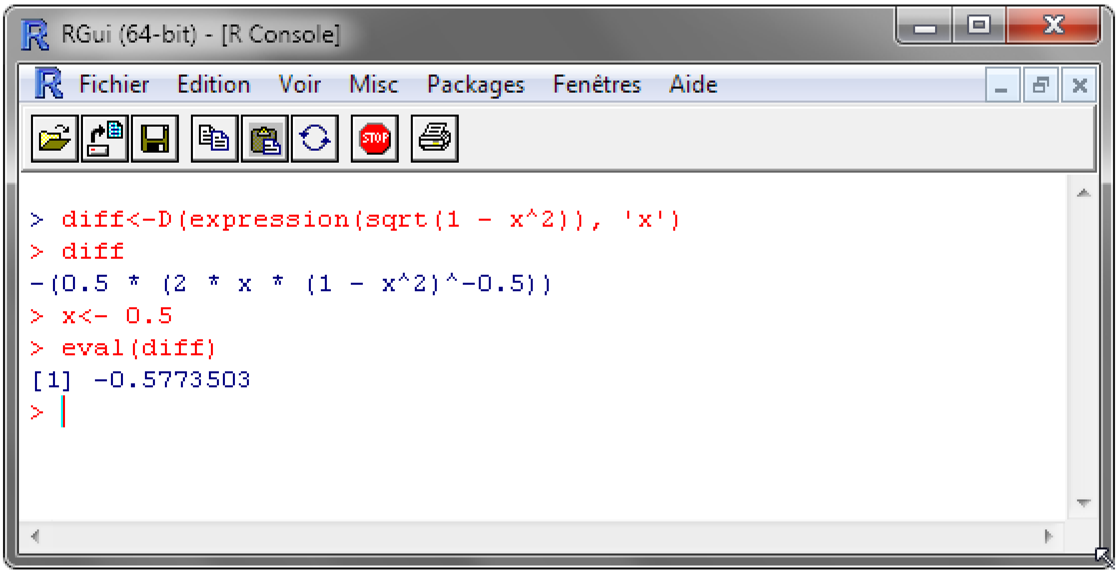


Figure 15 Grap avec fonction f(x) et ses racines

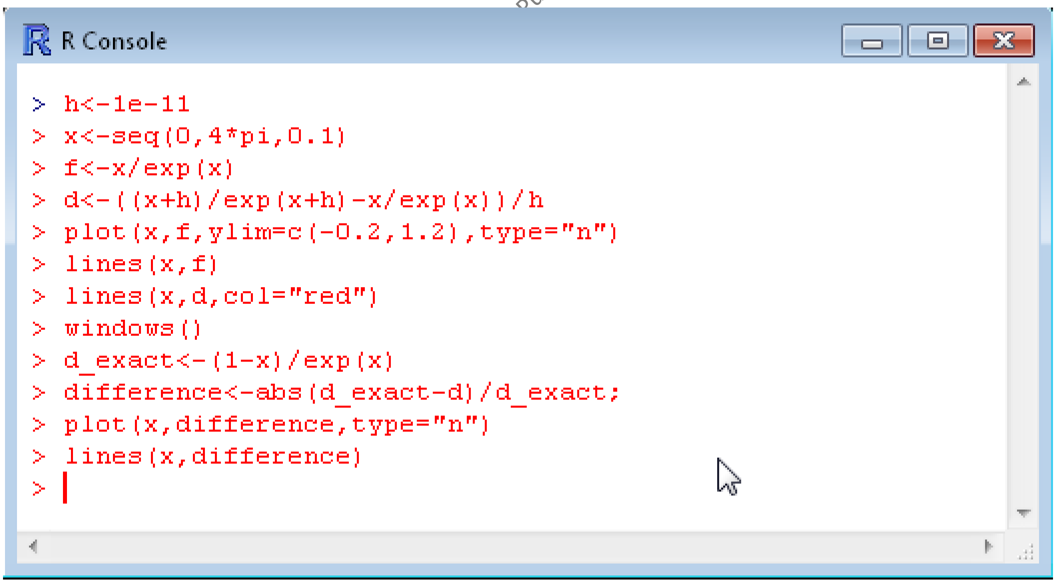
Exemple 29.: Dérivation algébrique/Dérivation numérique

R 3.0.2

R a nativement une commande pour dériver. Voici un exemple de dérivation avec l'estimation de la dérivée en un point:



Ensuite, pour la dérivée numérique nous allons reprendre l'exemple basique fait dans le cours MATLAB et comparer la différence... D'abord avec un pas relativement standard:



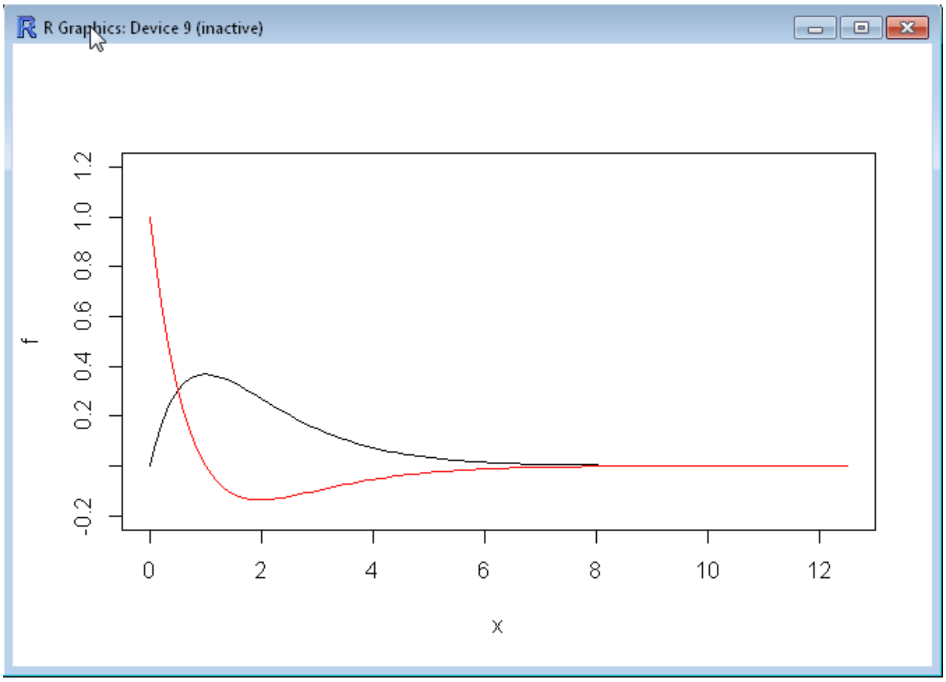
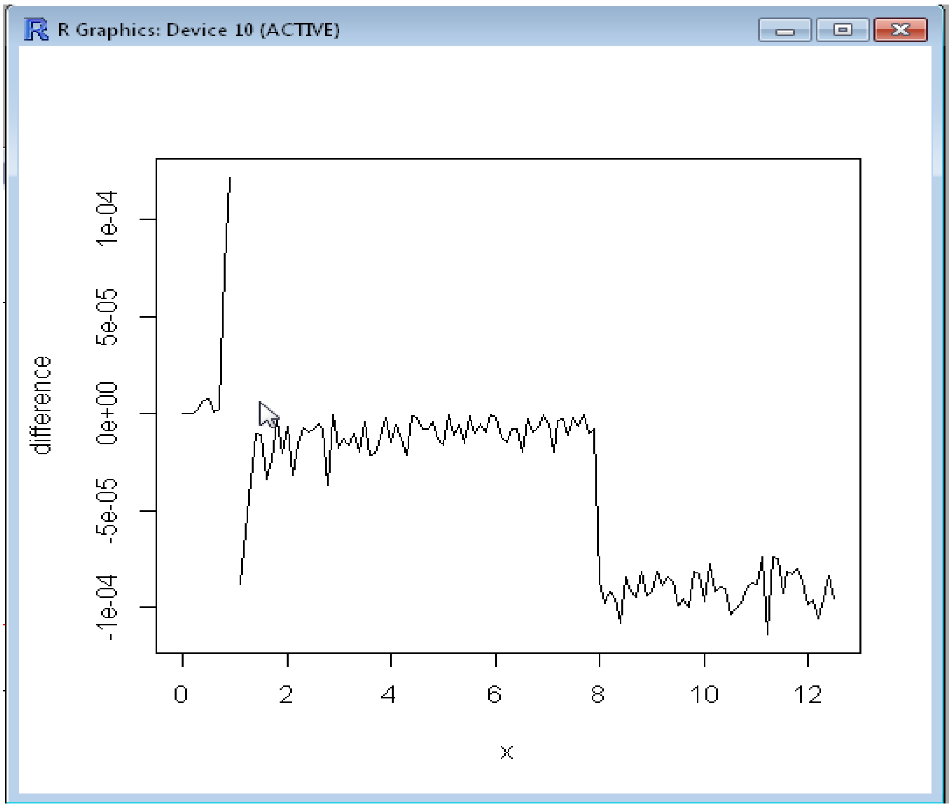
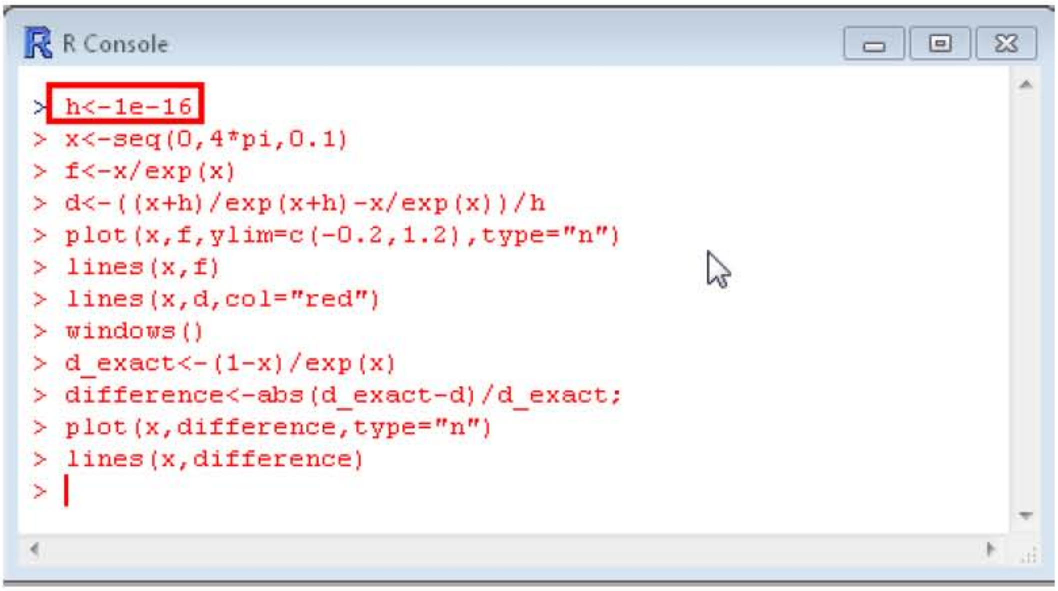
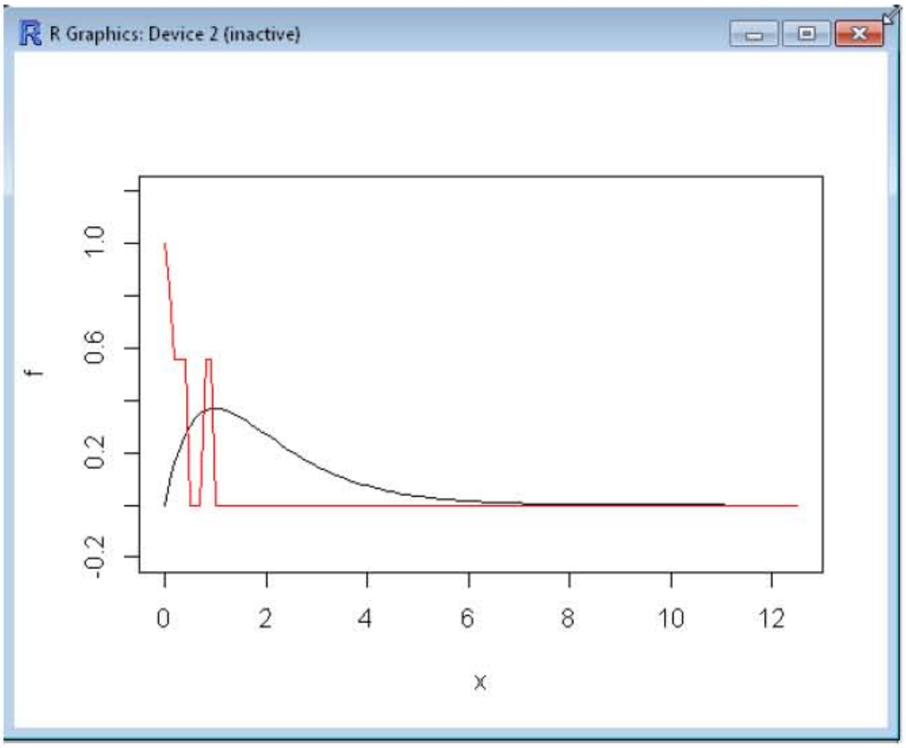


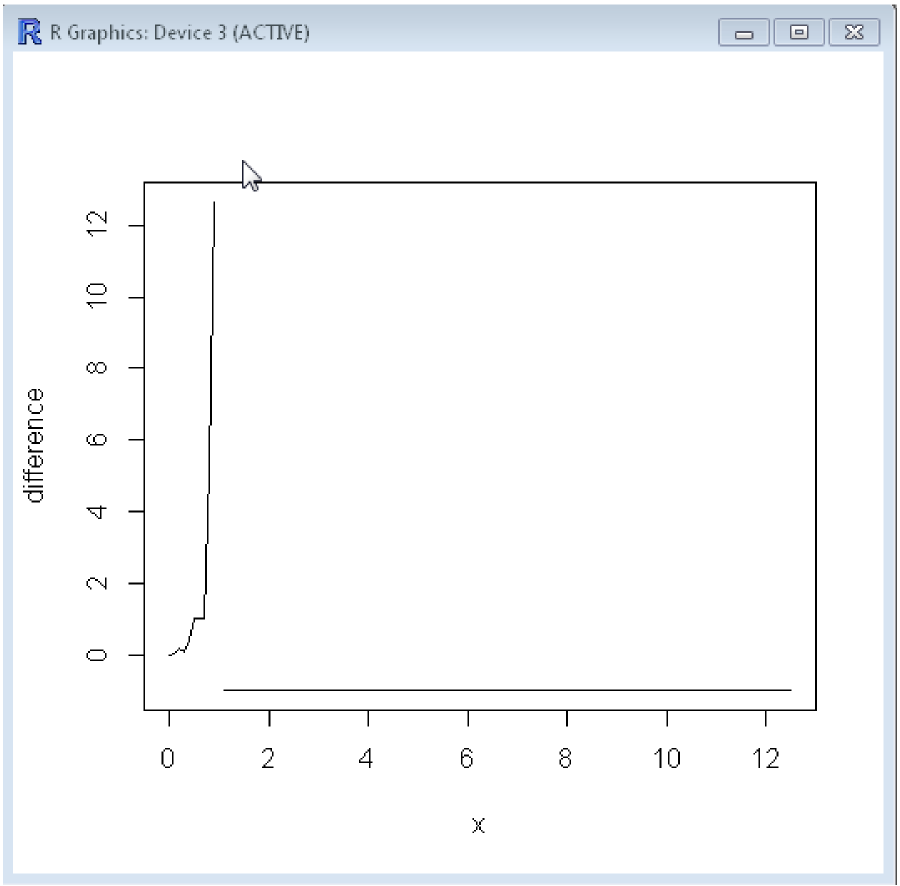
Figure 16 Graph d'une fonction f(x) et sa dérivée



Et ensuite avec la même petite valeur problématique:


R Console
> h<-1e-16
> x<-seq(0,4*pi,0.1)
> f<-x/exp(x)
> d<-((x+h)/exp(x+h)-x/exp(x))/h
> plot(x,f,ylim=c(-0.2,1.2),type="n")
> lines(x,f)
> lines(x,d,col="red")
> windows()
> d_exact<-(1-x)/exp(x)
> difference<-abs(d_exact-d)/d_exact;
> plot(x,difference,type="n")
> lines(x,difference)
>


R Graphics: Device 2 (inactive)
f
1.0
0.6
0.2
-0.2
0
2
4
6
8
10
12
x



Nous voyons donc apparaître le même problème de précision que dans MATLAB. Si nous tâtonnons un peu nous verrons que le meilleur pas est de l'ordre de 1E-8 ce qui amène à une erreur relative de l'ordre de 1E-7.

Voyons cependant ce que nous pouvons faire avec la package **numDeriv** de R et la commande **grad( )** en commençant d'abord par réobtenir le premier résultat ci-dessus en utilisant la méthode *simple*:

R Console

```
> library(numDeriv)
> x<-seq(0,4*pi,0.1)
> func1 <- function(x){x/exp(x)}
> f<-x/exp(x)
> numd1 <- grad(func1, x,method="simple",method.args=list(eps=1e-11))
> exact <- (1-x)/exp(x)
> plot(x,f,type='n',ylim=c(-0.2,1.2))
> lines(x,f)
> lines(x,numd1,col="red")
> windows()
> plot(x, (numd1 - exact)/exact)
> lines(x, (numd1 - exact)/exact,col="green")
>
```

Ce qui donne (nous retrouvons donc bien la même chose):

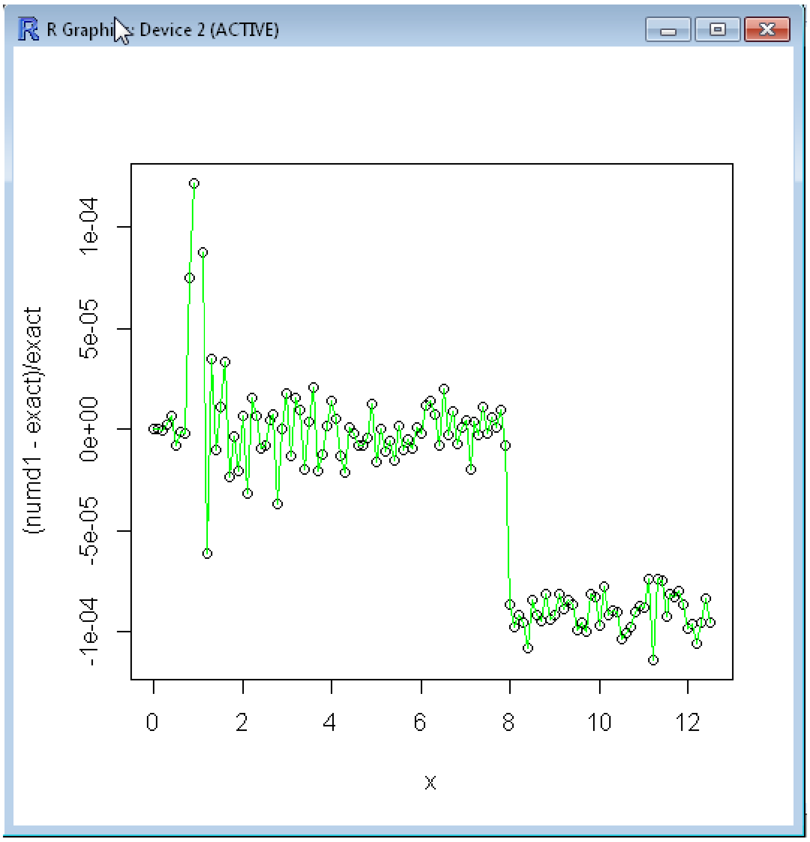


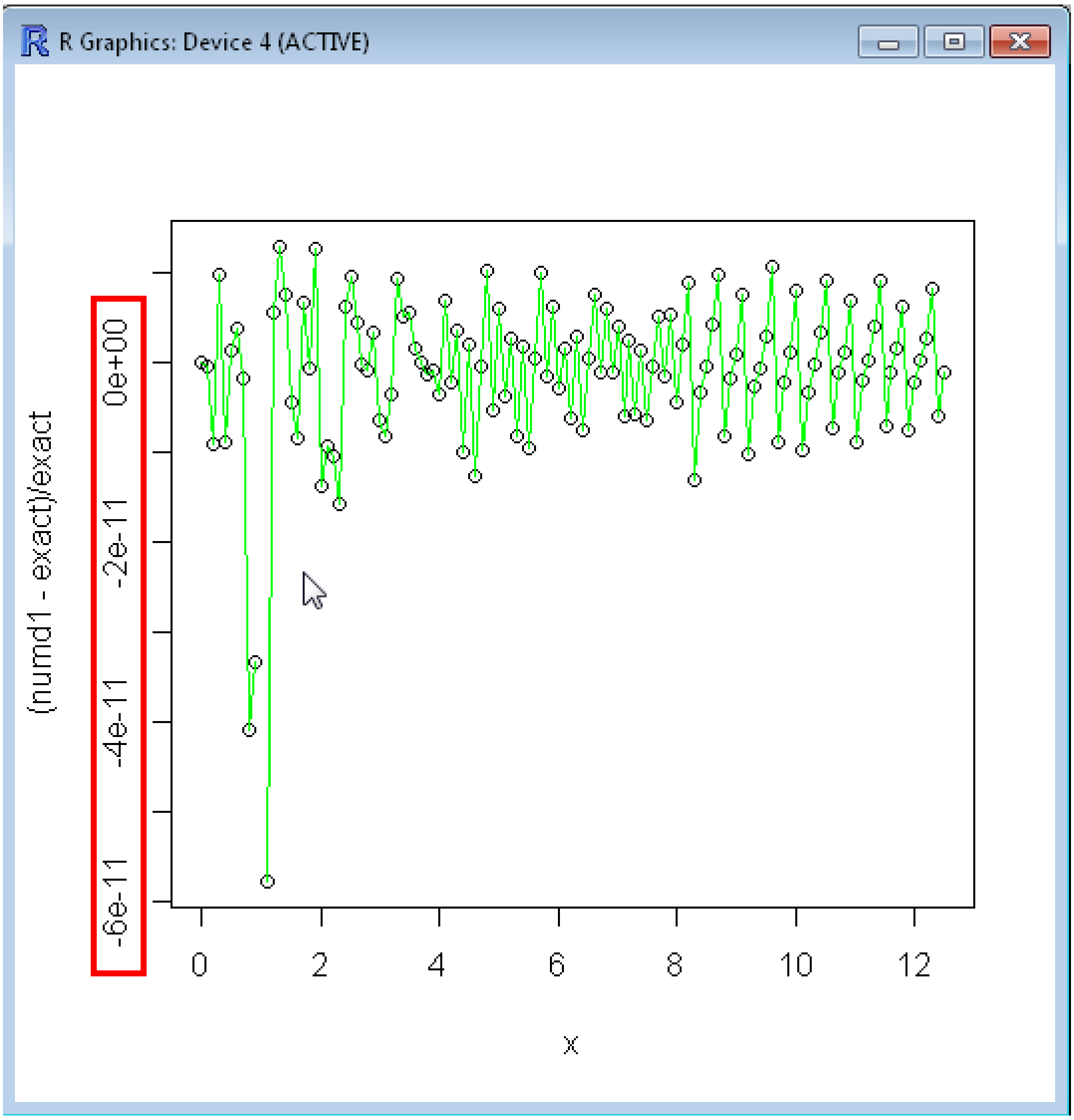
Figure 17 Graph d'une fonction f(x) avec ses point estimés

et nous aurons le même problème qu'avant si nous prenons le pas comme valant 1E-16.

Par contre, laissons ce package choisir sa méthode par défaut:

```
R Console
> library(numDeriv)
> x<-seq(0,4*pi,0.1)
> func1 <- function(x){x/exp(x)}
> f<-x/exp(x)
> numd1 <- grad(func1,x)
> exact <- (1-x)/exp(x)
> plot(x,f,type='n',ylim=c(-0.2,1.2))
> lines(x,f)
> lines(x,numd1,col="red")
> windows()
> plot(x, (numd1 - exact)/exact)
> lines(x, (numd1 - exact)/exact,col="green")
>
```

et là nous avons un résultat remarquable d'amélioration d'environ 8 ordres de grandeur!:



Qui trouvera mieux?

Exemple 30.: Intégration numérique

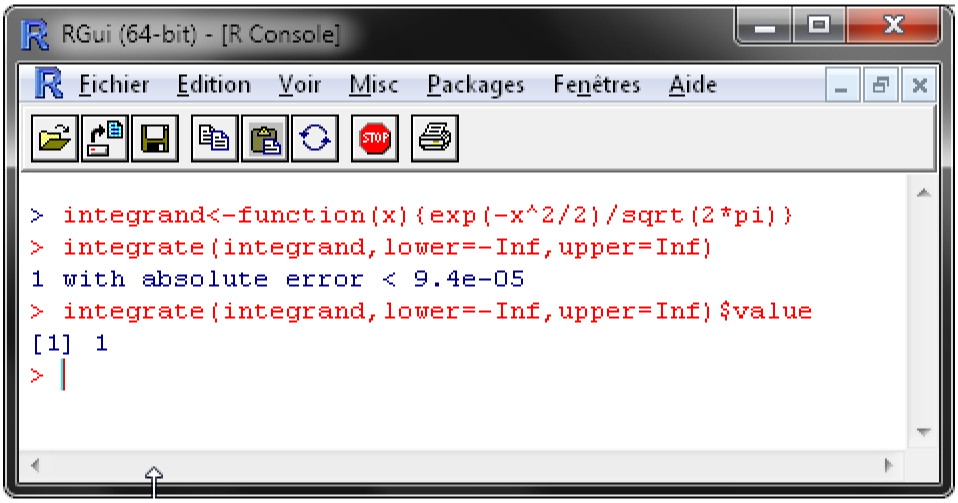
R 3.0.2

Pour l'intégration numérique (il existe un package – **Ryacas** - pour l'intégration symbolique mais il nécessite une connexion internet constante et de plus le serveur correspondant n'est pas toujours en ligne).

Nous verrons d'autres applications de l'intégration dans R avec la convolution dans le chapitre de traitement du signal.

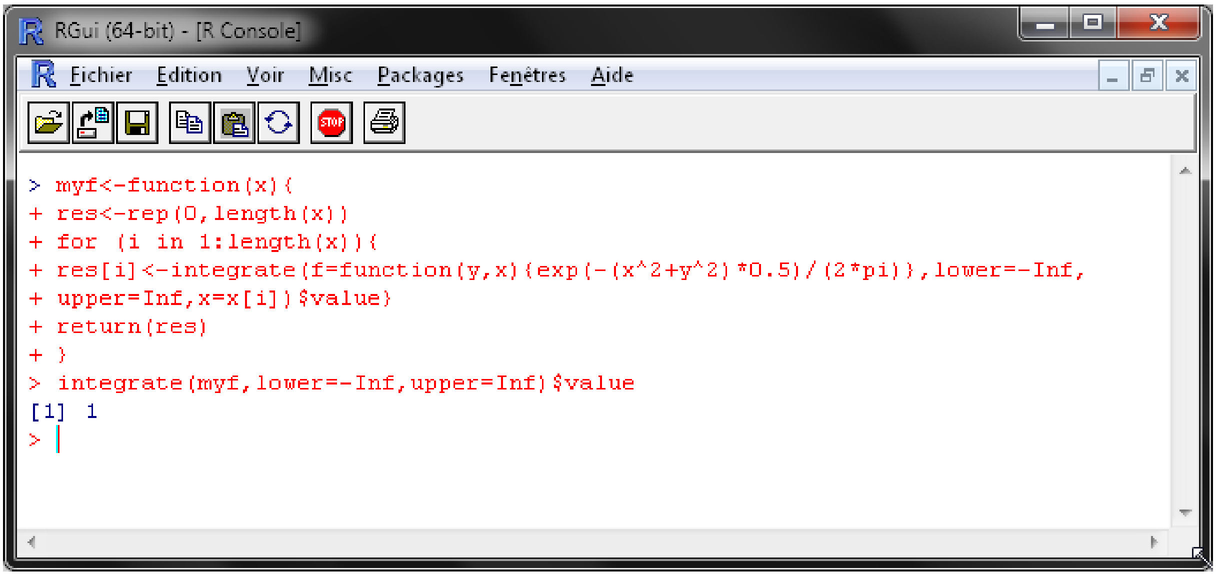
Intégrale simple

Un petit exemple avec la loi Normale centrée réduite univariée utilisant la fonction **integrate( )** :



Double Intégrale

Pour une double intégrale la syntaxe est un peu subtile dans R. Voici un petit exemple pour la loi Normale centrée réduite bivariée:



Intégrale avec somme

Et pour une intégrale avec une somme finie dans l'intégrande comme par exemple l'espérance d'un kernel smoothing avec noyau gaussien tronqué comme démontré et étudié dans le cours théorique:

$$E(S)=\int_{0}^{+\infty}\sum_{i=1}^{n}\frac{s}{nh\sqrt{2x}}e^{-\frac{1}{2}\left(\frac{s-X_i}{h}\right)^2}ds$$

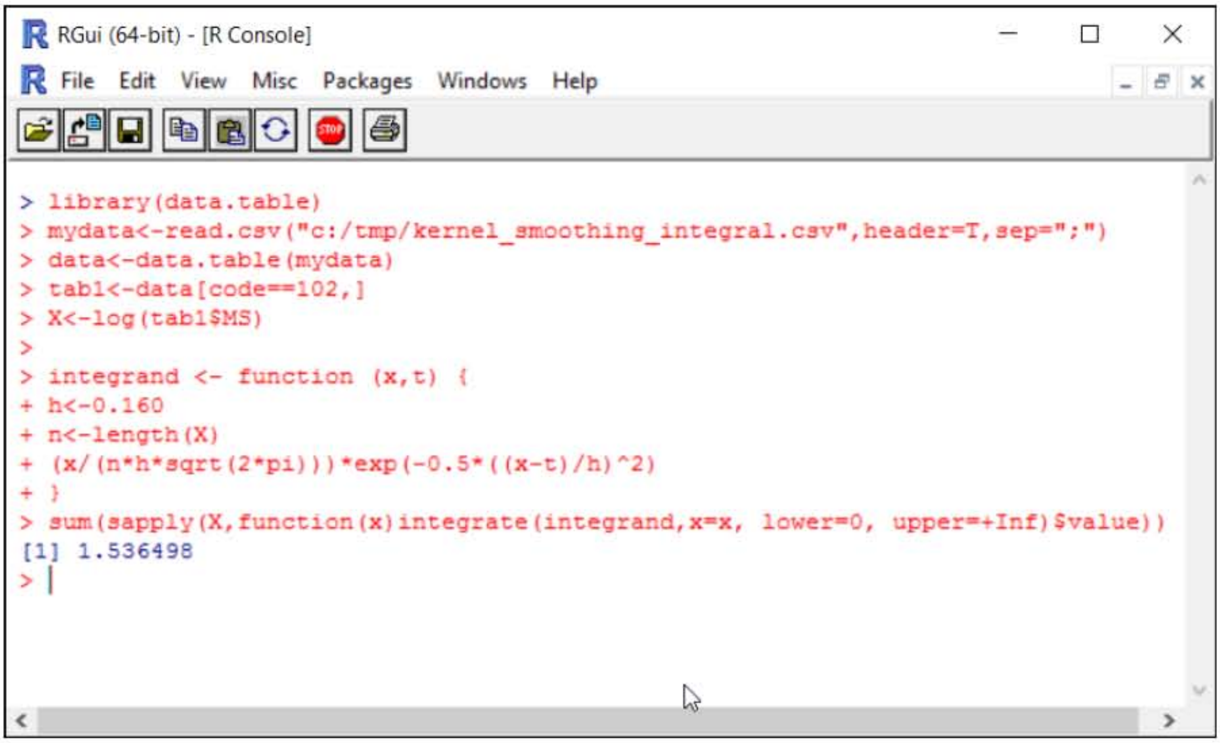
en utilisant et normalisant les données ci-dessous:

kernel_smoothing_integral.csv - Notepad

File Edit Format View Help

```
code;MP;MS
00102;1495.18;3840
00102;1495.18;16330
00102;1712;113208.67
00102;2990.36;8800
00102;1495.18;20217.29
00102;1495.18;341414.79
00102;1495.18;10837.77
00102;4485.54;6081.59
00102;4485.54;12530.1
00102;1121.39;63471.32
00102;2242.77;15503.41
00102;1495.18;22017.54
00102;5801.73;69798.76
00102;1495.18;8030
00102;1495.18;17376.74
00102;4485.54;46842.91
00102;1495.18;11194.63
00102;1495.18;3226.45
00102;1495.18;14517.15
00102;1495.18;4655
00102;1495.18;16034.81
00102;1495.18;10551.89
00102;1495.18;578385.37
00102;971.87;6230.25
00102;971.87;3000
00102;1495.18;3620
00102;2242.77;8925
00102;1495.18;1140
00102;1495.18;32273.46
```

Le script correspondant est alors:

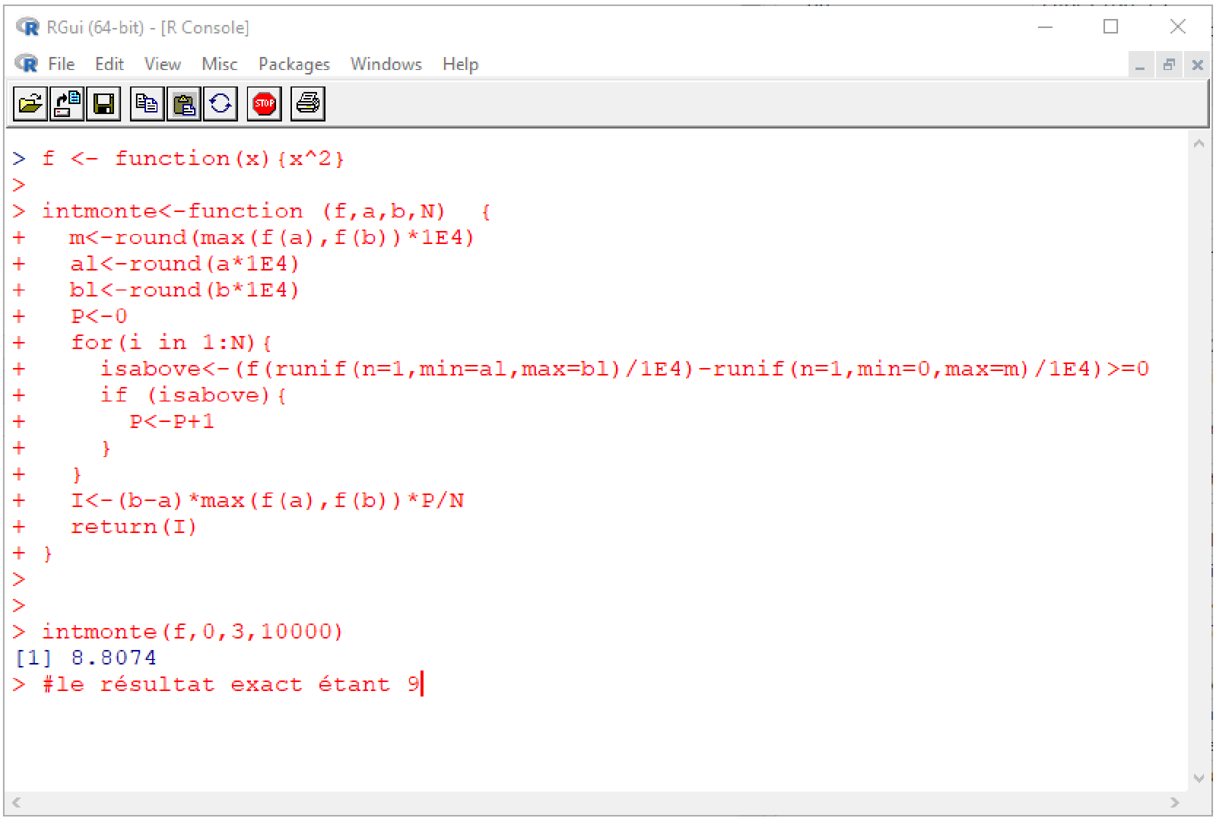

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(data.table)
> mydata<-read.csv("c:/tmp/kernel_smoothing_integral.csv",header=T,sep=";")
> data<-data.table(mydata)
> tab1<-data[code==102,]
> X<-log(tab1$MS)
>
> integrand <- function (x,t) {
+ h<-0.160
+ n<-length(X)
+ (x/(n*h*sqrt(2*pi)))*exp(-0.5*((x-t)/h)^2)
+ }
> sum(sapply(X,function(x)integrate(integrand,x=x, lower=0, upper=+Inf)$value))
[1] 1.536498
>

Intégrales par Monte Carlo

Remarque: Le lecteur doit se rappeler d'utiliser la fonctio **set.seed( )** s'il veut reproduire les résultats à chaque exécution ou même entre différentes machines!!

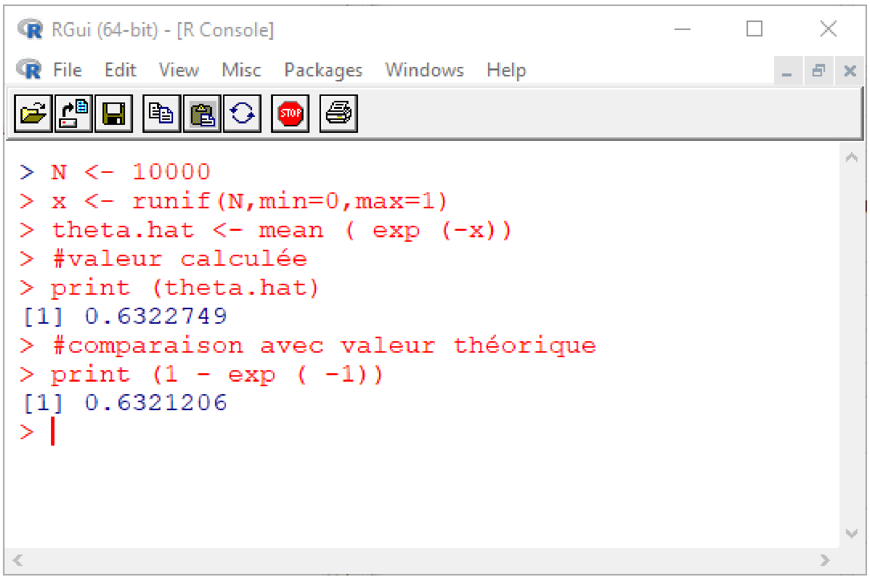
Nous allons reproduire ici les trois exemple vus dans le cours théorique.

D'abord le premier cas simple utilisant la méthode du "touché ou loupé":



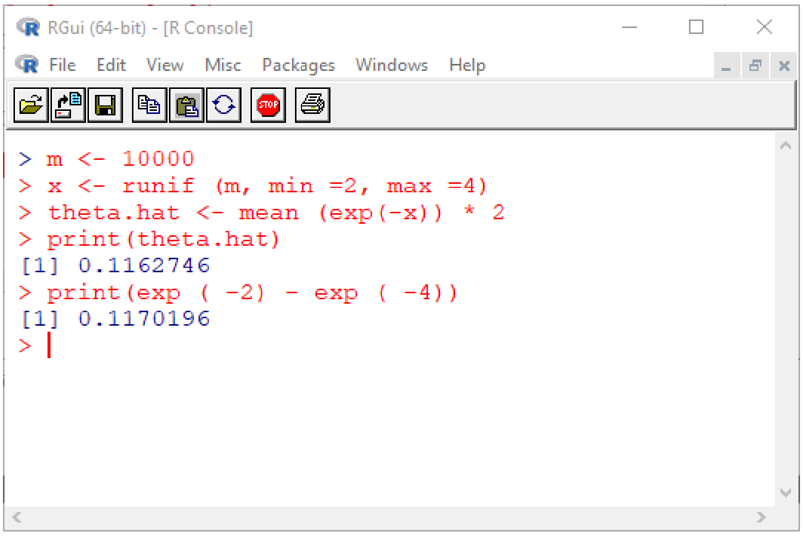
Et ensuite deuxième cas utilisant la méthode de la valeur moyenne pour le calcul de:

$$\theta = \int_0^1 e^{-x} dx$$



et pour calculer:

$$\theta = \int_2^4 e^{-x} dx$$



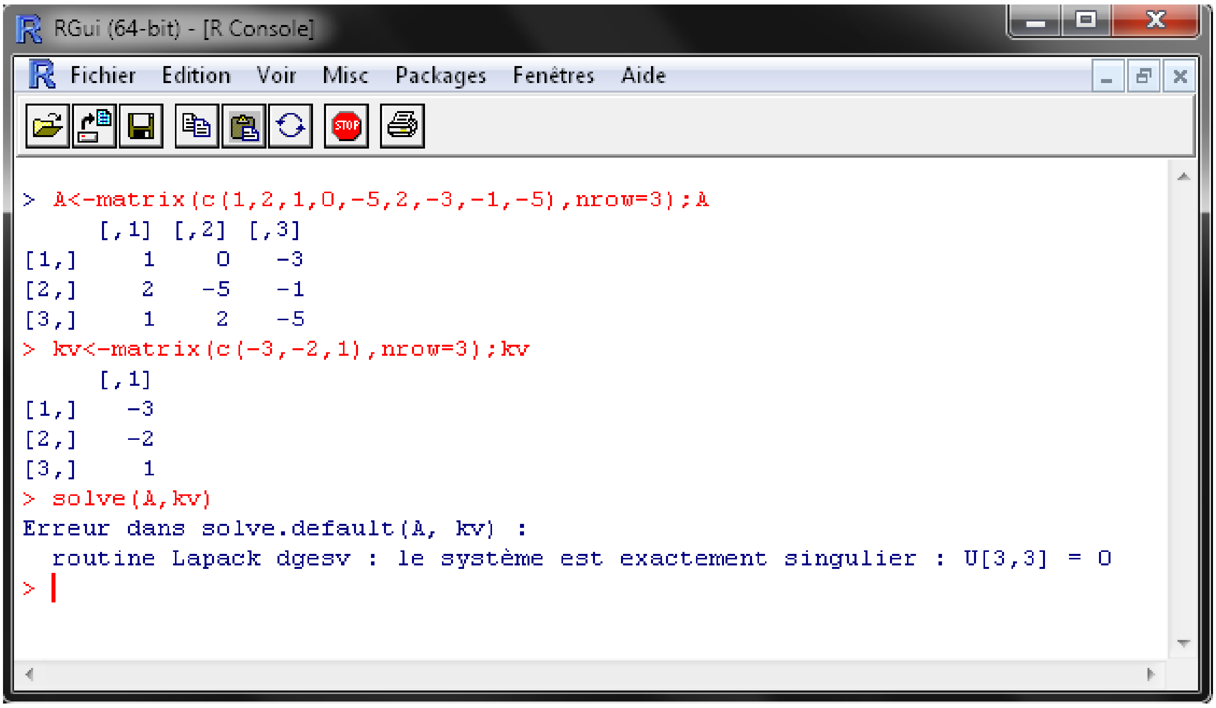
Mais il existe un package qui implémente toutes les méthodes vues dans le cours théorique. Ce package est nommé **SI** (stochastic integration) et comparons toutes les méthodes (quatre) vue dans le cours théorique:

```
> library("SI")
>
> set.seed(0)
> h <- function(x){
+     exp(x)
+ }
> N <- 100000
>
> #Méthode du touché ou loupé
> #Intégration de exp(x) de -1 à 1
> SPMresult <- SI.SPM(h,-1,1,exp(1),N)
> (I1 <- SPMresult[[1]])
[1] 2.334569
> (VarI1 <- SPMresult[[2]])
[1] 7.241821e-05
>
> #Méthode de la valeur moyenne
> #Intégration de exp(x) de -1 à 1
> N <- 100000
> MVMresult <- SI.MVM(h,-1,1,N)
> (I2 <- MVMresult[[1]])
[1] 2.346854
> (VarI2 <- MVMresult[[2]])
[1] 1.723418e-05
>
> #Méthode de l'échantillonnage préférentiel (importance sampling)
> #Intégration de exp(x) de -1 à 1
> #En utilisant la fonction complémentaire (3/2+x)/3
> g <- function(x){return((3/2+x)/3)}
> G_inv <- function(y){return(sqrt(6*y+1/4)-3/2)}
> ISMresult <- SI.ISM(h,g,G_inv,N)
> (I3 <- ISMresult[[1]])
[1] 2.349192
> (VarI3 <- ISMresult[[2]])
[1] 1.937335e-06
>
>
> #Méthode de l'échantillonnage stratifié
> #Intégration de exp(x) de -1 à 1
> N <- 100000
> SSMresult <- SI.SSM(h,-1,1,10,N)
     to
[1,]  1 1.2 1.4 1.6 1.8 2 2.2 2.4 2.6 2.8 3
> (I4 <- SSMresult[[1]])
[1] 17.36745
> (VarI4 <- SSMresult[[2]])
[1] 1.311457e-05
>
```

Exemple 31.: Résoudre un système linéaire

R 3.0.2

Nous allons voir ici comment résoudre le petit problème d'algèbre linéaire que nous avons fait dans le cours théorique sur les systèmes linéaires. Il s'agit simplement d'appliquer les commandes vues plus haut lors de notre découverte des variables matricielles.



Donc nous retrouvons la même concluions que dans le cours théorique: ce système n'a pas de solutions.

Exemple 32.: Générer une variable (pseudo-)aléatoire uniforme et reproductible

R 3.0.2

Nous allons voir par la suite comment gérer certaines fonctions de distributions classiques dans les petites classes et démontrées dans le cours théorique. Cependant, avant cela, il convient d'abord de voir une commande particulière que l'on écrit **set.seed( )** .

Considérons donc d'abord la génération suivante d'une variable aléatoire uniforme comprise entre 0 et 1:

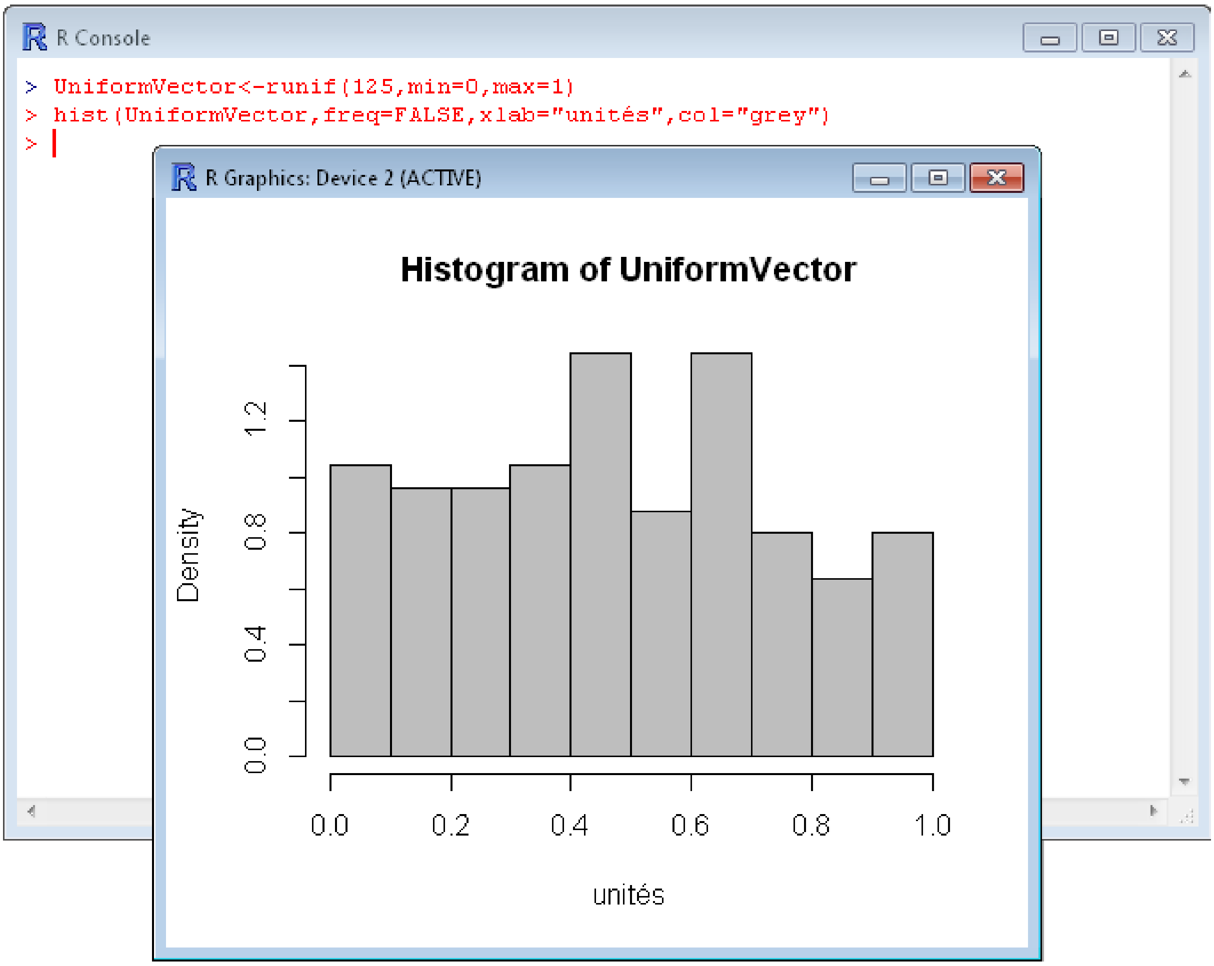
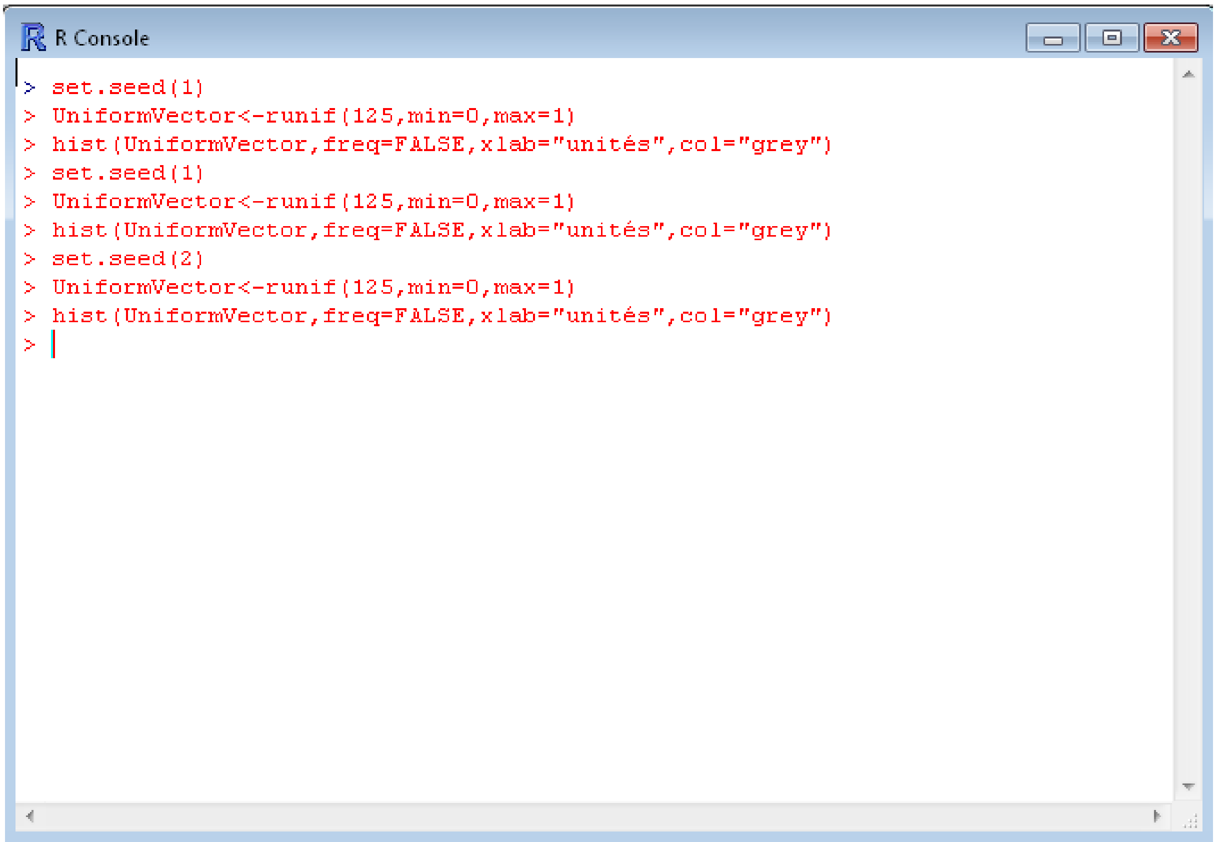


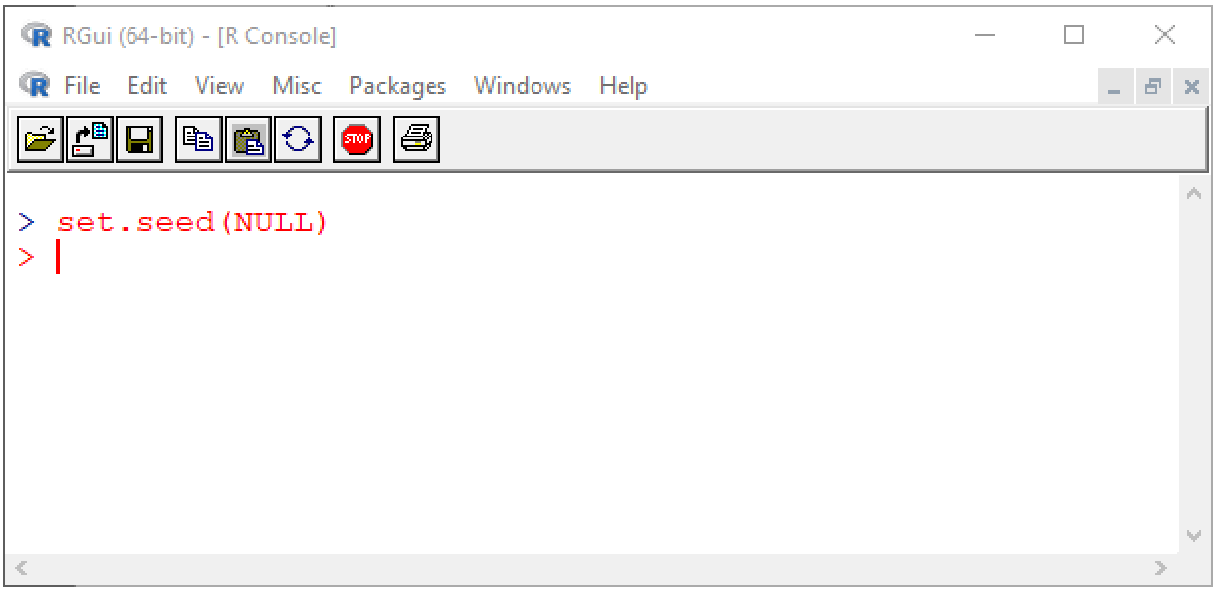
Figure 18 Graph de type Histogramme

À chaque fois que vous réexécuterez ces deux commandes, vous n'obtiendrez pas le même graphique. Par contre, si vous utilisez **set.seed( )** avec un identifiant, alors dans ce cas le résultat peut être reproduit:



Ainsi, les deux premiers graphs sont identiques par contre le troisième est différent car il utilise un autre vecteur aléatoire.

Pour annuler le seed, on écrira:



Optimisation (recherche opérationnelle)

Nous allons parcourir ici quelques techniques d'optimisation quadratique dont nous avons étudié les détails mathématiques dans le cours théorique. L'idée étant de monter graduellement en puissance (complexité mathématique sous-jacente).

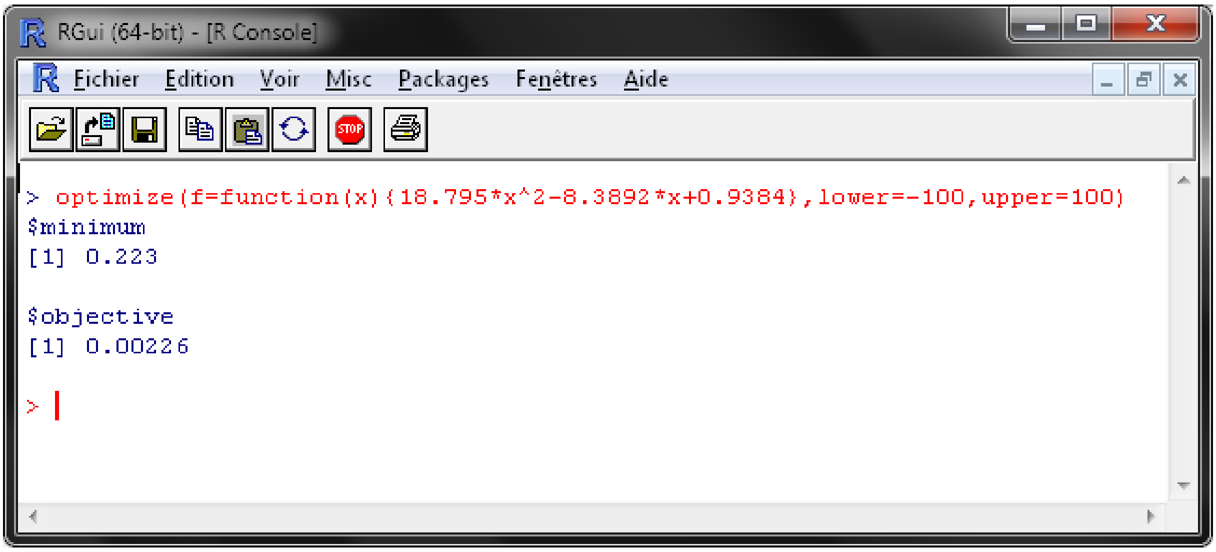
Nous verrons lors de notre étude des modèles de régression, comment utiliser ce type de techniques pour faire des régressions contraintes (voir plus loin!).

Exemple 1.: Optimisation unidimensionnelle (contrainte)

R 3.0.2

Nous allons voir ici encore un cas trivial de manipulation algébrique. Il s'agit de trouver l'optimisation d'une fonction univariée en utilisant la commande **optimize( )** dont la spécification d'un intervalle de recherche est obligatoire.

Prenons le cas de la parabole que nous avions obtenu lors de notre étude théorique du modèle de Markowitz (évidemment c'est un cas trivial mais cela marche aussi pour des fonctions univariées beaucoup plus complexes):



Donc le minimum se trouve en:

$$(x, y) = (0.223, 0.00226)$$

Exemple 2.: Optimisation multidimensionnelle (sans contrainte et local)

R 3.0.2

Utilisons la fonction baleine à bosse que nous avions utilisée dans le cours théorique lors de notre démonstration de la méthode d'optimisation de Newton-Raphson:

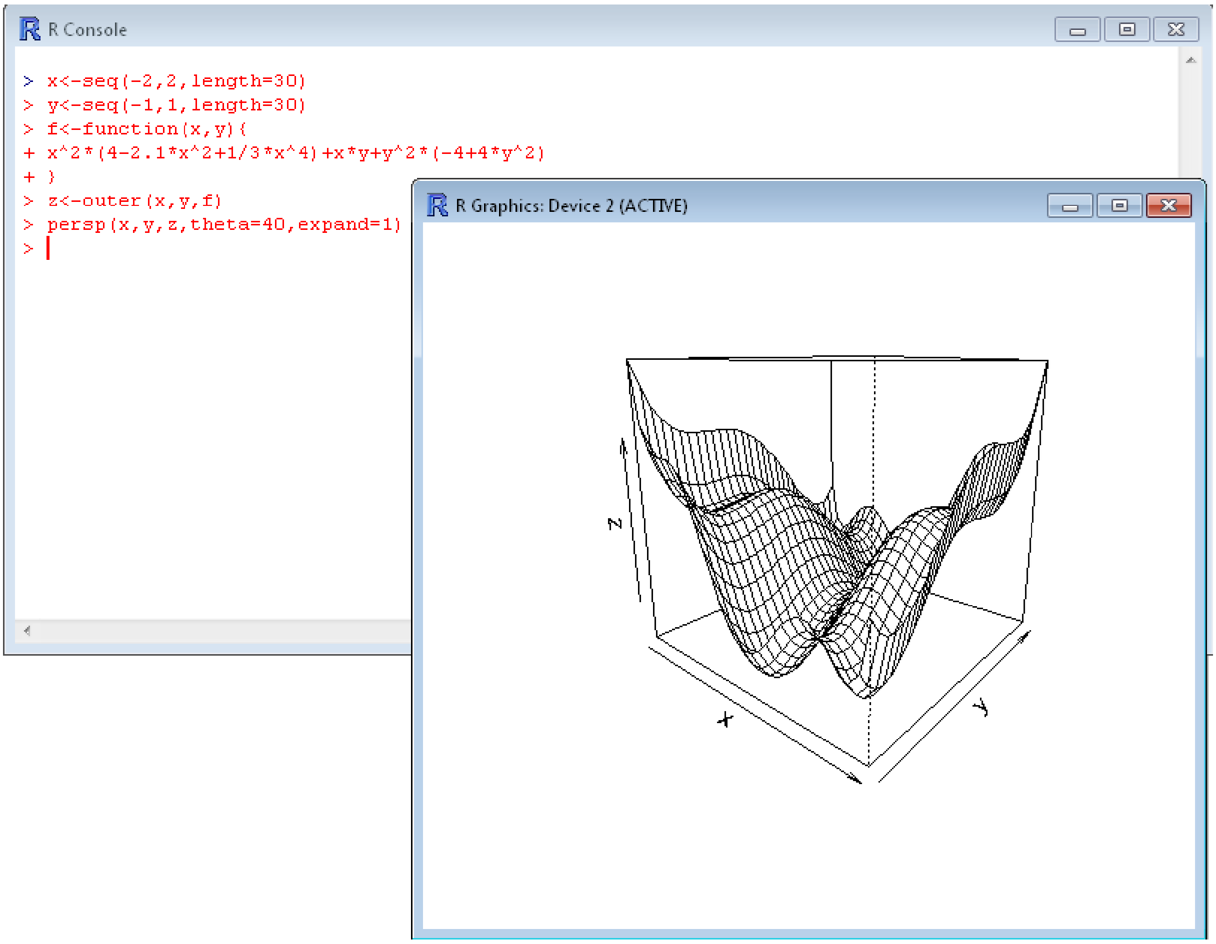
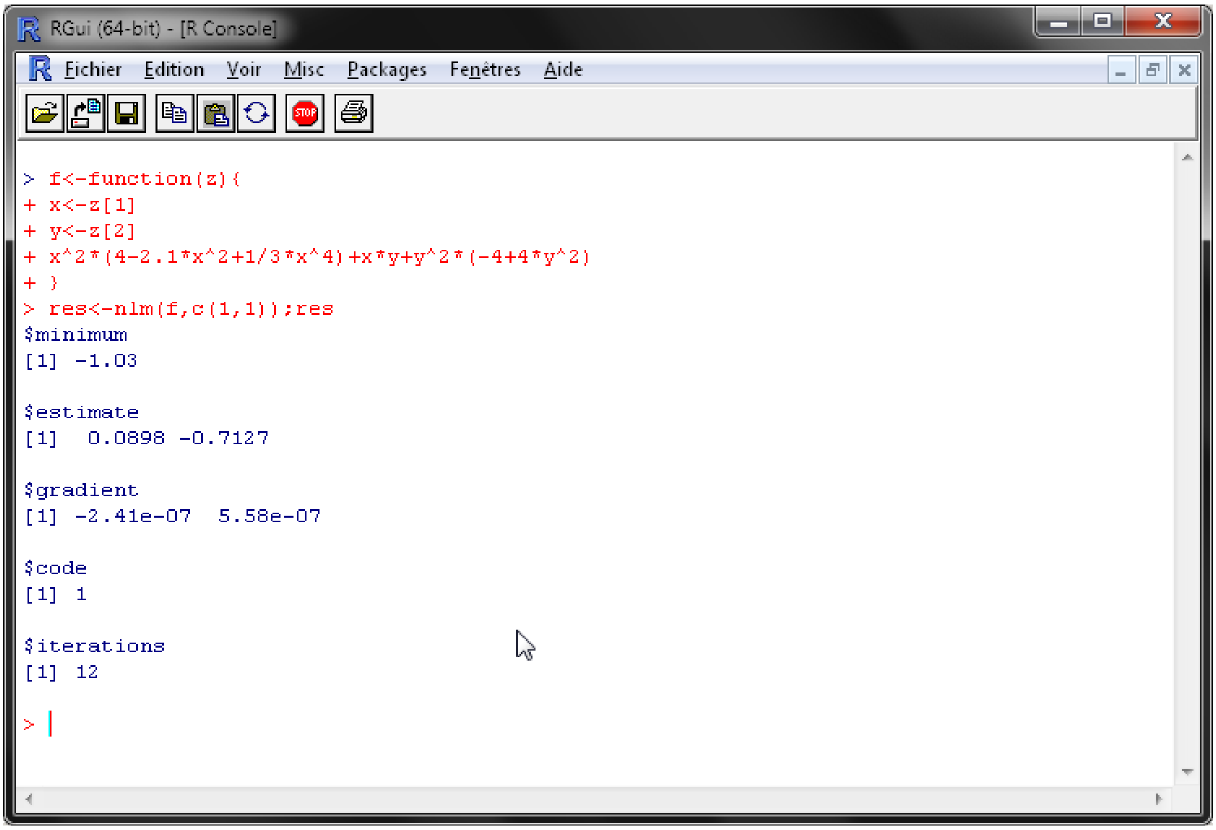


Figure 19 Graph 3D de fonction f(x,y) filaire

Nous avons alors en utilisant la commande **nlm( )** qui signifie **n**ewton **l**ocal **m**inimum:



Ce qui est effectivement conforme à ce que nous avions vu dans le cours théorique comme le rappelle ce plot en courbes de niveau ci-dessous:

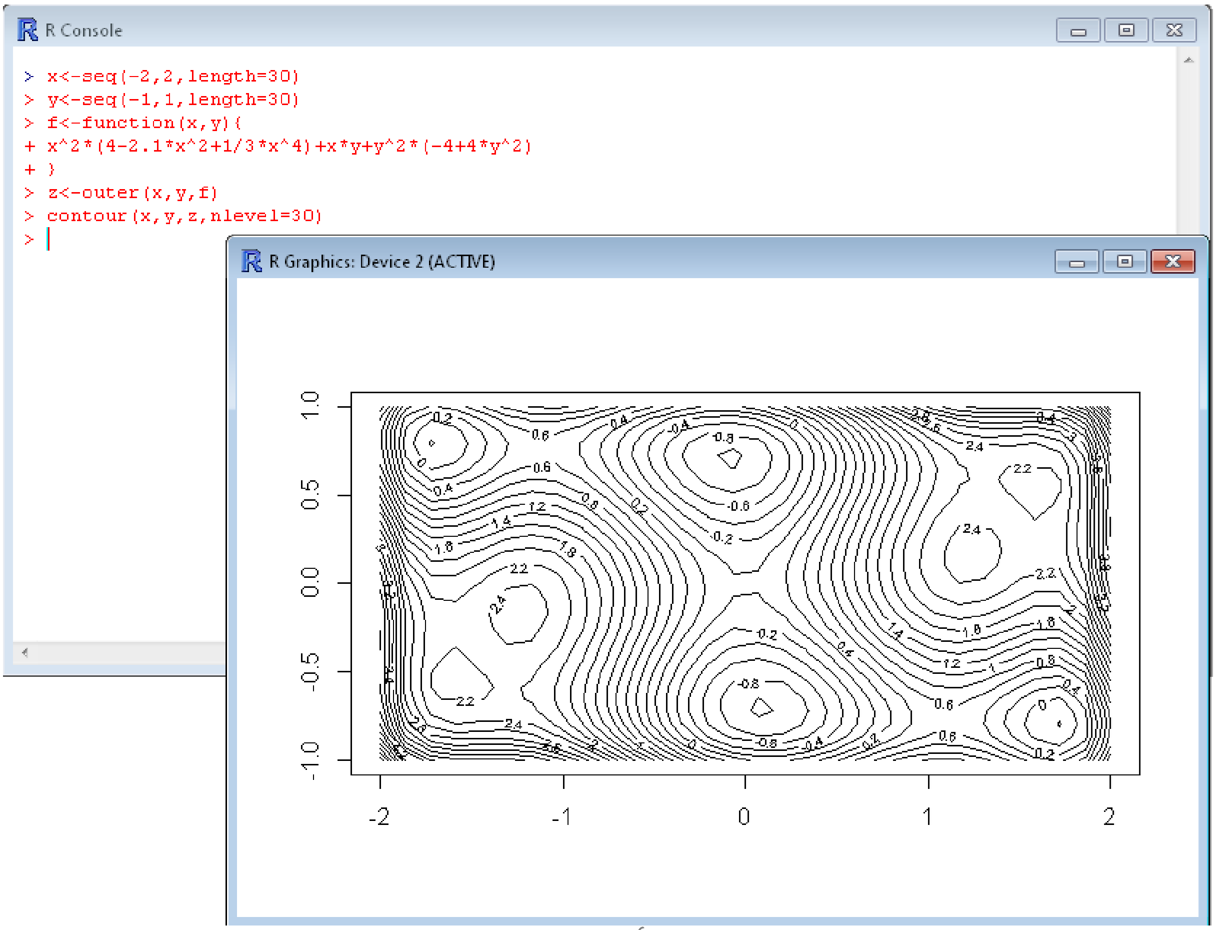
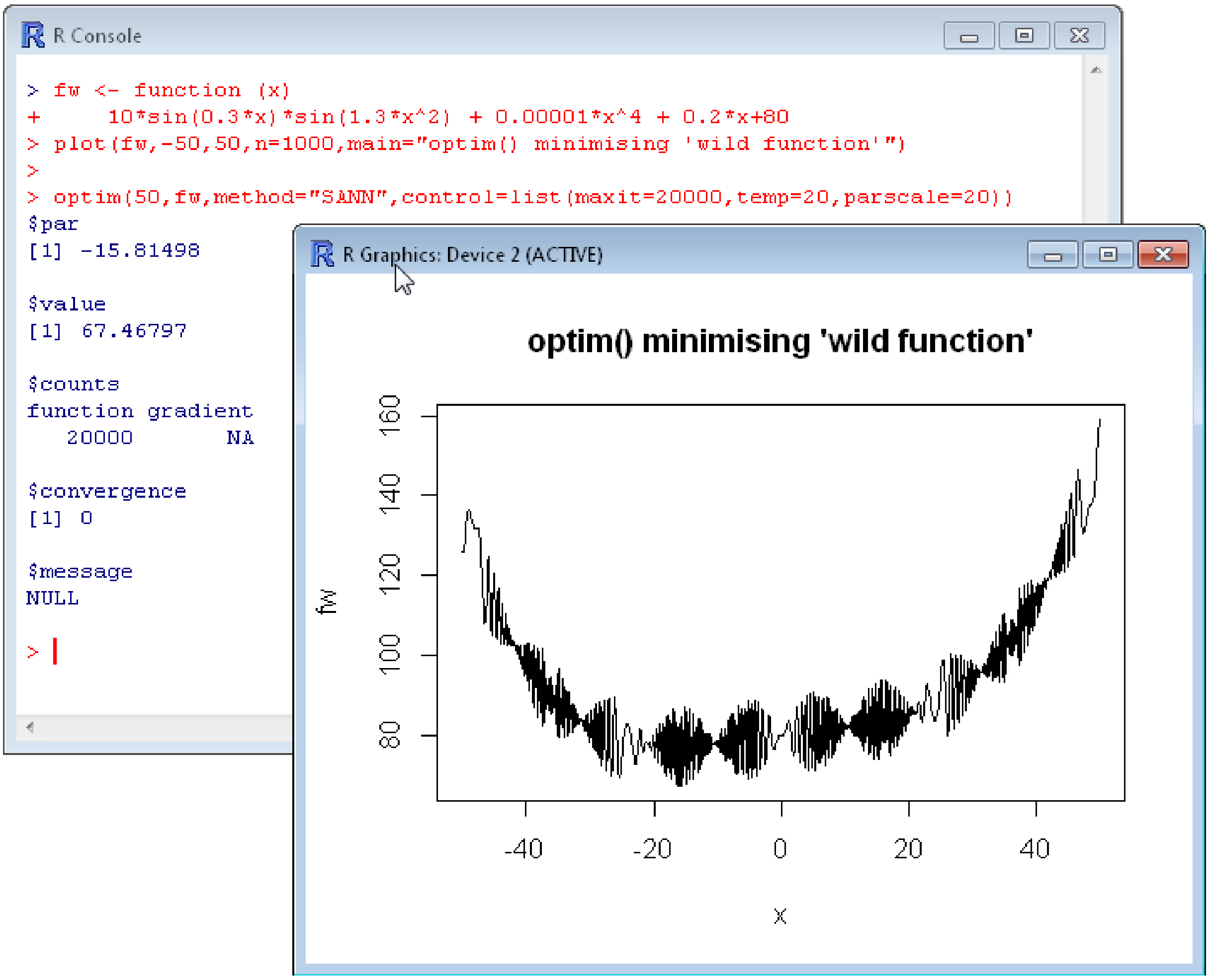


Figure 20 Graph de contour (isoclines) 2D de fonction 3D f(x,y)

Exemple 3.: Optimisation multidimensionnelle (sans contrainte et global)

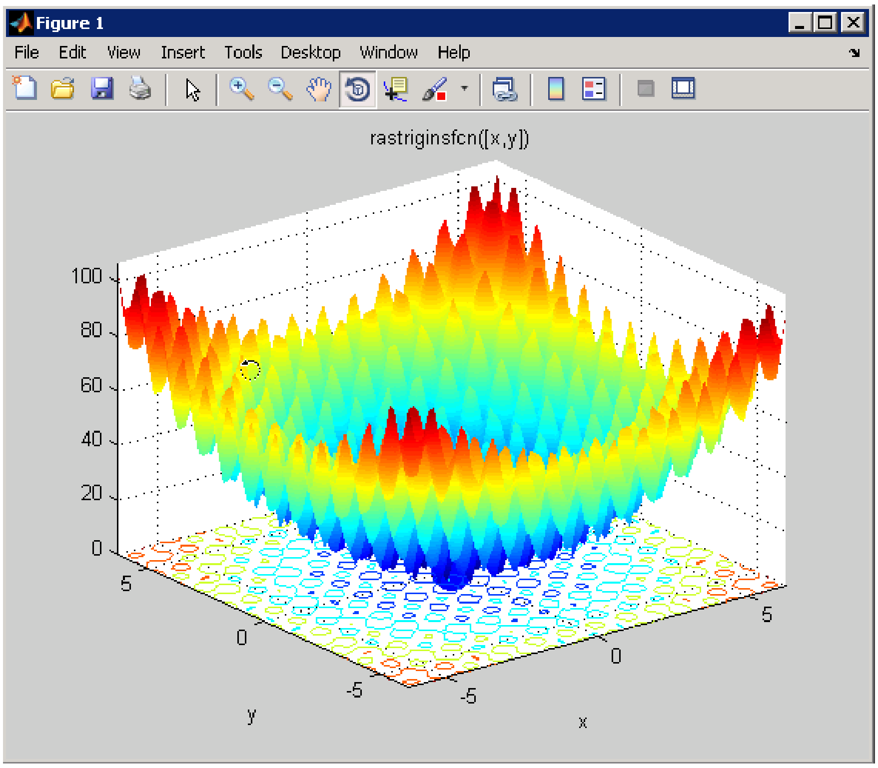
R 3.1.1

Nous avons parlé dans la formation des très nombreuses techniques empiriques qui existent pour chercher des optimums et nous avons démontré deux techniques typiques. Nous allons voir ici ce que propose R nativement avec la fonction **optim( )** :



Nous voyons qu'il trouve bien le minimum global en utilisant la méthode.

Voyons maintenant avec les algorithmes génétiques du package **GA**. Comme pour le cours MATLAB, nous allons chercher le minimum de la fonction Rastrigin qui visuellement donnait dans MATLAB:



Donc d'abord voyons ce qu'elle donne dans le logiciel R avec **persp3D( )** :

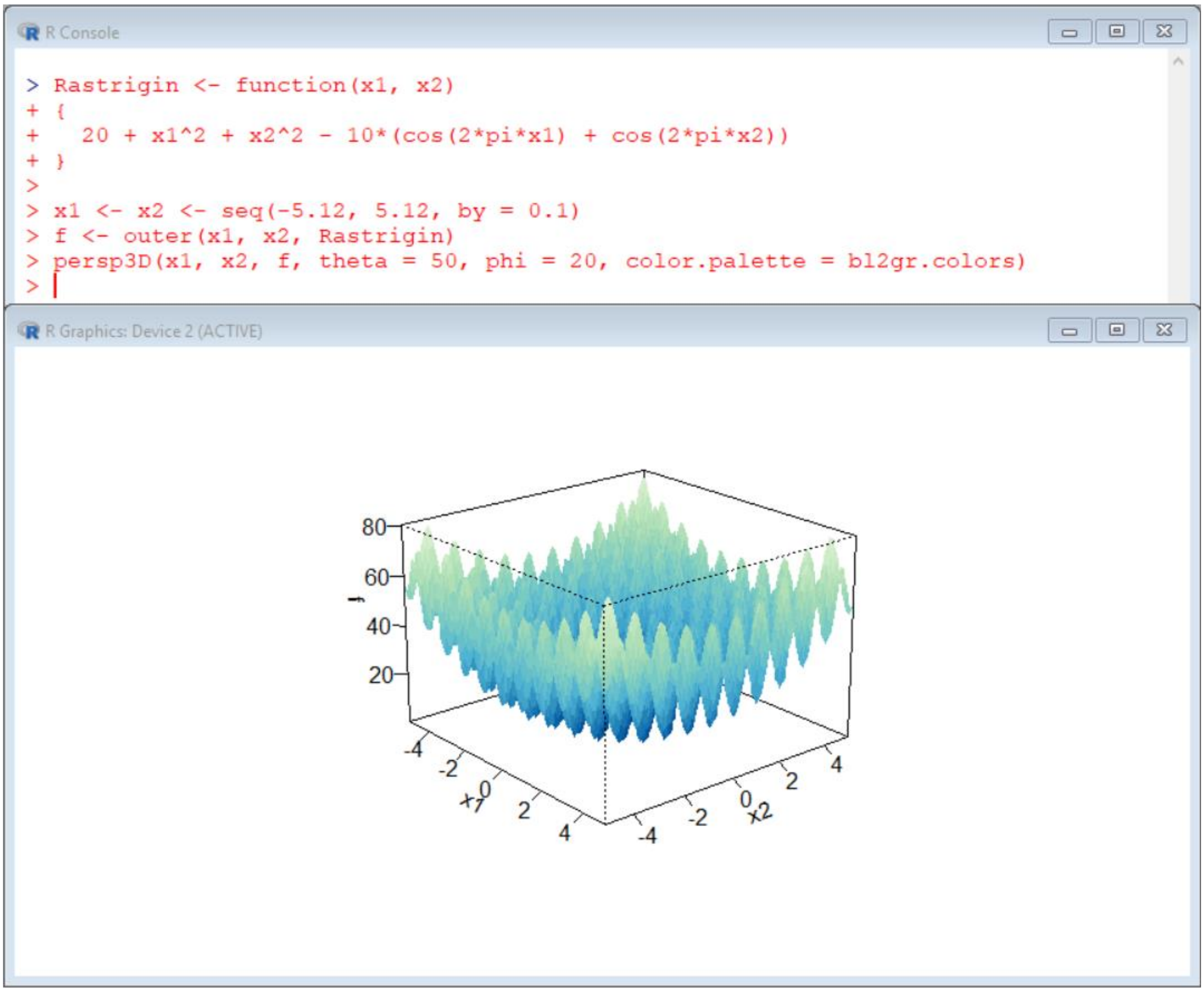


Figure 21 Graph de surface de fonction 3Df(x,y) avec couleurs

ou:

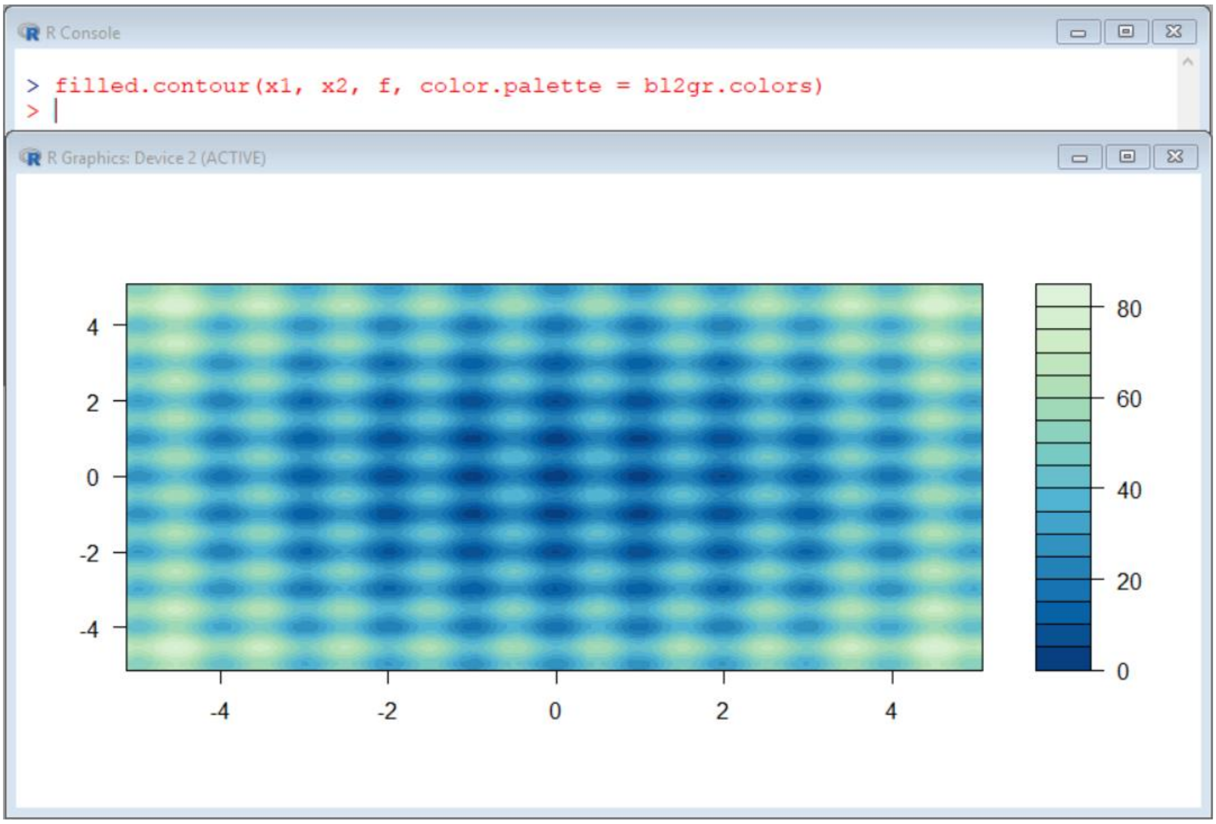
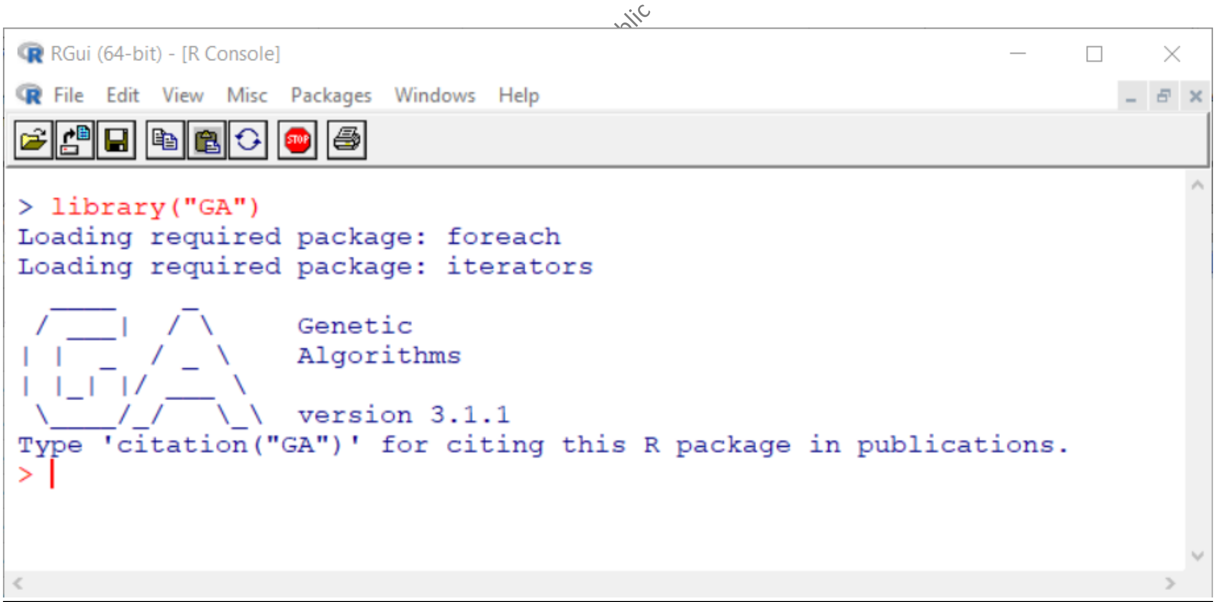


Figure 22 Graph de projection d'amplitude de fonction 3D f(x,y)

Maintenant, nous chargeons le package **GA**:



Ensuite, nous lançons l'optimisation avec la fonction **ga( )** :

```
> GA <- ga(type = "real-valued",
+          fitness =  function(x) -Rastrigin(x[1], x[2]),
+          lower = c(-5.12, -5.12), upper = c(5.12, 5.12),
+          popSize = 50, maxiter = 1000, run = 100)
GA | iter = 1 | Mean = -36.161573 | Best =  -7.686431
GA | iter = 2 | Mean = -27.948362 | Best =  -6.002614
GA | iter = 3 | Mean = -22.316413 | Best =  -2.888756
GA | iter = 4 | Mean = -20.6794735 | Best =  -0.5653031
GA | iter = 5 | Mean = -18.8994686 | Best =  -0.1055646
GA | iter = 6 | Mean = -21.5660694 | Best =  -0.1055646
GA | iter = 7 | Mean = -19.1140857 | Best =  -0.0774759
GA | iter = 8 | Mean = -18.4315878 | Best =  -0.0774759
GA | iter = 9 | Mean = -18.1302241 | Best =  -0.0774759
GA | iter = 10 | Mean = -13.0542181 | Best =  -0.0774759
GA | iter = 11 | Mean = -10.781931360 | Best =  -0.003010902
GA | iter = 12 | Mean = -8.755395489 | Best = -0.003010902
GA | iter = 13 | Mean = -8.301851794 | Best = -0.003010902
GA | iter = 14 | Mean = -6.903129155 | Best = -0.003010902
GA | iter = 15 | Mean = -7.461764555 | Best = -0.003010902
GA | iter = 16 | Mean = -9.390269468 | Best = -0.003010902
GA | iter = 17 | Mean = -4.523835575 | Best = -0.003010902
GA | iter = 18 | Mean = -3.367755554 | Best = -0.003010902
GA | iter = 19 | Mean = -5.228869316 | Best = -0.003010902
GA | iter = 20 | Mean = -1.761303495 | Best = -0.003010902
GA | iter = 21 | Mean = -4.103444352 | Best = -0.003010902
GA | iter = 22 | Mean = -2.583822137 | Best = -0.003010902
GA | iter = 23 | Mean = -4.300955092 | Best = -0.003010902
GA | iter = 24 | Mean = -3.278160031 | Best = -0.003010902
GA | iter = 25 | Mean = -5.372335517 | Best = -0.002718448
GA | iter = 26 | Mean = -1.939442555 | Best = -0.002718448
GA | iter = 27 | Mean = -3.066997863 | Best = -0.002718448
GA | iter = 28 | Mean = -2.790060541 | Best = -0.002701575
GA | iter = 29 | Mean = -2.732641520 | Best = -0.002191539
```

Et nous pouvons résumer l'analyse:

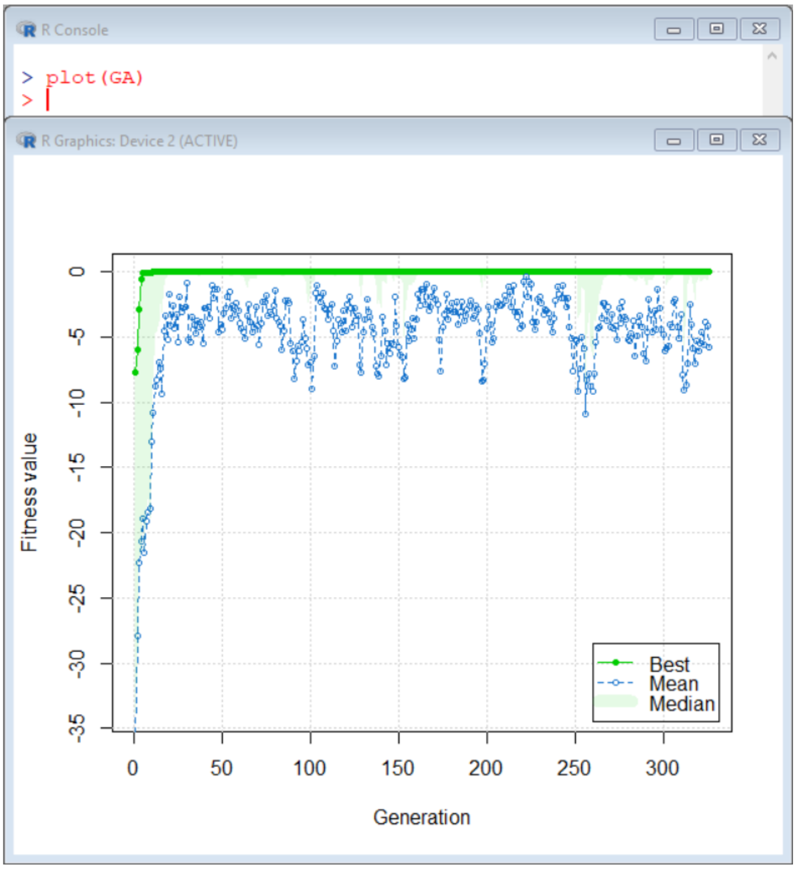
```
> summary(GA)
-- Genetic Algorithm -------------------

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  1000
Elitism               =  2
Crossover probability =  0.8
Mutation probability  =  0.1
Search domain =
         x1    x2
lower -5.12 -5.12
upper  5.12  5.12

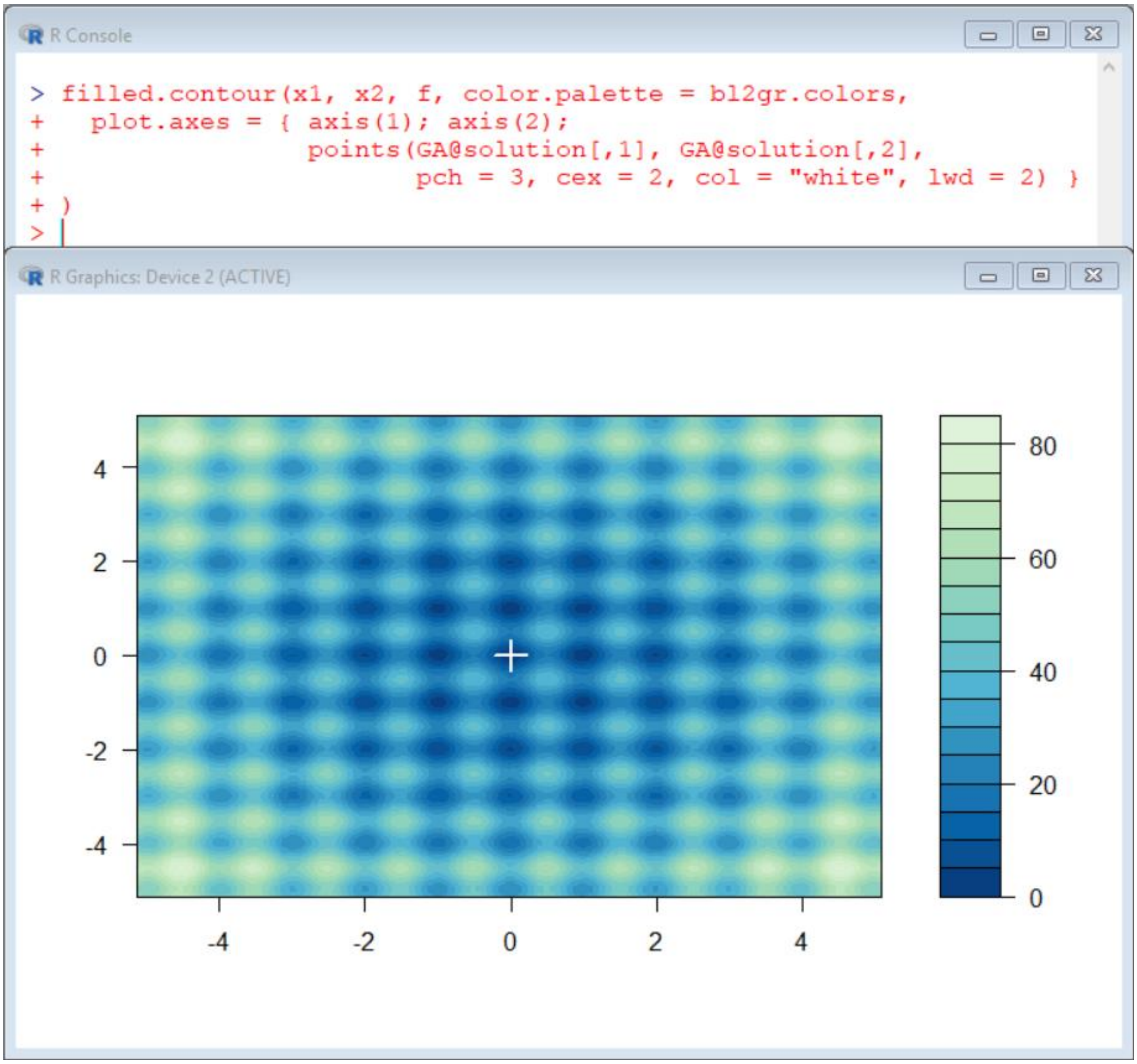
GA results:
Iterations             = 326
Fitness function value = -2.384393e-08
Solution =
                x1           x2
[1,] -6.444661e-06 8.868609e-06
>
```

On converge donc bien vers la coordonnée (0,0) qui est l'optimum global!

On peut analyser la convergence graphiquement:



On peut aussi plotter le résultat:



On peut même visualiser avec une animation, le fonctionnement de l'algorithme génétique:

R Console

```
> monitor <- function(obj)
+ {
+   contour(x1, x2, f, drawlabels = FALSE, col = grey(0.5))
+   title(paste("iteration =", obj@iter), font.main = 1)
+   points(obj@population, pch = 20, col = 2)
+   Sys.sleep(0.2)
+ }
>
> GA <- ga(type = "real-valued",
+          fitness =  function(x) -Rastrigin(x[1], x[2]),
+          lower = c(-5.12, -5.12), upper = c(5.12, 5.12),
+          popSize = 50, maxiter = 100,
+          monitor = monitor)
```

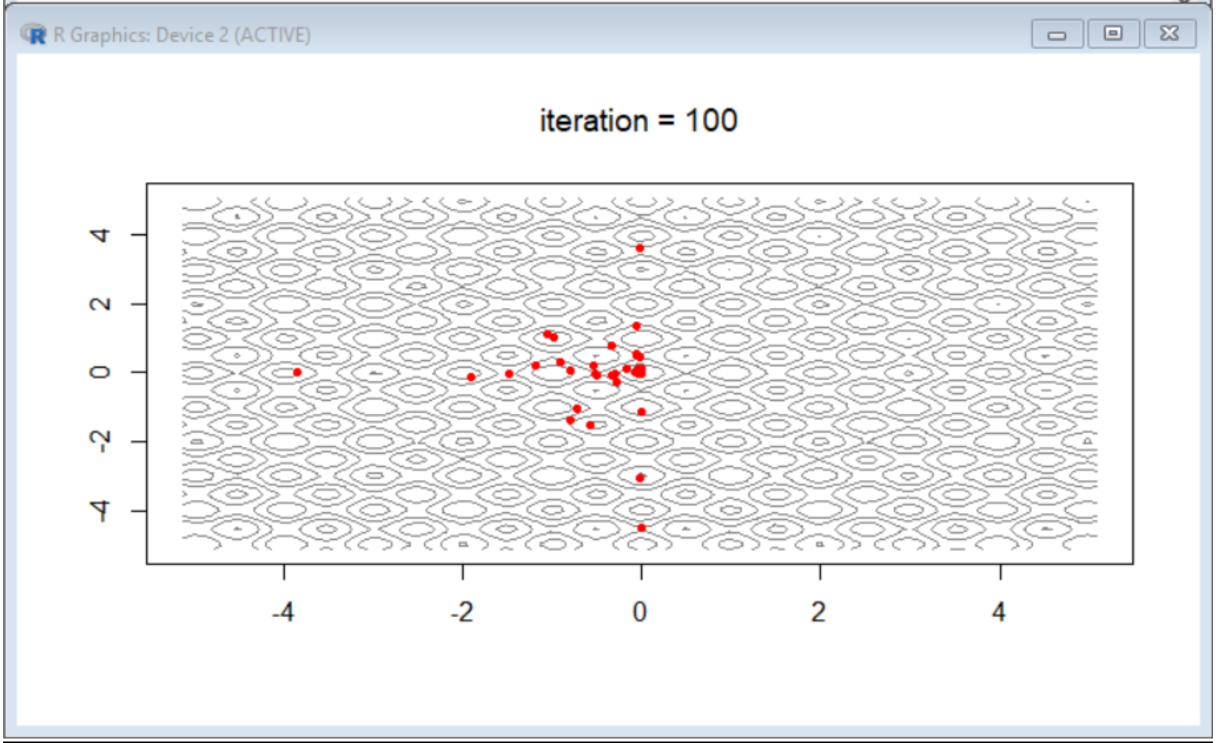


Figure 23 Graph de représentation de convergence d'algorithme génétique

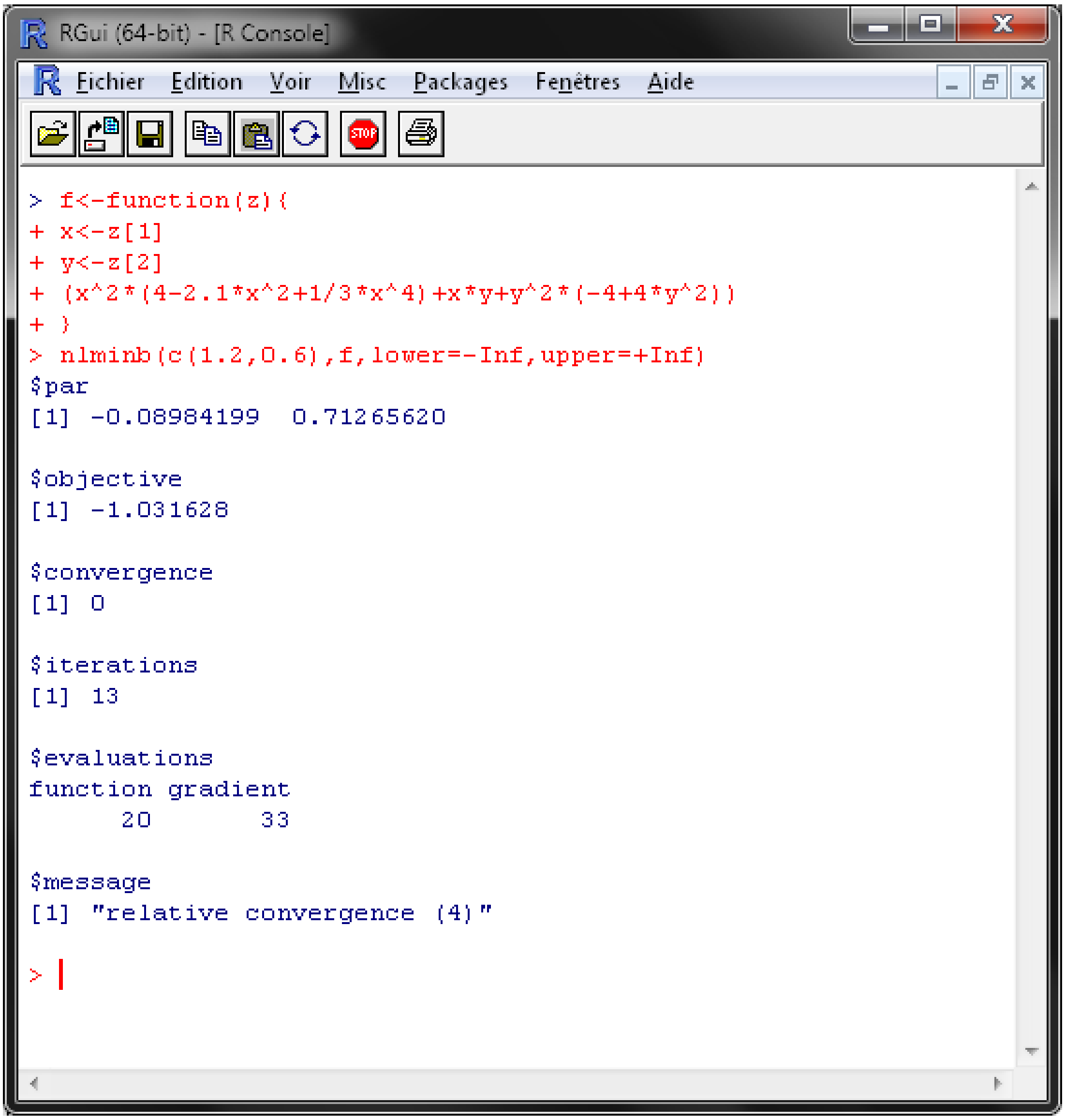
Exemple 4.: Optimisation multidimensionnelle (avec contraintes)

R 3.0.2

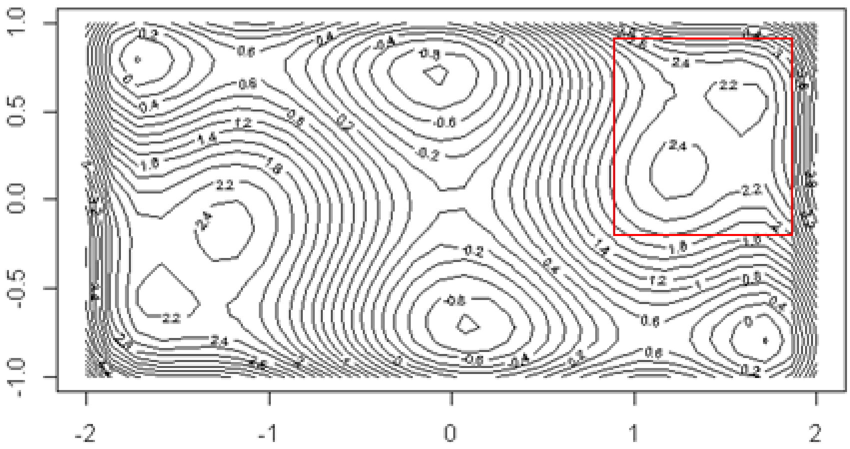
Nous allons utiliser la commande **nlmb( )** qui cherche un minimum local avec des contraintes bornées à l'infini en partant arbitrairement du point:

$$(x, y) = (1.2, 0.6)$$

d'abord pour vérifier que nous retrouvons bien les mêmes valeurs que la commande **nlm( )** :



Après supposons que nous cherchons le maximum dans la zone mise en évidence ci-dessous:



Dans ce cas nous écrivons (à finir d'écrire...):

..... (à terminer)

Exemple 5.: Optimisation (programmation) linéaire par la méthode du simplexe

R 3.0.2

Nous allons donc ici utiliser R pour résoudre le problème de programmation linéaire étudié en cours suite à la démonstration mathématique de la méthode du simplexe.

Rappelons que le problème était donné par le système canonique suivant:

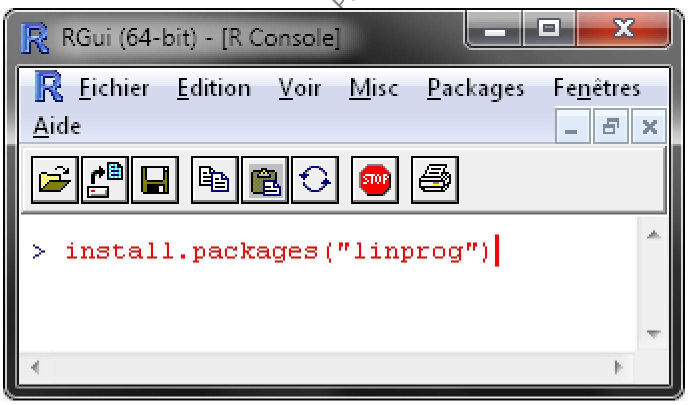
$$\begin{aligned} A1 &: 3x_1 + 4x_2 \leq 160 \\ A2 &: 6x_1 + 3x_2 \leq 180 \\ & x_1, x_2 \geq 0 \end{aligned}$$

avec la fonction d'utilité à maximiser:

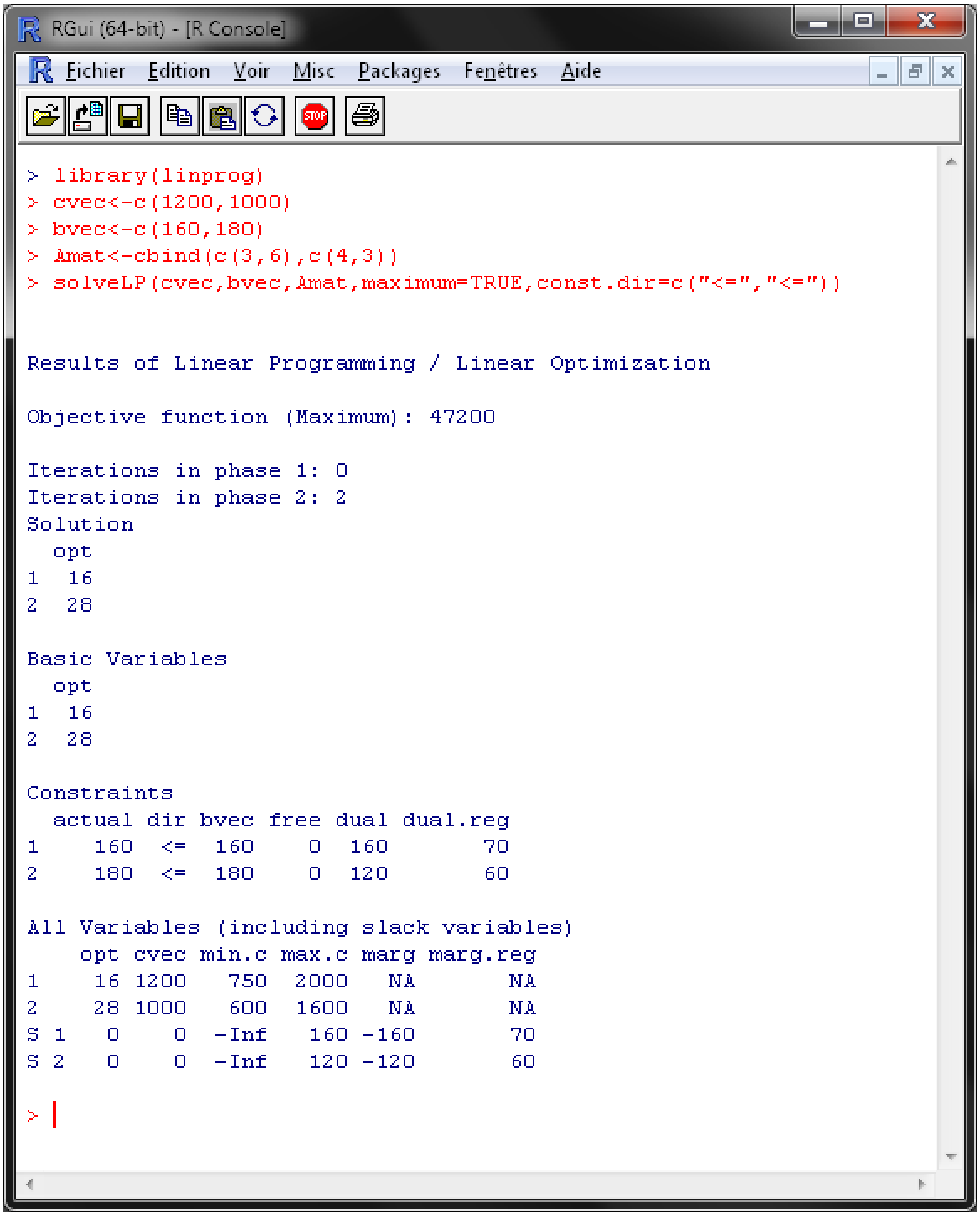
$$Z = 1'200x_1 + 1'000x_2$$

Nous avons vu que cela était relativement laborieux à résoudre à la main, très simple à résoudre dans les tableurs Open Office Calc et Microsoft. Excel. Eh bien voyons maintenant cela avec R.

D'abord, nous installons le package **linprog**:



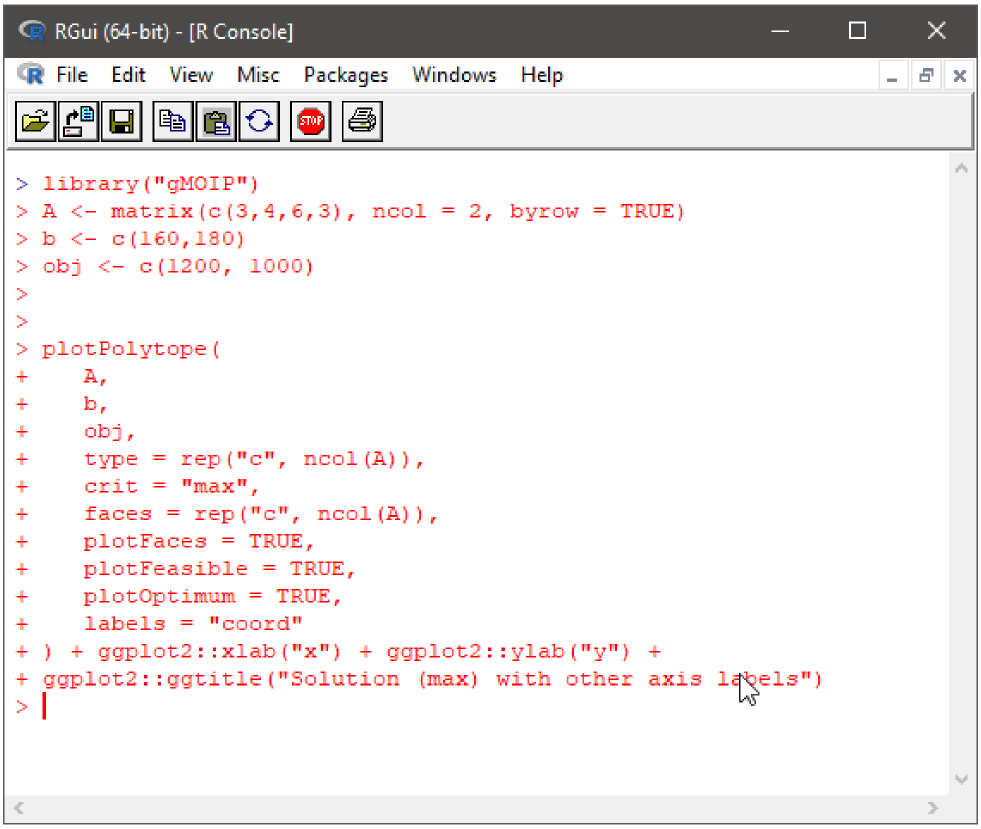
et ensuite nous écrivons (pas évident à deviner):



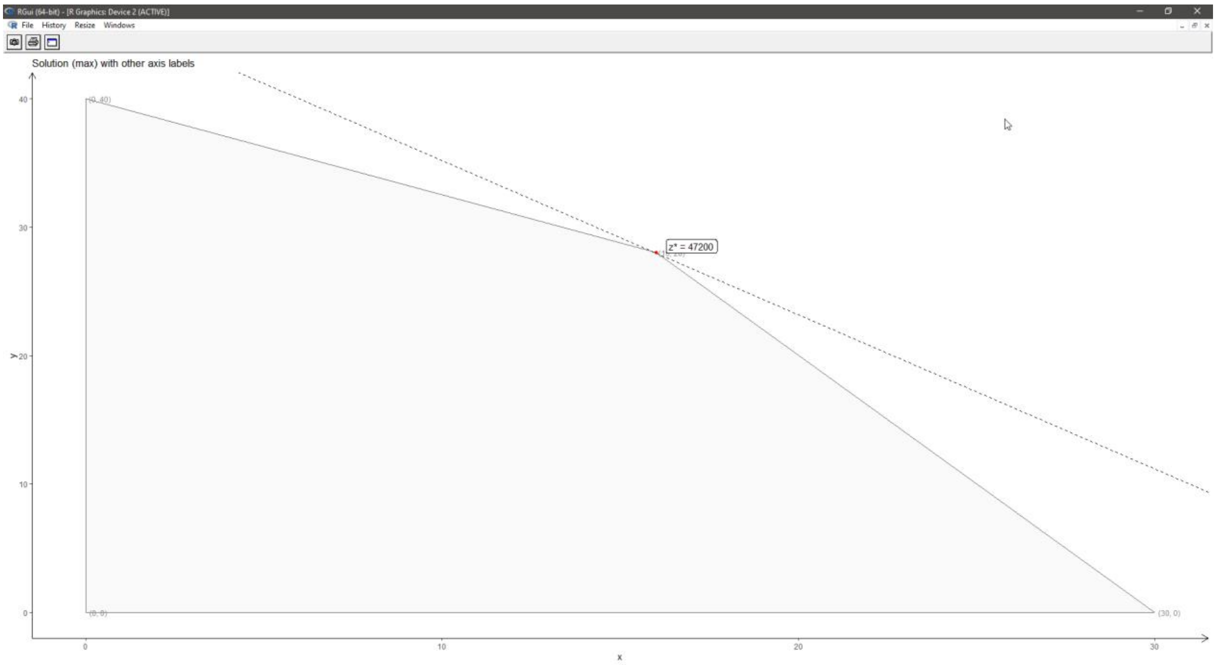
et nous retrouvons donc bien tous les résultats des calculs faits à la main et dans les tableurs dans le cours théoriques.

Par défaut le package **linprog** n'accepte que les résultats positifs (raison pour laquelle il n'y a pas d'options pour définir le signe des variables).

Avec le package **gMOIP** et sa fonction **plotPolytope( )** nous pouvons enfin depuis 2022 faire des graphes 2D et pseudo-3D de programmes linéaires de type simplexe! Voyons comment faire le graphique (polytope) du problème ci-dessus (même si ce n'est pas parfait… c'est mieux que rien!):


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("gMOIP")
> A <- matrix(c(3,4,6,3), ncol = 2, byrow = TRUE)
> b <- c(160,180)
> obj <- c(1200, 1000)
>
>
> plotPolytope(
+ A,
+ b,
+ obj,
+ type = rep("c", ncol(A)),
+ crit = "max",
+ faces = rep("c", ncol(A)),
+ plotFaces = TRUE,
+ plotFeasible = TRUE,
+ plotOptimum = TRUE,
+ labels = "coord"
+) + ggplot2::xlab("x") + ggplot2::ylab("y") +
+ ggplot2::ggtitle("Solution (max) with other axis labels")
>

Ce qui donne:


Solution (max) with other axis labels
z* = 47200

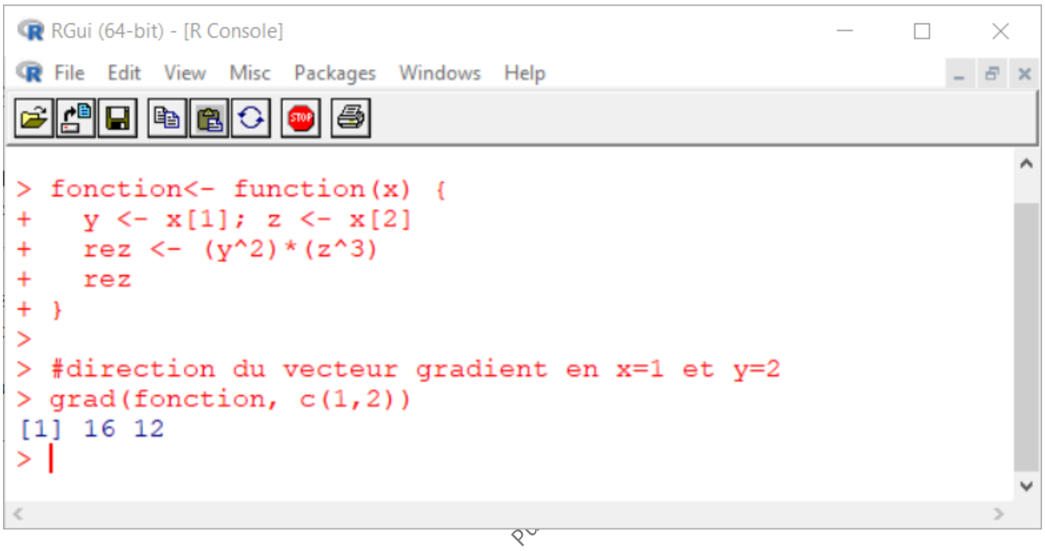
Exemple 6.: Gradient et Matrice Hessienne

R 3.4.2

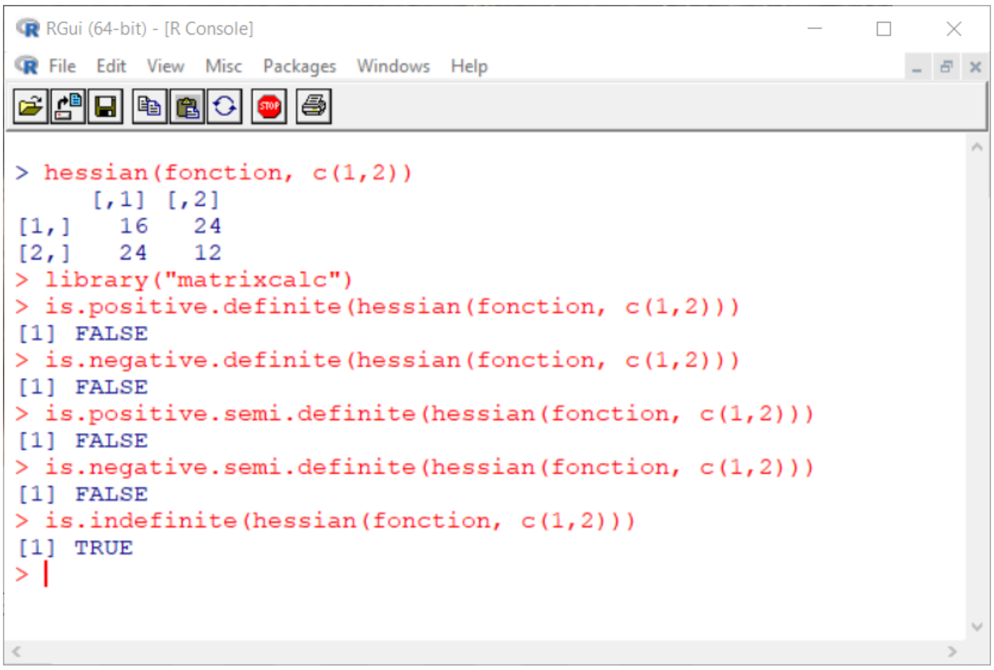
Nous avons vu dans le cours théorique l'importance de la Hessienne (et donc implicitement du gradient aussi!) dans le cadre de l'optimisation non linéaire!

Voyons donc ici un exemple en utilisant le package **pracma**.

Calculons d'abord la direction du vecteur gradient au point $x = 1$ et $y = 2$ en utilisant la fonction **grad( )** :



Et voyons maintenant si on peut obtenir la matrice hessienne au même point à l'aide de la fonction **hessian( )** et comment extraire rapidement le type de matrice à laquelle nous avons affaire (définie semi-positive ou autre) au besoin avec le package **matrixcalc**:



Méthode de Gauss-Newton (logarithme du gradient)

On ne peut pas parler du gradient et de la matrice Hessienne en faisant abstraction de l'algorithme du gradient qui a une importance majeure en Machine Learning (apprentissage machine) et dans de nombreux domaines de l'ingénierie ou la physique appliquées.

Nous allons reprendre ici l'exemple du cours théorique et utiliser la fonction **grad.desc( )** du package **animation**:

```
> library(animation)
> ani.options(nmax = 50)
animation option 'nmax' changed: 20 --> 50
> f<-function(x, y) x^2*(4-2.1*x^2+1/3*x^4)+x*y+y^2*(-4+4*y^2)
> grad.desc(f,c(-2,-1.3,2,1.2),c(1,1),gamma = 0.01)
> |
```

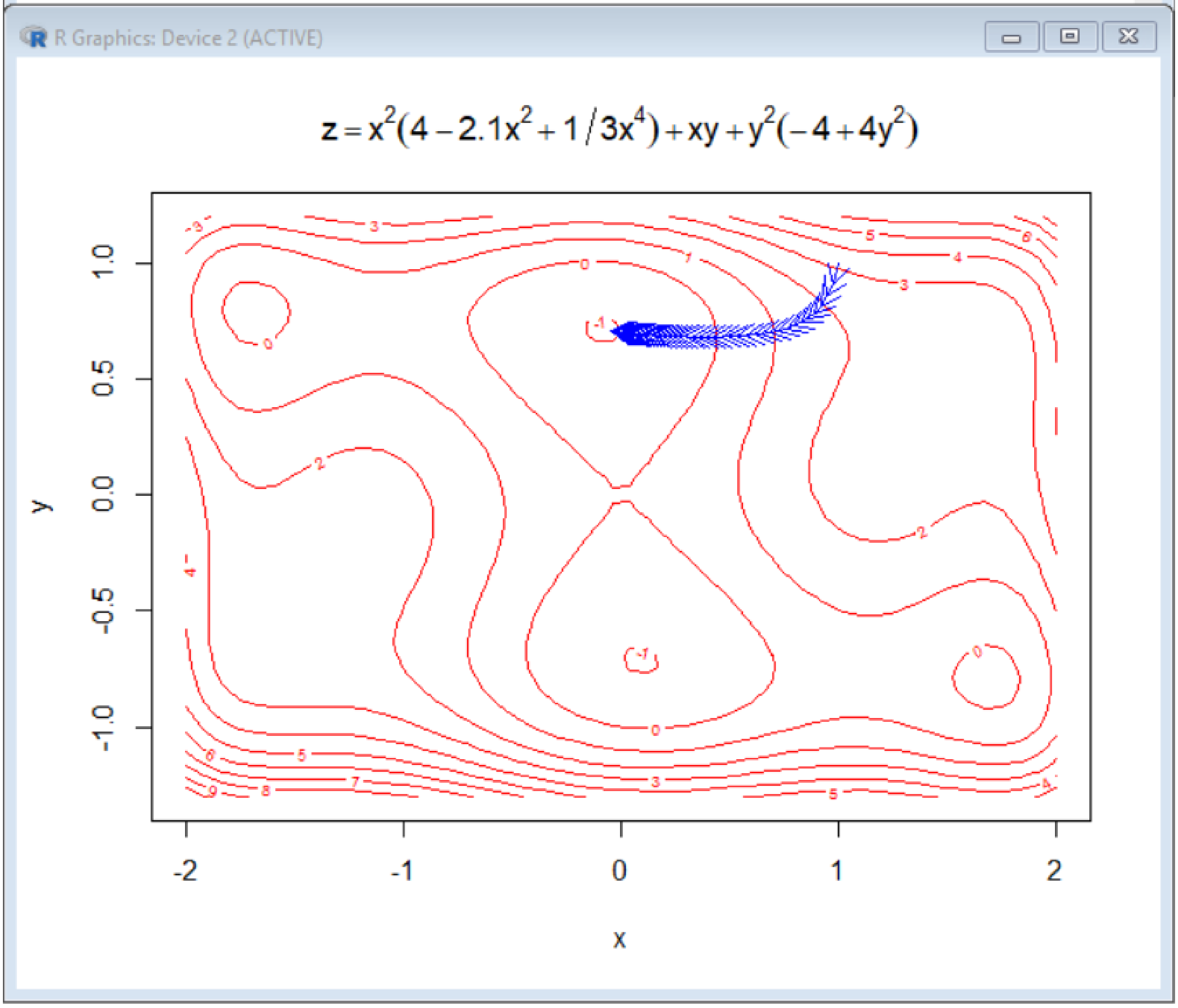


Figure 24 Graph de la convergence de le méthode de Gauss-Newton d'une fonction f(x,y)

Ou pour reprendre l'exemple avec l'exponentielle faite aussi dans le cours théorique:

```
> library(animation)
> ani.options(nmax = 100)
> f = function(x, y) (x*exp(y*1)-3.2939)^2+(x*exp(y*2)-4.2699)^2+(x*exp(y*4)-7.1749)^2+
+ (x*exp(y*5)-9.3008)^2+(x*exp(y*8)-20.259)^2
> gd<- grad.desc(f,init =c(2.5, 0.25), gamma = 0.00001)
Warning messages:
1: In arrows(xy[1:i, 1], xy[1:i, 2], newxy[1:i, 1], newxy[1:i, 2],  :
  zero-length arrow is of indeterminate angle and so skipped
2: In arrows(xy[1:i, 1], xy[1:i, 2], newxy[1:i, 1], newxy[1:i, 2],  :
  zero-length arrow is of indeterminate angle and so skipped
3: In arrows(xy[1:i, 1], xy[1:i, 2], newxy[1:i, 1], newxy[1:i, 2],  :
  zero-length arrow is of indeterminate angle and so skipped
> gd$par
        x         y 
2.5006480 0.2614825 
> 
```

9. Importer/Exporter des données (Data Collection)

Contenu

Pour ceux qui préfèrent utiliser des interfaces visuelles pour l'import de données, nous les renvoyons à des outils comme **RStudio** ou **SpeedR**. Mais attention il faut avoir le Java Runtime installé sur sa machine (ce qui est toujours le cas dans les grandes organisations de toute façon).

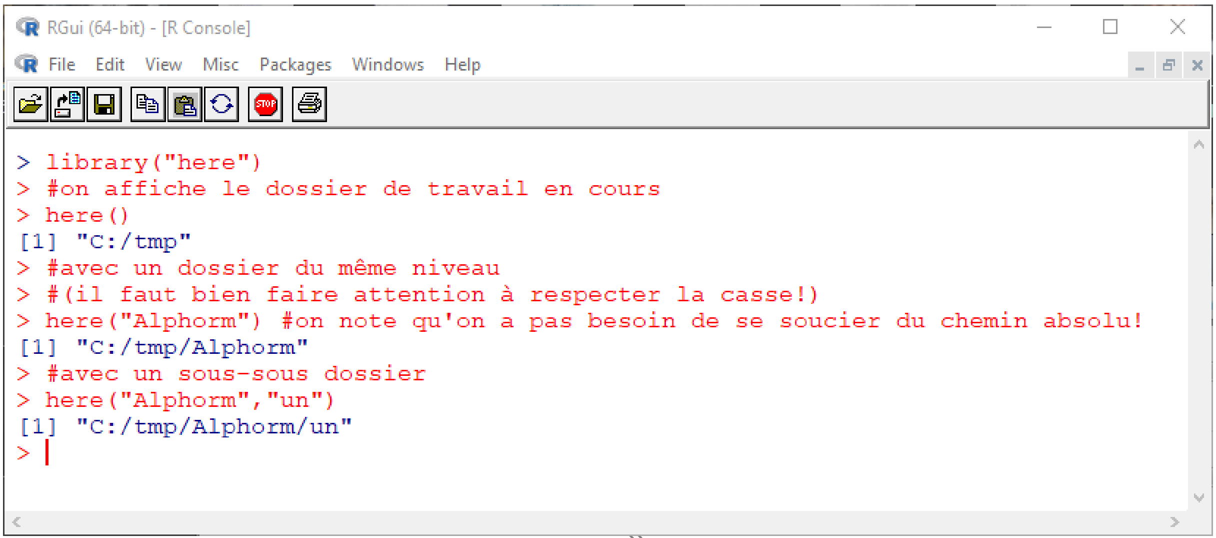
Dans ce chapitre, nous allons nous concentrer sur des méthodes se focalisant sur l'utilisation de l'interface native de R comme à l'habitude.

Exemple 33.: À propos du package "here"

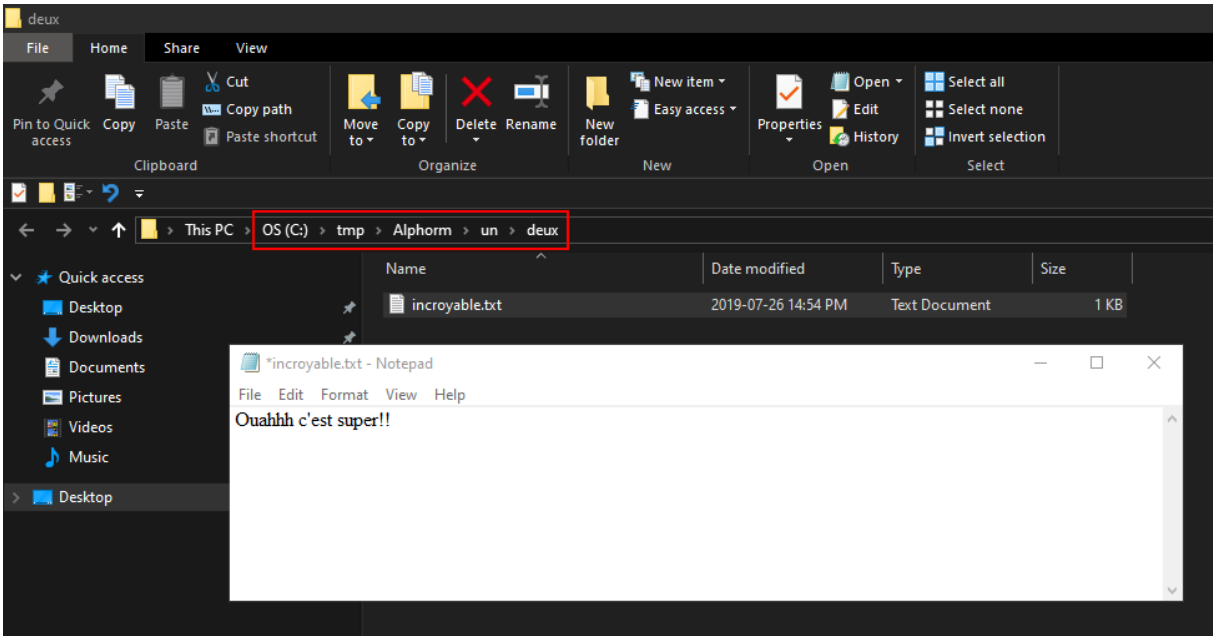
R 3.6.2

Avant de commencer, nous souhaiterions ouvrir une parenthèse sur le package **here** et sa fonction **here( )** qui est utilisée par certains consultants, statisticiens et data scientists car elle permet de travailler plus simplement avec des dossiers en relatif (c'est-à-dire sans avoir à se soucier de la structure des dossiers parents du dossier de travail en cours!).

Voyons en quoi cela consiste



Maintenant considérons la structure de dossiers suivante ainsi qu'un fichier texte:



Voyons comment utiliser le package **here** pour accéder à ce fichier:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> #pour accéder à un fichier dans "/Alphorm/un/deux/" on utilisera par exemple
> print(readLines(here("Alphorm","un","deux","incroyable.txt")))
[1] "Ouahhh c'est super!!"
>

Exemple 34.: Importer/Exporter des *.csv en local

R 3.0.2

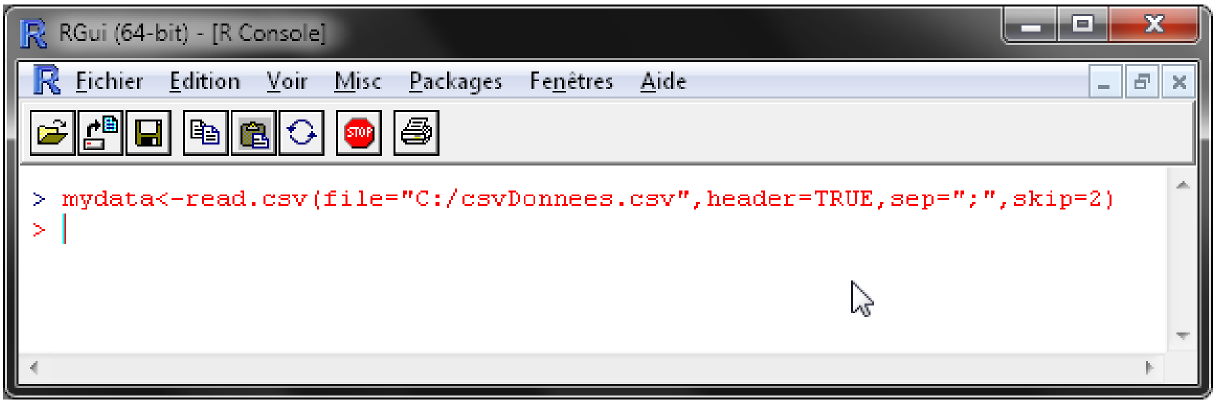
Ici nous allons vouloir d'abord importer au propre le fichier *.csv suivant qui contient une petite complication assez courante dans les entreprises:

```
csvDonnees.csv - Bloc-notes
Fichier Edition Format Affichage ?
Une ligne de début avec du texte inutile
et une deuxième ligne inutile...
N° Client;Activité;N° de Commande;Date de commande;Article;Quantité;Prix par pièc
100;Assurances;1;03.01.2000;Compaq Presario 100;12;1650;1.50%;19503;Oui
123;Machines/Outils;2;03.01.2000;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000;Compaq Presario 100;13;1650;1.50%;21128.25;Oui
103;Assurances;6;04.01.2000;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000;Compaq Presario 100;4;1650;0.00%;6600;Oui
116;Pharmaceutique;10;04.01.2000;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000;Compaq Presario 100;6;1650;1.50%;9751.5;Oui
123;Machines/Outils;13;05.01.2000;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000;Compaq Presario 100;8;1650;1.50%;13002;Oui
124;Éducation;16;05.01.2000;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000;Compaq Presario 100;11;1650;1.50%;17877.75;Oui
106;Construction;18;05.01.2000;AST Intel 200;11;3190;1.50%;34563.65;Oui
101;Construction;19;05.01.2000;Compaq Presario 100;14;1650;1.50%;22753.5;Non
116;Pharmaceutique;20;06.01.2000;IBM 500;7;2299;1.50%;15851.605;Non
112;Machines/Outils;21;06.01.2000;AST Intel 150;6;2690;1.50%;15897.9;Oui
```

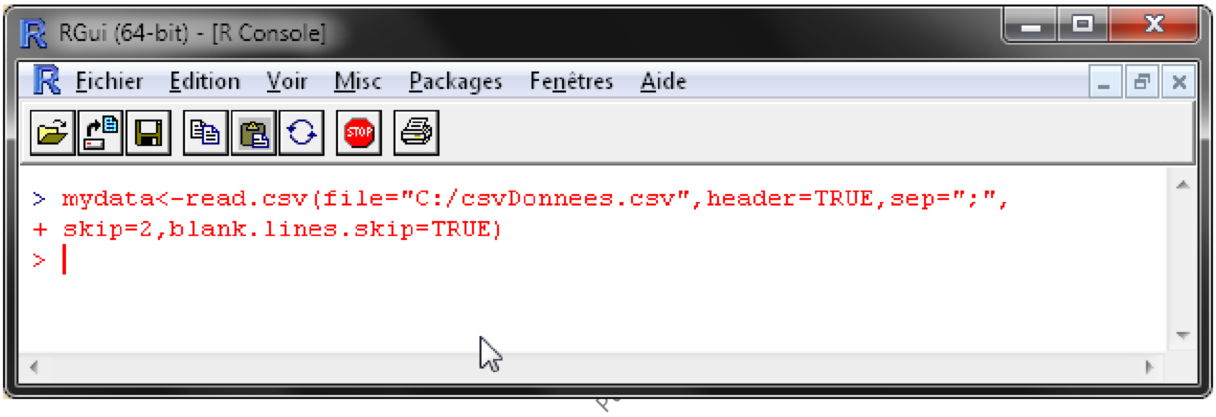
Pour ce faire voici nous allons utiliser la commande native **read.csv( )** et ignorer les deux premières lignes avec le paramètre *skip*:

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> setwd("C:/")
> mydata=read.csv("csvDonnees.csv",header=T,sep=";",skip=2)
> mydata
   N..Client        Activité N..de.Commande Date.de.commande             Article
1        100      Assurances              1       03.01.2000 Compaq Presario 100
2        123 Machines/Outils              2       03.01.2000             IBM 500
3        109       Éducation              3       03.01.2000       AST Intel 150
4        104       Éducation              4       03.01.2000       AST Intel 200
5        117         Banques              5       04.01.2000 Compaq Presario 100
6        103      Assurances              6       04.01.2000       AST Intel 150
7        104       Éducation              7       04.01.2000       AST Intel 200
8        111     Alimentaire              8       04.01.2000             IBM 500
9        113    Construction              9       04.01.2000 Compaq Presario 100
10       116  Pharmaceutique             10       04.01.2000             IBM 500
11       110    Distribution             11       05.01.2000       AST Intel 200
12       112 Machines/Outils             12       05.01.2000 Compaq Presario 100
13       123 Machines/Outils             13       05.01.2000             IBM 500
14       113    Construction             14       05.01.2000       AST Intel 150
15       115    Distribution             15       05.01.2000 Compaq Presario 100
16       124       Éducation             16       05.01.2000       AST Intel 200
17       124       Éducation             17       05.01.2000 Compaq Presario 100
18       106    Construction             18       05.01.2000       AST Intel 200
```

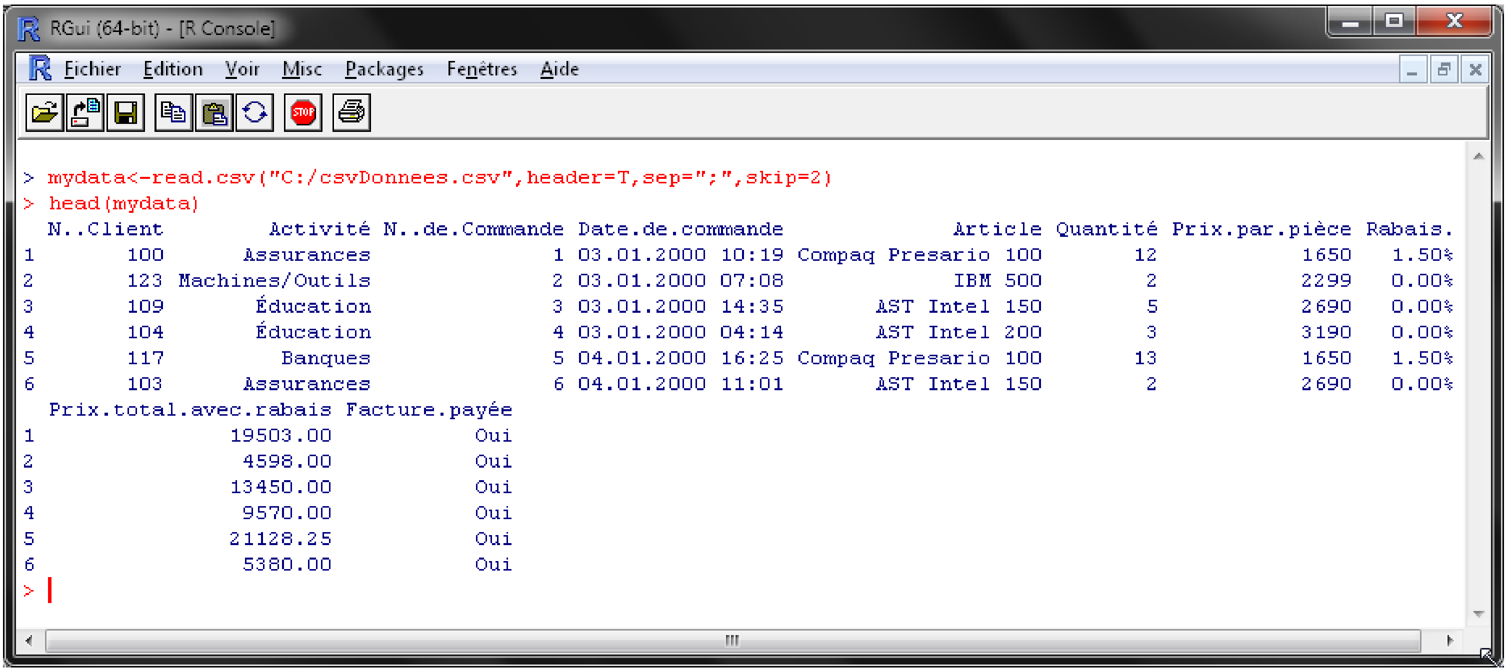
Ou plus explicitement:



Et s'il y a des lignes vides, nous pouvons nous en débarrasser automatiquement:

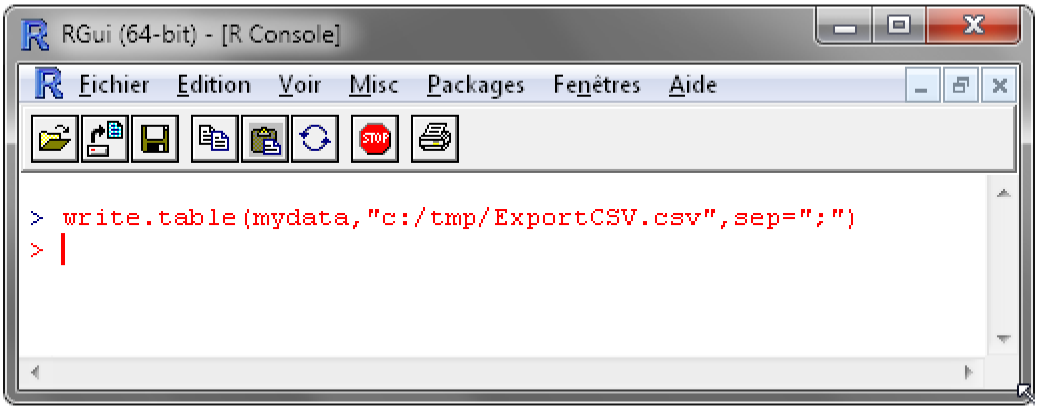


Nous pouvons afficher les 6 premières lignes du data frame avec la commande **head( )** (ou les 6 dernières avec la commande **tail( )**):

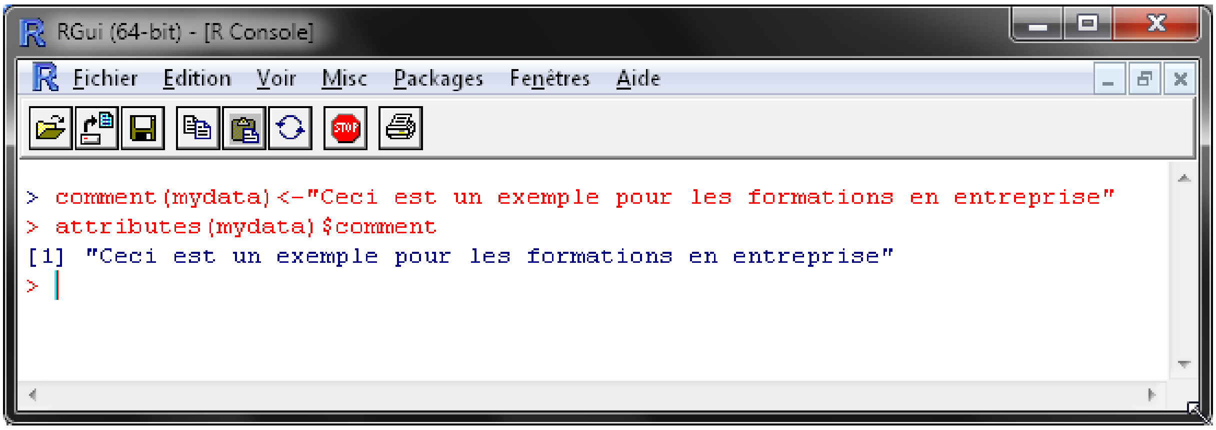


Nous pouvons importer aussi tout un dossier comme cela est montré via un exemple à la page 2313.

Une fois des opérations/calculs/manipulations effectuées sur le data frame nous pouvons l'exporter à nouveau en *.csv avec la commande **write.table( )** :



Nous pouvons aussi associer un commentaire à un data frame pour y mettre une petite remarque:



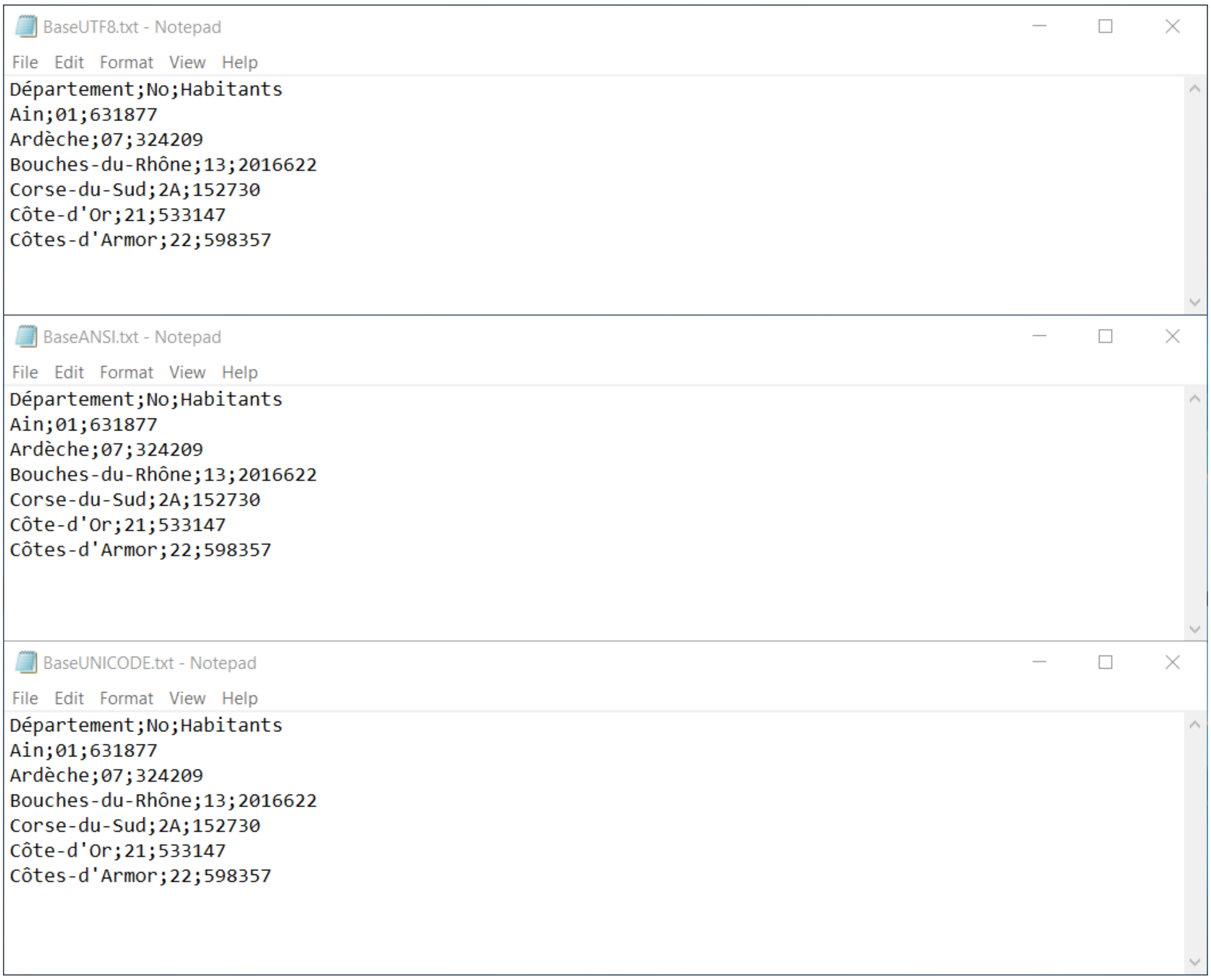
Nous pouvons aussi importer des fichiers textes avec le package **readr** qui contient relativement plus d'options et qui en plus génère des objets de type tibble en sortie. Voyons cela:

```
> library("readr")
> #read_csv2 lit les fichiers avec des ";"
> #read_csv lit les fichiers avec des ","
> options(tibble.print_min = 15)
> (mydata<-read_csv2(file="C:/tmp/csvDonnees.csv",col_names=TRUE,skip=2,skip_empty_rows=TRUE,
+ ,locale(encoding = "CP1254"),progress = show_progress()))
Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
Parsed with column specification:
cols(
  `N° Client` = col_double(),
  Activité = col_character(),
  `N° de Commande` = col_double(),
  `Date de commande` = col_character(),
  Article = col_character(),
  Quantité = col_double(),
  `Prix par pièce` = col_double(),
  `Rabais%` = col_character(),
  `Prix total avec rabais` = col_number(),
  `Facture payée` = col_character()
)
# A tibble: 109 x 10
   `N° Client` Activité `N° de Commande` `Date de comman~ Article Quantité `Prix par pièce`
         <dbl> <chr>               <dbl> <chr>            <chr>      <dbl>            <dbl>
 1         100 Assuran~                1 03.01.2000 10:19 Compaq~       12             1650
 2         123 Machine~                2 03.01.2000 07:08 IBM 500        2             2299
 3         109 Éducati~                3 03.01.2000 14:35 AST In~        5             2690
 4         104 Éducati~                4 03.01.2000 04:14 AST In~        3             3190
 5         117 Banques                 5 04.01.2000 16:25 Compaq~       13             1650
 6         103 Assuran~                6 04.01.2000 11:01 AST In~        2             2690
 7         104 Éducati~                7 04.01.2000 21:30 AST In~        2             3190
 8         111 Aliment~                8 04.01.2000 02:20 IBM 500        4             2299
 9         113 Constru~                9 04.01.2000 23:06 Compaq~        4             1650
10         116 Pharmac~               10 04.01.2000 16:14 IBM 500        2             2299
11         110 Distrib~               11 05.01.2000 14:21 AST In~        6             3190
12         112 Machine~               12 05.01.2000 03:29 Compaq~        6             1650
13         123 Machine~               13 05.01.2000 16:22 IBM 500        6             2299
14         113 Constru~               14 05.01.2000 16:09 AST In~        3             2690
15         115 Distrib~               15 05.01.2000 01:47 Compaq~        8             1650
# ... with 94 more rows, and 3 more variables: `Rabais%` <chr>, `Prix total avec
#   rabais` <dbl>, `Facture payée` <chr>
> class(mydata)
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
> #on convertit le tibble en data frame
> mydata<-data.frame(mydata)
> 
```

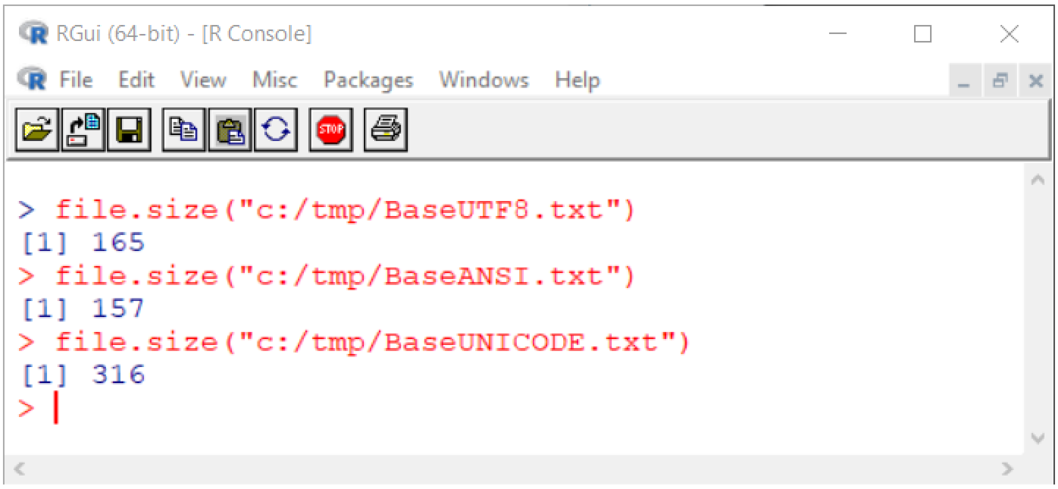
Exemple 35.: Import avec gestion des encodages

R 3.5

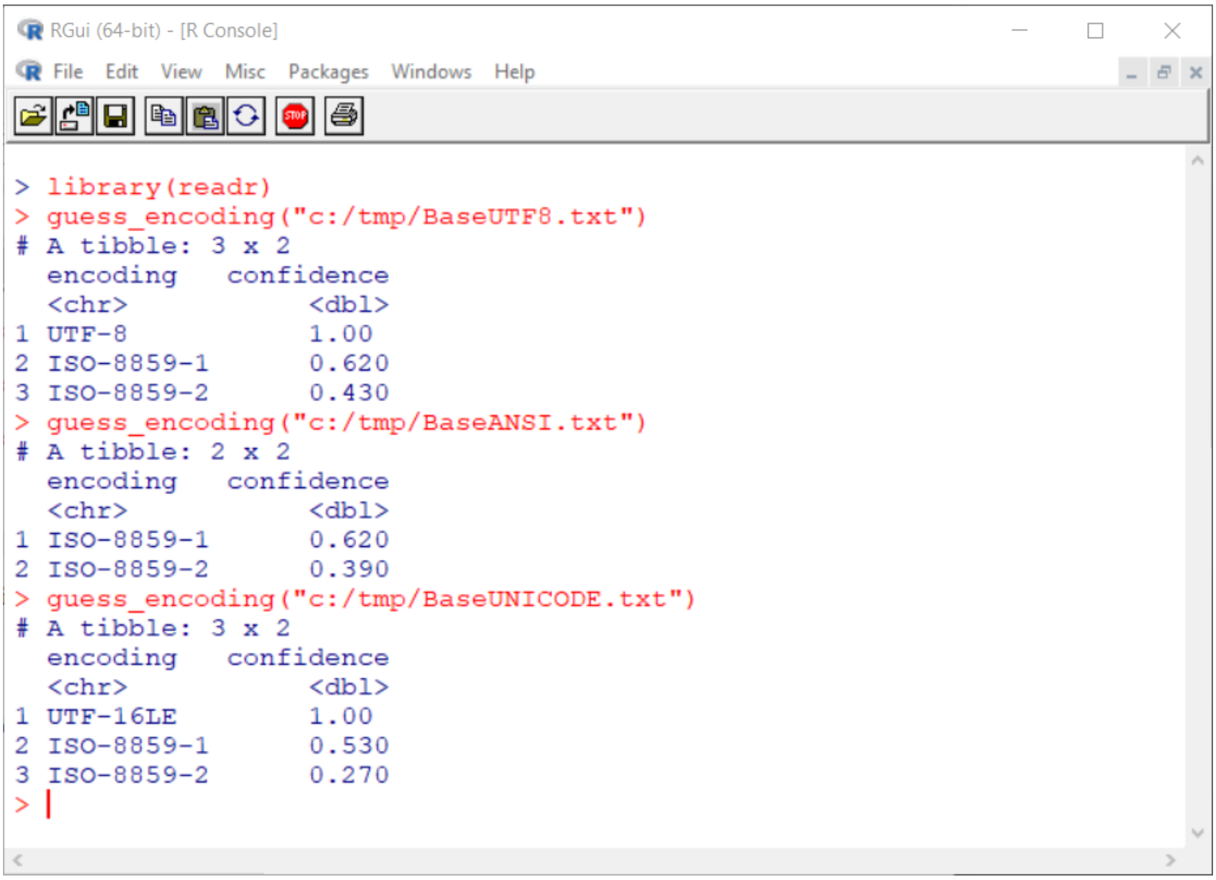
L'encodage avec les dates est un des cauchemars des informaticiens (et donc du Data Scientist). Pour l'illustrer, considérons les trois fichiers suivants qui ont l'air a priori identiques mais comme leur nom l'indique… sont en réalité encodés différemment:



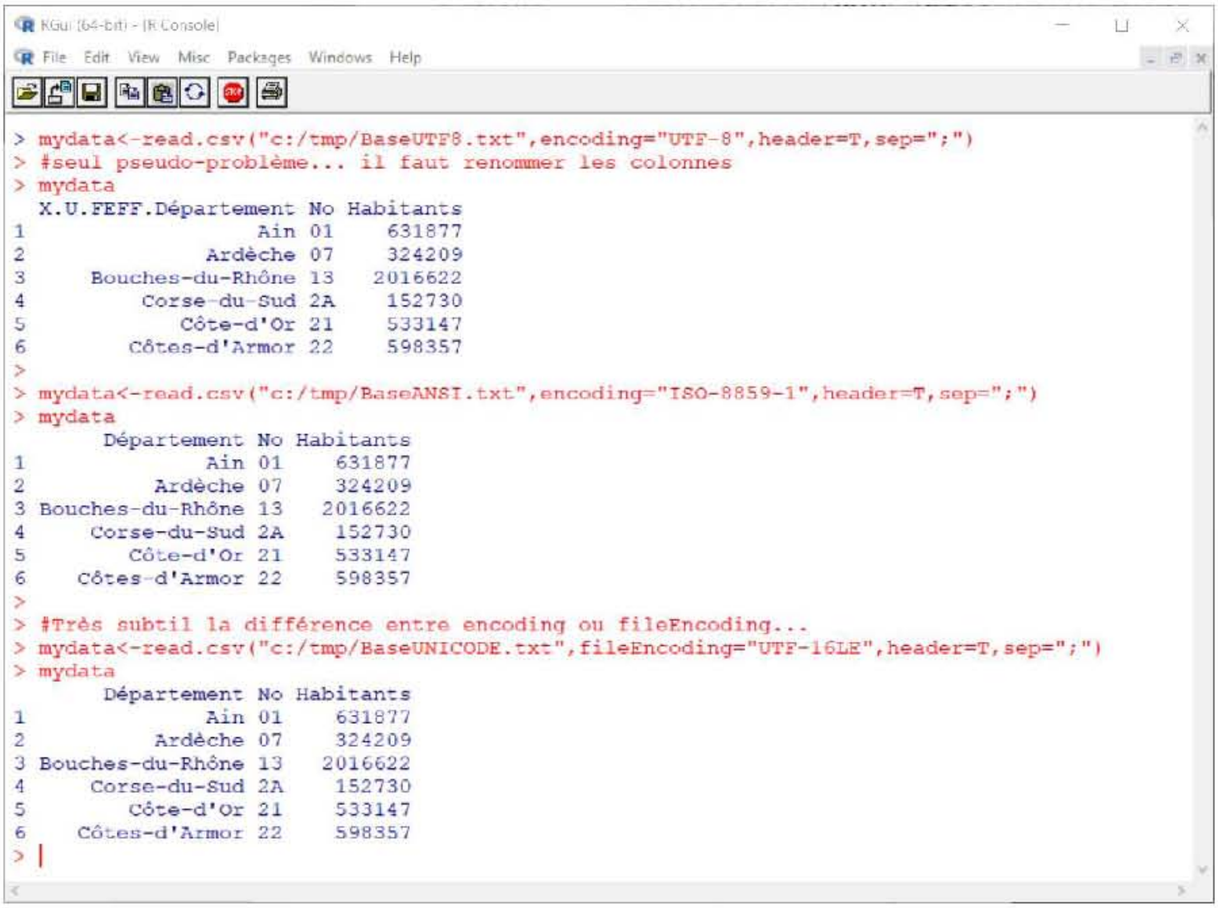
Déjà… vérifions leur taille mémoire afin de vérifier indirectement que leur encodage est différent et ce à l'aide de la fonction **file.size( )** :



Donc oui… effectivement cela n'annonce rien de bon! Allons un peu plus loin en détectant l'encodage à l'aide de la fonction **guess_encoding( )** du package **readr**:



Ayant ces informations, allons-y pour l'import en utilisant certains paramètres proposés par **read.csv( )**:

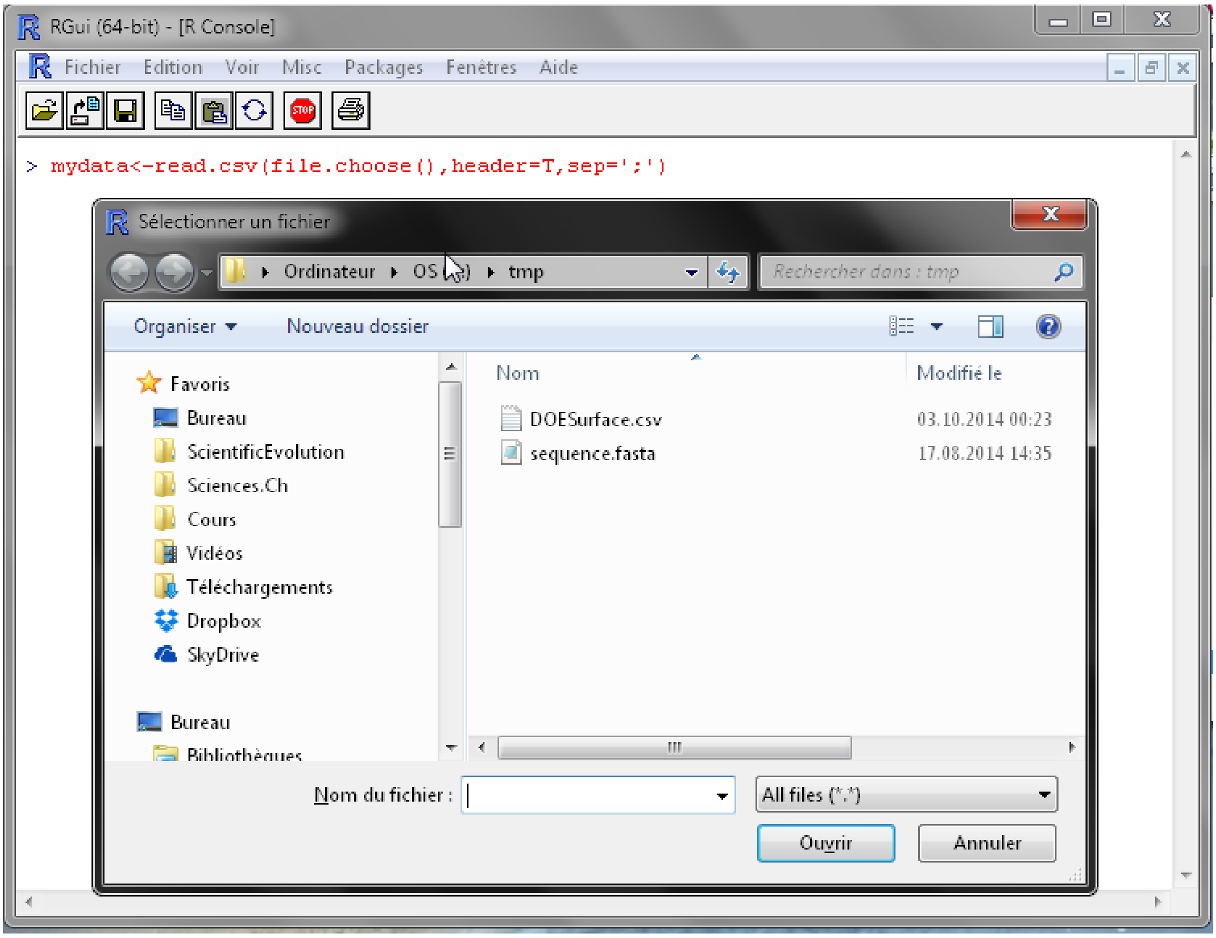


```
> mydata<-read.csv("c:/tmp/BaseUTF8.txt",encoding="UTF-8",header=T,sep=";")
> #seul pseudo-problème... il faut renommer les colonnes
> mydata
  X.U.FEFF.Département No Habitants
1                  Ain 01    631877
2              Ardèche 07    324209
3     Bouches-du-Rhône 13   2016622
4        Corse-du-Sud 2A    152730
5          Côte-d'Or 21    533147
6        Côtes-d'Armor 22    598357
>
> mydata<-read.csv("c:/tmp/BaseANSI.txt",encoding="ISO-8859-1",header=T,sep=";")
> mydata
       Département No Habitants
1              Ain 01    631877
2          Ardèche 07    324209
3 Bouches-du-Rhône 13   2016622
4     Corse-du-Sud 2A    152730
5        Côte-d'Or 21    533147
6    Côtes-d'Armor 22    598357
>
> #Très subtil la différence entre encoding ou fileEncoding...
> mydata<-read.csv("c:/tmp/BaseUNICODE.txt",fileEncoding="UTF-16LE",header=T,sep=";")
> mydata
       Département No Habitants
1              Ain 01    631877
2          Ardèche 07    324209
3 Bouches-du-Rhône 13   2016622
4     Corse-du-Sud 2A    152730
5        Côte-d'Or 21    533147
6    Côtes-d'Armor 22    598357
> 
```

Exemple 36.: Importer interactivement des données

R 3.1.2

Le but ici est simplement de découvrir la fonction **file.choose( )** qui injectée dans **read.csv( )** dans des plus gros scripts (que ce soit en pur R ou en C# avec R.Net) permet à l'utilisateur de choisir son fichier avec une boîte de dialogue:



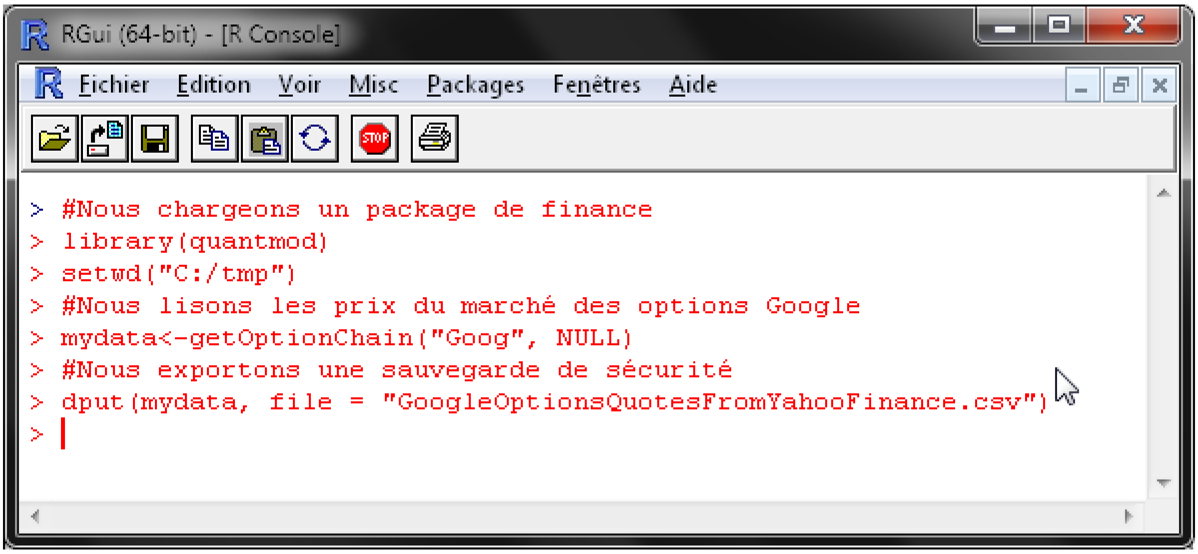
Exemple 37.: Importer/Exporter des *.csv en local pour les listes

R 3.0.2

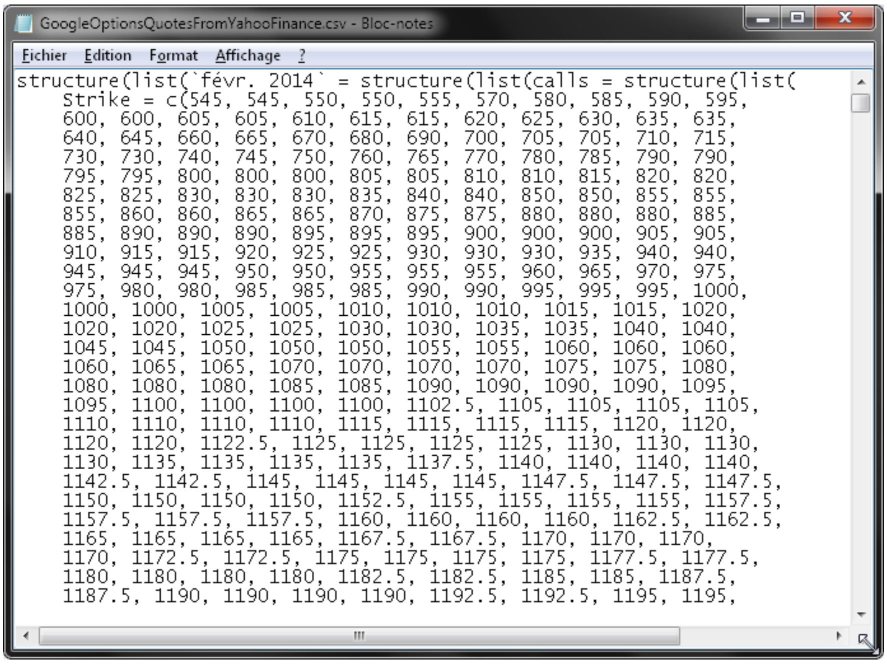
Quand nous commencerons à faire des scripts R complexes pour les domaines des SIG (systèmes d'information géographiques) ou la finance quantitative, nous nous retrouverons avec des listes importées d'Internet.

Il est parfois plus prudent de les sauvegarder en local car parfois les serveurs les fournissant sont en maintenant ou disparaissent tout simplement.

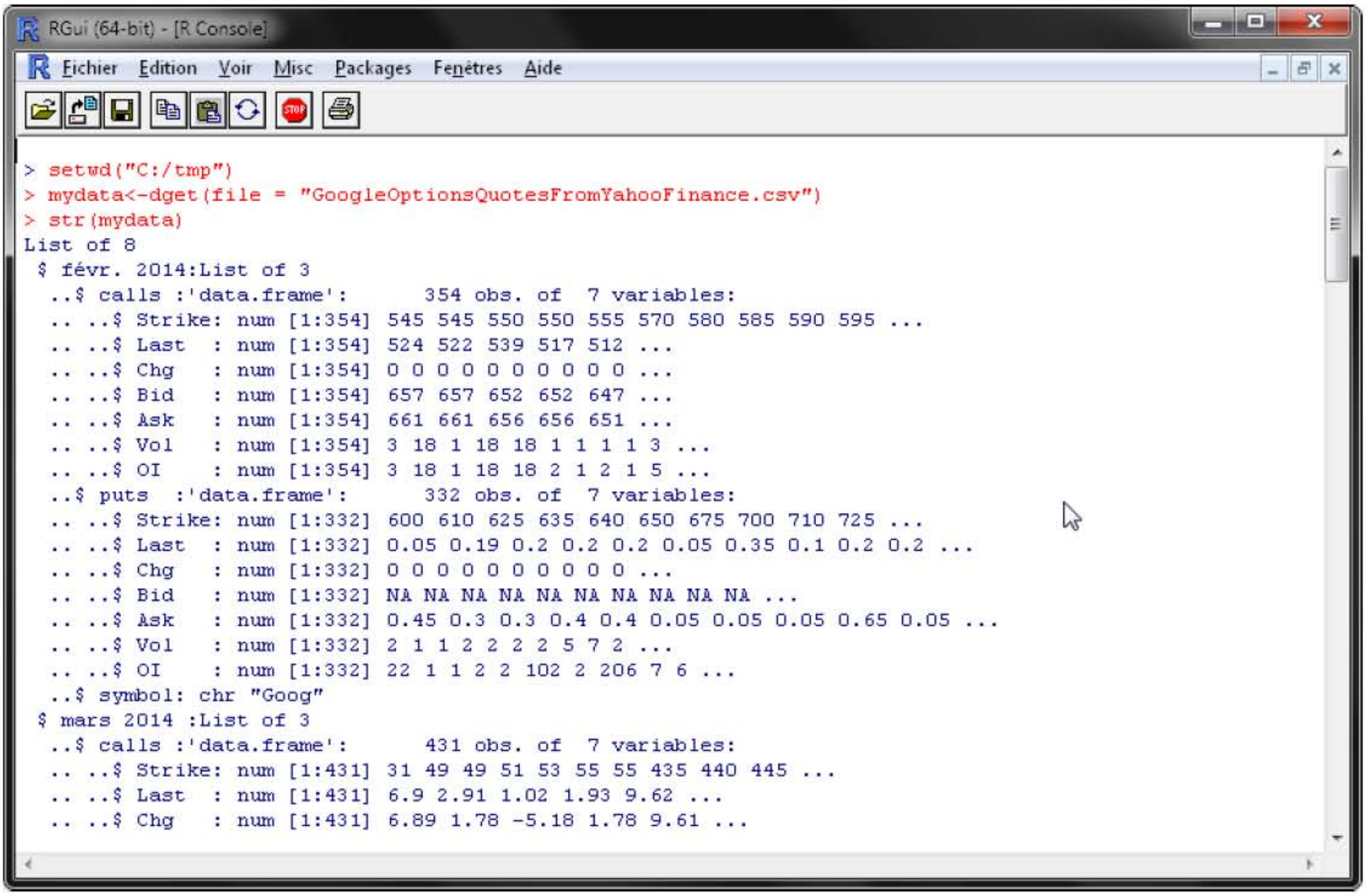
Voyons un exemple d'export en *.csv d'une liste fournie par Yahoo Finance avec le package **quantmod** et la commande d'export général à R pour les listes qui est **dput( )** :



Ce qui donne un fichier *.csv avec un extrait de contenu tel que visible ci-dessous:



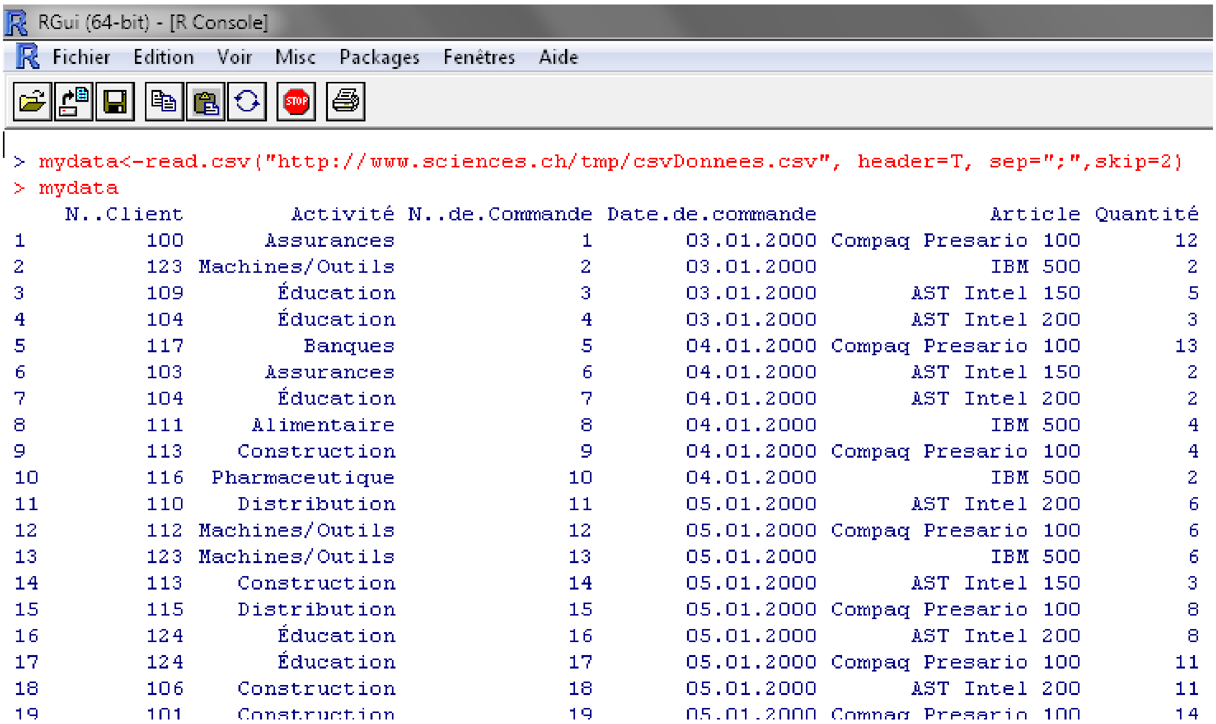
Ensuite, pour importer il suffit d'utiliser la commande **dget( )** :



Exemple 38.: Importer des *.csv du web

R 3.0.2

Un cas très important dans le domaine financier est de pouvoir aller chercher des données en temps réel sur des sociétés fournisse ce type de service. La technique n'est pas bien différente d'avec la technique locale:

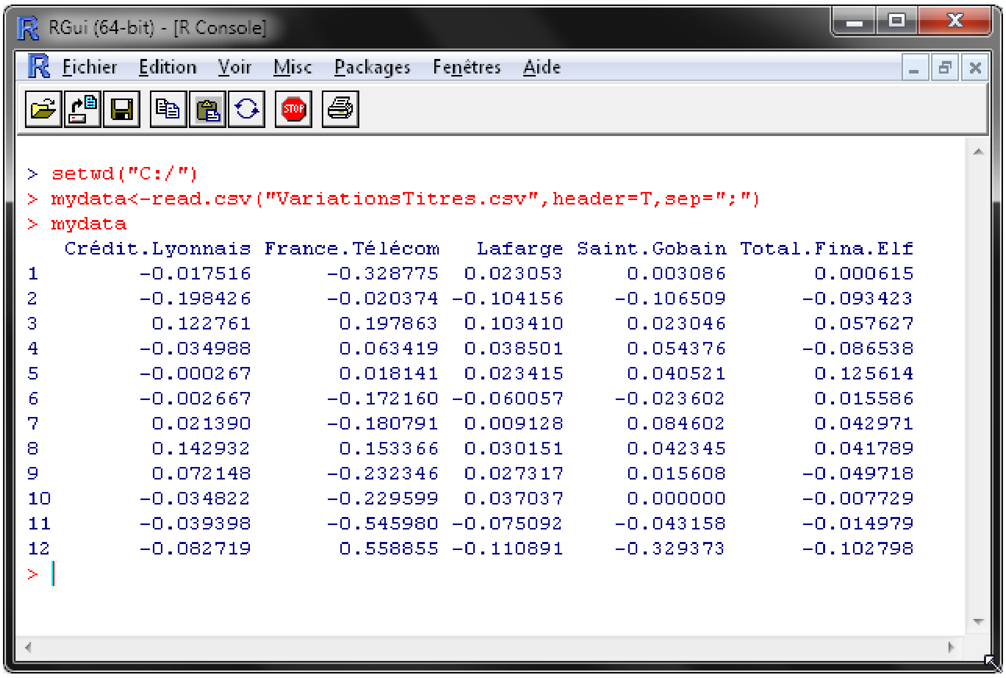


Exemple 39.: Exporter pour les forums

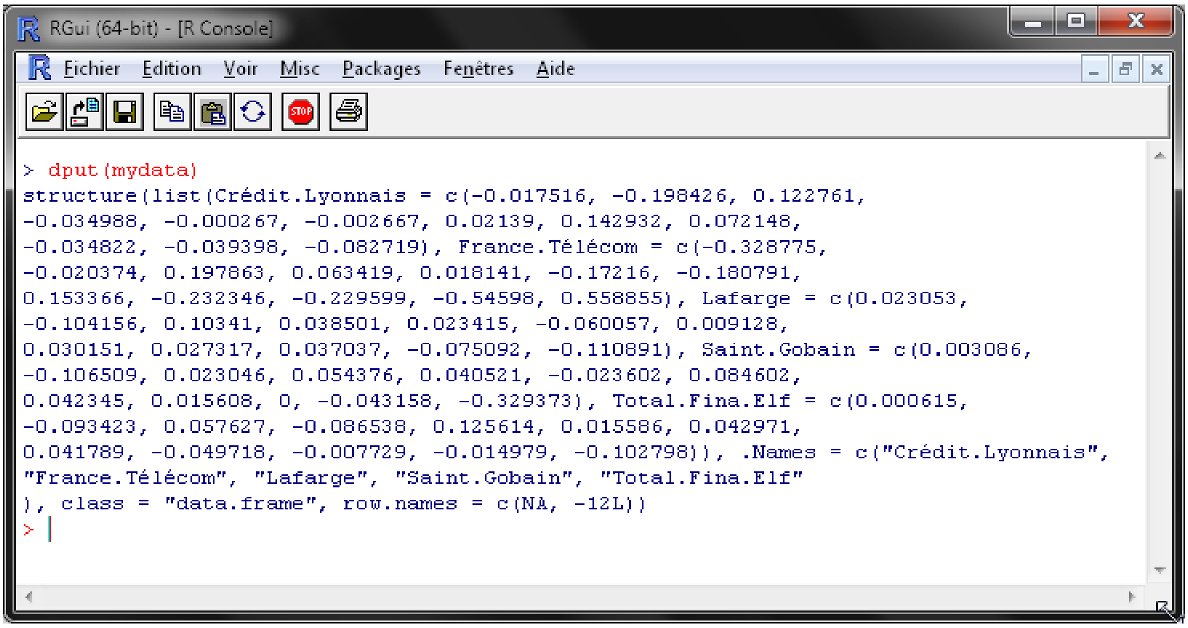
R 3.0.2

Souvent les étudiants demandent comment ils peuvent rapidement copier/coller un jeu reproductible de leurs données de travail sur un forum pour bien montrer où se situe leur problème avec le jeu de données réelles.

Considérons alors l'import de données suivant:



Pour communiquer rapidement ce petit jeu de données sur un forum, vous pouvez alors utiliser la commande **dput( )** :



et ensuite il n'y a qu'à copier/coller sur le forum de votre choix!

Exemple 40.: Fusionner des fichiers *.csv "un par un"

R 3.0.2

Le but ici va être de voir une manipulation fréquente dans les laboratoires qui consiste à fusionner des fichiers découlant typiquement de LabView à l'aide de la commande **rbind( )** .

Considérons au moins deux fichiers *.csv:

Ventes_01.csv - Bloc-notes

Fichier Edition Format Affichage ?

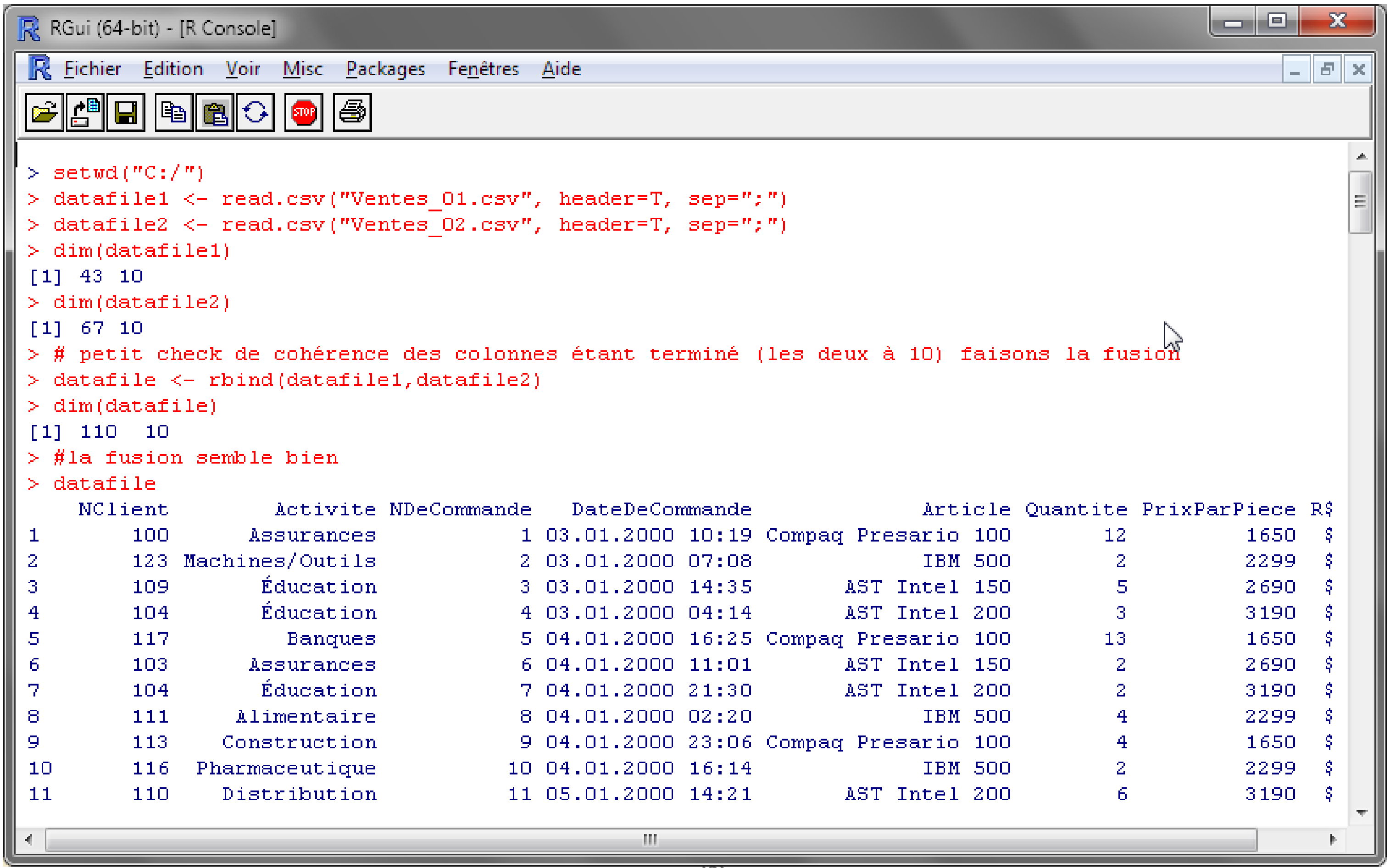
```
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Rab
100;Assurances;1;03.01.2000 10:19;Compaq Presario 100;12;1650;1.50%;19503;Ou
123;Machines/Outils;2;03.01.2000 07:08;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000 14:35;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000 04:14;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000 16:25;Compaq Presario 100;13;1650;1.50%;21128.25;Ou
103;Assurances;6;04.01.2000 11:01;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000 21:30;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000 02:20;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000 23:06;Compaq Presario 100;4;1650;0.00%;6600;Ou
116;Pharmaceutique;10;04.01.2000 16:14;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000 14:21;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000 03:29;Compaq Presario 100;6;1650;1.50%;975
123;Machines/Outils;13;05.01.2000 16:22;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000 16:09;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000 01:47;Compaq Presario 100;8;1650;1.50%;13002;
124;Éducation;16;05.01.2000 21:13;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000 01:48;Compaq Presario 100;11;1650;1.50%;17877.75
```


Ventes_02.csv - Bloc-notes

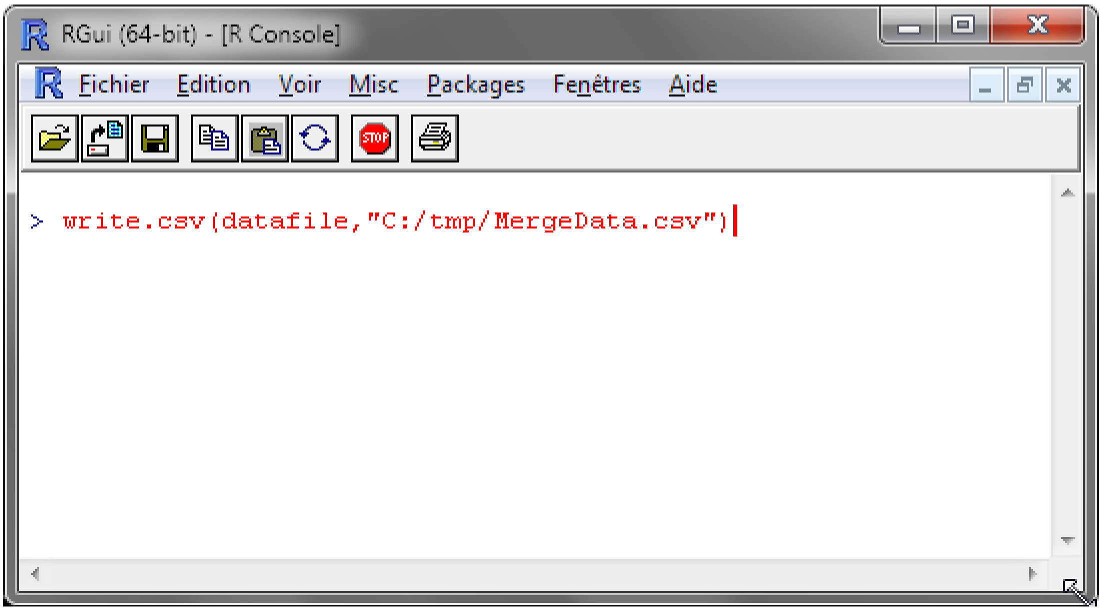
Fichier Edition Format Affichage ?

```
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Rab
113;Construction;43;12.01.2000 09:33;IBM 500;6;2299;1.50%;13587.09;Oui
127;Alimentaire;44;12.01.2000 20:11;AST Intel 200;2;3190;0.00%;6380;Oui
100;Assurances;45;12.01.2000 17:05;AST Intel 200;4;3190;0.00%;12760;Oui
109;Éducation;46;13.01.2000 11:12;AST Intel 150;1;2690;0.00%;2690;Non
113;Construction;47;13.01.2000 02:28;IBM 500;11;2299;1.50%;24909.665;Oui
114;Distribution;48;13.01.2000 03:13;Compaq Presario 100;8;1650;1.50%;13002;
122;Distribution;49;13.01.2000 14:28;AST Intel 200;7;3190;1.50%;21995.05;Oui
100;Assurances;50;13.01.2000 11:02;AST Intel 150;13;2690;1.50%;34445.45;Oui
119;Distribution;51;14.01.2000 08:33;IBM 500;2;2299;0.00%;4598;Oui
102;Machines/Outils;52;14.01.2000 02:22;Compaq Presario 100;5;1650;0.00%;825
112;Machines/Outils;53;17.01.2000 23:03;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;54;17.01.2000 12:43;IBM 500;5;2299;0.00%;11495;Oui
103;Assurances;55;18.01.2000 03:13;IBM 500;7;2299;1.50%;15851.605;Oui
109;Éducation;56;18.01.2000 16:51;AST Intel 150;21;2690;3.00%;54795.3;Oui
110;Distribution;57;18.01.2000 18:56;AST Intel 200;7;3190;1.50%;21995.05;Oui
127;Alimentaire;58;18.01.2000 20:20;Compaq Presario 100;7;1650;1.50%;11376.7
105;Banques;59;18.01.2000 04:05;IBM 500;9;2299;1.50%;20380.635;Oui
```

Nous allons les fusionner dans R tout en sachant qu'au delà de plusieurs centaines de millions de lignes R n'arrive plus à gérer la fusion de fichier *.csv (je montrerai une technique pour cela après).

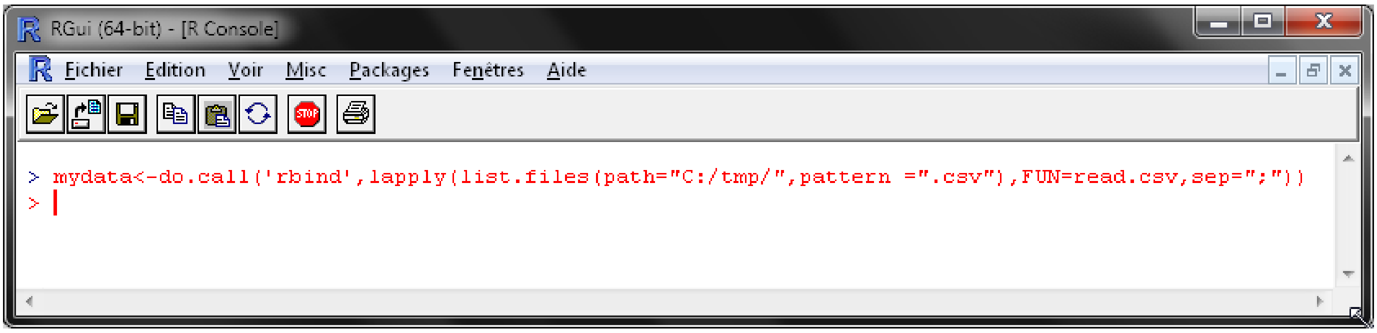


et ensuite nous pouvons exporter le tout:



Vous pourrez alors vérifier dans ce fichier d'export que bien que nous ayons fusionné le deuxième fichier aussi avec la ligne de titre, celle-ci a été éliminée a priori par R dès la fusion dans R avec la commande **rbind( )** .

Pour fusionner plusieurs fichiers d'un dossier sans avoir à spécifier leur nom, une possibilité est:



Si les fichiers sont trop gros (centaines de millions de lignes chacun) il vaut mieux si vous travaillez sur le système d'exploitation Microsoft Windows utiliser le shell MS DOS avec la commande suivante qui est extrêmement efficace (je l'ai testée jusqu'à obtenir un fichier final de 11GB en moins de 3 minutes):

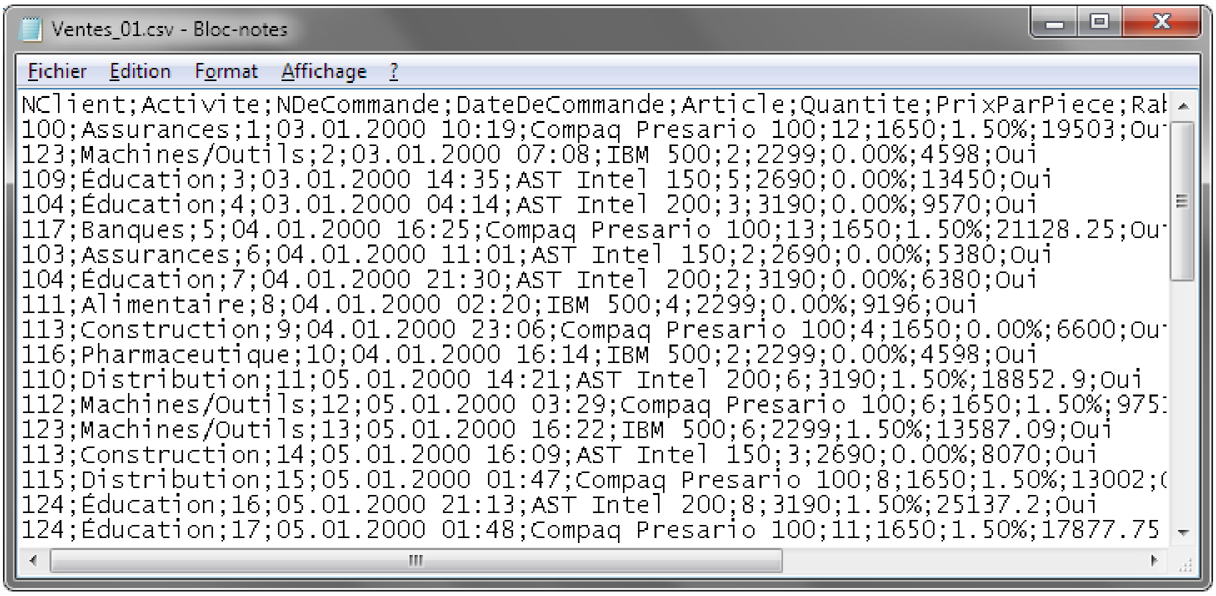


Mais attention au petit piège!!! Il faudra au préalable enlever la ligne de titre à tous les fichiers excepté le premier (sinon quoi la ligne de titre va se répéter plusieurs fois dans le fichier final).

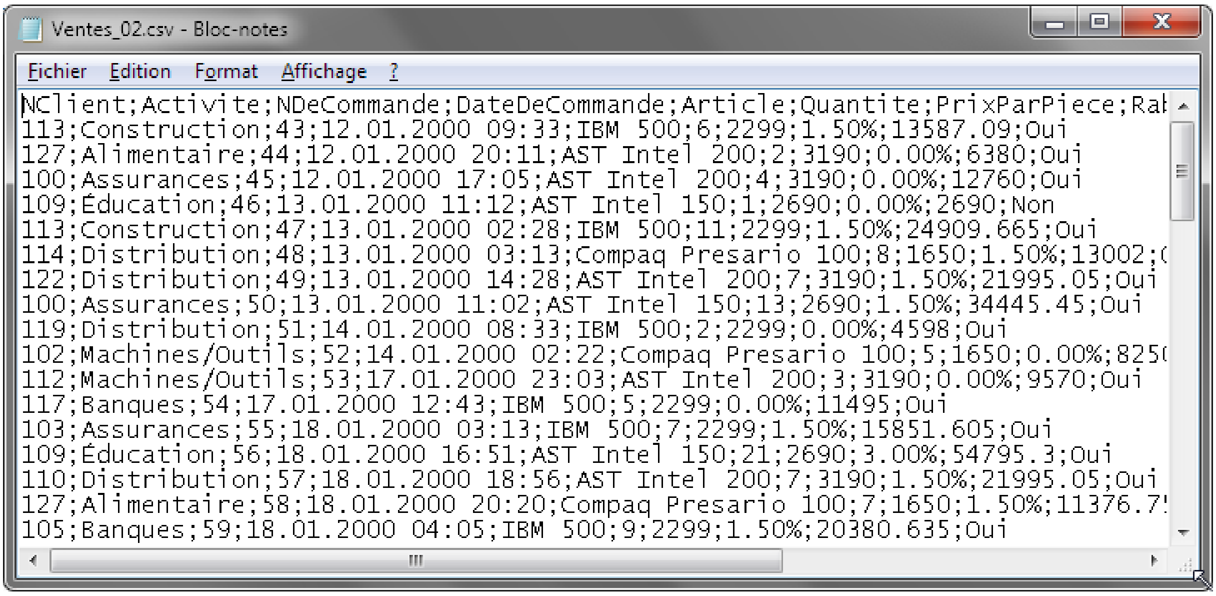
Exemple 41.: Fusionner des fichiers *.csv d'un dossier

R 3.4.5

Dans de nombreuses situations il est utile de savoir fusionner tous les fichiers se trouvant dans un dossier. Considérons donc un dossier contenant à nouveau nos deux fichiers:



```
Ventes_01.csv - Bloc-notes
Fichier Edition Format Affichage ?
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Ral
100;Assurances;1;03.01.2000 10:19;Compaq Presario 100;12;1650;1.50%;19503;Ou
123;Machines/Outils;2;03.01.2000 07:08;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000 14:35;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000 04:14;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000 16:25;Compaq Presario 100;13;1650;1.50%;21128.25;Ou
103;Assurances;6;04.01.2000 11:01;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000 21:30;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000 02:20;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000 23:06;Compaq Presario 100;4;1650;0.00%;6600;Ou
116;Pharmaceutique;10;04.01.2000 16:14;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000 14:21;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000 03:29;Compaq Presario 100;6;1650;1.50%;975
123;Machines/Outils;13;05.01.2000 16:22;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000 16:09;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000 01:47;Compaq Presario 100;8;1650;1.50%;13002;(
124;Éducation;16;05.01.2000 21:13;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000 01:48;Compaq Presario 100;11;1650;1.50%;17877.75
```



```
Ventes_02.csv - Bloc-notes
Fichier Edition Format Affichage ?
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Ral
113;Construction;43;12.01.2000 09:33;IBM 500;6;2299;1.50%;13587.09;Oui
127;Alimentaire;44;12.01.2000 20:11;AST Intel 200;2;3190;0.00%;6380;Oui
100;Assurances;45;12.01.2000 17:05;AST Intel 200;4;3190;0.00%;12760;Oui
109;Éducation;46;13.01.2000 11:12;AST Intel 150;1;2690;0.00%;2690;Non
113;Construction;47;13.01.2000 02:28;IBM 500;11;2299;1.50%;24909.665;Oui
114;Distribution;48;13.01.2000 03:13;Compaq Presario 100;8;1650;1.50%;13002;(
122;Distribution;49;13.01.2000 14:28;AST Intel 200;7;3190;1.50%;21995.05;Oui
100;Assurances;50;13.01.2000 11:02;AST Intel 150;13;2690;1.50%;34445.45;Oui
119;Distribution;51;14.01.2000 08:33;IBM 500;2;2299;0.00%;4598;Oui
102;Machines/Outils;52;14.01.2000 02:22;Compaq Presario 100;5;1650;0.00%;825(
112;Machines/Outils;53;17.01.2000 23:03;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;54;17.01.2000 12:43;IBM 500;5;2299;0.00%;11495;Oui
103;Assurances;55;18.01.2000 03:13;IBM 500;7;2299;1.50%;15851.605;Oui
109;Éducation;56;18.01.2000 16:51;AST Intel 150;21;2690;3.00%;54795.3;Oui
110;Distribution;57;18.01.2000 18:56;AST Intel 200;7;3190;1.50%;21995.05;Oui
127;Alimentaire;58;18.01.2000 20:20;Compaq Presario 100;7;1650;1.50%;11376.7!
105;Banques;59;18.01.2000 04:05;IBM 500;9;2299;1.50%;20380.635;Oui
```

Pour fusionner cela, nous écrirons à l'aide de **file.list( )** :


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("data.table")
> files<-list.files("C:/tmp/Ventes/",pattern="*.csv",full.name = TRUE)
> files
[1] "C:/tmp/Ventes/Ventes_01.csv" "C:/tmp/Ventes/Ventes_02.csv"
> merge<-do.call(rbind,lapply(files,read.csv))
> nrow(merge)
[1] 109
>

Exemple 42.: R et LaTeX

R 3.1.2

Exporter un dataframe en LaTeX

Tous ceux qui travaillent avec TeX/LaTeX savent comme il est pénible de transformer une table de données en table LaTeX. Pour ceux qui travaillent avec R il existe le package **xtable** qui permet d'automatiser l'écriture du code à l'aide de la fonction **xtable( )** :

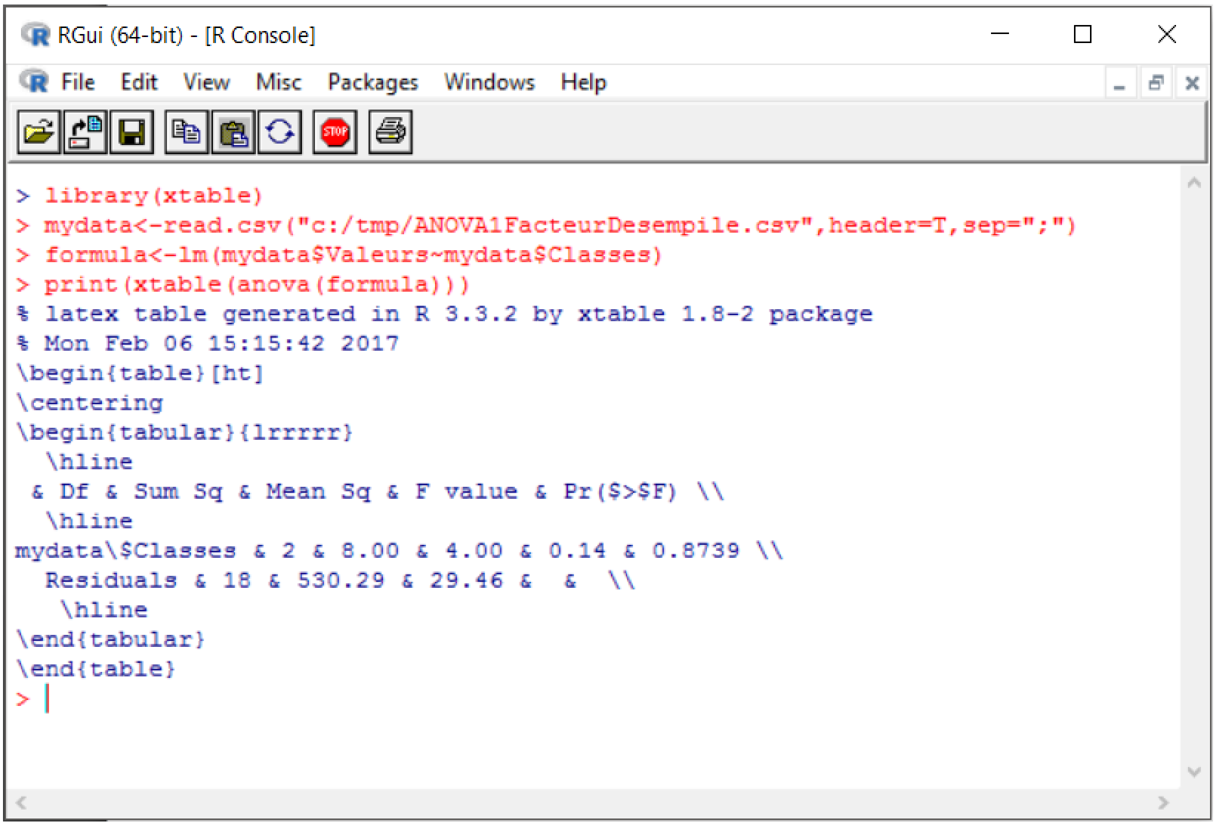
```
RGui (32-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library(xtable)
> mydata<-read.csv("C:/tmp/RtoLaTeX.csv",header=T,sep=";")
> print(xtable(mydata[,-1], display = c("f","f","f","f","f"), digits = 2),
+ format.args = list(big.mark = "'", decimal.mark = "."))
% latex table generated in R 3.2.1 by xtable 1.7-4 package
% Thu Jul 09 20:23:08 2015
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
  \hline
 & Couts & CoutA & CoutB & CoutC \\
  \hline
1 & 44'439.00 & 515.00 & 541.00 & 928.00 \\
  2 & 43'936.00 & 929.00 & 692.00 & 711.00 \\
  3 & 44'464.00 & 800.00 & 710.00 & 824.00 \\
  4 & 41'533.00 & 979.00 & 675.00 & 758.00 \\
  5 & 46'343.00 & 1'165.00 & 1'147.00 & 635.00 \\
  6 & 44'922.00 & 651.00 & 939.00 & 901.00 \\
  7 & 43'203.00 & 847.00 & 755.00 & 580.00 \\
  8 & 43'000.00 & 942.00 & 908.00 & 589.00 \\
  9 & 40'967.00 & 630.00 & 738.00 & 682.00 \\
  10 & 48'582.00 & 1'113.00 & 1'175.00 & 1'050.00 \\
  11 & 45'003.00 & 1'086.00 & 1'075.00 & 984.00 \\
  12 & 44'303.00 & 843.00 & 640.00 & 828.00 \\
  13 & 42'070.00 & 500.00 & 752.00 & 708.00 \\
  14 & 44'353.00 & 813.00 & 989.00 & 804.00 \\
  15 & 45'968.00 & 1'190.00 & 823.00 & 904.00 \\
  16 & 47'781.00 & 1'200.00 & 1'108.00 & 1'120.00 \\
  17 & 43'202.00 & 731.00 & 590.00 & 1'065.00 \\
  18 & 44'074.00 & 1'089.00 & 607.00 & 1'132.00 \\
  19 & 44'610.00 & 786.00 & 513.00 & 839.00 \\
   \hline
\end{tabular}
\end{table}
> 
```

Il ne reste plus qu'à copier coller!

Exporter une analyse statistique en LaTeX

De nombreuses analyses statistiques peuvent être automatiquement générées et exportées pour LaTeX. Voyons un exemple:



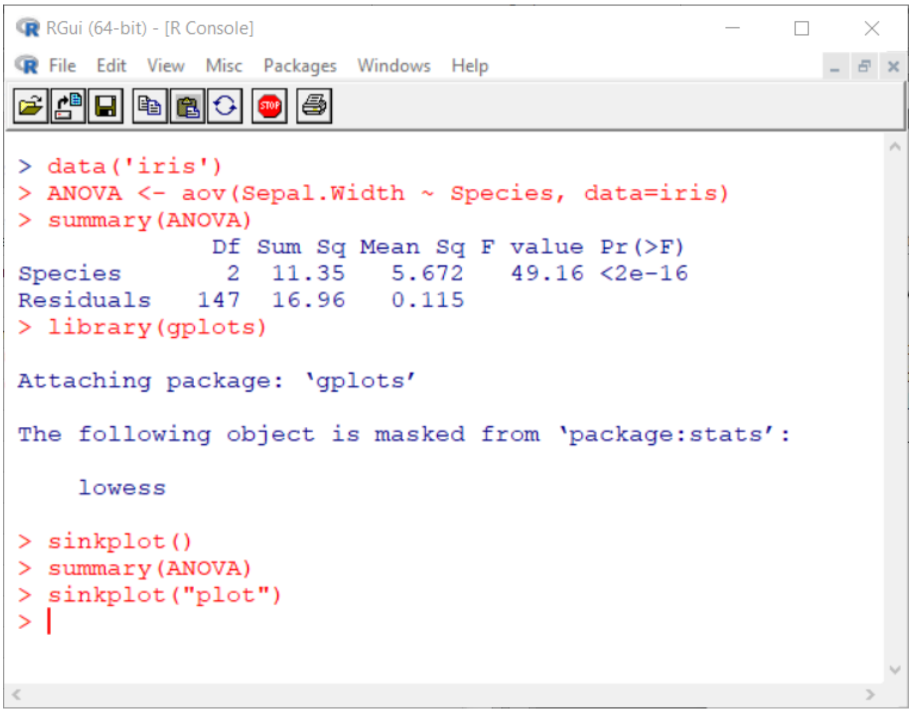
Ce qui donne compilé dans LaTeX:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
mydata$Classes	2	8.00	4.00	0.14	0.8739
Residuals	18	530.29	29.46		

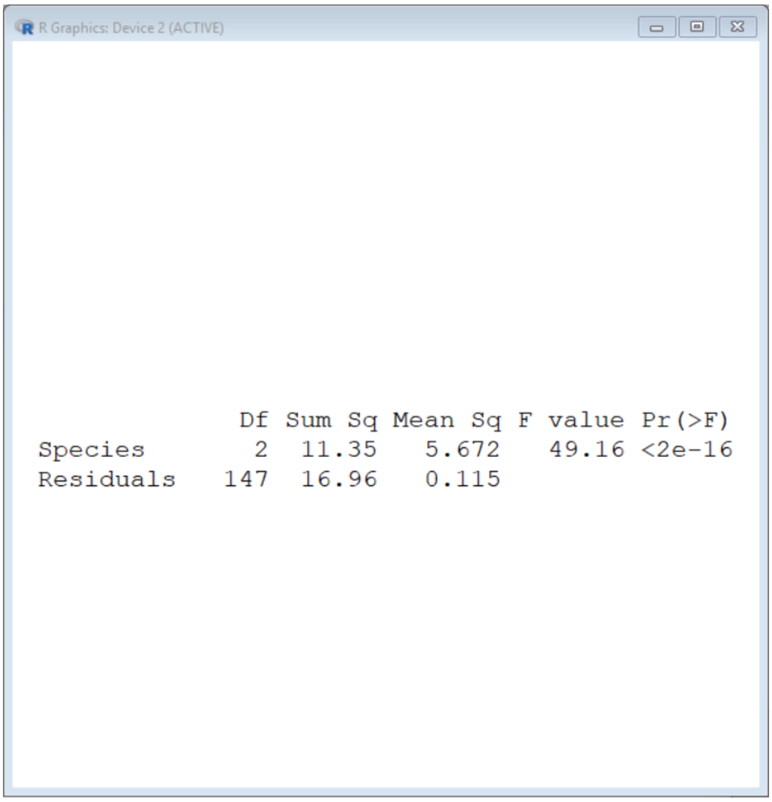
Avouons que c'est pas mal ;-)

Exporter une analyse statistique dans une fenêtre graphique

Il est très utile dans le cadre de la génération de graphique, d'intégrer parfois des analyses statistiques poussées (ou simples!) ou de le faire pour ceux qui veulent imprimer sur du papier ou sur un PDF un résultat d'analyse avec le script plutôt que le script tout seul (comme le fait le menu **File/Print**). Voyons un cas au plus simple (très utile pour l'usage de Power BI) à l'aide du package **gplots** et de sa fonction **sinkplot( )** :



Ce qui donne:



	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Species	2	11.35	5.672	49.16	<2e-16
Residuals	147	16.96	0.115		

Figure 25 Graph d'un tableau statistique

Exporter l'équation d'un modèle statistique en LaTeX

Pour certains modèles statistiques, extraire l'équation du modèle automatiquement permet de gagner du temps lors de la rédaction d'articles scientifiques. Il en est de même lors de l'usage de R Markdown où il peut être utile de générer le code LaTeX des modèles.

Pour cela il existe un package nommé **equatiomatic** et sa fonction extract_eq( ) qui fait ce travail pour un petit nombre de modèles dont voici la liste au jour du mois d'octobre 2020:

Model	Packages/Functions
linear regression	`stats::lm`
logistic regression	`stats::glm(family = binomial(link = 'logit'))`
probit regression	`stats::glm(family = binomial(link = 'probit'))`
ordinal logistic regression	`MASS::polr(method = 'logistic')`; `ordinal::clm(link = 'logit')`
ordinal probit regression	`MASS::polr(method = 'probit')`; `ordinal::clm(link = 'probit')`

Voyons un premier exemple:

```
> library("equatiomatic")
> fit <- lm(mpg ~ cyl + disp, mtcars)
> extract_eq(fit)
$$
\operatorname{mpg} = \alpha + \beta_{1}(\operatorname{cyl}) + \beta_{2}(\operatorname{disp}) + \epsilon
$$>
```

Ce qui donne:

$$
\operatorname{mpg} = \alpha + \beta_{1}(\operatorname{cyl}) + \beta_{2}(\operatorname{disp}) + \epsilon
$$

ou encore:

```
$$> extract_eq(fit, use_coefs = TRUE)
$$
\operatorname{mpg} = 34.66 - 1.59(\operatorname{cyl}) - 0.02(\operatorname{disp}) + \epsilon
$$>
```

Ce qui donne:

$$\mathrm{mpg} = 34.66 - 1.59(\mathrm{cyl}) - 0.02(\mathrm{disp}) + \epsilon$$

Ou encore:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

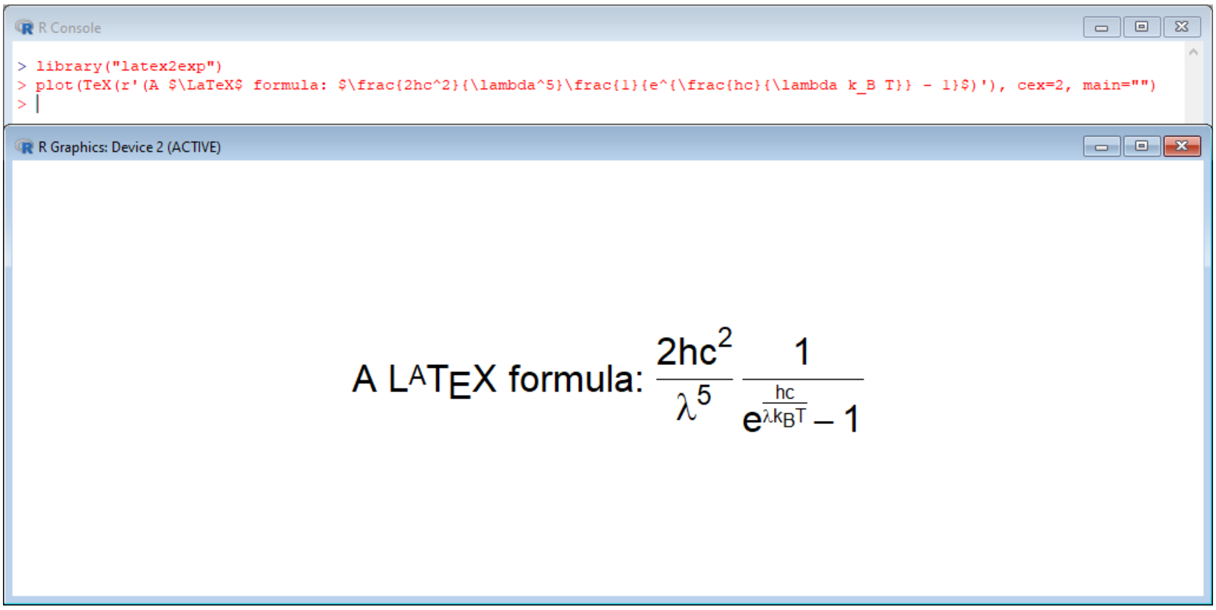
> library("lme4")
> str(sleepstudy)
'data.frame':   180 obs. of  3 variables:
 $ Reaction: num  250 259 251 321 357 ...
 $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
 $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1 1 1 1 1 1 1 1 1 ...
> fit <- lmer(Reaction~Days + (1|Subject), sleepstudy)
> extract_eq(fit, use_coefs=TRUE, wrap = TRUE)
```

Ce qui donne:

$$\begin{aligned} \mathrm{Reaction} &\sim N\left(\alpha_{j[i]} + \beta_1(\mathrm{Days}), \sigma^2\right) \\ \alpha_j &\sim N\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right), \text{for Subject } j = 1, \dots, J \end{aligned}$$

Plotter une équation LaTeX

Ceci est facilement faisable avec le package latex2exp de la manière suivante:



Importer une sortie R dans LaTeX

Considérons maintenant la sortie R suivante qui a été copiée dans un fichier * .txt:

ROutput.txt - Notepad

File Edit Format View Help

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model  : ARFIMA(0,0,5)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
ma1     0.000000          NA       NA       NA
ma2     0.000000          NA       NA       NA
ma3     0.000000          NA       NA       NA
ma4     0.000000          NA       NA       NA
ma5    -0.041598    0.020328  -2.0463 0.040727
omega   0.000006    0.000001   4.2148 0.000025
alpha1  0.092063    0.011337   8.1205 0.000000
beta1   0.892993    0.012696  70.3366 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ma1     0.000000          NA       NA       NA
ma2     0.000000          NA       NA       NA
ma3     0.000000          NA       NA       NA
ma4     0.000000          NA       NA       NA
ma5    -0.041598    0.020374  -2.0417 0.041178
omega   0.000006    0.000002   3.0914 0.001992
alpha1  0.092063    0.018209   5.0560 0.000000
beta1   0.892993    0.019863  44.9586 0.000000

LogLikelihood : 6757.949

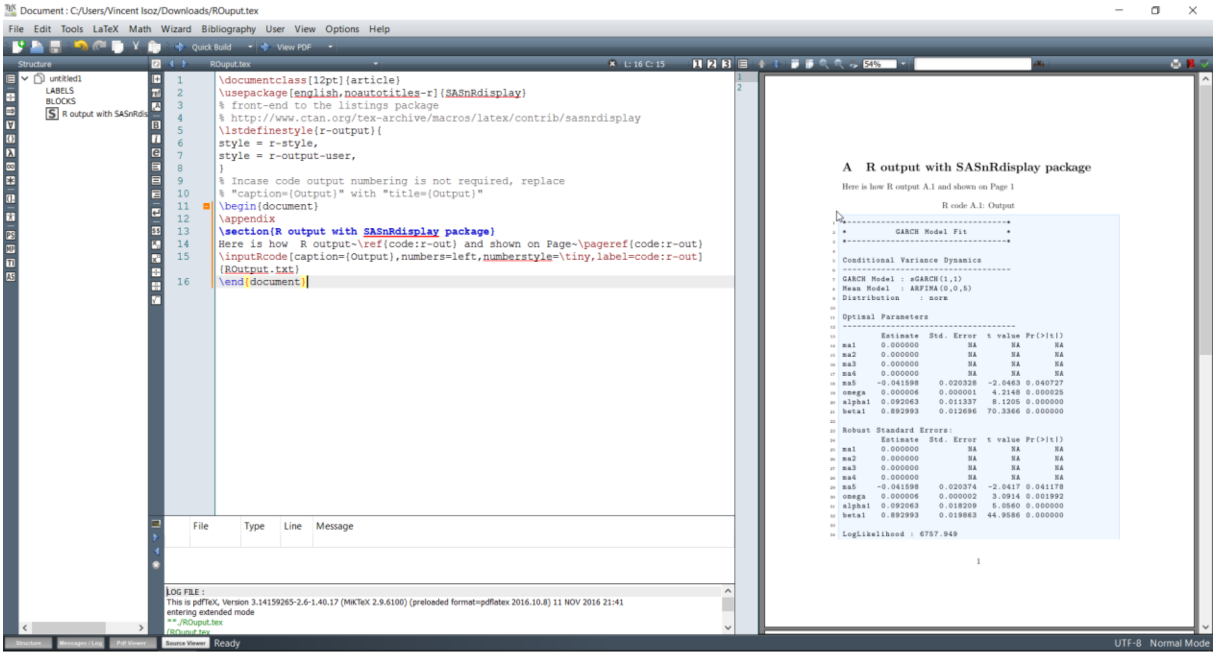
Information Criteria
------------------------------------

Akaike       -5.2356
Bayes        -5.2265
Shibata      -5.2356
Hannan-Quinn -5.2323

Q-Statistics on Standardized Residuals
------------------------------------
```

La question est de savoir s'il existe un moyen rapide et pas trop salissant de le rendre dans un document LaTeX?

Nous utilisons TeXMaker 4.1.2 avec:



où il est fait usage du package **SASnRdisplay**:

```
\documentclass[12pt]{article}
\usepackage[english,noautotitles-r]{SASnRdisplay}
% front-end to the listings package
% http://www.ctan.org/tex-archive/macros/latex/contrib/sasnrdisplay
\lstdefinestyle{r-output}{
style = r-style,
style = r-output-user,
}
% Incase code output numbering is not required, replace
% "caption={Output}" with "title={Output}"
\begin{document}
\appendix
\section{R output with SASnRdisplay package}
Here is how  R output~\ref{code:r-out} and shown on Page~\pageref{code:r-out}
\inputRcode[caption={Output},numbers=left,numberstyle=\tiny,label=code:r-out]
{ROutput.txt}
\end{document}
```

Ce qui donne donc:

A R output with SASnRdisplay package

Here is how R output A.1 and shown on Page 1

R code A.1: Output

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model  : ARFIMA(0,0,5)
Distribution    : norm

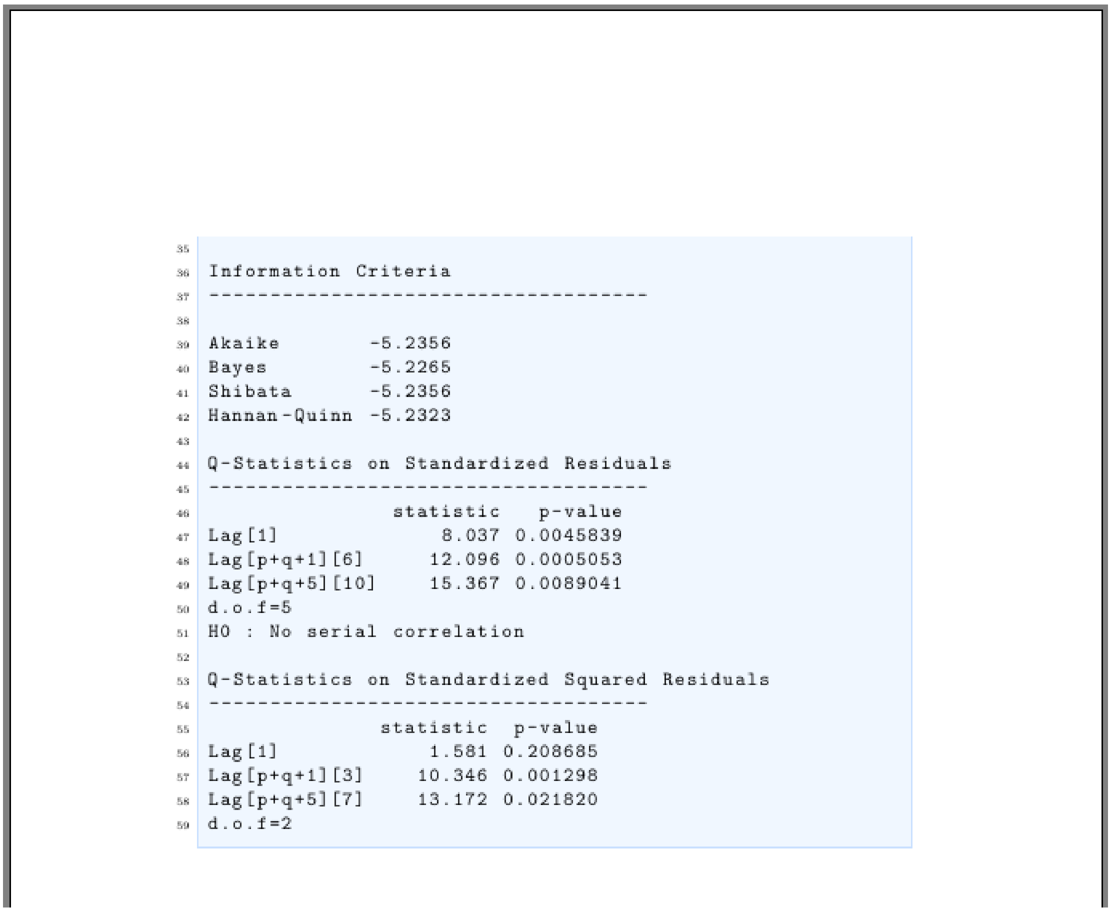
Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
ma1     0.000000          NA       NA       NA
ma2     0.000000          NA       NA       NA
ma3     0.000000          NA       NA       NA
ma4     0.000000          NA       NA       NA
ma5    -0.041598    0.020328  -2.0463 0.040727
omega   0.000006    0.000001   4.2148 0.000025
alpha1  0.092063    0.011337   8.1205 0.000000
beta1   0.892993    0.012696  70.3366 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ma1     0.000000          NA       NA       NA
ma2     0.000000          NA       NA       NA
ma3     0.000000          NA       NA       NA
ma4     0.000000          NA       NA       NA
ma5    -0.041598    0.020374  -2.0417 0.041178
omega   0.000006    0.000002   3.0914 0.001992
alpha1  0.092063    0.018209   5.0560 0.000000
beta1   0.892993    0.019863  44.9586 0.000000

LogLikelihood : 6757.949
```

1

et sur la seconde page:

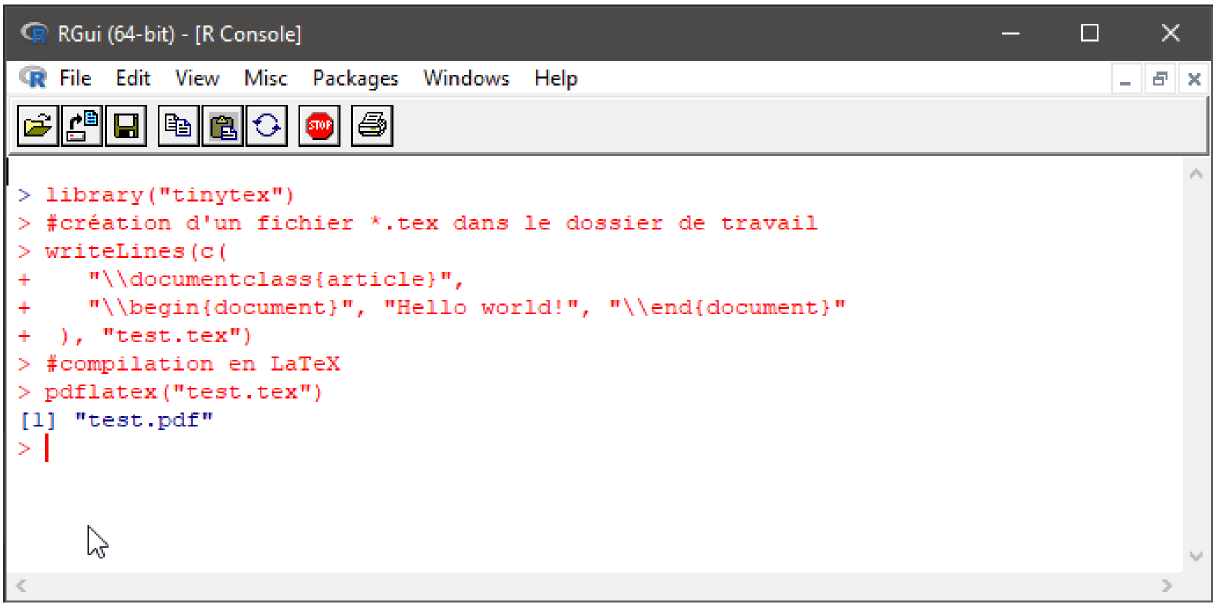


```
35
36 Information Criteria
37 ------------------------------------
38
39 Akaike       -5.2356
40 Bayes        -5.2265
41 Shibata      -5.2356
42 Hannan-Quinn -5.2323
43
44 Q-Statistics on Standardized Residuals
45 ------------------------------------
46                statistic   p-value
47 Lag[1]              8.037 0.0045839
48 Lag[p+q+1][6]      12.096 0.0005053
49 Lag[p+q+5][10]     15.367 0.0089041
50 d.o.f=5
51 H0 : No serial correlation
52
53 Q-Statistics on Standardized Squared Residuals
54 ------------------------------------
55               statistic  p-value
56 Lag[1]             1.581 0.208685
57 Lag[p+q+1][3]     10.346 0.001298
58 Lag[p+q+5][7]     13.172 0.021820
59 d.o.f=2
```

dont la qualité est comme à l'habitude avec LaTeX: remarquable!!!

Compiler une sortie R LaTeX en un PDF directement depuis R

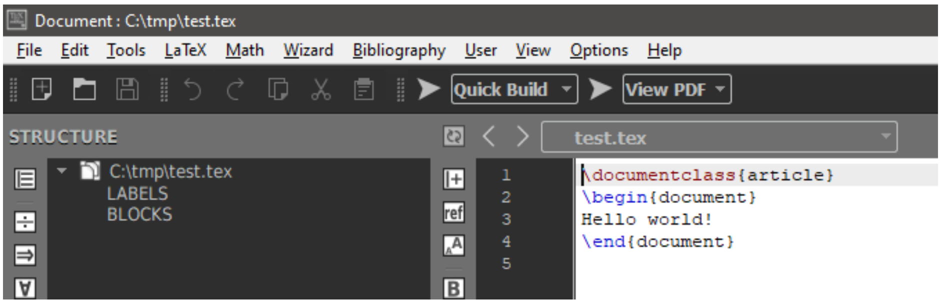
Si MikTeX est installé sur votre ordinateur, alors vous pouvez compiler un fichier *.tex directement depuis R à l'aide du package **tinytex** et sa fonction pdflatex( ) comme le montre l'exemple ci-dessous:



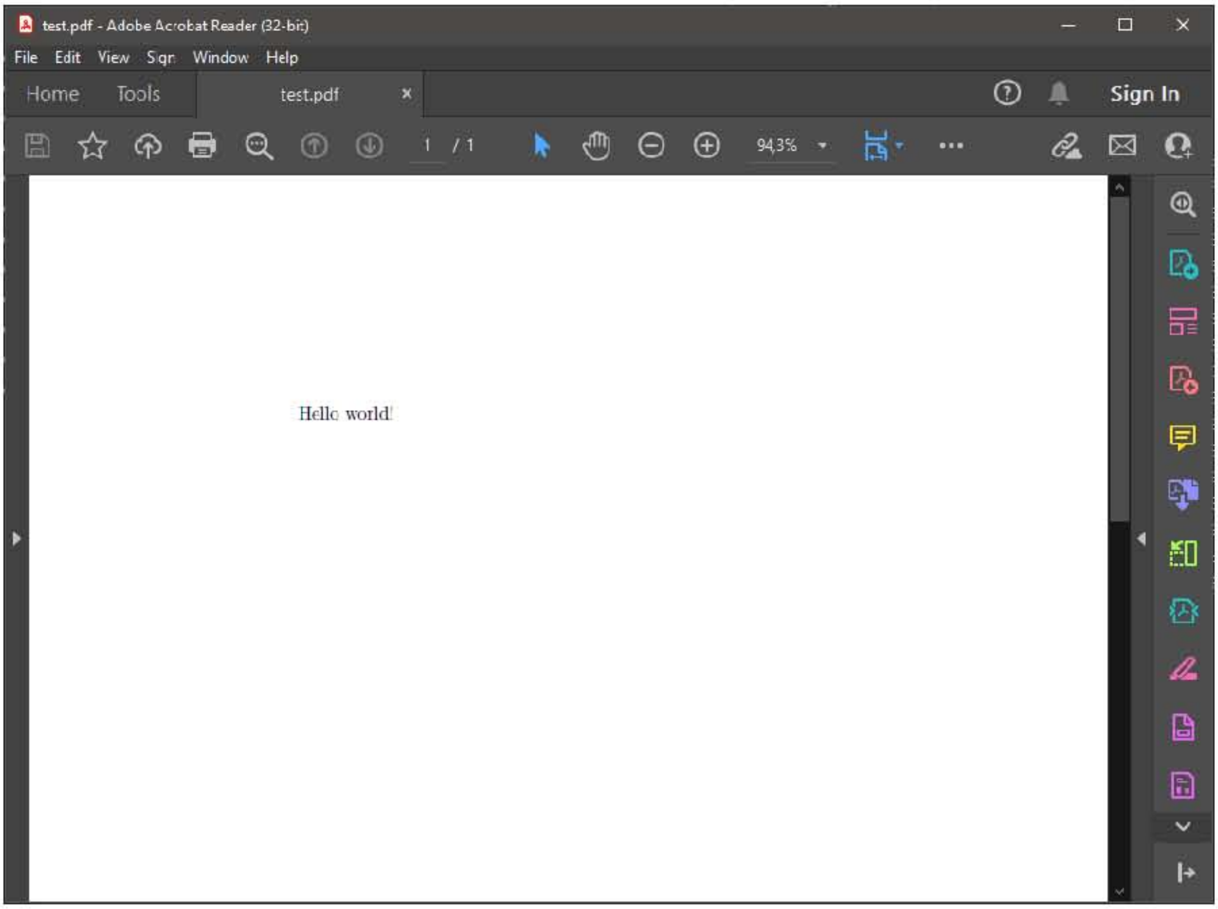
Ce qui donne:



Le fichier text.tex contient comme attendu:



Et le pdf lui contient bien:


test.pdf - Adobe Acrobat Reader (32-bit)
File Edit View Sign Window Help
Home Tools test.pdf
Sign In
1 / 1
94,3%
Hello world!

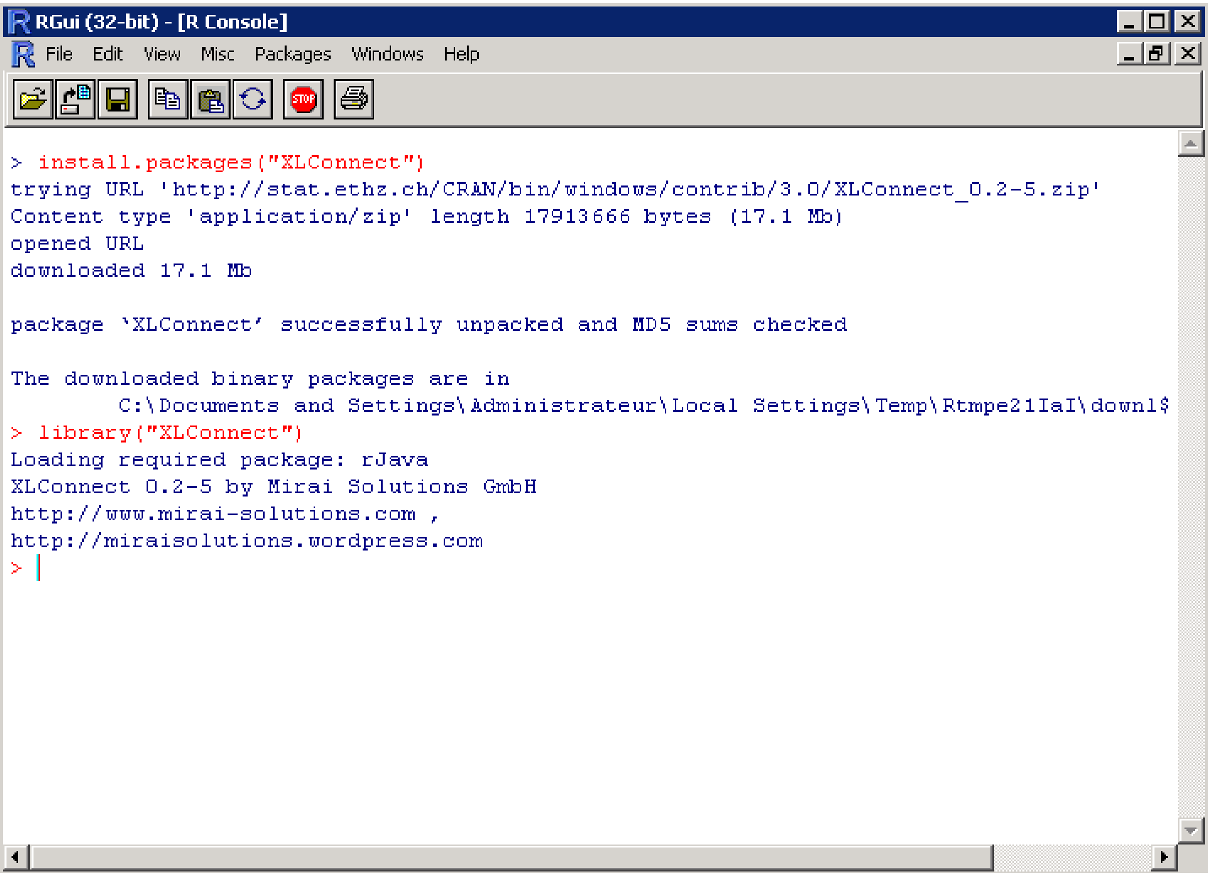
Exemples 43.: Importer/Exporter au format Microsoft Excel *.xlsx (2007-2019)

R 3.0.2

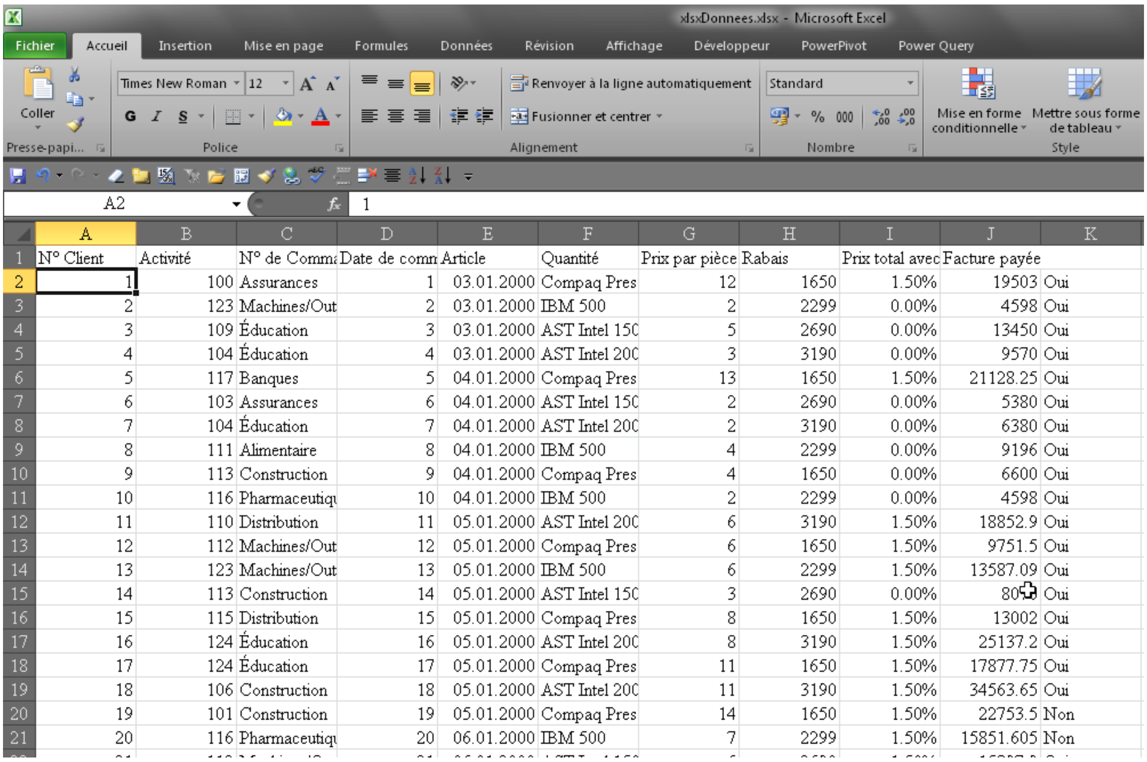
De nombreux packages de lecture/écriture XLSX ne marchent pas actuellement avec la version x64 de R. Il faut exécuter la version x32 et de plus avoir la version x32 du Runtime Java d'installé. De plus, parfois l'installation de ce package cause beaucoup d'ennuis concernant le chargement de la dépendance avec le package rJava. Si c'est le cas, on doit s'assurer de définir les chemins d'accès aux dossiers bin ("C:\Program Files\Java\jre 1.8.0_71\bin") et server ("C:\Program Files\Java\jre 1.8.0_71\server") de Java comme variables d'environnement et redémarrer RStudio ou R Visual Studio (voir plus bas). Ou simplement, comme ici, utiliser la version 32 bits de R permet de ne rien avoir à modifier comme paramètres dans le système d'exploitation!

En 32 bits

Nous installons maintenant le package **XLConnect** en apprenant directement au passage comment le faire avec la commande **install.packages**( ) au lieu de passer par un menu:



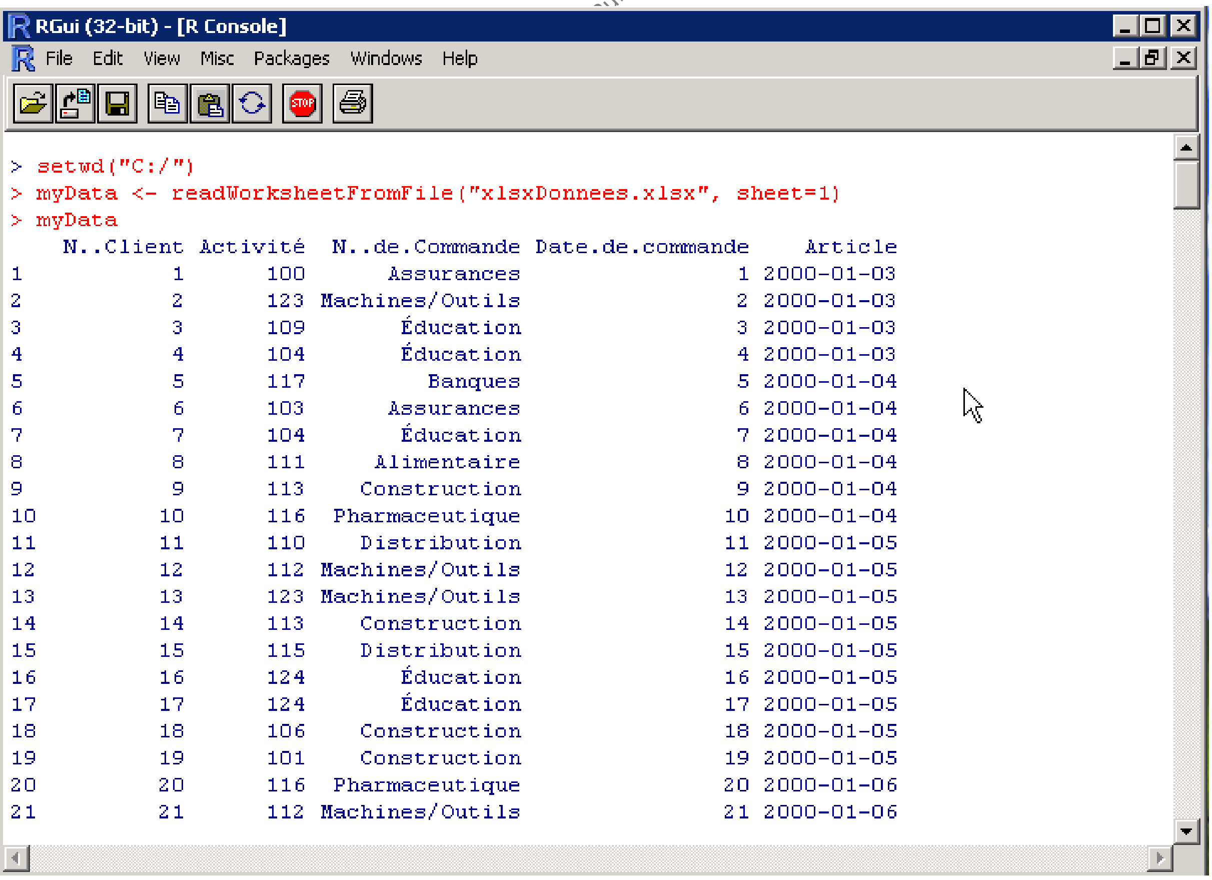
Ensuite le but est d'importer le contenu du fichier suivant *xlsxDonnees.xlsx* dans un data frame:



	A	B	C	D	E	F	G	H	I	J	K
1	N° Client	Activité	N° de Comma	Date de comm	Article	Quantité	Prix par pièce	Rabais	Prix total avec	Facture payée	
2	1	100	Assurances	1	03.01.2000	Compaq Pres	12	1650	1.50%	19503	Oui
3	2	123	Machines/Out	2	03.01.2000	IBM 500	2	2299	0.00%	4598	Oui
4	3	109	Éducation	3	03.01.2000	AST Intel 150	5	2690	0.00%	13450	Oui
5	4	104	Éducation	4	03.01.2000	AST Intel 200	3	3190	0.00%	9570	Oui
6	5	117	Banques	5	04.01.2000	Compaq Pres	13	1650	1.50%	21128.25	Oui
7	6	103	Assurances	6	04.01.2000	AST Intel 150	2	2690	0.00%	5380	Oui
8	7	104	Éducation	7	04.01.2000	AST Intel 200	2	3190	0.00%	6380	Oui
9	8	111	Alimentaire	8	04.01.2000	IBM 500	4	2299	0.00%	9196	Oui
10	9	113	Construction	9	04.01.2000	Compaq Pres	4	1650	0.00%	6600	Oui
11	10	116	Pharmaceutiq	10	04.01.2000	IBM 500	2	2299	0.00%	4598	Oui
12	11	110	Distribution	11	05.01.2000	AST Intel 200	6	3190	1.50%	18852.9	Oui
13	12	112	Machines/Out	12	05.01.2000	Compaq Pres	6	1650	1.50%	9751.5	Oui
14	13	123	Machines/Out	13	05.01.2000	IBM 500	6	2299	1.50%	13587.09	Oui
15	14	113	Construction	14	05.01.2000	AST Intel 150	3	2690	0.00%	80[illegible]	Oui
16	15	115	Distribution	15	05.01.2000	Compaq Pres	8	1650	1.50%	13002	Oui
17	16	124	Éducation	16	05.01.2000	AST Intel 200	8	3190	1.50%	25137.2	Oui
18	17	124	Éducation	17	05.01.2000	Compaq Pres	11	1650	1.50%	17877.75	Oui
19	18	106	Construction	18	05.01.2000	AST Intel 200	11	3190	1.50%	34563.65	Oui
20	19	101	Construction	19	05.01.2000	Compaq Pres	14	1650	1.50%	22753.5	Non
21	20	116	Pharmaceutiq	20	06.01.2000	IBM 500	7	2299	1.50%	15851.605	Non

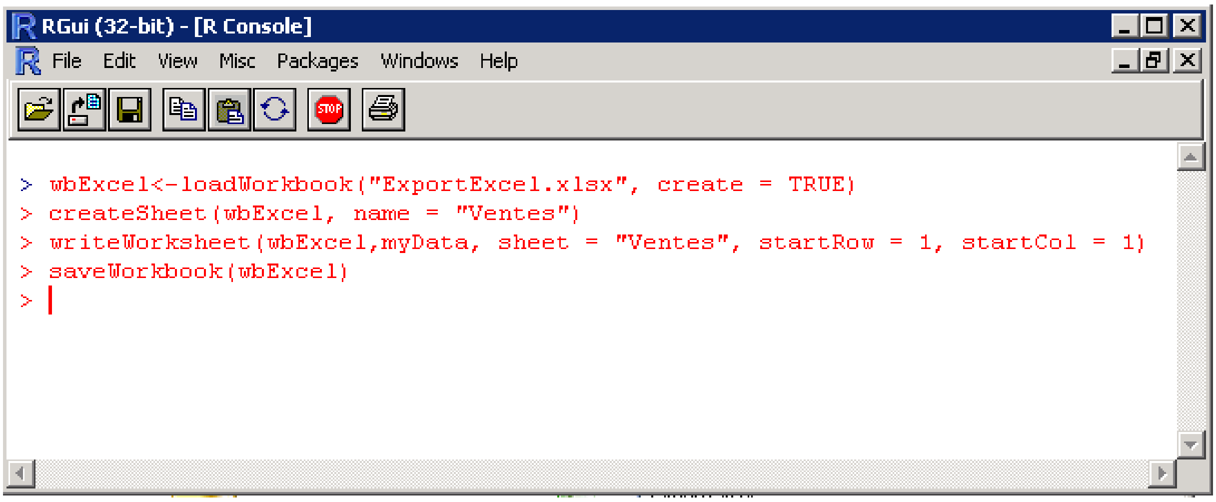
PS: Remarquez la dernière colonne n'a pas de titre (légende) pour voir quel va être le comportement du package dans cette situation.

Pour cela, nous allons utiliser la commande **readWorkhseetFromFile( )** comme ci-dessous:

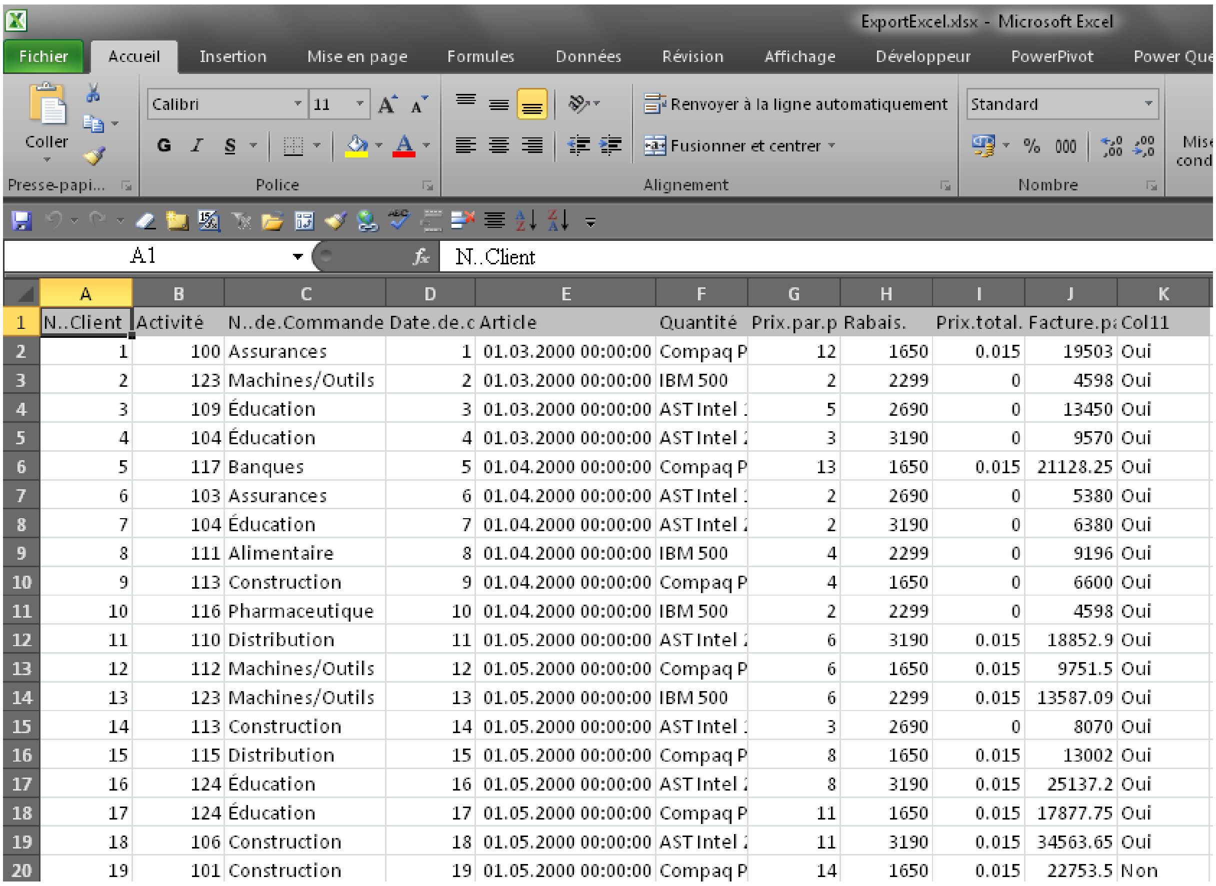


```
> setwd("C:/")
> myData <- readWorksheetFromFile("xlsxDonnees.xlsx", sheet=1)
> myData
   N..Client Activité  N..de.Commande Date.de.commande    Article
1          1      100      Assurances                1 2000-01-03
2          2      123 Machines/Outils                2 2000-01-03
3          3      109       Éducation                3 2000-01-03
4          4      104       Éducation                4 2000-01-03
5          5      117         Banques                5 2000-01-04
6          6      103      Assurances                6 2000-01-04
7          7      104       Éducation                7 2000-01-04
8          8      111     Alimentaire                8 2000-01-04
9          9      113    Construction                9 2000-01-04
10        10      116  Pharmaceutique               10 2000-01-04
11        11      110    Distribution               11 2000-01-05
12        12      112 Machines/Outils               12 2000-01-05
13        13      123 Machines/Outils               13 2000-01-05
14        14      113    Construction               14 2000-01-05
15        15      115    Distribution               15 2000-01-05
16        16      124       Éducation               16 2000-01-05
17        17      124       Éducation               17 2000-01-05
18        18      106    Construction               18 2000-01-05
19        19      101    Construction               19 2000-01-05
20        20      116  Pharmaceutique               20 2000-01-06
21        21      112 Machines/Outils               21 2000-01-06
```

et ensuite on exporte à nouveau le tout dans un fichier *.xlsx (bon évidemment dans la réalité on fera des manipulations dessus entre temps!) dans le dossier courant de R (qui pour rappel peut être changé par la commande **setwd( )**):



Ce qui nous donne:



	A	B	C	D	E	F	G	H	I	J	K
1	N..Client	Activité	N..de.Commande	Date.de.c	Article	Quantité	Prix.par.p	Rabais.	Prix.total.	Facture.p	Col11
2	1	100	Assurances	1	01.03.2000 00:00:00	Compaq P	12	1650	0.015	19503	Oui
3	2	123	Machines/Outils	2	01.03.2000 00:00:00	IBM 500	2	2299	0	4598	Oui
4	3	109	Éducation	3	01.03.2000 00:00:00	AST Intel :	5	2690	0	13450	Oui
5	4	104	Éducation	4	01.03.2000 00:00:00	AST Intel :	3	3190	0	9570	Oui
6	5	117	Banques	5	01.04.2000 00:00:00	Compaq P	13	1650	0.015	21128.25	Oui
7	6	103	Assurances	6	01.04.2000 00:00:00	AST Intel :	2	2690	0	5380	Oui
8	7	104	Éducation	7	01.04.2000 00:00:00	AST Intel :	2	3190	0	6380	Oui
9	8	111	Alimentaire	8	01.04.2000 00:00:00	IBM 500	4	2299	0	9196	Oui
10	9	113	Construction	9	01.04.2000 00:00:00	Compaq P	4	1650	0	6600	Oui
11	10	116	Pharmaceutique	10	01.04.2000 00:00:00	IBM 500	2	2299	0	4598	Oui
12	11	110	Distribution	11	01.05.2000 00:00:00	AST Intel :	6	3190	0.015	18852.9	Oui
13	12	112	Machines/Outils	12	01.05.2000 00:00:00	Compaq P	6	1650	0.015	9751.5	Oui
14	13	123	Machines/Outils	13	01.05.2000 00:00:00	IBM 500	6	2299	0.015	13587.09	Oui
15	14	113	Construction	14	01.05.2000 00:00:00	AST Intel :	3	2690	0	8070	Oui
16	15	115	Distribution	15	01.05.2000 00:00:00	Compaq P	8	1650	0.015	13002	Oui
17	16	124	Éducation	16	01.05.2000 00:00:00	AST Intel :	8	3190	0.015	25137.2	Oui
18	17	124	Éducation	17	01.05.2000 00:00:00	Compaq P	11	1650	0.015	17877.75	Oui
19	18	106	Construction	18	01.05.2000 00:00:00	AST Intel :	11	3190	0.015	34563.65	Oui
20	19	101	Construction	19	01.05.2000 00:00:00	Compaq P	14	1650	0.015	22753.5	Non

Nous voyons alors ce que deviennent les espaces initialement dans les titres des colonnes ainsi que la colonne qui n'avait initialement pas de titre.

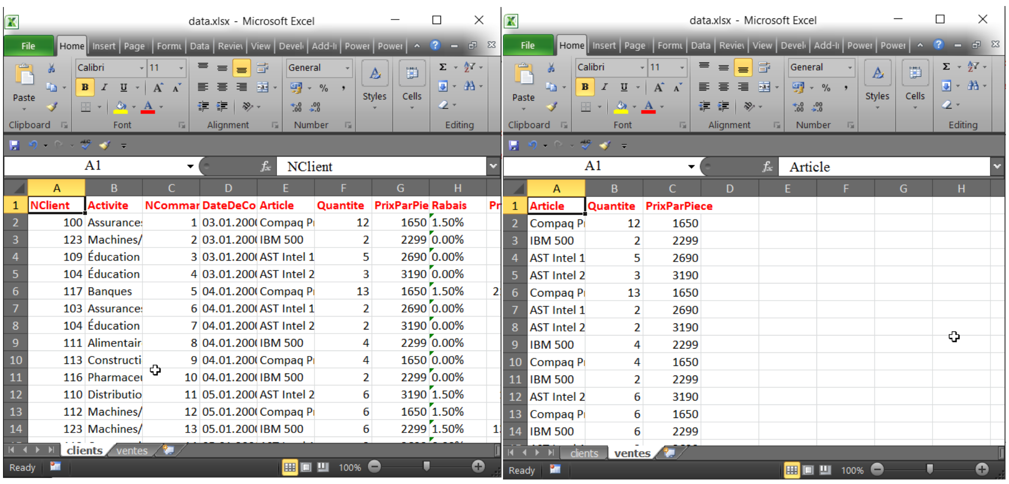
Nous pouvons aller plus loin en termes de manipulation de fichiers Microsoft Excel avec le package **xlsx** toujours avec une version 32 bits de R comme le montre l'exemple ci-dessous:

```
> library(xlsx)
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=";")
> str(mydata)
'data.frame':   109 obs. of  10 variables:
 $ NClient            : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Activite           : Factor w/ 8 levels "Alimentaire",..: 2 7 6 6 3 2 6 1 4 8 ...
 $ NCommande          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande     : Factor w/ 21 levels "03.01.2000","04.01.2000",..: 1 1 1 1 2 2 2 2 2 2 ...
 $ Article            : Factor w/ 4 levels "AST Intel 150",..: 3 4 1 2 3 1 2 4 3 4 ...
 $ Quantite           : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece       : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais             : Factor w/ 4 levels "0.00%","1.50%",..: 2 1 1 1 2 1 1 1 1 1 ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21128 ...
 $ FacturePayee       : Factor w/ 2 levels "Non","Oui": 2 2 2 2 2 2 2 2 2 2 ...
>
```

Ensuite, nous écrivons:

```
> data.xlsx<-createWorkbook()
> client<-createSheet(wb=data.xlsx,sheetName="clients")
> ventes<-createSheet(wb=data.xlsx,sheetName="ventes")
> #définir le style des en-têtes des colonnes
> style<-CellStyle(data.xlsx)+Font(data.xlsx,isBold=TRUE,color="red")
> #remplir ces feuilles avec une partie des dataframes et sans la colonne des numéros de ligne
> addDataFrame(x=mydata,sheet=client,row.names=FALSE,colnamesStyle=style)
> addDataFrame(x=mydata[c("Article","Quantite","PrixParPiece")],sheet=ventes,row.names=FALSE,colnamesStyle=style)
> saveWorkbook(data.xlsx,file="data.xlsx")
>
```

Ce qui donne:



NClient	Activite	NCommar	DateDeCo	Article	Quantite	PrixParPie	Rabais
100	Assurance:	1	03.01.200(	Compaq Pı	12	1650	1.50%
123	Machines/	2	03.01.200(	IBM 500	2	2299	0.00%
109	Éducation	3	03.01.200(	AST Intel 1	5	2690	0.00%
104	Éducation	4	03.01.200(	AST Intel 2	3	3190	0.00%
117	Banques	5	04.01.200(	Compaq Pı	13	1650	1.50%
103	Assurance:	6	04.01.200(	AST Intel 1	2	2690	0.00%
104	Éducation	7	04.01.200(	AST Intel 2	2	3190	0.00%
111	Alimentair	8	04.01.200(	IBM 500	4	2299	0.00%
113	Constructi	9	04.01.200(	Compaq Pı	4	1650	0.00%
116	Pharmaceı	10	04.01.200(	IBM 500	2	2299	0.00%
110	Distributio	11	05.01.200(	AST Intel 2	6	3190	1.50%
112	Machines/	12	05.01.200(	Compaq Pı	6	1650	1.50%
123	Machines/	13	05.01.200(	IBM 500	6	2299	1.50%

Article	Quantite	PrixParPiece
Compaq Pı	12	1650
IBM 500	2	2299
AST Intel 1	5	2690
AST Intel 2	3	3190
Compaq Pı	13	1650
AST Intel 1	2	2690
AST Intel 2	2	3190
IBM 500	4	2299
Compaq Pı	4	1650
IBM 500	2	2299
AST Intel 2	6	3190
Compaq Pı	6	1650
IBM 500	6	2299

Enfin, voyons le package qui pose le moins de problèmes en matière d'import de fichier Microsoft Excel et qui est le package **readxl** avec sa fonction **read_excel( )**:

```
> library("readxl")
> #cette fonction permet aussi de choisir une plange donnée dans la feuille!!!
> myData<-read_excel(path="c:/tmp/xlsDonnees.xlsx",sheet=1)
New names:
* `` -> ...10
> #notez que le résultat un tibble...
> head(myData)
# A tibble: 6 x 10
     ID `N° Client` Activité      `Date de commande`  Article         Quantité `Prix par pièce` `Rabais%` `Prix total avec rab~ ...10
  <dbl>       <dbl> <chr>         <dttm>              <chr>              <dbl>            <dbl>     <dbl>                 <dbl> <chr>
1     1         100 Assurances    2003-06-03 00:00:00 Compaq Presar~        12             1650      0.01                 19503  Oui
2     2         123 Machines/Out~ 2009-10-17 00:00:00 IBM 500                2             2299      0                     4598  Oui
3     3         119 Alimentaire   2004-05-09 00:00:00 IBM 500               22             2299      0.03                 49061. Oui
4     4         101 Construction  2003-01-14 00:00:00 Compaq Presar~        11             1650      0                    18150  Oui
5     5         117 Pharmaceutiq~ 2005-03-14 00:00:00 IBM 500                8             2299      0.01                 18116. Oui
6     6         121 Distribution  2009-09-11 00:00:00 AST Intel 200         23             3190      0.03                 71169. Oui
> #on convertit en un dataframe
> myData<-as.data.frame(myData)
> head(myData)
  ID N° Client         Activité Date de commande            Article Quantité Prix par pièce Rabais% Prix total avec rabais ...10
1  1        100       Assurances       2003-06-03 Compaq Presario 100       12           1650    0.01               19503.00   Oui
2  2        123 Machines/Outils       2009-10-17             IBM 500        2           2299    0.00                4598.00   Oui
3  3        119      Alimentaire       2004-05-09             IBM 500       22           2299    0.03               49060.66   Oui
4  4        101     Construction       2003-01-14 Compaq Presario 100       11           1650    0.00               18150.00   Oui
5  5        117   Pharmaceutique       2005-03-14             IBM 500        8           2299    0.01               18116.12   Oui
6  6        121     Distribution       2009-09-11       AST Intel 200       23           3190    0.03               71168.90   Oui
> dim(myData)
[1] 1048575      10
>
```

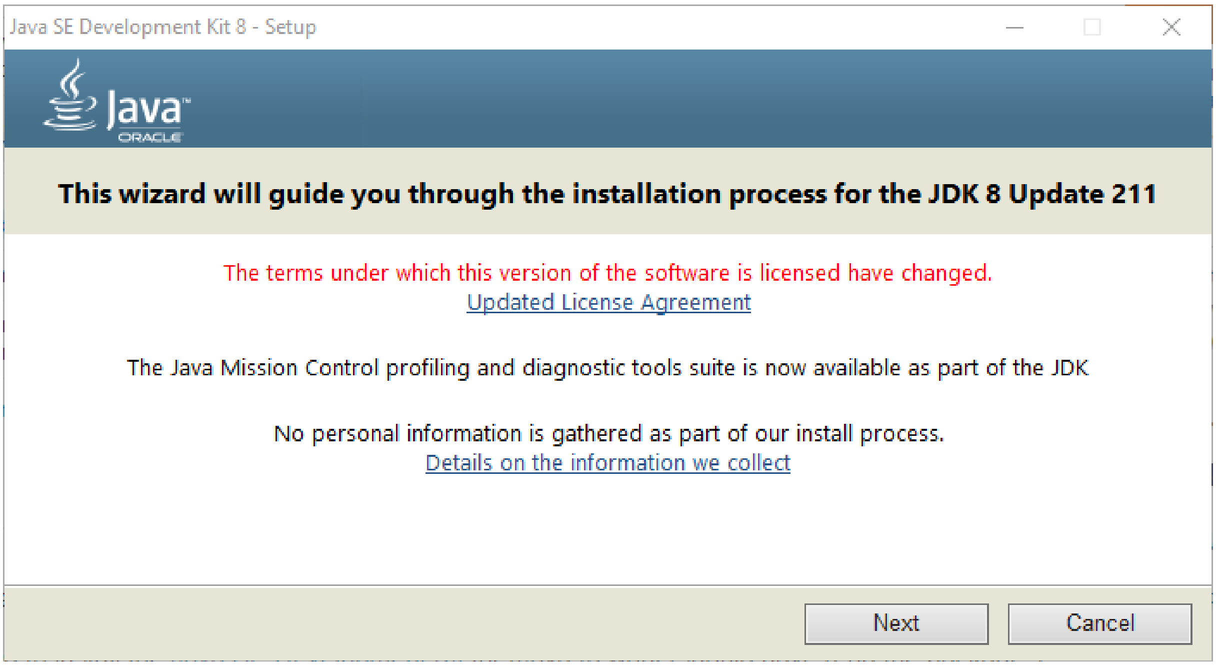
Notez que ce package fonctionne dans la version 64 bit mais ne sait faire uniquement de l'import!!!!!!

En 64 bits

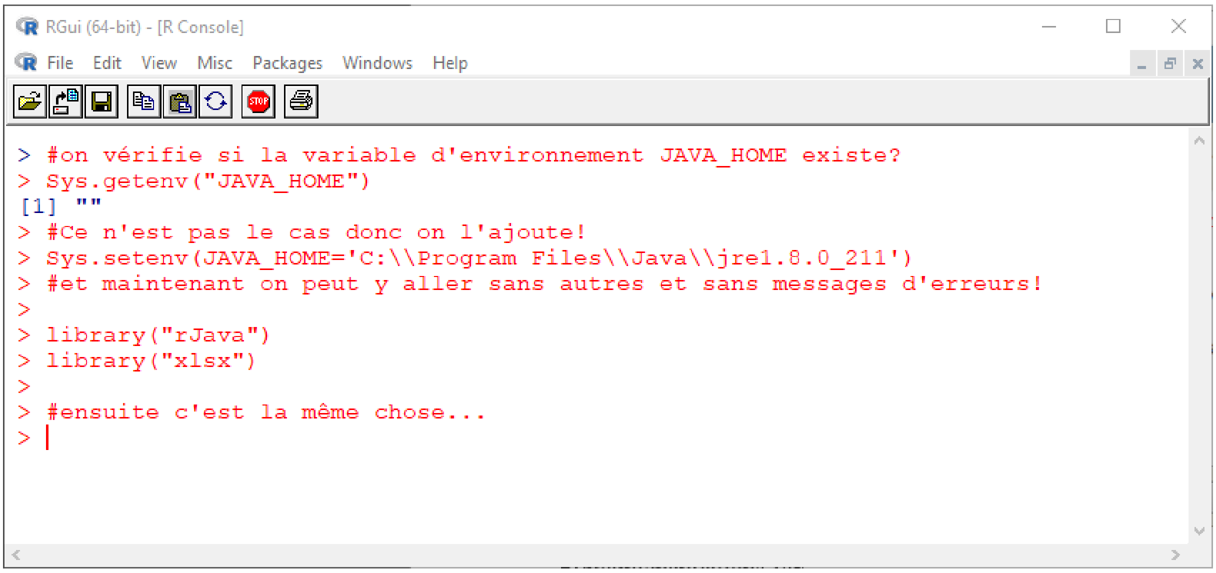
Maintenant voyons comment faire marche le package xlsx dans l'environnement 64 bits! D'abord on se rendra sur la page web:

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

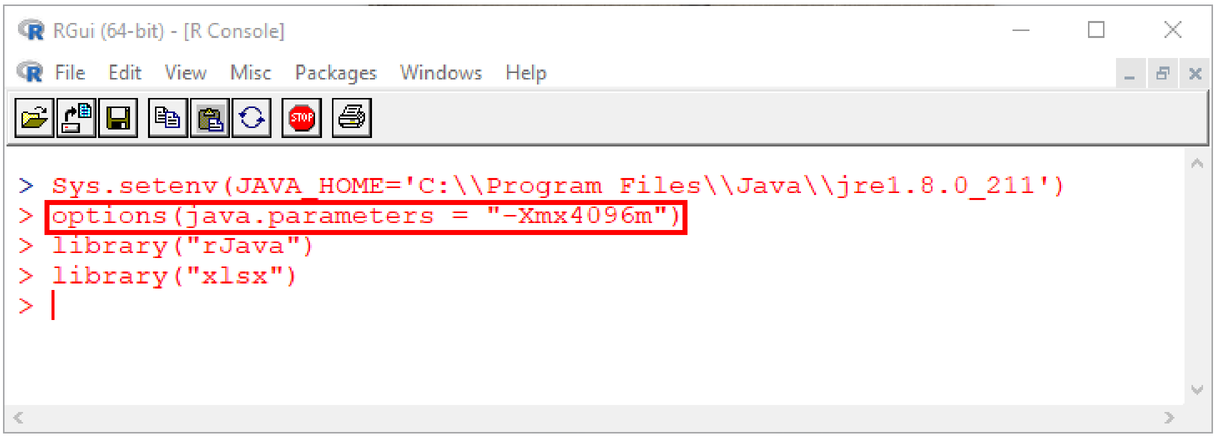
pour télécharger et installer le jdk8 pour Microsoft Windows et installer ce dernier de façon classique:



Et maintenant allons-y:

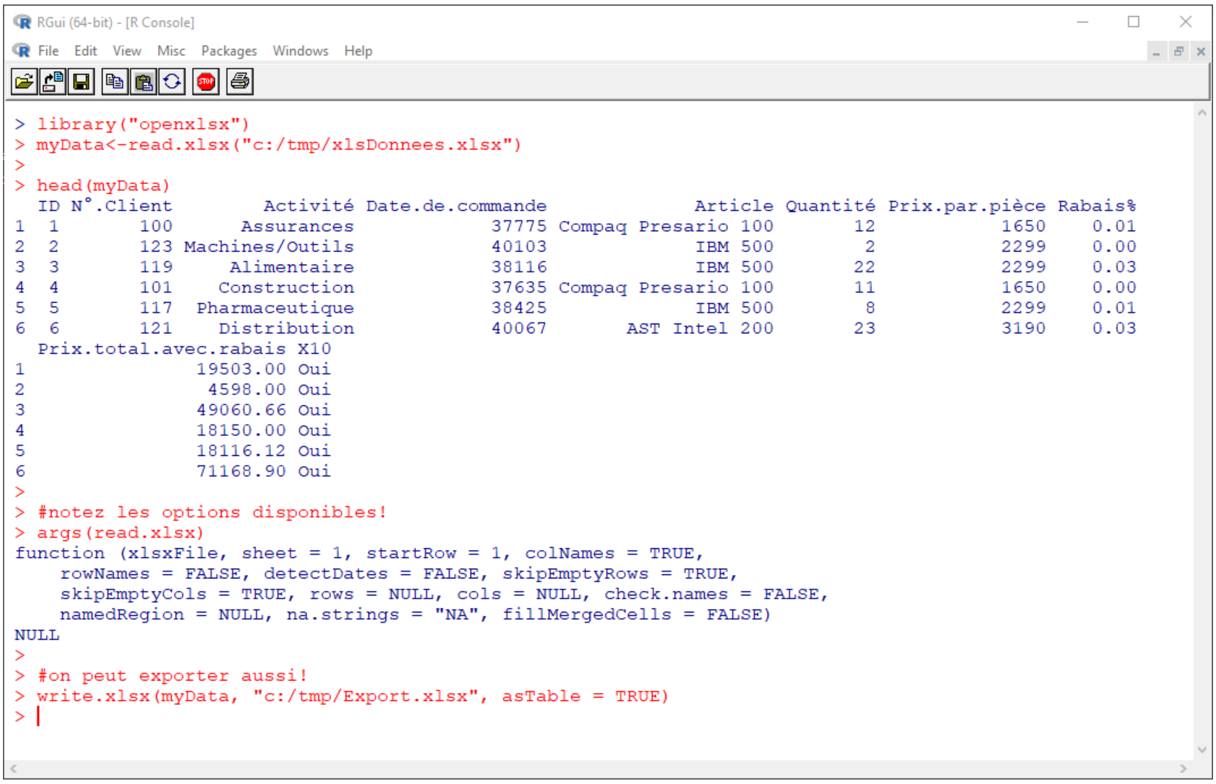


Si les jeux de données Microsoft Excel à importer sont conséquents, un message d'erreur de dépassement de capacité de mémoire apparaîtra, pour l'éviter (quand la mémoire est suffisante) on ajoutera:



Ensuite… bonne chance!

Sinon **indépendamment de toute package relatif à Java**, on a le package **openxlsx**[6] et ses fonctions **read.xlsx( )** et **write.xlsx( )**:



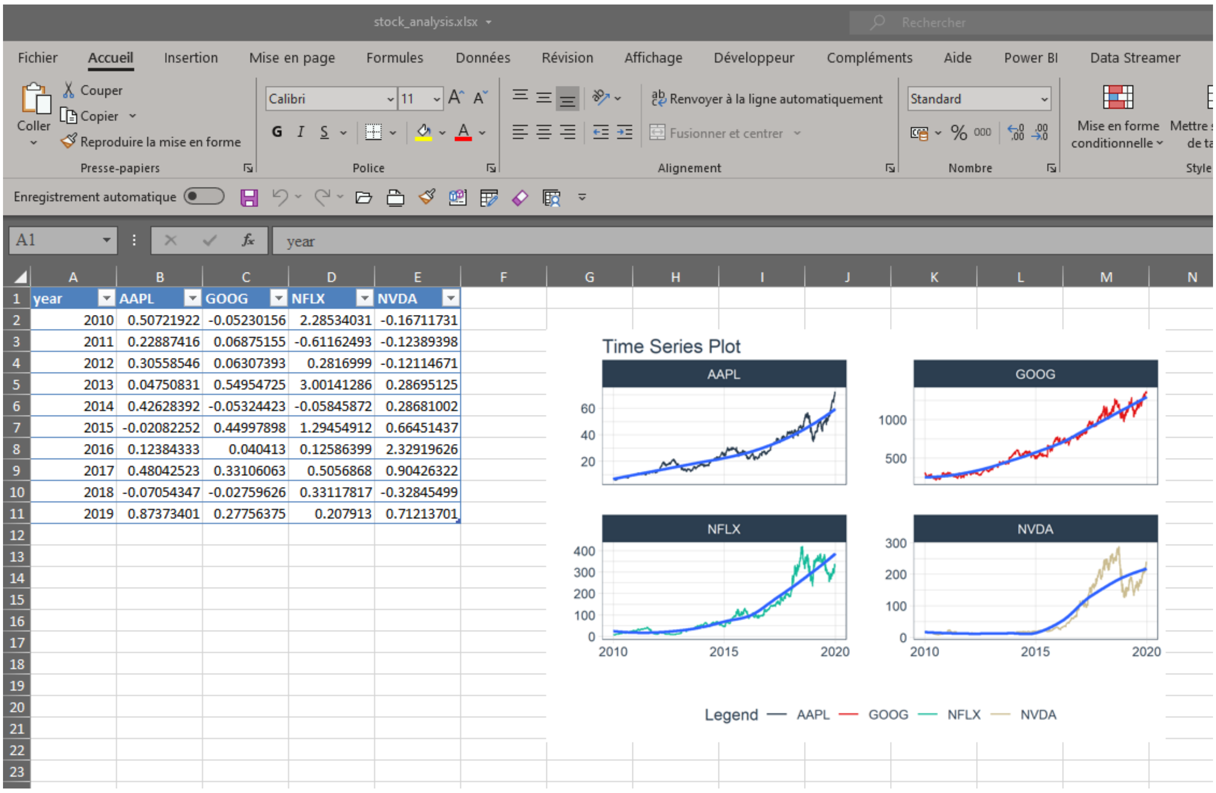
[6] Un package alternatif est en cours de développement, nommé **openxlsx2** qui devrait résoudre en théorie certains problèmes du package **openxlsx**.

Pour ceux qui souhaite un exemple plus complet en voici un:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #Exemple pris de: https://www.youtube.com/watch?v=EMSkZOF-ZG8&feature=youtu.be
>
> library("openxlsx")
> library("tidyquant")
> library("tidyverse")
> library("timetk")
>
> #Obtention des données
> stock_data_tbl <- c("AAPL", "GOOG", "NFLX", "NVDA") %>%
+     tq_get(from = "2010-01-01", to = "2019-12-31")
>
> #Préparation des données
> stock_pivot_table <- stock_data_tbl %>%
+     pivot_table(
+         .rows    = ~ YEAR(date),
+         .columns = ~ symbol,
+         .values  = ~ PCT_CHANGE_FIRSTLAST(adjusted)
+     ) %>%
+     rename(year = 1)
>
> stock_plot <- stock_data_tbl %>%
+     group_by(symbol) %>%
+     plot_time_series(date, adjusted, .color_var = symbol,
+ .facet_ncol = 2, .interactive = FALSE)
>
> #Création du classeur Microsoft Excel
> wb <- createWorkbook()
>
> #Ajout d'une feuille
> addWorksheet(wb, sheetName = "stock_analysis")
>
> # Ajout d'une graphique
> print(stock_plot)
> wb %>% insertPlot(sheet = "stock_analysis", startCol = "G", startRow = 3)
>
> #Ajout de données
> writeDataTable(wb, sheet = "stock_analysis", x = stock_pivot_table)
>
> #On sauve le fichier Microsoft Excel
> saveWorkbook(wb, "stock_analysis.xlsx", overwrite = TRUE)
>
> #On l'ouvre immédiatement derrière
> openXL("stock_analysis.xlsx")
> |
```

Ce qui donne une fois le fichier Microsoft Excel ouvert:

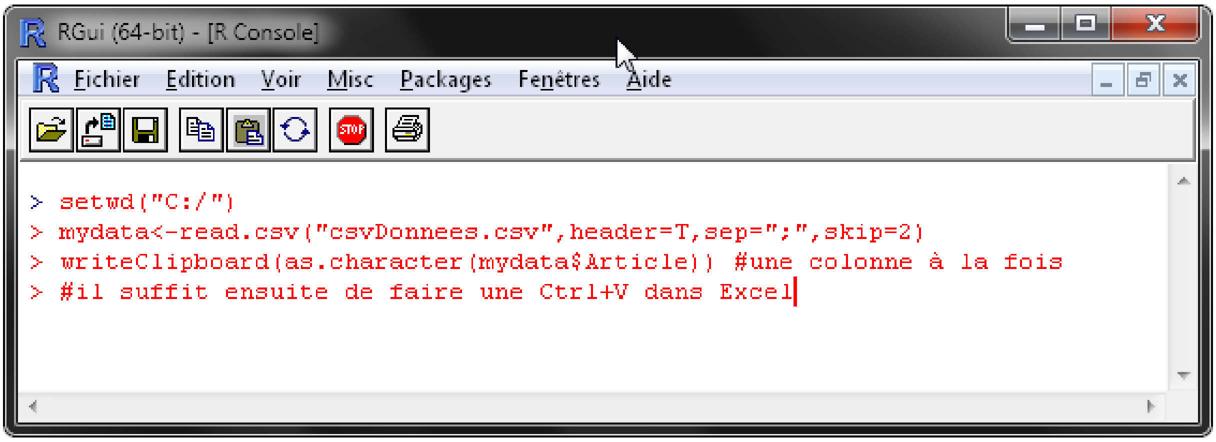


year	AAPL	GOOG	NFLX	NVDA
2010	0.50721922	-0.05230156	2.28534031	-0.16711731
2011	0.22887416	0.06875155	-0.61162493	-0.12389398
2012	0.30558546	0.06307393	0.2816999	-0.12114671
2013	0.04750831	0.54954725	3.00141286	0.28695125
2014	0.42628392	-0.05324423	-0.05845872	0.28681002
2015	-0.02082252	0.44997898	1.29454912	0.66451437
2016	0.12384333	0.040413	0.12586399	2.32919626
2017	0.48042523	0.33106063	0.5056868	0.90426322
2018	-0.07054347	-0.02759626	0.33117817	-0.32845499
2019	0.87373401	0.27756375	0.207913	0.71213701

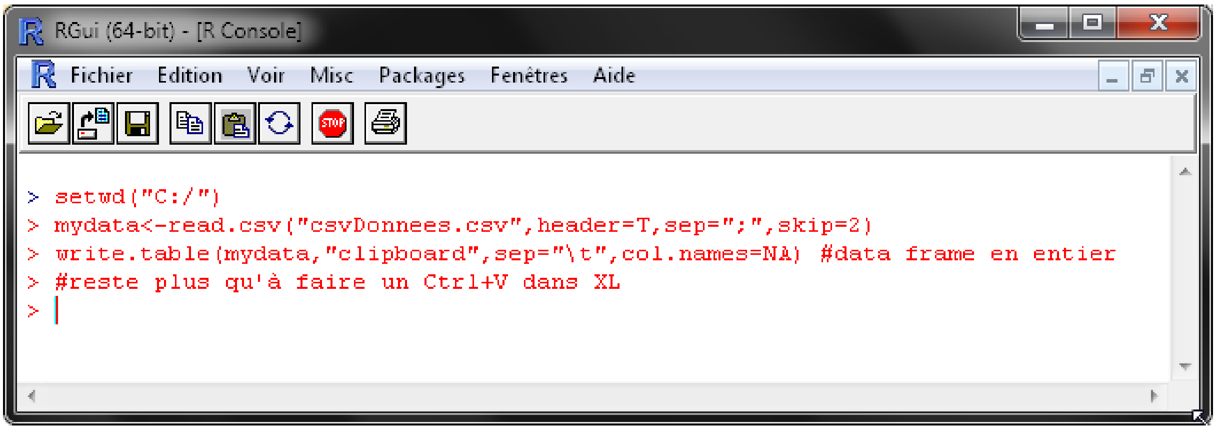
Utiliser le presse-papiers

Il est possible de mettre un data frame dans la mémoire (presse-papier) pour pouvoir ensuite par un simple copier/coller se retrouver avec une un tableau type dans Microsoft Excel(ou autre).

Voici cela avec un exemple d'abord pour des colonnes une par une:



et pour un data frame en entier:



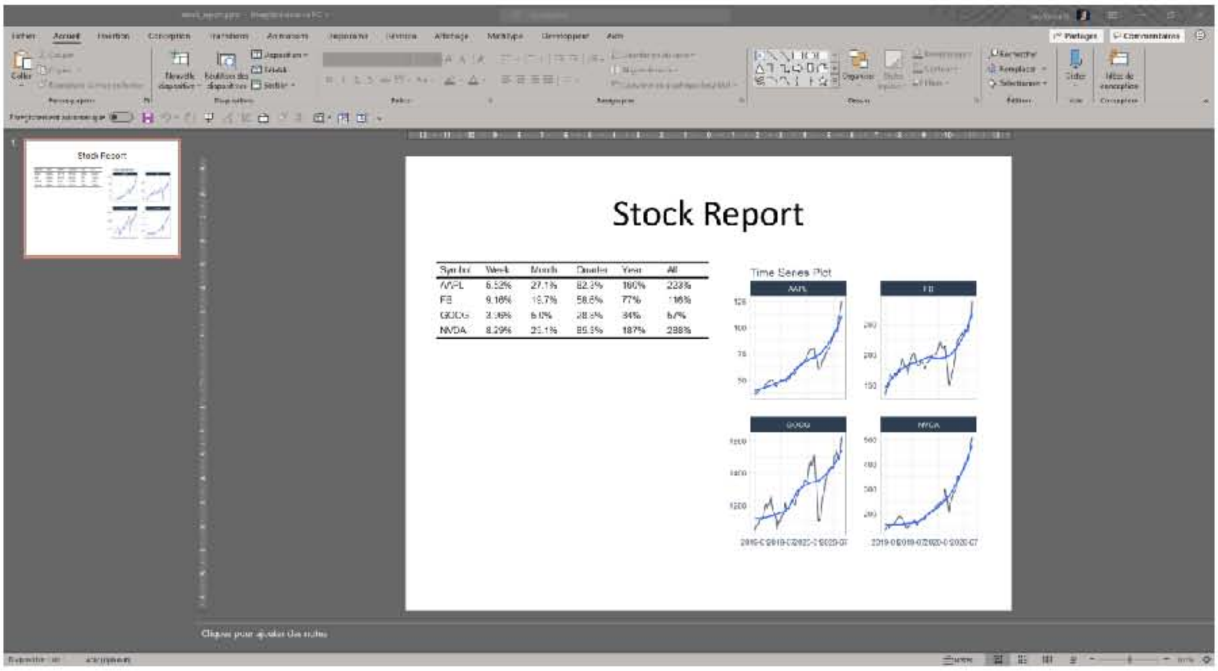
Exemple 44.: Exporter des données vers Microsoft PowerPoint

R 3.6.2

Beaucoup de multinationales ont besoin d'automatiser la génération automatisée de présentation Microsoft PowerPoint. Voyons donc comment un exemple simple d'un telle situation qui utilise la fameux package **officer**:

```
> library("officer")
> library("flextable")
> library("tidyverse")
> library("tidyquant")
> library("timetk")
>
> #Obtention des données
> stock_data_tbl <- c("AAPL", "GOOG", "FB", "NVDA") %>%
+     tq_get(from = "2019-01-01", to = "2020-08-31")
>
> #Préparation des données
> stock_returns_tbl <- stock_data_tbl %>%
+     select(symbol, date, adjusted) %>%
+     group_by(symbol) %>%
+     summarise(
+         week    = last(adjusted) / first(tail(adjusted, 7)) - 1,
+         month   = last(adjusted) / first(tail(adjusted, 30)) - 1,
+         quarter = last(adjusted) / first(tail(adjusted, 90)) - 1,
+         year    = last(adjusted) / first(tail(adjusted, 365)) - 1,
+         all     = last(adjusted) / first(adjusted) - 1
+     )
`summarise()` ungrouping output (override with `.groups` argument)
>
> #Préparation du graphique
> stock_plot <- stock_data_tbl %>%
+     group_by(symbol) %>%
+     summarize_by_time(adjusted = AVERAGE(adjusted), .by = "week") %>%
+     plot_time_series(date, adjusted, .facet_ncol = 2, .interactive = FALSE)
.date_var is missing. Using: date
> stock_plot
>
> #Préparation de la table
> stock_table <- stock_returns_tbl %>%
+     rename_all(.funs = str_to_sentence) %>%
+     mutate_if(is.numeric, .funs = scales::percent) %>%
+     flextable::flextable()
> stock_table
>
> #Création du fichier Microsoft PowerPoint
> doc <- read_pptx()
> doc <- add_slide(doc)
> doc <- ph_with(doc, value = "Stock Report", location = ph_location_type(type = "title"))
> doc <- ph_with(doc, value = stock_table, location = ph_location_left())
> doc <- ph_with(doc, value = stock_plot, location = ph_location_right())
>
> print(doc, target = "stock_report.pptx")
>
```

Ce qui donne:


Stock Report
Time Series Plot

Exemple 45.: Exporter des données vers Microsoft Word

R 4.1.1

L'exemple qui suit utilise aussi **officer** et est un copier/coller de l'exemple fournit ici par Sang-Heon Lee:

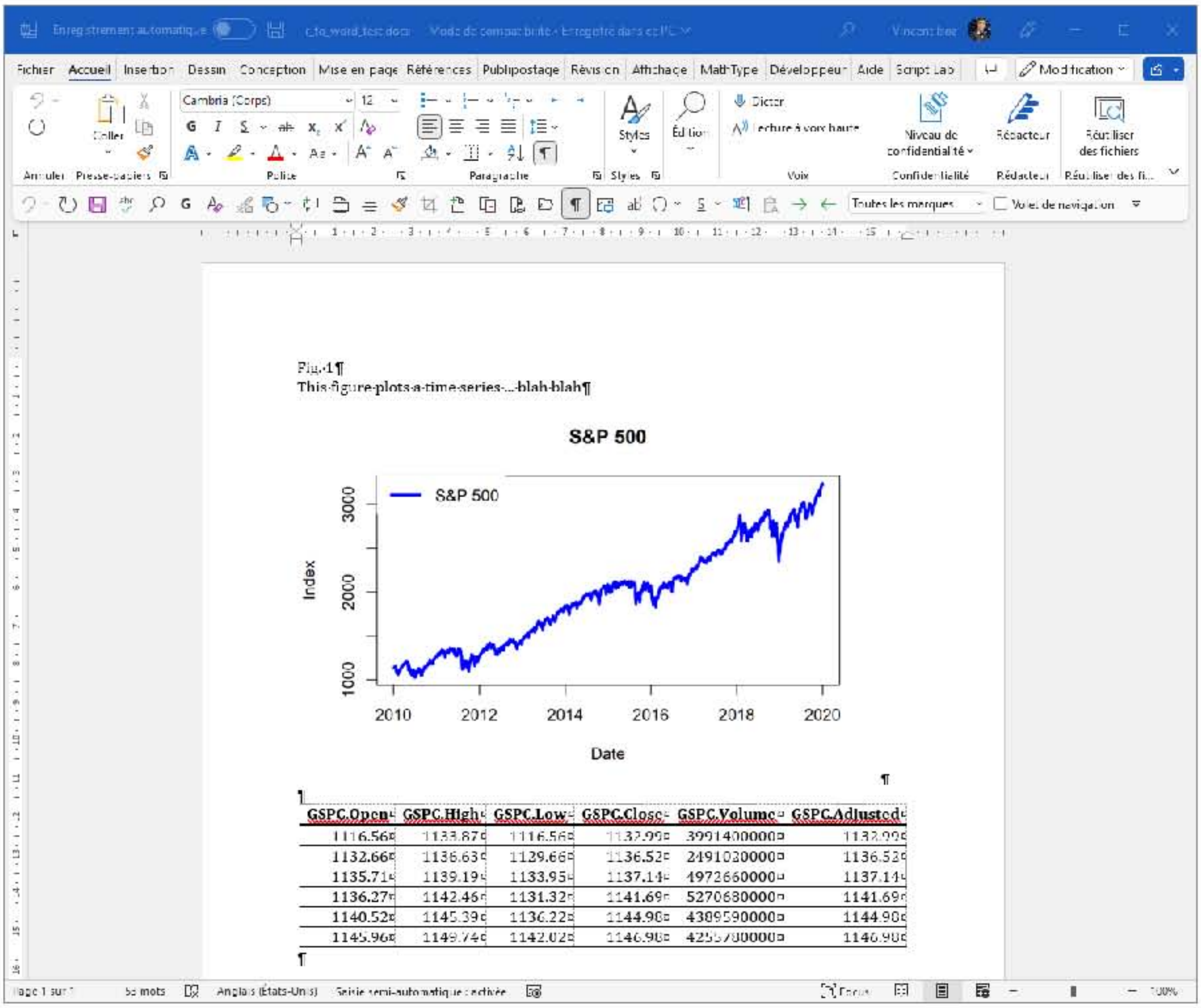
https://kiandlee.blogspot.com/2022/08/paste-r-plot-image-into-ms-word.html

Le code simplifié à son strict minimum est:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #========================================================#
> # Quantitative ALM, Financial Econometrics & Derivatives
> # ML/DL using R, Python, Tensorflow by Sang-Heon Lee
> #
> # https://kiandlee.blogspot.com
> #========================================================#
>
> graphics.off(); rm(list = ls())
>
> library(quantmod) # getSymbols
> library(officer)
> setwd("c:/tmp/")
>
> #-----------------------------------------------
> # read S&P 500 and NASDAQ index
> #-----------------------------------------------
>
> sdate <- as.Date("2010-01-01")
> edate <- as.Date("2020-01-01")
> getSymbols("^GSPC", from=sdate, to=edate)
[1] "^GSPC"
> getSymbols("^IXIC", from=sdate, to=edate)
[1] "^IXIC"
> df <- data.frame( time(GSPC[,6]), GSPC[,6], IXIC[,6])
>
> rownames(df) <- NULL
> colnames(df) <- c("date", "GSPC", "NDAQ")
>
> #------------------------------------------------------------
> # paste images to a WORD file with additional information
> #------------------------------------------------------------
>
> ### create a new document object
> doc <- read_docx()
>
> ### add title and description
> doc <- body_add(doc, "Fig. 1", style = "Normal")
> desc <- "This figure plots a time series ... blah blah"
> doc <- body_add(doc, desc, style = "Normal")
>
>
> ### various method to add figure into WORD
>
> # 1) by direct commands
> doc <- body_add(doc, plot_instr(code = {
+     matplot(df$date, df$GSPC, main = "S&P 500", type = "l",
+             lwd = 3, col = "blue", xlab="Date", ylab="Index")
+     legend('topleft',legend="S&P 500", col="blue", lty=1,lwd=3, box.lty=0)
+ }), height = 4)
>
> ### add table if necessary
> doc <- body_add(doc, "", style = "Normal")
> doc <- body_add(doc, head(as.data.frame(GSPC)), style = "table_template")
>
> ### save word file
> print(doc, target="r_to_word_test.docx")
>
```

Ce qui donne:


Fig. 1
This figure plots a time series ... blah blah
S&P 500
S&P 500
Index
Date
3000
2000
1000
2010
2012
2014
2016
2018
2020
GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1116.56 1133.87 1116.56 1132.99 3991400000 1132.99
1132.66 1136.63 1129.66 1136.52 2491020000 1136.52
1135.71 1139.19 1133.95 1137.14 4972660000 1137.14
1136.27 1142.46 1131.32 1141.69 5270680000 1141.69
1140.52 1145.39 1136.22 1144.98 4389590000 1144.98
1145.96 1149.74 1142.02 1146.98 1146.98

Exemple 46.: Importer/Exporter des données au format SPSS

R 3.0.2

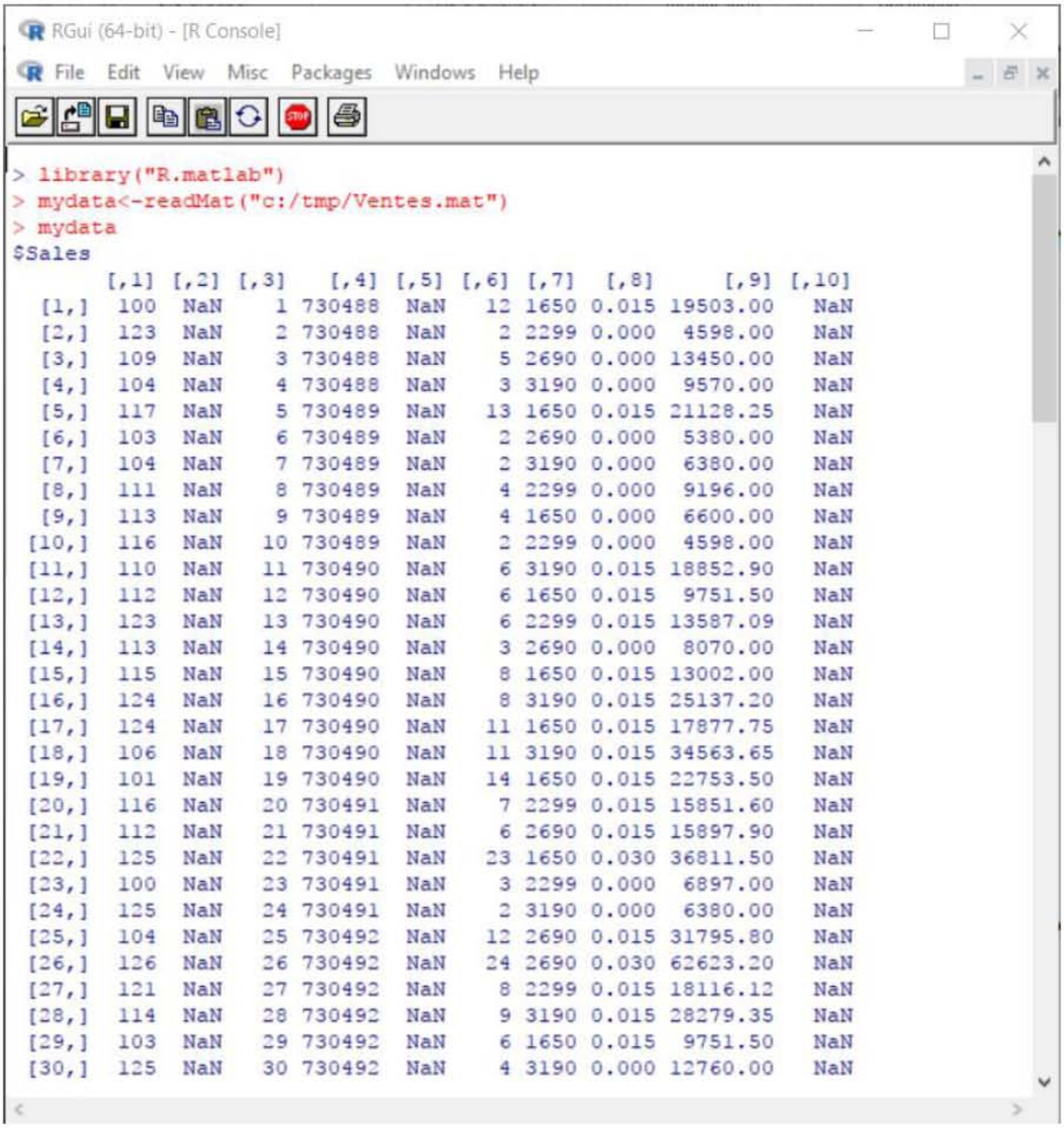
De nombreux étudiants universitaires doivent passer leurs anciennes données SPSS en R pour pouvoir utiliser toute la puissance de R. Cela est possible avec la commande **read.spss( )** du package **foreign** qui en plus gère magnifique bien les labels SPSS qui sont normalement une sa***** à gérer et à transformer avec d'autres packages:

```
> library("foreign")
> mydata<-read.spss(file="c:/tmp/SPSS.sav",use.value.labels=TRUE,max.value.labels=Inf,to.data.frame=TRUE)
> str(mydata)
'data.frame':   54 obs. of  16 variables:
 $ ID         : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Sex        : Factor w/ 2 levels "Female","Male": 1 1 1 1 2 1 1 1 2 2 ...
 $ Age        : num  22 22 21 21 20 18 22 22 20 20 ...
 $ Ethnicity  : Factor w/ 6 levels "European American",..: 1 1 1 1 1 1 1 1 3 2 ...
 $ Cl_St      : Factor w/ 5 levels "Freshman","Sophomore",..: 3 3 3 3 4 2 3 4 2 3 ...
 $ Fam_Income : Factor w/ 10 levels "0 to 15,000",..: 5 5 5 6 6 6 6 6 7 7 ...
 $ Candy      : Factor w/ 2 levels "Skittles","None": 1 1 1 1 1 1 1 1 1 1 ...
 $ Stimuli    : Factor w/ 3 levels "Printed","Spoken",..: 1 1 1 1 1 1 1 1 1 2 ...
 $ Distraction: Factor w/ 3 levels "No Distraction",..: 1 1 1 2 2 2 3 3 3 1 ...
 $ Recall1    : num  9 9 11 8 7 6 10 8 9 8 ...
 $ Recall2    : num  8.28 8.28 10.12 7.36 6.02 ...
 $ Skittles   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Printed    : num  1 1 1 1 1 1 1 1 1 0 ...
 $ Spoken     : num  0 0 0 0 0 0 0 0 0 1 ...
 $ CellPhone  : num  0 0 0 1 1 1 0 0 0 0 ...
 $ LightBulb  : num  0 0 0 0 0 0 1 1 1 0 ...
 - attr(*, "variable.labels")= Named chr  "Participant Identification Number" "Gender" "Age" "Ethnicity" ...
  ..- attr(*, "names")= chr  "ID" "Sex" "Age" "Ethnicity" ...
> head(mydata)
  ID    Sex Age         Ethnicity     Cl_St        Fam_Income    Candy Stimuli
1  1 Female  22 European American    Junior 45,001 to 55,000 Skittles Printed
2  2 Female  22 European American    Junior 45,001 to 55,000 Skittles Printed
3  3 Female  21 European American    Junior 45,001 to 55,000 Skittles Printed
4  4 Female  21 European American    Junior 55,001 to 65,000 Skittles Printed
5  5   Male  20 European American    Senior 55,001 to 65,000 Skittles Printed
6  6 Female  18 European American Sophomore 55,001 to 65,000 Skittles Printed
      Distraction Recall1 Recall2 Skittles Printed Spoken CellPhone LightBulb
1  No Distraction       9    8.28        1       1      0         0         0
2  No Distraction       9    8.28        1       1      0         0         0
3  No Distraction      11   10.12        1       1      0         0         0
4 Cell Phone Ring       8    7.36        1       1      0         1         0
5 Cell Phone Ring       7    6.02        1       1      0         1         0
6 Cell Phone Ring       6    5.52        1       1      0         1         0
>
```

Exemple 47.: Importer des données au format MATLAB

R 3.4.2

L'import de données MATLAB se fait qu'avec des matrices contenues dans des fichiers *.mat avec le package **R.matlab** et sa fonction **readMat( )** :

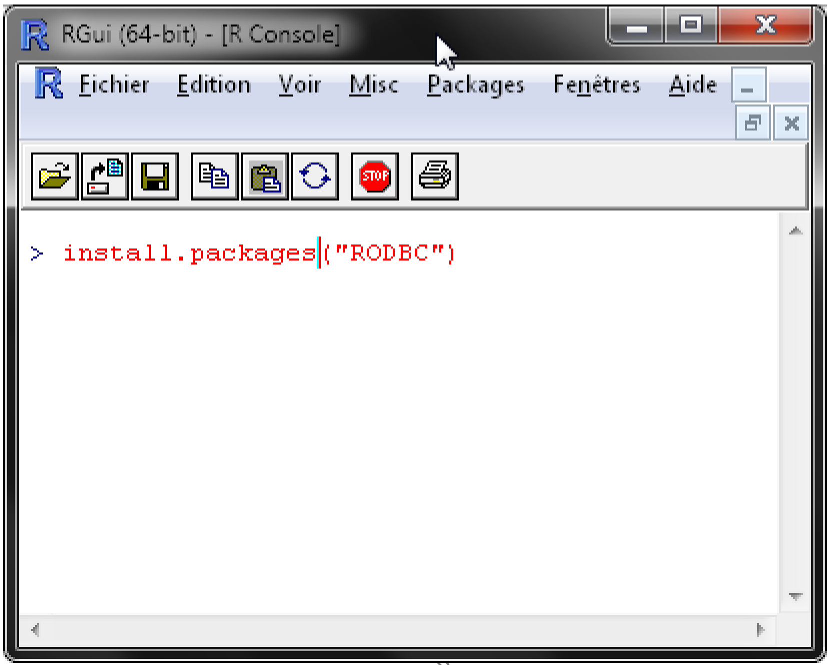



```
> library("R.matlab")
> mydata<-readMat("c:/tmp/Ventes.mat")
> mydata
$Sales
      [,1] [,2] [,3]   [,4] [,5] [,6] [,7]  [,8]     [,9] [,10]
 [1,]  100  NaN    1 730488  NaN   12 1650 0.015 19503.00   NaN
 [2,]  123  NaN    2 730488  NaN    2 2299 0.000  4598.00   NaN
 [3,]  109  NaN    3 730488  NaN    5 2690 0.000 13450.00   NaN
 [4,]  104  NaN    4 730488  NaN    3 3190 0.000  9570.00   NaN
 [5,]  117  NaN    5 730489  NaN   13 1650 0.015 21128.25   NaN
 [6,]  103  NaN    6 730489  NaN    2 2690 0.000  5380.00   NaN
 [7,]  104  NaN    7 730489  NaN    2 3190 0.000  6380.00   NaN
 [8,]  111  NaN    8 730489  NaN    4 2299 0.000  9196.00   NaN
 [9,]  113  NaN    9 730489  NaN    4 1650 0.000  6600.00   NaN
[10,]  116  NaN   10 730489  NaN    2 2299 0.000  4598.00   NaN
[11,]  110  NaN   11 730490  NaN    6 3190 0.015 18852.90   NaN
[12,]  112  NaN   12 730490  NaN    6 1650 0.015  9751.50   NaN
[13,]  123  NaN   13 730490  NaN    6 2299 0.015 13587.09   NaN
[14,]  113  NaN   14 730490  NaN    3 2690 0.000  8070.00   NaN
[15,]  115  NaN   15 730490  NaN    8 1650 0.015 13002.00   NaN
[16,]  124  NaN   16 730490  NaN    8 3190 0.015 25137.20   NaN
[17,]  124  NaN   17 730490  NaN   11 1650 0.015 17877.75   NaN
[18,]  106  NaN   18 730490  NaN   11 3190 0.015 34563.65   NaN
[19,]  101  NaN   19 730490  NaN   14 1650 0.015 22753.50   NaN
[20,]  116  NaN   20 730491  NaN    7 2299 0.015 15851.60   NaN
[21,]  112  NaN   21 730491  NaN    6 2690 0.015 15897.90   NaN
[22,]  125  NaN   22 730491  NaN   23 1650 0.030 36811.50   NaN
[23,]  100  NaN   23 730491  NaN    3 2299 0.000  6897.00   NaN
[24,]  125  NaN   24 730491  NaN    2 3190 0.000  6380.00   NaN
[25,]  104  NaN   25 730492  NaN   12 2690 0.015 31795.80   NaN
[26,]  126  NaN   26 730492  NaN   24 2690 0.030 62623.20   NaN
[27,]  121  NaN   27 730492  NaN    8 2299 0.015 18116.12   NaN
[28,]  114  NaN   28 730492  NaN    9 3190 0.015 28279.35   NaN
[29,]  103  NaN   29 730492  NaN    6 1650 0.015  9751.50   NaN
[30,]  125  NaN   30 730492  NaN    4 3190 0.000 12760.00   NaN
```

Exemple 48.: Importer des données de fichiers MS Access (*.mdb/*.accdb)

R 3.0.2

Nous allons d'abord installer le package suivant **RODBC**:

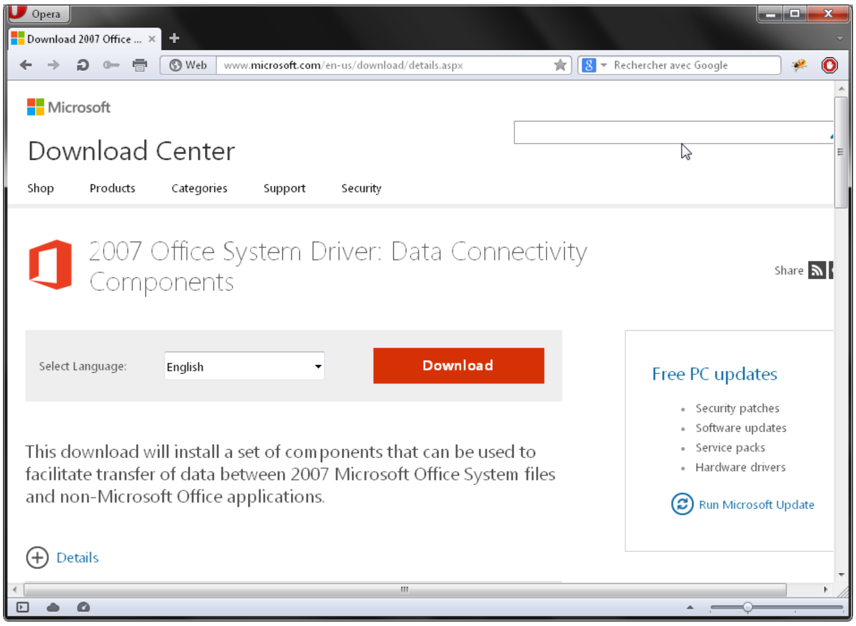


Attention!!! Ce package ne permet pas que d'accéder aux données mais également des les modifier, de créer des tables, etc.!

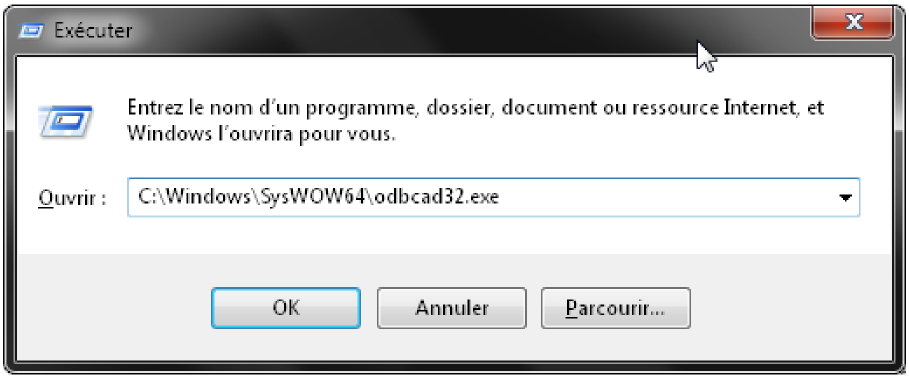
Ensuite, il faut suivant la manière dont vos informaticiens ont installé Microsoft Windows dans votre organisation aller télécharger la bonne version des drivers ODBC Access de Microsoft:

http://www.microsoft.com/en-us/download/details.aspx?id=23734

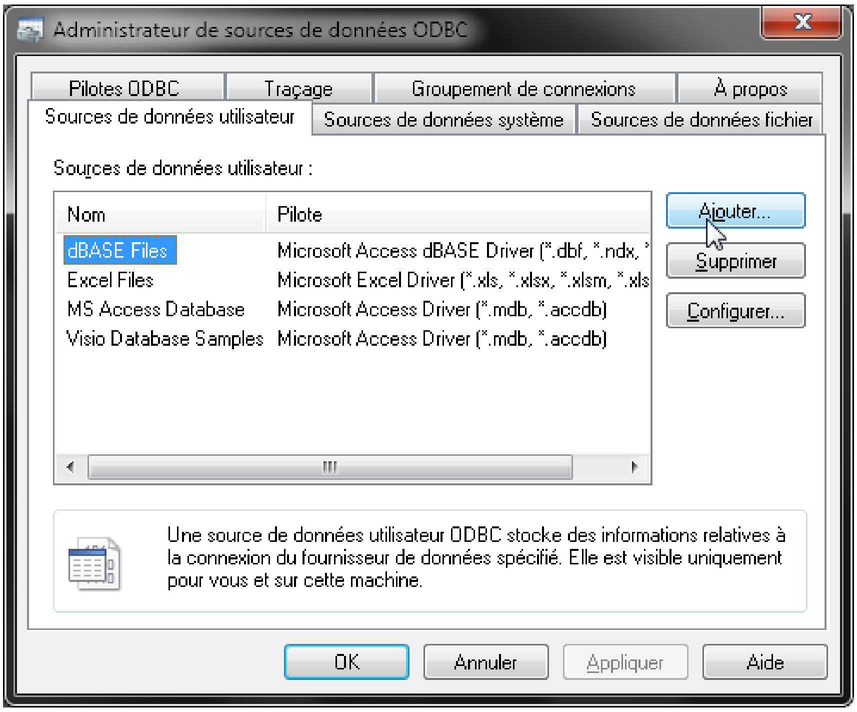
ce qui donnait au jour où j'écris ces lignes, la page Internet suivante avec un exécutable à installer:



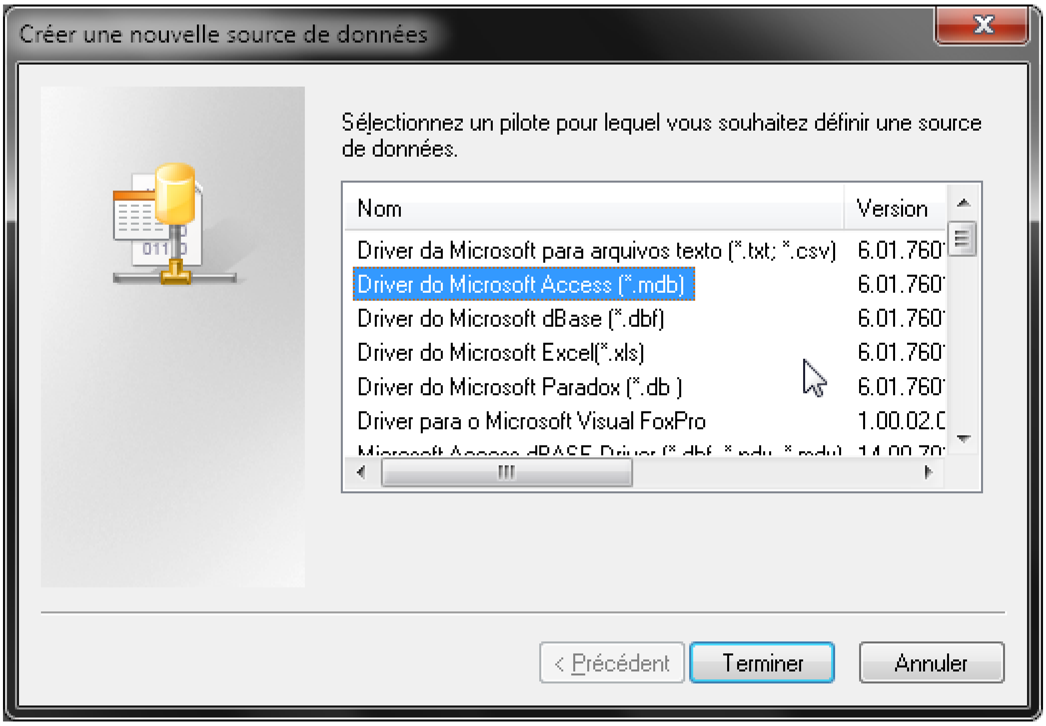
Ensuite, nous lançons le gestionnaire ODBC de Windows en 32 bits car le package **ODBC** ne marche pas correctement en x64:



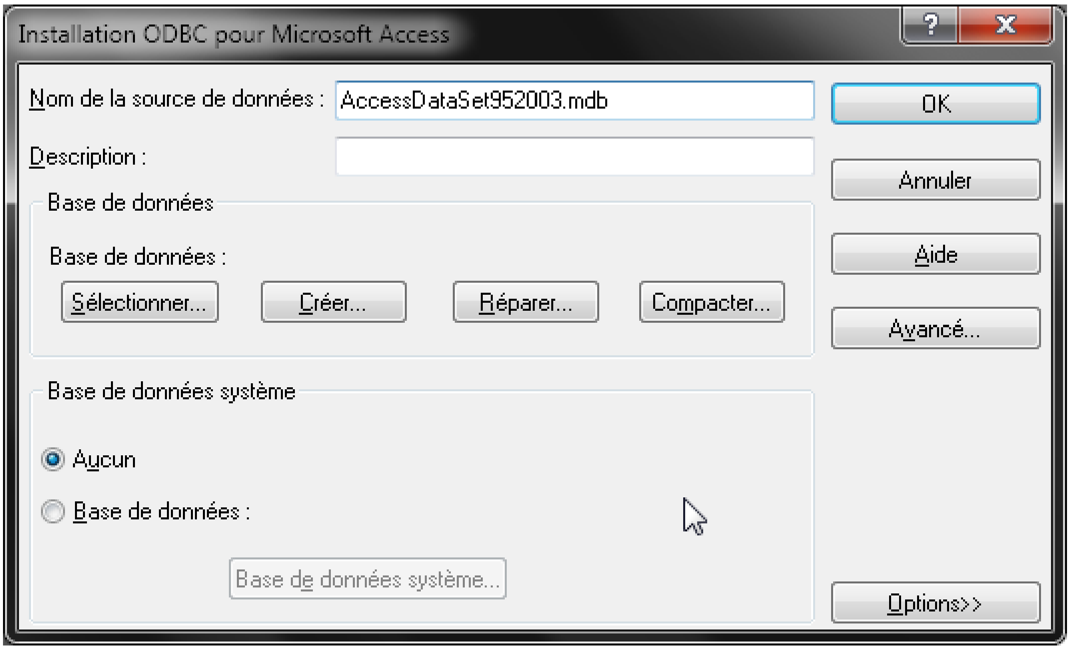
S'ouvre alors la boîte de dialogue suivante et nous cliquons sur **Ajouter**:



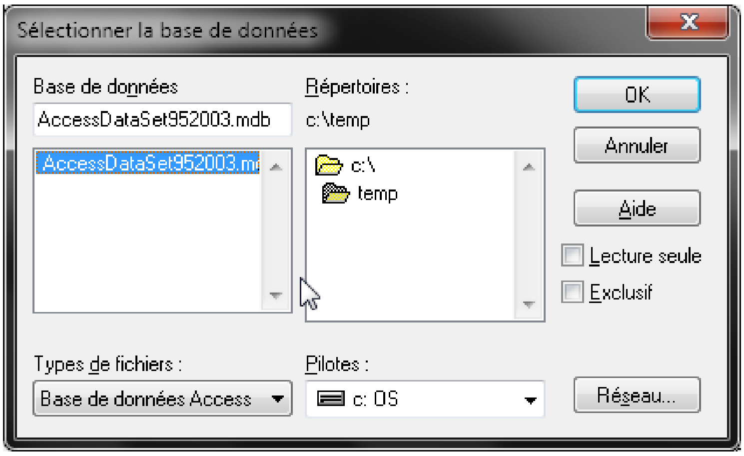
Nous prenons ensuite le driver qui correspond au premier fichier que nous voulons utiliser (nous commençons par un *.mdb et nous verrons ensuite pour le *.accdb):



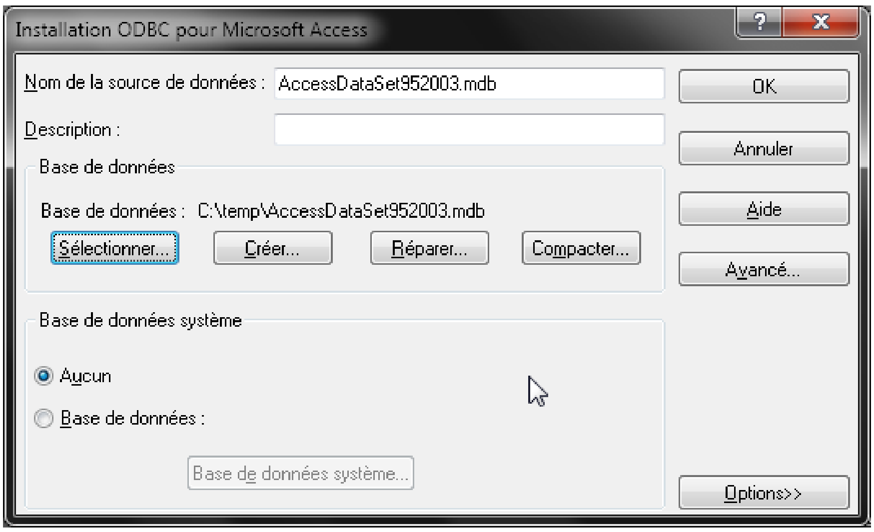
Nous cliquons ensuite sur **Terminer** et dans la boîte de dialogue qui apparaît il faut absolument donner au nom de la source de données le nom du fichier Access lui-même auquel vous voulez par la suite vous connecter:



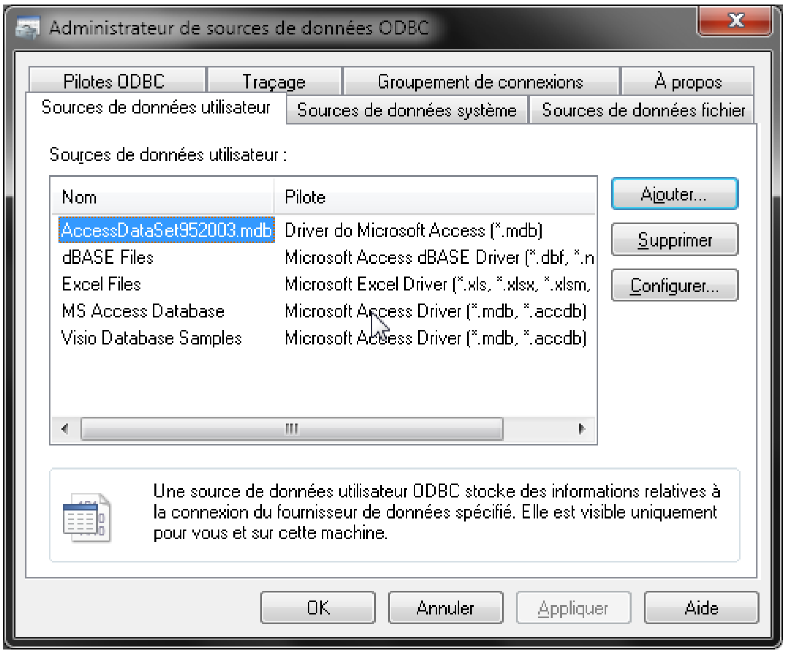
Ensuite, nous cliquons sur le bouton **Sélectionner...**:



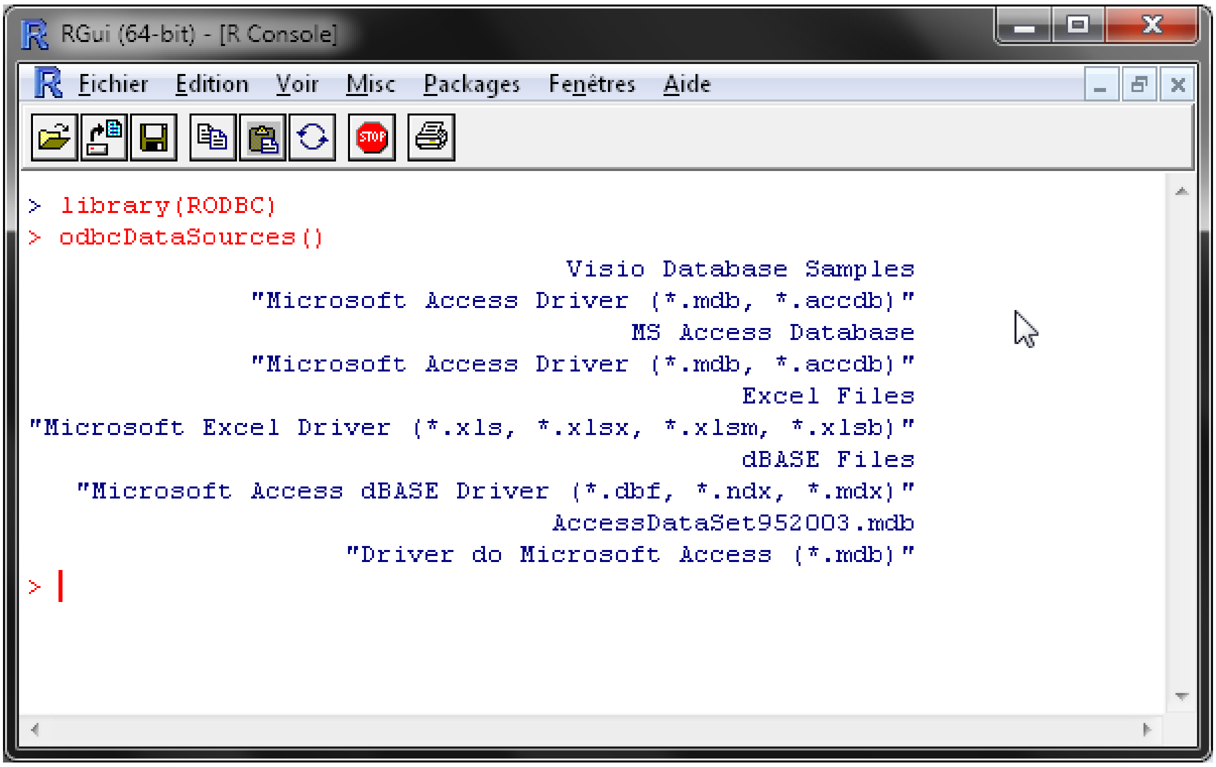
et nous allons chercher le dossier où se trouve le Access qui nous intéresse. Nous validons par **OK** une première fois:



et encore par **OK**:



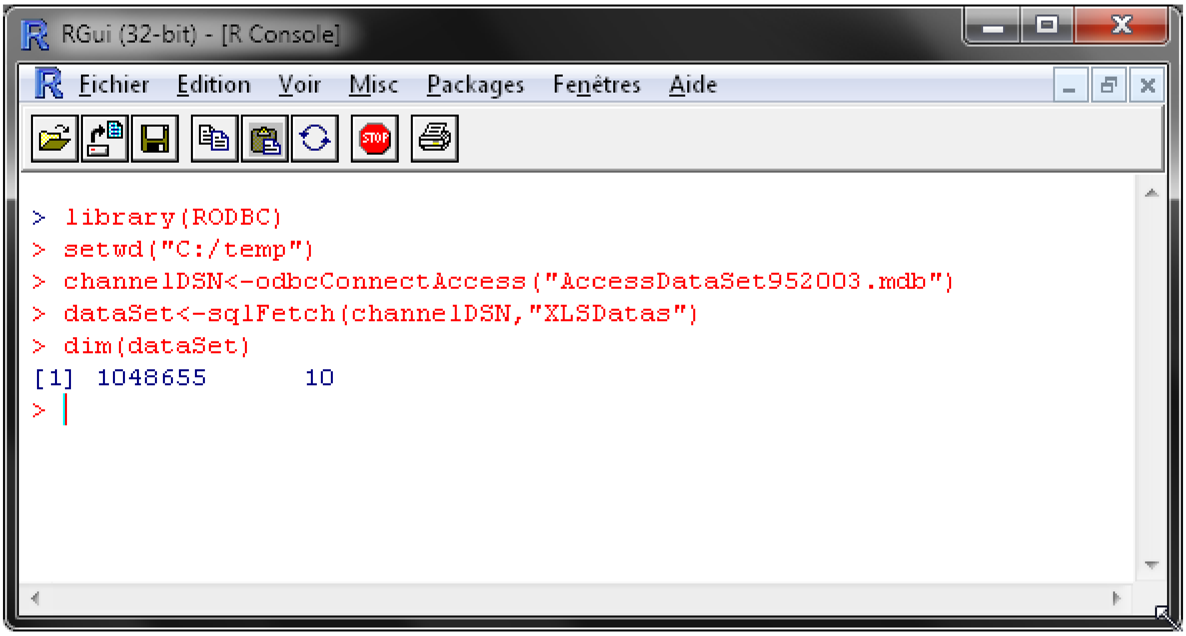
et nous validons ensuite sur **OK** et retournons dans R. Nous chargeons comme à l'habitude la librairie et avec la commande **odbcDataSources( )** nous testons si R voit bien notre driver:



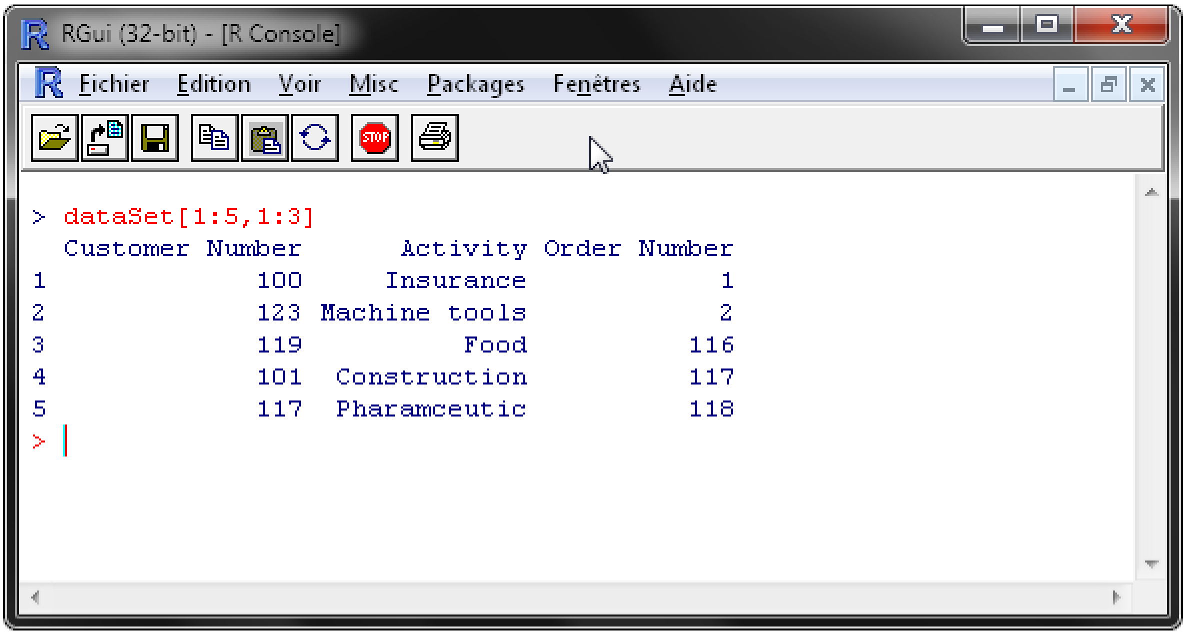
et c'est donc bien le cas.

Pour la suite nous sommes obligés d'utiliser (du moins au jour où j'écris ces lignes) la version x32 de R pour pouvoir poursuivre et nous utilisons la commande **odbcConnectAccess( )** comme il s'agit d'un fichier Access 2003 ou antérieur et ensuite la commande **sqlFetch**( )

pour chercher la table *XLSData* qui nous intéresse et qui contient effectivement environ 1 million de lignes:



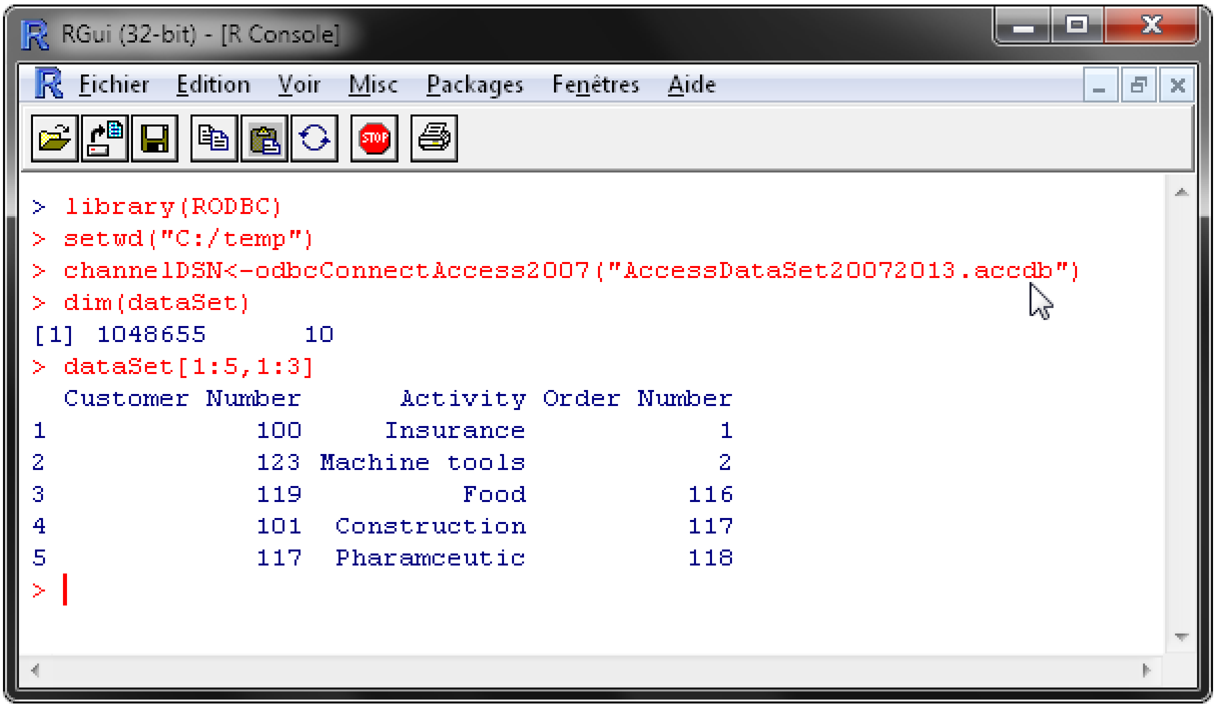
affichons quelques colonnes et quelques lignes pour voir que tout est ok:



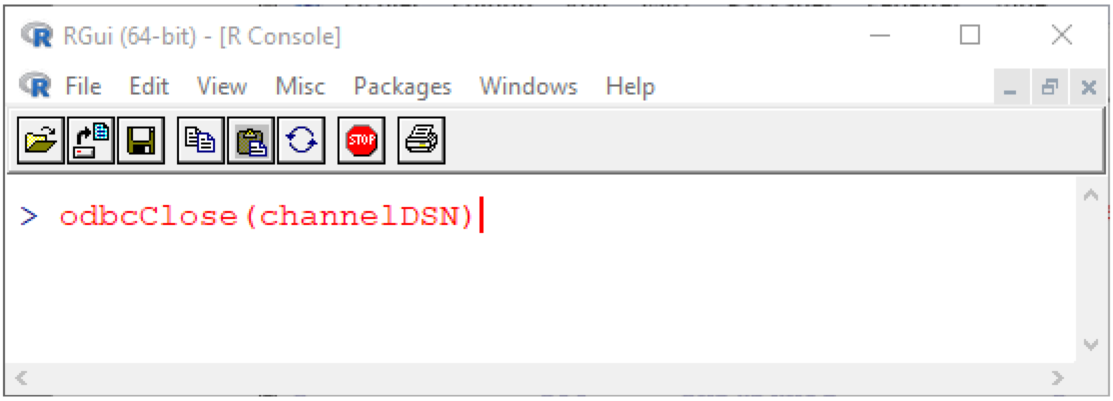
Donc c'est tout bon!

Attention!!! Si le fichier Microsoft Access est en lecture seule ou ouvert par un autre utilisateur l'accès par R ne sera pas possible.

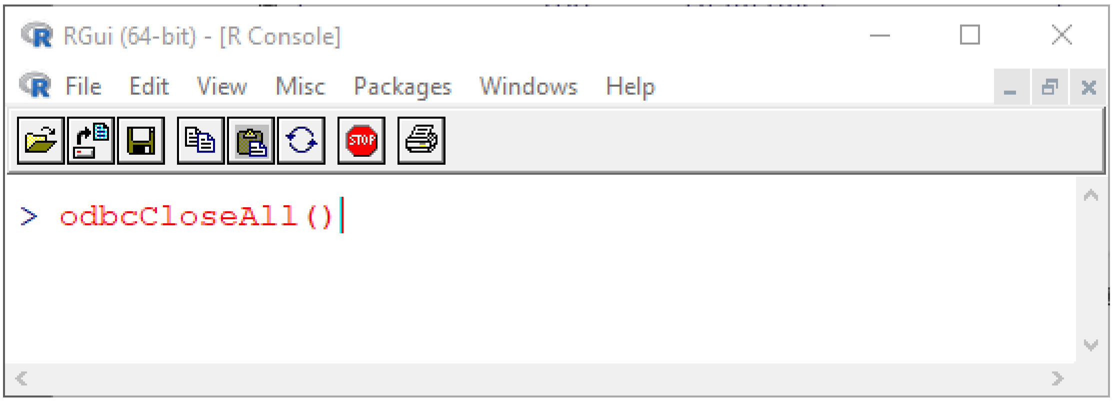
La méthode est presque exactement la même pour des *.accdb à part qu'on prendra au début un driver *.odbc pour les fichiers *.accdb et non *.mdb et que le code change un peu pour car nous devrons utiliser la commande **odbcConnectAccess2007( )** (même si c'est pour des fichiers d'Access 2010 ou du 2013):



Pour fermer la connexion, on utilisera:



Ou:



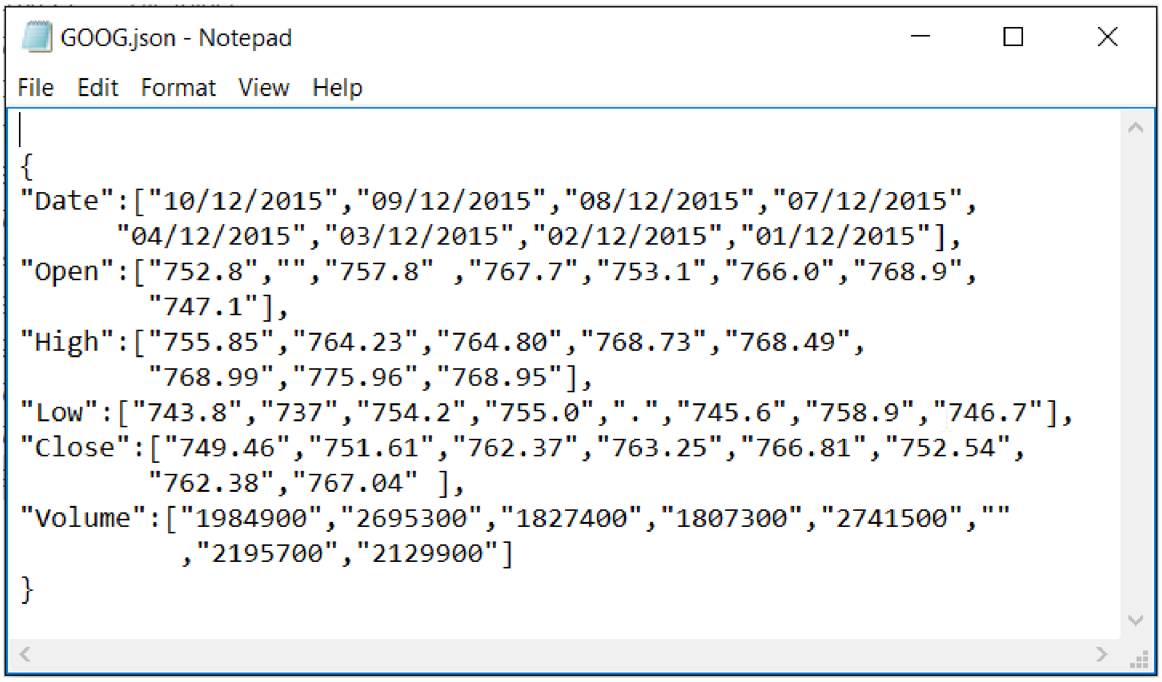
Enfin, gardez à l'esprit que vous ne pourrez pas importer plus de données que la quantité de mémoire vive (RAM) qui est libre sur votre ordinateur!!!!

Exemple 49.: Importer des données JSON

R 3.2.1

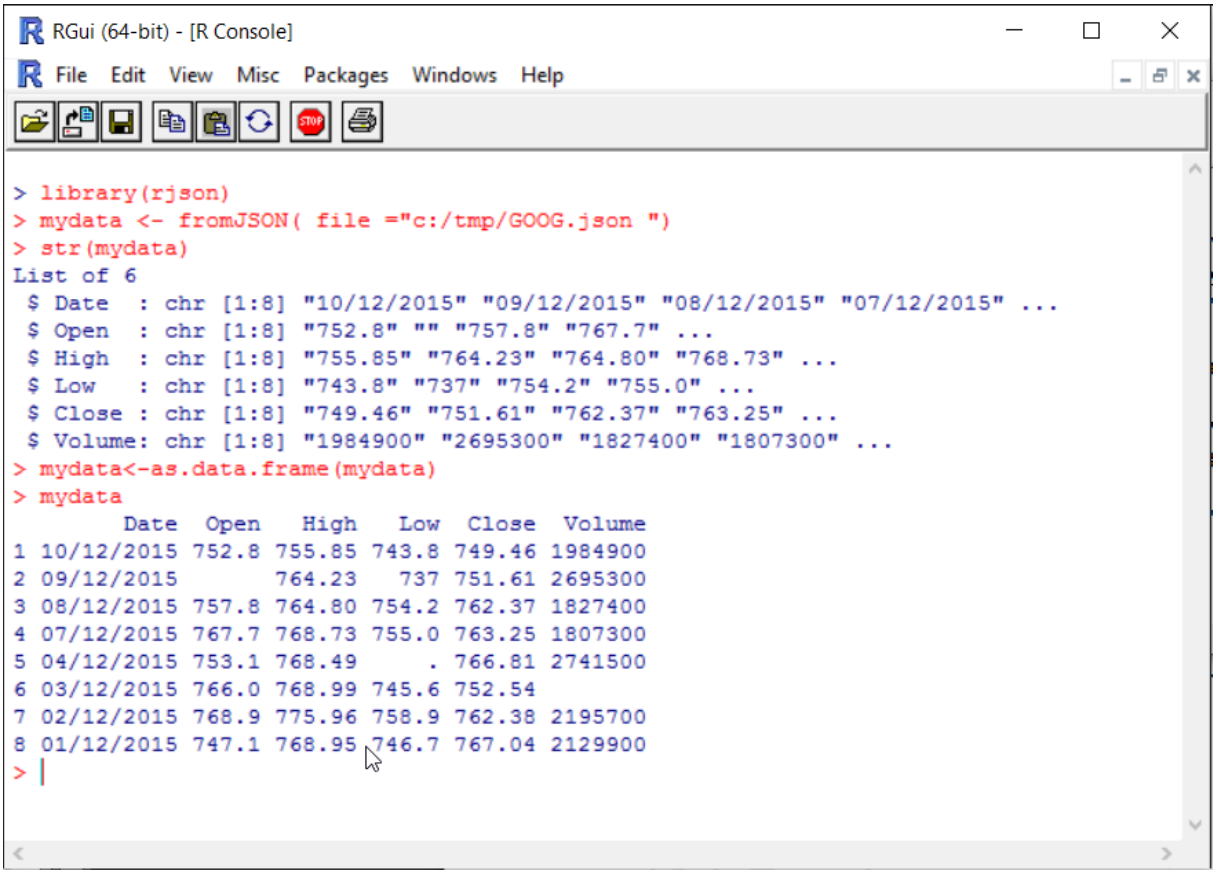
Un fichier JSON (JavaScript Object Notation) est un fichier contenant des données structurées comme du XML mais qui dérive de la notation des objets javascript. Pour importer des données d'un fichier *.json, il faut d'abor installer et charger le package **rjson** et avec la fonction **fromJSON( ),** on importe les données sous forme de liste.

Considérons maintenant le fichier *.json suivant:

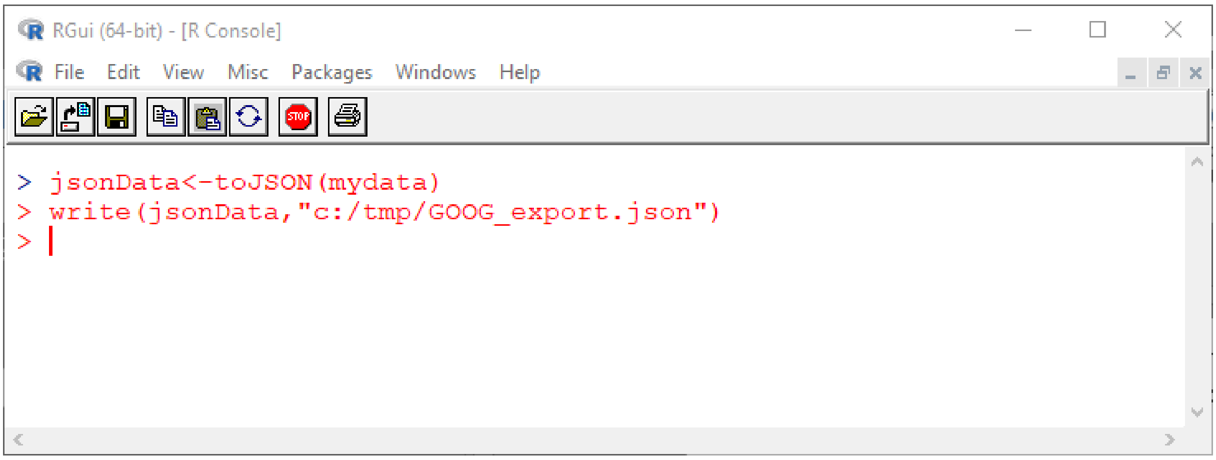



```
{
"Date":["10/12/2015","09/12/2015","08/12/2015","07/12/2015",
      "04/12/2015","03/12/2015","02/12/2015","01/12/2015"],
"Open":["752.8","","757.8" ,"767.7","753.1","766.0","768.9",
        "747.1"],
"High":["755.85","764.23","764.80","768.73","768.49",
        "768.99","775.96","768.95"],
"Low":["743.8","737","754.2","755.0",".","745.6","758.9","746.7"],
"Close":["749.46","751.61","762.37","763.25","766.81","752.54",
        "762.38","767.04" ],
"Volume":["1984900","2695300","1827400","1807300","2741500",""
          ,"2195700","2129900"]
}
```

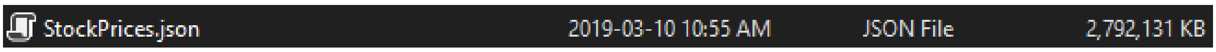

L'import se fait relativement facilement avec la fonction **fromJSON( )**:



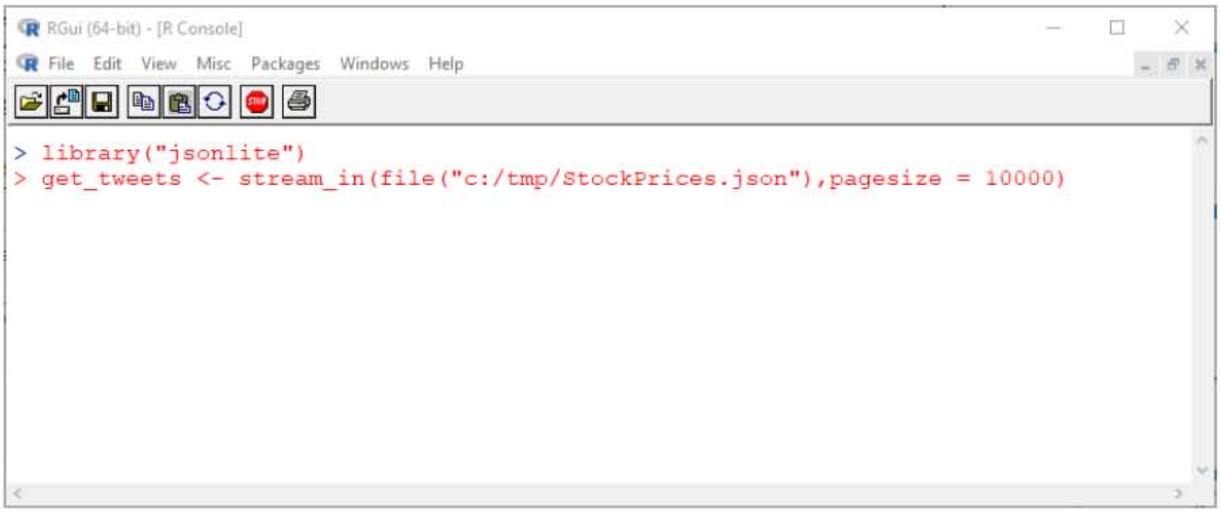
On peut inverser aussi la procédure en exportant du JSON à parti d'un dataframe avec la fonction **toJSON( )**:



Si le fichier est nettement plus gros, comme le suivant:



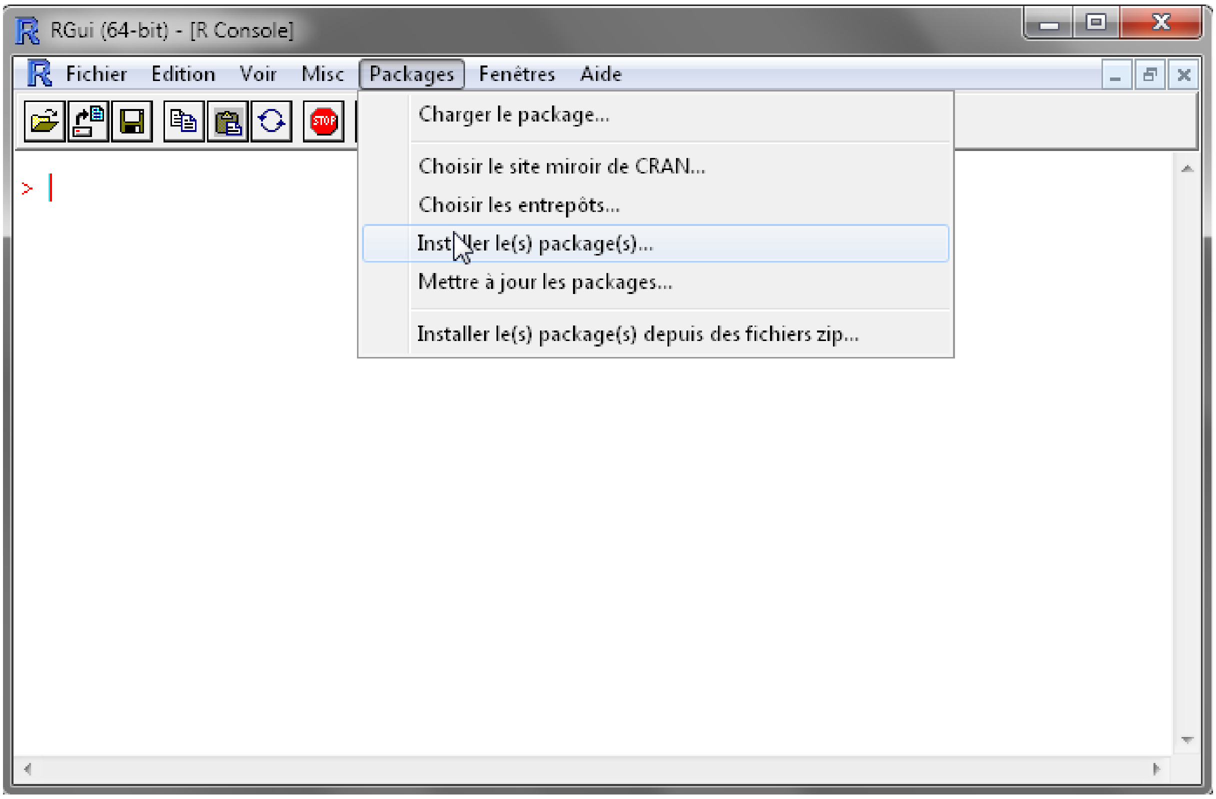
Alors il faudra changer de package pour passer à **jsonlite** et sa fonction **stream_in( )** et être beaucoup plus patient (entre 5 et 10 minutes sur un ordinateur contemporain):


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("jsonlite")
> get_tweets <- stream_in(file("c:/tmp/StockPrices.json"),pagesize = 10000)

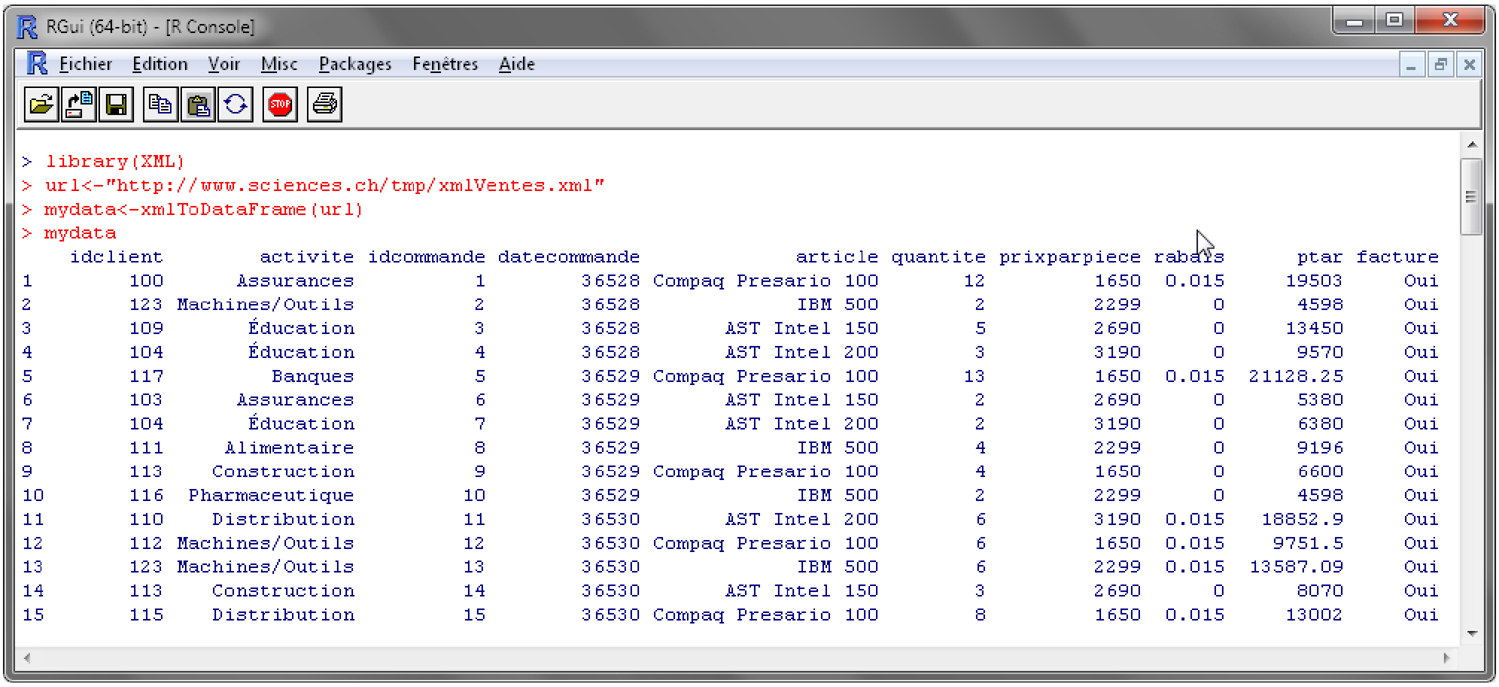
Exemple 50.: Importer des données XML ou HTML du web

R 3.0.2

D'abord il va nous falloir installer le package **XML**:

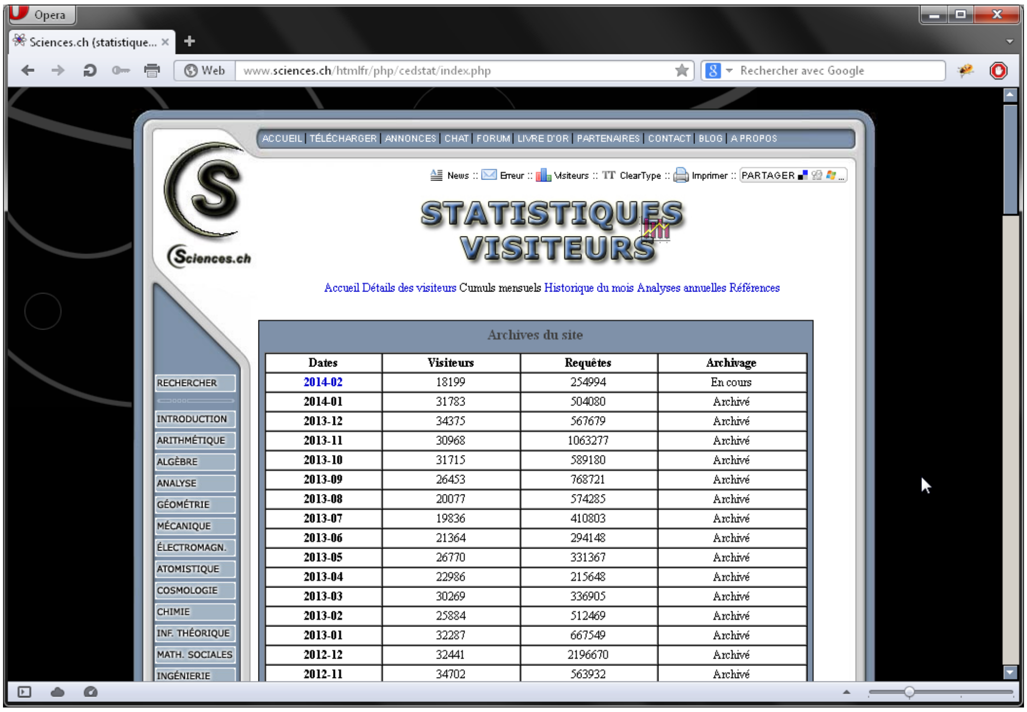


Enfin, nous utilisons la commande **xmlToDataFrame( )** :

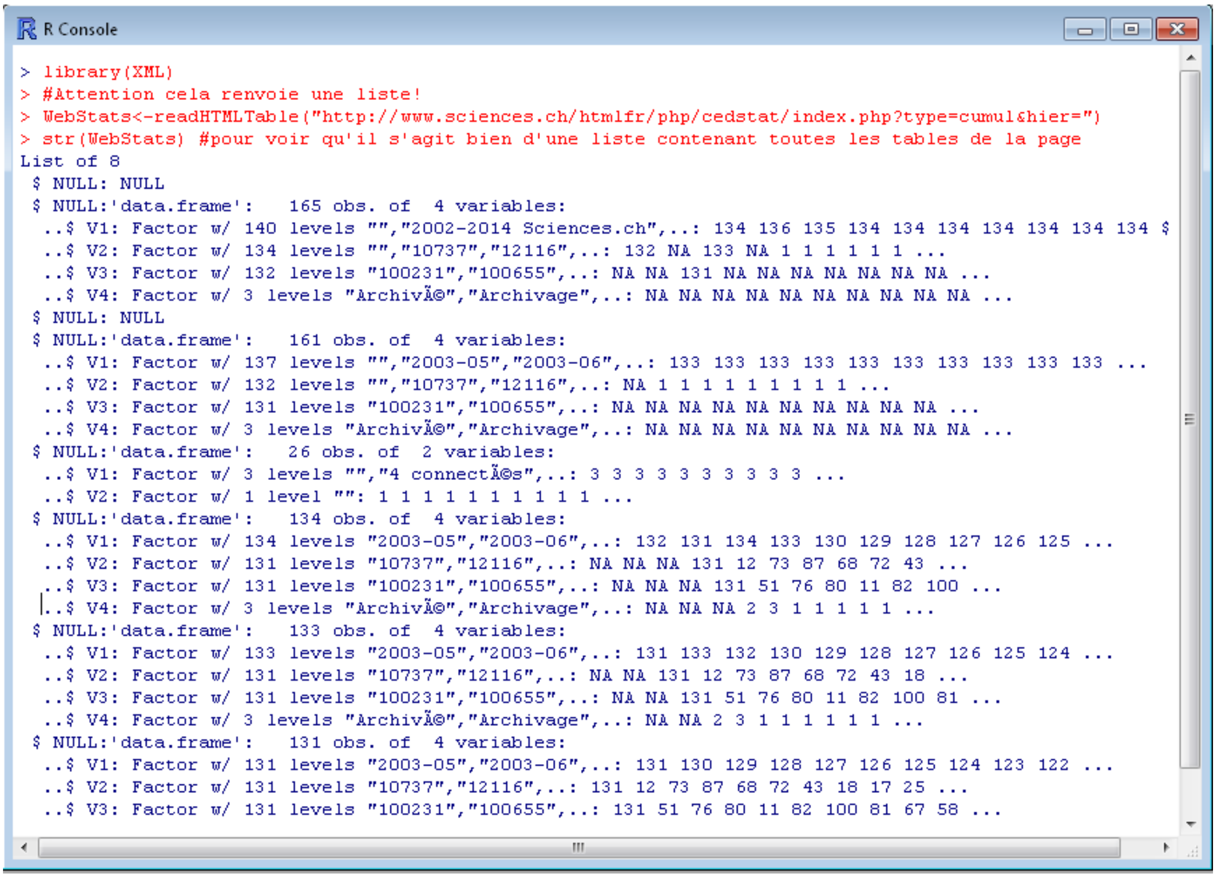


Un cas de plus en plus fréquent consiste à aspirer des données provenant des tables HTML et nous pouvons faire cela toujours avec le même package XML.

Considérons la page Internet suivante:



Nous voulons importer les données dans R.

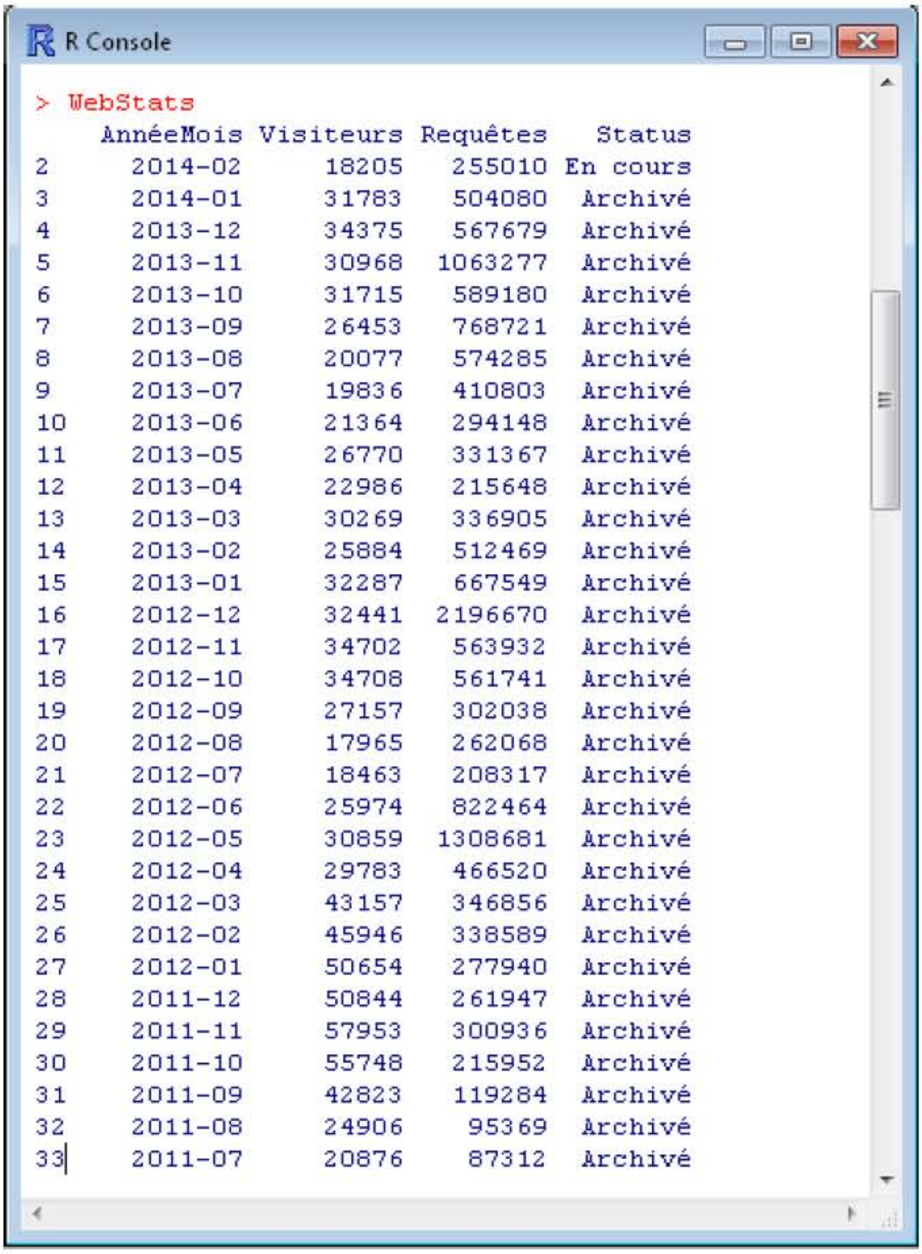


```
> library(XML)
> #Attention cela renvoie une liste!
> WebStats<-readHTMLTable("http://www.sciences.ch/htmlfr/php/cedstat/index.php?type=cumul&hier=")
> str(WebStats) #pour voir qu'il s'agit bien d'une liste contenant toutes les tables de la page
List of 8
 $ NULL: NULL
 $ NULL:'data.frame':   165 obs. of  4 variables:
  ..$ V1: Factor w/ 140 levels "","2002-2014 Sciences.ch",..: 134 136 135 134 134 134 134 134 134 134 $
  ..$ V2: Factor w/ 134 levels "","10737","12116",..: 132 NA 133 NA 1 1 1 1 1 1 ...
  ..$ V3: Factor w/ 132 levels "100231","100655",..: NA NA 131 NA NA NA NA NA NA NA ...
  ..$ V4: Factor w/ 3 levels "ArchivÃ©","Archivage",..: NA NA NA NA NA NA NA NA NA NA ...
 $ NULL: NULL
 $ NULL:'data.frame':   161 obs. of  4 variables:
  ..$ V1: Factor w/ 137 levels "","2003-05","2003-06",..: 133 133 133 133 133 133 133 133 133 133 ...
  ..$ V2: Factor w/ 132 levels "","10737","12116",..: NA 1 1 1 1 1 1 1 1 1 ...
  ..$ V3: Factor w/ 131 levels "100231","100655",..: NA NA NA NA NA NA NA NA NA NA ...
  ..$ V4: Factor w/ 3 levels "ArchivÃ©","Archivage",..: NA NA NA NA NA NA NA NA NA NA ...
 $ NULL:'data.frame':   26 obs. of  2 variables:
  ..$ V1: Factor w/ 3 levels "","4 connectÃ©s",..: 3 3 3 3 3 3 3 3 3 3 ...
  ..$ V2: Factor w/ 1 level "": 1 1 1 1 1 1 1 1 1 1 ...
 $ NULL:'data.frame':   134 obs. of  4 variables:
  ..$ V1: Factor w/ 134 levels "2003-05","2003-06",..: 132 131 134 133 130 129 128 127 126 125 ...
  ..$ V2: Factor w/ 131 levels "10737","12116",..: NA NA NA 131 12 73 87 68 72 43 ...
  ..$ V3: Factor w/ 131 levels "100231","100655",..: NA NA NA 131 51 76 80 11 82 100 ...
  ..$ V4: Factor w/ 3 levels "ArchivÃ©","Archivage",..: NA NA NA 2 3 1 1 1 1 1 ...
 $ NULL:'data.frame':   133 obs. of  4 variables:
  ..$ V1: Factor w/ 133 levels "2003-05","2003-06",..: 131 133 132 130 129 128 127 126 125 124 ...
  ..$ V2: Factor w/ 131 levels "10737","12116",..: NA NA 131 12 73 87 68 72 43 18 ...
  ..$ V3: Factor w/ 131 levels "100231","100655",..: NA NA 131 51 76 80 11 82 100 81 ...
  ..$ V4: Factor w/ 3 levels "ArchivÃ©","Archivage",..: NA NA 2 3 1 1 1 1 1 1 ...
 $ NULL:'data.frame':   131 obs. of  4 variables:
  ..$ V1: Factor w/ 131 levels "2003-05","2003-06",..: 131 130 129 128 127 126 125 124 123 122 ...
  ..$ V2: Factor w/ 131 levels "10737","12116",..: 131 12 73 87 68 72 43 18 17 25 ...
  ..$ V3: Factor w/ 131 levels "100231","100655",..: 131 51 76 80 11 82 100 81 67 58 ...
```

Nous poursuivons:

```
R Console
> #Après observation de la liste on voit que c'est le tableau de la web qui nous intéresse
> WebStats<-as.data.frame(WebStats[[8]])
> #on regarde le résultat
> str(WebStats)
'data.frame':	131 obs. of  4 variables:
 $ V1: Factor w/ 131 levels "2003-05","2003-06",..: 131 130 129 128 127 126 125 124 123 122 ...
 $ V2: Factor w/ 131 levels "10737","12116",..: 131 12 73 87 68 72 43 18 17 25 ...
 $ V3: Factor w/ 131 levels "100231","100655",..: 131 51 76 80 11 82 100 81 67 58 ...
 $ V4: Factor w/ 3 levels "ArchivÃ©","Archivage",..: 2 3 1 1 1 1 1 1 1 1 ...
> #On enlève la première ligne de données qui contient les en-têtes du tableau
> WebStats<-WebStats[-1,]
> #On renomme les colonnes
> names(WebStats)<-c("AnnéeMois","Visiteurs","Requêtes","Status")
> #On nettoie un problème d'encodage de caractères
> WebStats$Status<-gsub("Ã©","é",WebStats$Status)
> #et enfin on change le type de certaines colonnes (ce qui n'est pas toujours trivial...)
> WebStats$AnnéeMois<-as.character(WebStats$AnnéeMois)
> WebStats$Visiteurs<-as.numeric(levels(WebStats$Visiteurs))[WebStats$Visiteurs]
Message d'avis :
NAs introduits lors de la conversion automatique
> WebStats$Requêtes<-as.numeric(levels(WebStats$Requêtes))[WebStats$Requêtes]
Message d'avis :
NAs introduits lors de la conversion automatique
> str(WebStats)
'data.frame':	130 obs. of  4 variables:
 $ AnnéeMois: chr  "2014-02" "2014-01" "2013-12" "2013-11" ...
 $ Visiteurs: num  18205 31783 34375 30968 31715 ...
 $ Requêtes : num  255010 504080 567679 1063277 589180 ...
 $ Status   : chr  "En cours" "Archivé" "Archivé" "Archivé" ...
> 
```

On regarde le résultat qui est nickel:



```
> WebStats
   AnnéeMois Visiteurs Requêtes   Status
2    2014-02     18205   255010 En cours
3    2014-01     31783   504080  Archivé
4    2013-12     34375   567679  Archivé
5    2013-11     30968  1063277  Archivé
6    2013-10     31715   589180  Archivé
7    2013-09     26453   768721  Archivé
8    2013-08     20077   574285  Archivé
9    2013-07     19836   410803  Archivé
10   2013-06     21364   294148  Archivé
11   2013-05     26770   331367  Archivé
12   2013-04     22986   215648  Archivé
13   2013-03     30269   336905  Archivé
14   2013-02     25884   512469  Archivé
15   2013-01     32287   667549  Archivé
16   2012-12     32441  2196670  Archivé
17   2012-11     34702   563932  Archivé
18   2012-10     34708   561741  Archivé
19   2012-09     27157   302038  Archivé
20   2012-08     17965   262068  Archivé
21   2012-07     18463   208317  Archivé
22   2012-06     25974   822464  Archivé
23   2012-05     30859  1308681  Archivé
24   2012-04     29783   466520  Archivé
25   2012-03     43157   346856  Archivé
26   2012-02     45946   338589  Archivé
27   2012-01     50654   277940  Archivé
28   2011-12     50844   261947  Archivé
29   2011-11     57953   300936  Archivé
30   2011-10     55748   215952  Archivé
31   2011-09     42823   119284  Archivé
32   2011-08     24906    95369  Archivé
33   2011-07     20876    87312  Archivé
```

Exemple 51.: Importer des données XES (*.xes)

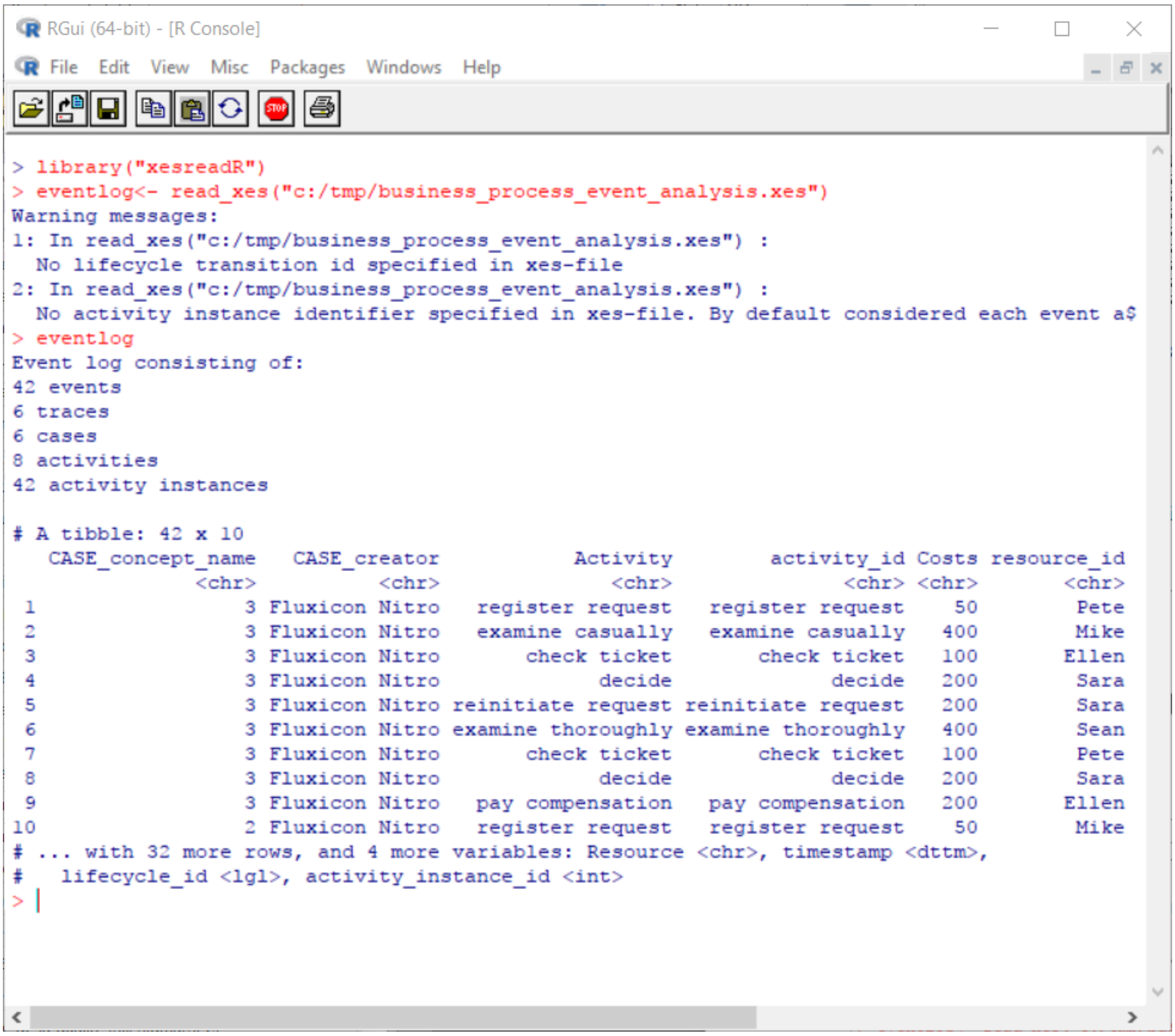
R 3.4.1

Nous allons pour importer un fichier un fichier *.xes (eXtensible Event Stream[7]) utiliser le package **xesreadR** avec le contenu suivant:

	A	B	C	D	E	F
1	Case ID	Event ID	dd-MM-yyy	Activity	Resource	Costs
2	1	35654423	30-12-2010:1	register req	Pete	50
3	1	35654424	31-12-2010:1	examine th	Sue	400
4	1	35654425	05-01-2011:1	check ticke	Mike	100
5	1	35654426	06-01-2011:1	decide	Sara	200
6	1	35654427	07-01-2011:1	reject reque	Pete	200
7	2	35654483	30-12-2010:1	register req	Mike	50
8	2	35654485	30-12-2010:1	check ticke	Mike	100
9	2	35654487	30-12-2010:1	examine ca	Sean	400
10	2	35654488	05-01-2011:1	decide	Sara	200
11	2	35654489	08-01-2011:1	pay compe	Ellen	200
12	3	35654521	30-12-2010:1	register req	Pete	50
13	3	35654522	30-12-2010:1	examine ca	Mike	400
14	3	35654524	30-12-2010:1	check ticke	Ellen	100
15	3	35654525	06-01-2011:0	decide	Sara	200
16	3	35654526	06-01-2011:1	reinitiate re	Sara	200
17	3	35654527	06-01-2011:1	examine th	Sean	400
18	3	35654530	08-01-2011:1	check ticke	Pete	100
19	3	35654531	09-01-2011:0	decide	Sara	200
20	3	35654533	15-01-2011:1	pay compe	Ellen	200
21	4	35654641	06-01-2011:1	register req	Pete	50
22	4	35654643	07-01-2011:1	check ticke	Mike	100
23	4	35654644	08-01-2011:1	examine th	Sean	400
24	4	35654645	09-01-2011:1	decide	Sara	200
25	4	35654647	12-01-2011:1	reject reque	Ellen	200
26	5	35654711	06-01-2011:0	register req	Ellen	50
27	5	35654712	07-01-2011:1	examine ca	Mike	400
28	5	35654714	08-01-2011:1	check ticke	Pete	100
29	5	35654715	10-01-2011:1	decide	Sara	200
30	5	35654716	11-01-2011:16	reinitiate re	Sara	200
31	5	35654718	14-01-2011:1	check ticke	Ellen	100
32	5	35654719	16-01-2011:1	examine ca	Mike	400
33	5	35654720	19-01-2011:11	decide	Sara	200
34	5	35654721	20-01-2011:1	reinitiate re	Sara	200
35	5	35654722	21-01-2011:0	examine ca	Sue	400
36	5	35654724	21-01-2011:11	check ticke	Pete	100
37	5	35654725	23-01-2011:1	decide	Sara	200
38	5	35654726	24-01-2011:1	reject reque	Mike	200
39	6	35654871	06-01-2011:1	register req	Mike	50
40	6	35654873	06-01-2011:1	examine ca	Ellen	400
41	6	35654874	07-01-2011:1	check ticke	Mike	100
42	6	35654875	07-01-2011:1	decide	Sara	200
43	6	35654877	16-01-2011:11	pay compe	Mike	200

Donc nous chargeons d'abord les packages et ensuite le fichier *.xes d'exemple à l'aide de la fonction **read_xes( )** :

[7] Il s'agit du format normalisé des logiciels qui stockent en temps réel ou manuellement les étapes d'avancement de processus business.



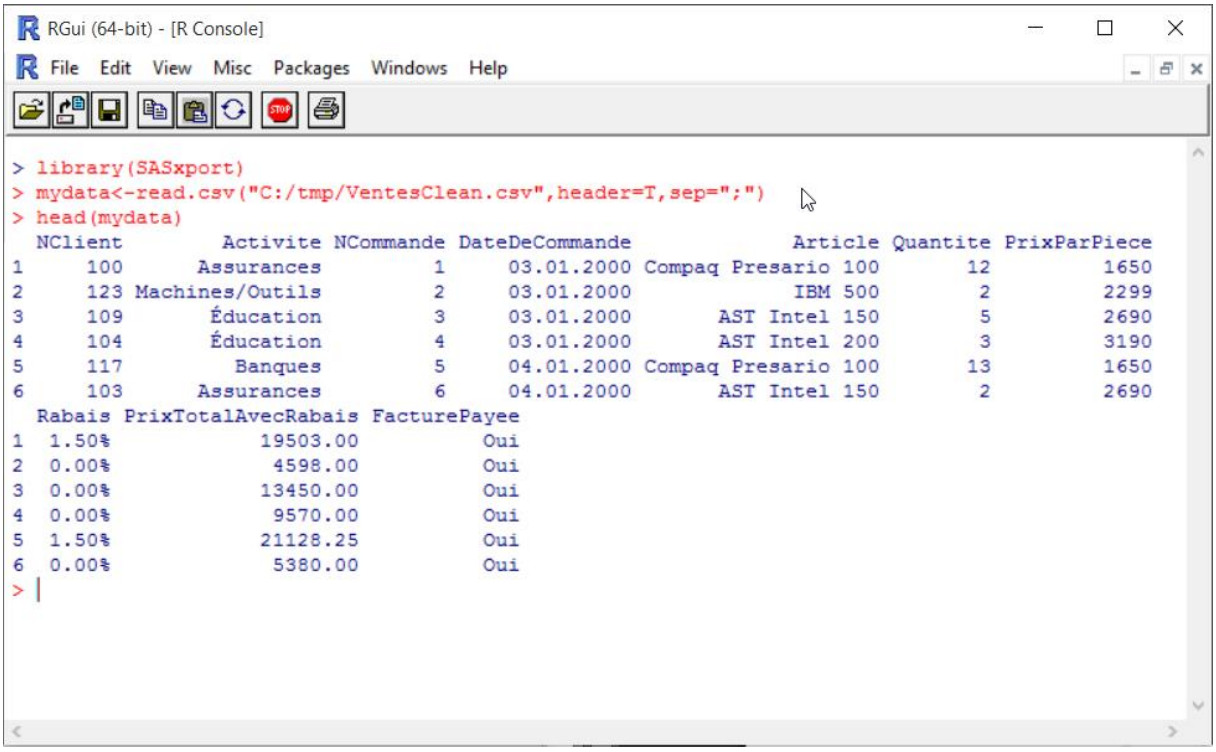
et voilà! Pour en savoir plus sur l'utilité de ce type de données, le lecteur pourra se référer aux exemples sur la business process analysis donnés plus loin à la page 1986.

Exemple 52.: Exporter/Lire des fichiers SAS (*.dat)

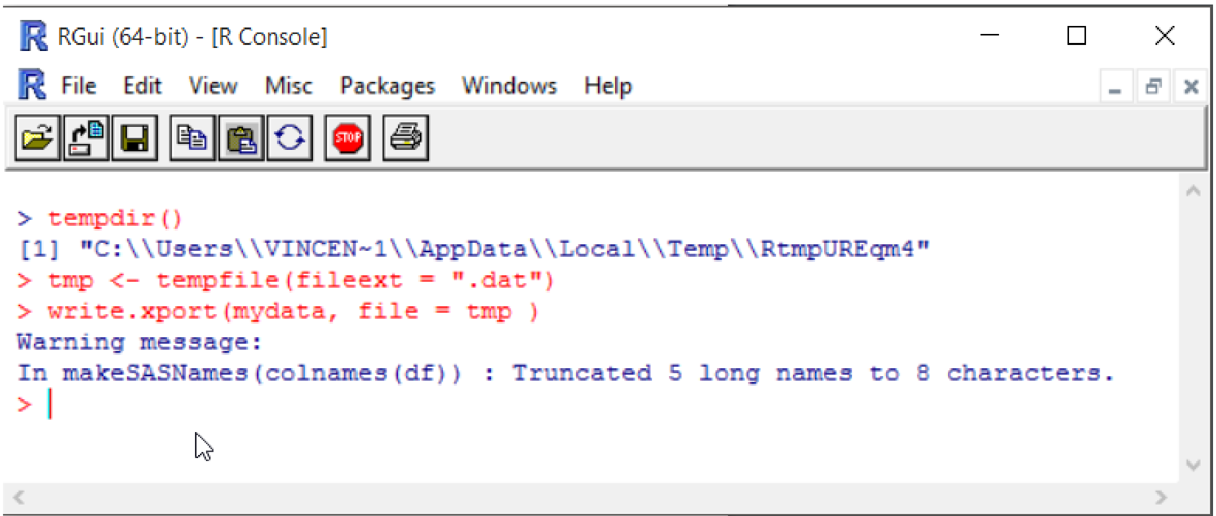
R 3.2.3

Ceux qui travaillent dans le pharma connaissent bien l'exigence de la FDA (Federal Drug Administration) à propos du format des données analysées. Heureusement et bien évidemment, il existe un package R pour écrire et lire ce type de fichier: le package **SASxport**.

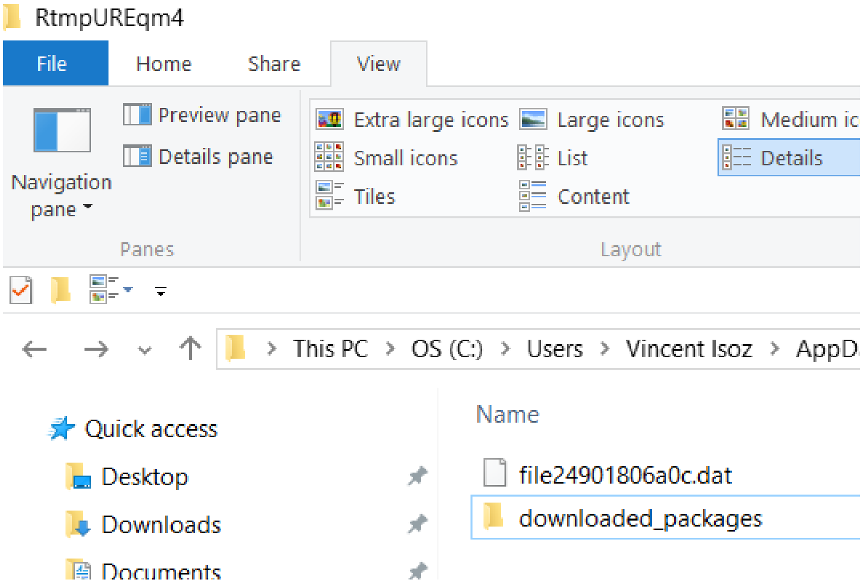
D'abord nous chargeons le package avec un jeu de données se trouvant dans un *.csv:



Ensuite, nous exportons avec la fonction **write.xport( )**:

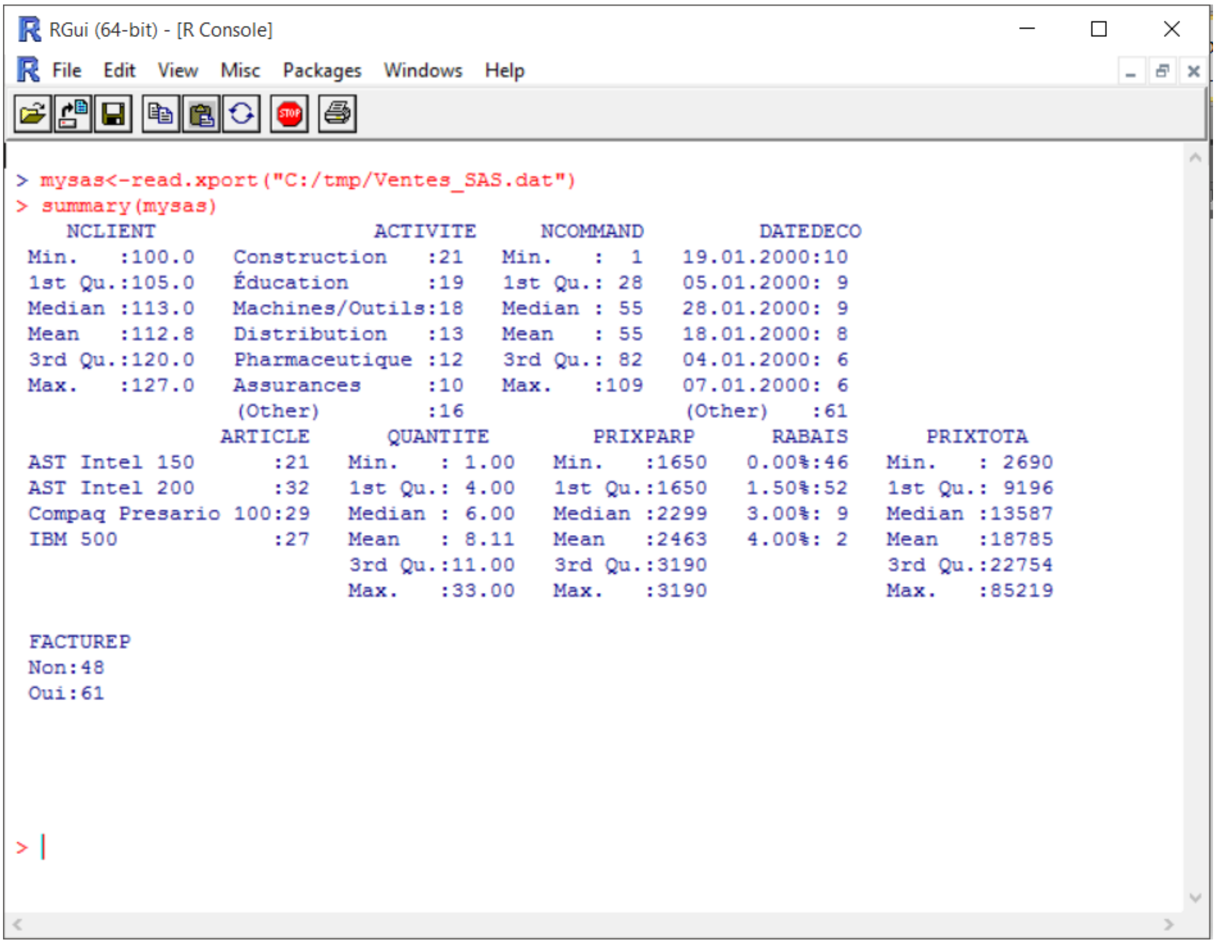


Nous allons dans le dossier concerné pour voir que le fichier *.dat est bien là:



Ce qui est bien le cas (c'est un fichier binaire donc inutile de penser à l'ouvrir dans le bloc-notes)!

Après avoir renommé et déplacer el fichier, nous pouvons aussi le lire avec la fonction **read.xport( ) :**



Exemple 53.: Exporter/Lire des fichiers SAS (*.xpt)

R 3.6.0

Ceux qui travaillent dans le pharma connaissent bien les fichiers *.xpt de SAS (Transport File Format) qui restent actuellement la référence du BFSS (Behavioral Risk Factor Surveillance System). Heureusement et bien évidemment, il existe un package R pour écrire et lire ce type de fichier: le package **foreign** et sa fonction **read.xport( )**.

Voyons ceci:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("foreign")
> BFSS<-read.xport("LLCP2014.xpt")
> str(BFSS)
'data.frame':   464664 obs. of  279 variables:
 $ X_STATE  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ FMONTH   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ IDATE    : Factor w/ 399 levels "01022014","01022015",..: 31 11 15 11 29 1 9 19 1 13 ...
 $ IMONTH   : Factor w/ 12 levels "01","02","03",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ IDAY     : Factor w/ 31 levels "01","02","03",..: 17 7 9 7 16 2 6 11 2 8 ...
 $ IYEAR    : Factor w/ 2 levels "2014","2015": 1 1 1 1 1 1 1 1 1 1 ...
 $ DISPCODE : num  1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 ...
 $ SEQNO    : num  2.01e+09 2.01e+09 2.01e+09 2.01e+09 2.01e+09 ...
 $ X_PSU    : num  2.01e+09 2.01e+09 2.01e+09 2.01e+09 2.01e+09 ...
 $ CTELENUM : num  1 1 1 1 1 1 1 1 1 1 ...
 $ PVTRESD1 : num  1 1 1 1 1 1 1 1 1 1 ...
 $ COLGHOUS : num  NA NA NA NA NA NA NA NA NA NA ...
 $ STATERES : num  1 1 1 1 1 1 1 1 1 1 ...
 $ LADULT   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ NUMADULT : num  2 1 2 1 2 2 2 1 2 2 ...
 $ NUMMEN   : num  1 1 1 0 1 1 1 0 1 1 ...
 $ NUMWOMEN : num  1 0 1 1 1 1 1 1 1 1 ...
 $ GENHLTH  : num  5 3 1 3 3 3 2 3 2 1 ...
 $ PHYSHLTH : num  25 7 88 77 2 5 2 3 88 1 ...
 $ MENTHLTH : num  5 14 88 77 88 4 88 88 14 88 ...
 $ POORHLTH : num  88 7 NA 88 88 6 88 88 88 88 ...
 $ HLTHPLN1 : num  1 1 1 1 1 1 1 1 2 1 ...
 $ PERSDOC2 : num  1 1 1 1 1 2 1 1 1 1 ...
 $ MEDCOST  : num  1 2 2 2 2 2 2 2 1 2 ...
 $ CHECKUP1 : num  1 1 1 1 1 1 1 1 1 1 ...
 $ EXERANY2 : num  2 1 1 2 2 1 1 1 1 1 ...
 $ SLEPTIM1 : num  9 6 8 8 8 7 7 7 10 8 ...
 $ CVDINFR4 : num  1 2 2 2 2 2 2 2 2 2 ...
 $ CVDCRHD4 : num  2 2 2 2 1 2 2 2 2 2 ...
 $ CVDSTRK3 : num  1 1 2 2 2 2 2 2 2 2 ...
 $ ASTHMA3  : num  2 2 2 1 2 1 2 2 2 2 ...
 $ ASTHNOW  : num  NA NA NA NA 1 NA 2 NA NA NA NA ...
 $ CHCSCNCR : num  2 2 2 2 2 2 2 2 2 2 ...
 $ CHCOCNCR : num  2 2 2 2 1 1 2 2 2 2 ...
 $ CHCCOPD1 : num  1 1 2 1 2 2 2 2 2 1 ...
 $ HAVARTH3 : num  1 1 2 1 1 2 1 1 1 2 ...
 $ ADDEPEV2 : num  2 1 2 1 2 2 2 2 1 2 ...
 $ CHCKIDNY : num  1 2 2 2 2 2 2 2 2 2 ...
 $ DIABETE3 : num  1 3 3 1 3 1 3 3 3 3 ...
 $ DIABAGE2 : num  61 NA NA 32 NA 70 NA NA NA NA ...
 $ LASTDEN3 : num  2 1 1 1 1 2 1 2 2 1 ...
 $ RMVTETH3 : num  2 1 1 8 8 2 1 1 3 1 ...
 $ VETERAN3 : num  2 2 2 2 2 1 2 2 2 1 ...
 $ MARITAL  : num  1 1 1 3 1 1 1 2 1 1 ...
 $ CHILDREN : num  88 88 88 88 88 88 1 88 88 88 ...
 $ EDUCA    : num  5 4 6 6 5 6 6 4 3 5 ...
 $ EMPLOY1  : num  8 8 1 7 7 7 7 8 8 1 ...
```

Exemple 54.: Importer des données de Stata (*.dta)

R 3.6.0

Comme pour SPSS, nous allons utiliser ici le package **foreign** mais sa fonction **read.dta( )**. Voyons-en un exemple:

Exemple 55.: Export/Importer des données parquet

R 3.6.0

Nous allons utiliser ici le package **arrow** mais et ses fonctions **write_parquet( )** pour créer un fichier parquet à partir d'un fichier CSV et ensuite **read_parquet( )** pour lire le fichier créé. Voyons-en un exemple:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

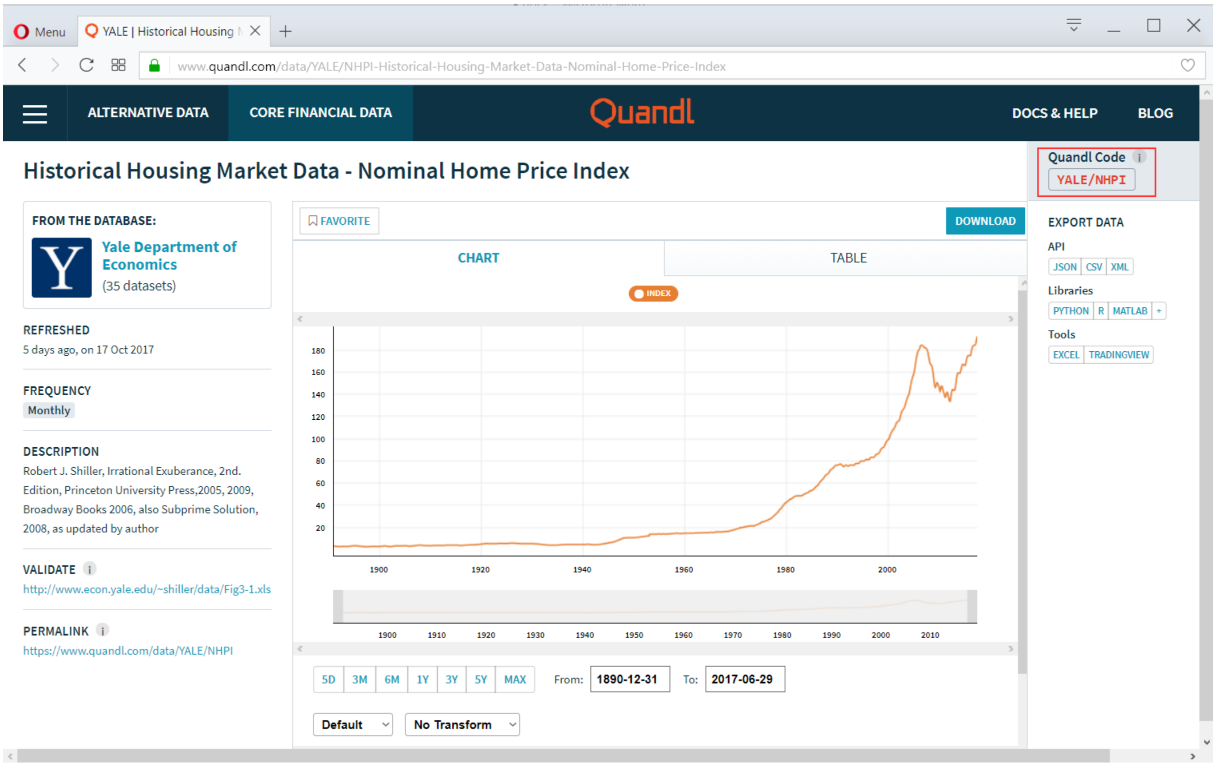
> library("arrow")
>
> start_time <- Sys.time()
> mydata<-read.csv("c:/tmp/Source_CSV_Ventes.csv",header=T,sep=";")
> end_time <- Sys.time()
> end_time - start_time
Time difference of 32.88602 secs
>
> start_time <- Sys.time()
> write_parquet(mydata, "c:/tmp/Source_Parquet_Ventes.parquet")
> end_time <- Sys.time()
> end_time - start_time
Time difference of 20.02654 secs
>
> start_time <- Sys.time()
> df <- read_parquet("c:/tmp/Source_Parquet_Ventes.parquet")
> end_time <- Sys.time()
> end_time - start_time
Time difference of 1.385324 secs
>
> str(df)
'data.frame':   12044587 obs. of  10 variables:
 $ ID                     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ N..Client              : num  100 123 119 101 117 121 102 122 104 107 ...
 $ Activité               : chr  "Assurances" "Machines/Outils" "Alimentaire" "Construction" ...
 $ Date.de.commande       : chr  "2003-6-3 00:00:00" "2009-10-17 00:00:00" "2004-5-9 00:00:00" "2003-$
 $ Article                : chr  "Compaq Presario 100" "IBM 500" "IBM 500" "Compaq Presario 100" ...
 $ Quantité               : num  12 2 22 11 8 23 24 23 4 24 ...
 $ Prix.par.pièce         : num  1650 2299 2299 1650 2299 ...
 $ Rabais.                : num  0.01 0 0.03 0 0.01 0.03 0 0 0.03 0.03 ...
 $ Prix.total.avec.rabais: num  19503 4598 49061 18150 18116 ...
 $ Facture.payée          : chr  "Oui" "Oui" "Oui" "Oui" ...
> |
```

Exemple 56.: Importer des données de Quandl

R 3.4.1

L'objectif ici va être d'importer des données de Quandl qui est un site tout aussi fameux que Yahoo en ce qui concerne les données (entre autres…), financières.

Nous souhaitons importer les données de la page suivante:



Observez bien sur la page web ci-dessus le *Quandl Code* que nous avons mis en évidence par un rectangle rouge. C'est ce que nous allons utiliser pour importer les données dans R avec le package **Quandl**.

Attention!!! Sans compte Quandl, le nombre d'appels depuis R est limité par jour!

```
> library(Quandl)
> mydata<-Quandl("YALE/NHPI", type="ts", start_date="1990-01-01")

> mydata
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1990  76.53  76.59  76.79  77.04  77.30  77.50  77.56  77.47  77.22  76.91  76.38  75.97
1991  75.53  75.25  75.14  75.32  75.77  76.25  76.51  76.59  76.57  76.29  76.04  75.83
1992  75.70  75.65  75.81  76.08  76.40  76.60  76.71  76.74  76.63  76.60  76.59  76.47
1993  76.40  76.33  76.39  76.67  77.04  77.52  77.89  78.14  78.20  78.18  78.18  78.12
1994  78.21  78.21  78.36  78.71  79.22  79.72  80.08  80.33  80.35  80.31  80.18  80.09
1995  80.04  80.01  80.10  80.39  80.71  81.09  81.43  81.68  81.76  81.75  81.66  81.54
1996  81.45  81.43  81.69  82.15  82.63  83.08  83.45  83.68  83.73  83.66  83.59  83.52
1997  83.56  83.62  83.94  84.35  84.90  85.43  85.88  86.19  86.34  86.44  86.67  86.88
1998  87.20  87.41  87.91  88.59  89.41  90.23  90.89  91.44  91.78  92.03  92.23  92.47
1999  92.75  93.01  93.64  94.47  95.40  96.40  97.23  97.93  98.44  98.87  99.19  99.58
2000 100.00 100.57 101.47 102.54 103.70 104.86 105.72 106.52 107.14 107.73 108.29 108.79
2001 109.22 109.64 110.40 111.25 112.20 113.27 114.23 114.99 115.47 115.68 115.84 116.06
2002 116.44 116.92 117.93 119.21 120.79 122.33 123.69 124.73 125.49 126.14 126.64 127.15
2003 127.65 128.33 129.31 130.49 131.84 133.23 134.65 135.96 137.08 137.98 138.77 139.63
2004 140.71 142.04 144.09 146.18 148.34 150.52 152.34 153.81 155.11 156.30 157.50 158.67
2005 160.14 161.93 164.58 167.00 169.54 172.01 174.10 175.92 177.61 178.75 179.68 180.11
2006 180.83 181.51 182.75 183.65 184.38 184.55 184.62 184.42 184.21 184.06 183.64 183.24
2007 182.73 182.48 182.20 182.14 181.89 181.55 181.00 180.25 179.13 177.54 175.18 173.36
2008 171.09 169.21 167.92 167.34 167.03 166.55 165.73 164.30 161.93 159.17 156.08 152.56
2009 149.37 147.63 146.53 146.96 148.19 149.81 150.76 150.68 149.64 148.60 147.96 146.69
2010 145.00 143.06 143.60 145.40 147.03 147.69 147.54 146.40 144.58 143.12 141.82 140.64
2011 139.05 137.75 137.79 139.15 140.68 141.93 142.33 141.77 140.16 138.40 136.65 135.16
2012 134.16 134.00 135.88 138.50 141.08 143.19 144.30 144.72 144.37 143.98 143.98 143.89
2013 144.33 145.18 147.99 151.00 153.90 156.46 158.32 159.43 159.71 159.61 159.42 159.34
2014 159.43 159.93 161.25 163.03 164.74 166.26 167.19 167.50 167.30 166.97 166.73 166.53
2015 166.32 166.70 168.17 170.05 171.94 173.54 174.59 175.04 175.16 175.17 175.28 175.26
2016 175.19 175.44 176.75 178.65 180.51 182.10 183.22 183.88 184.19 184.29 184.52 184.72
2017 184.95 185.33 186.80 188.78 190.79 192.60
>
```

et ensuite on peut s'amuser:

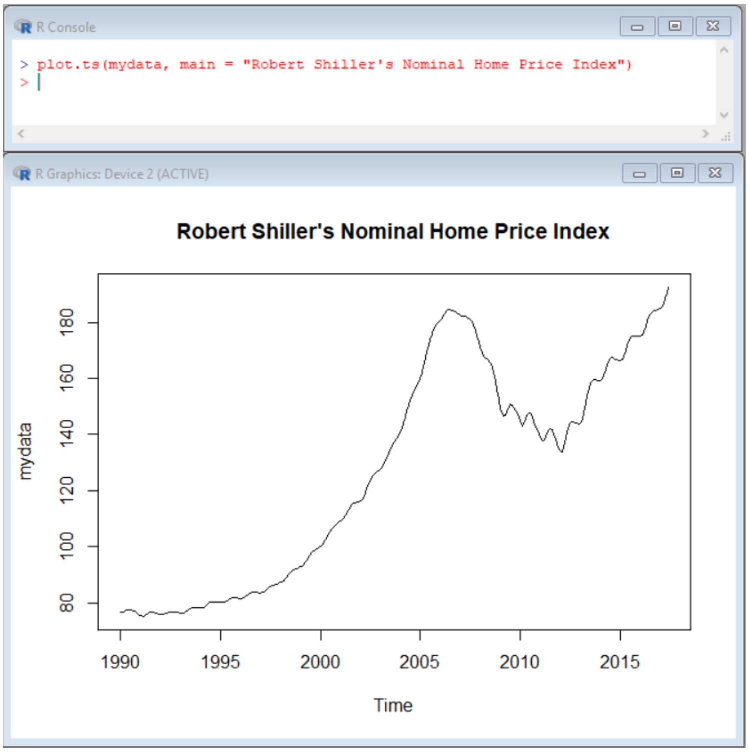
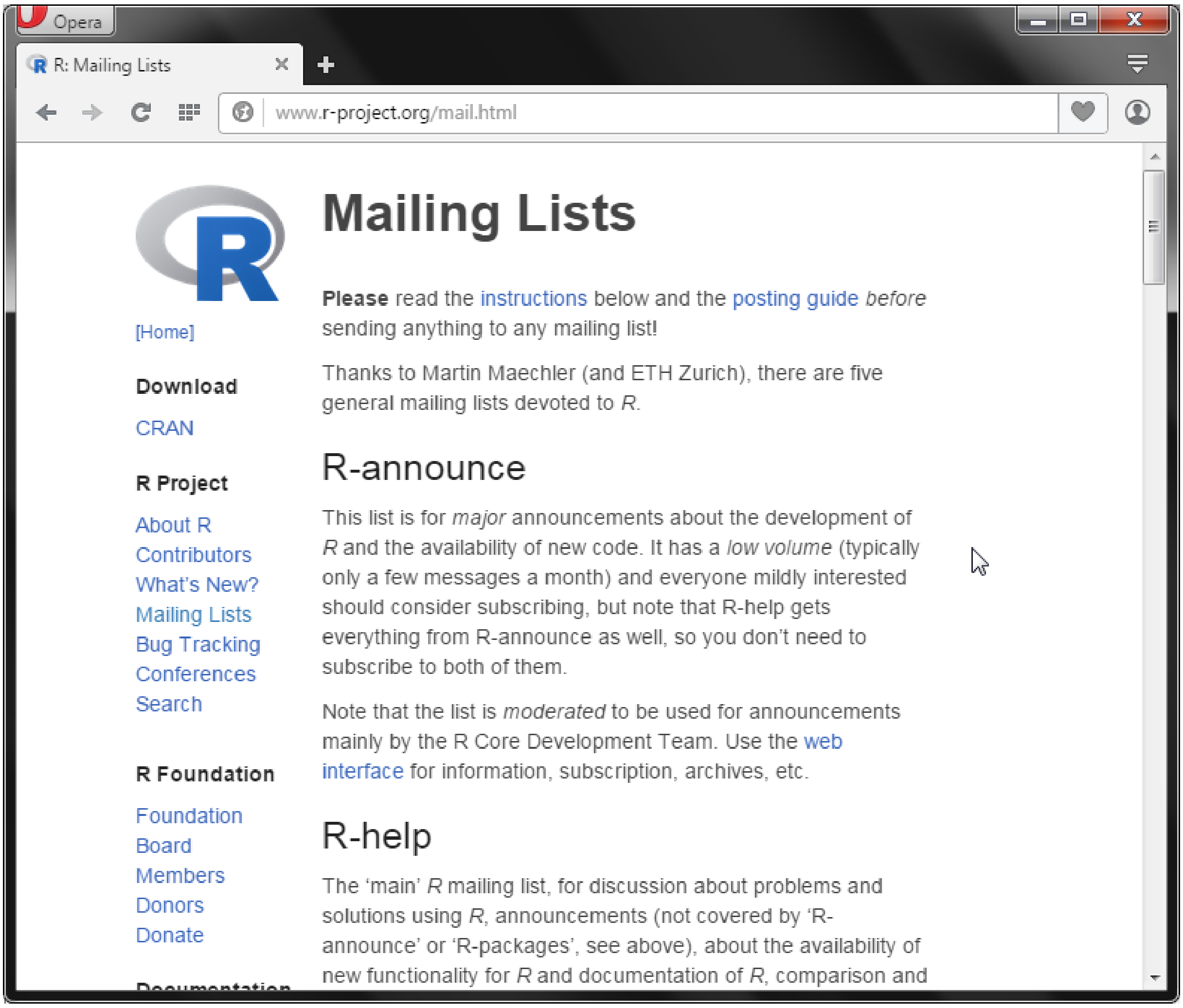


Figure 26 Graph d'une série temporelle avec données longitudinales

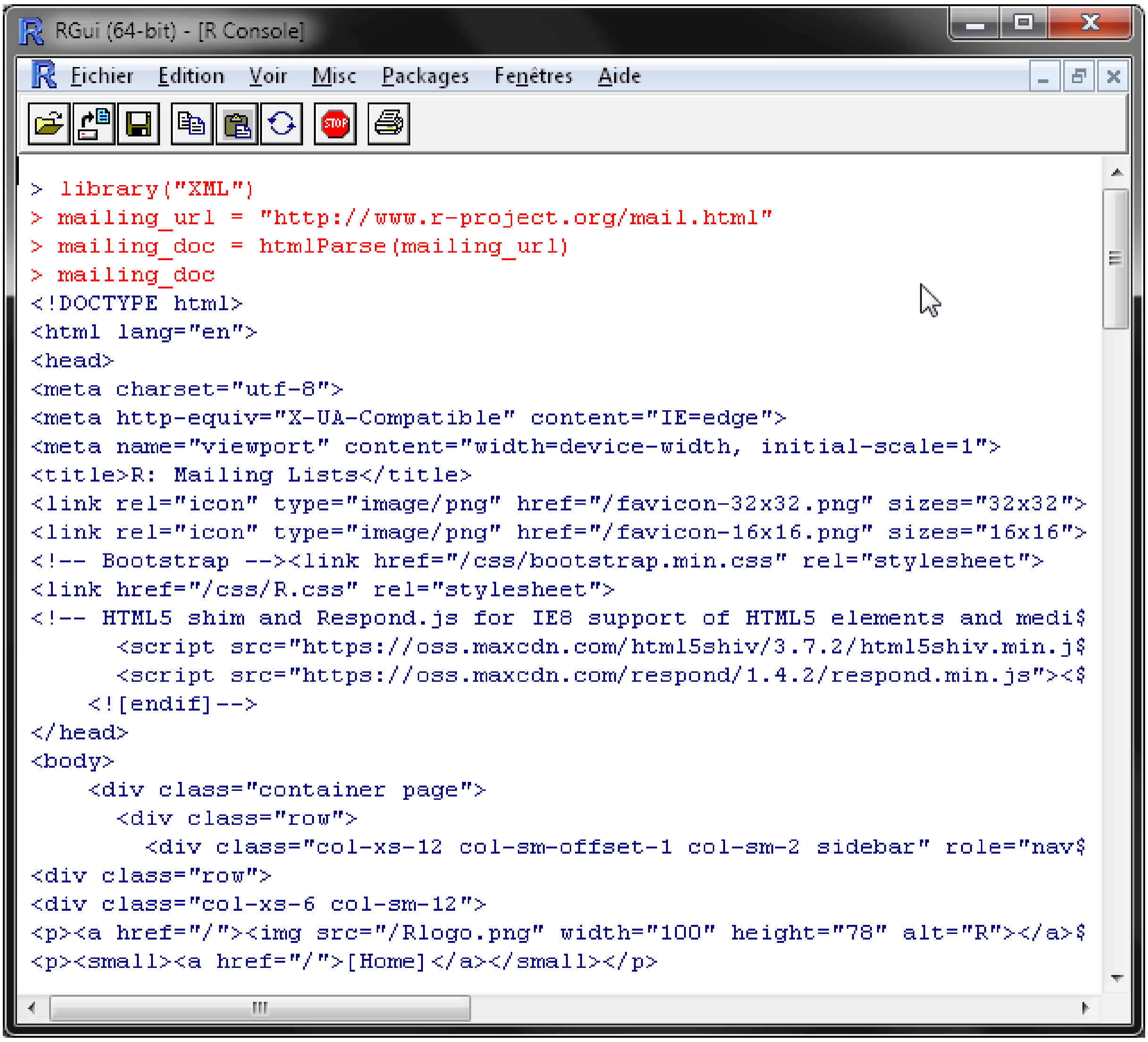
Exemple 57.: Parser du HTML (web scraping de base)

R 3.1.1

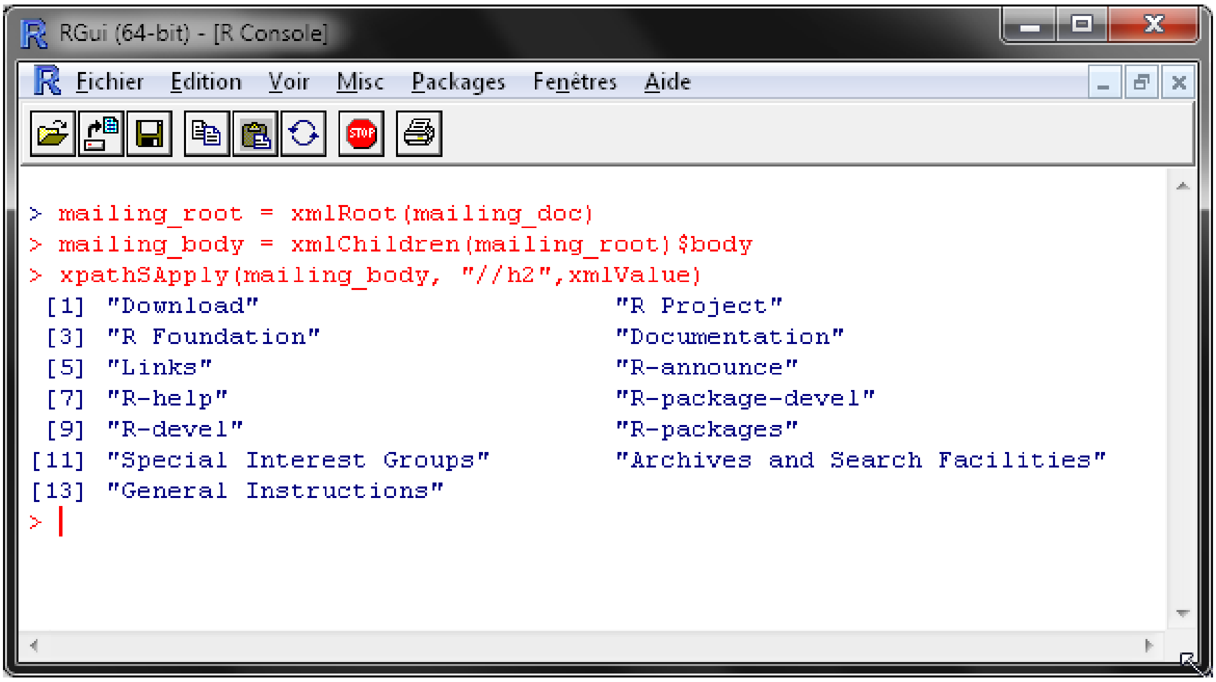
Toujours avec le package **XML**, nous parsons la page suivante:



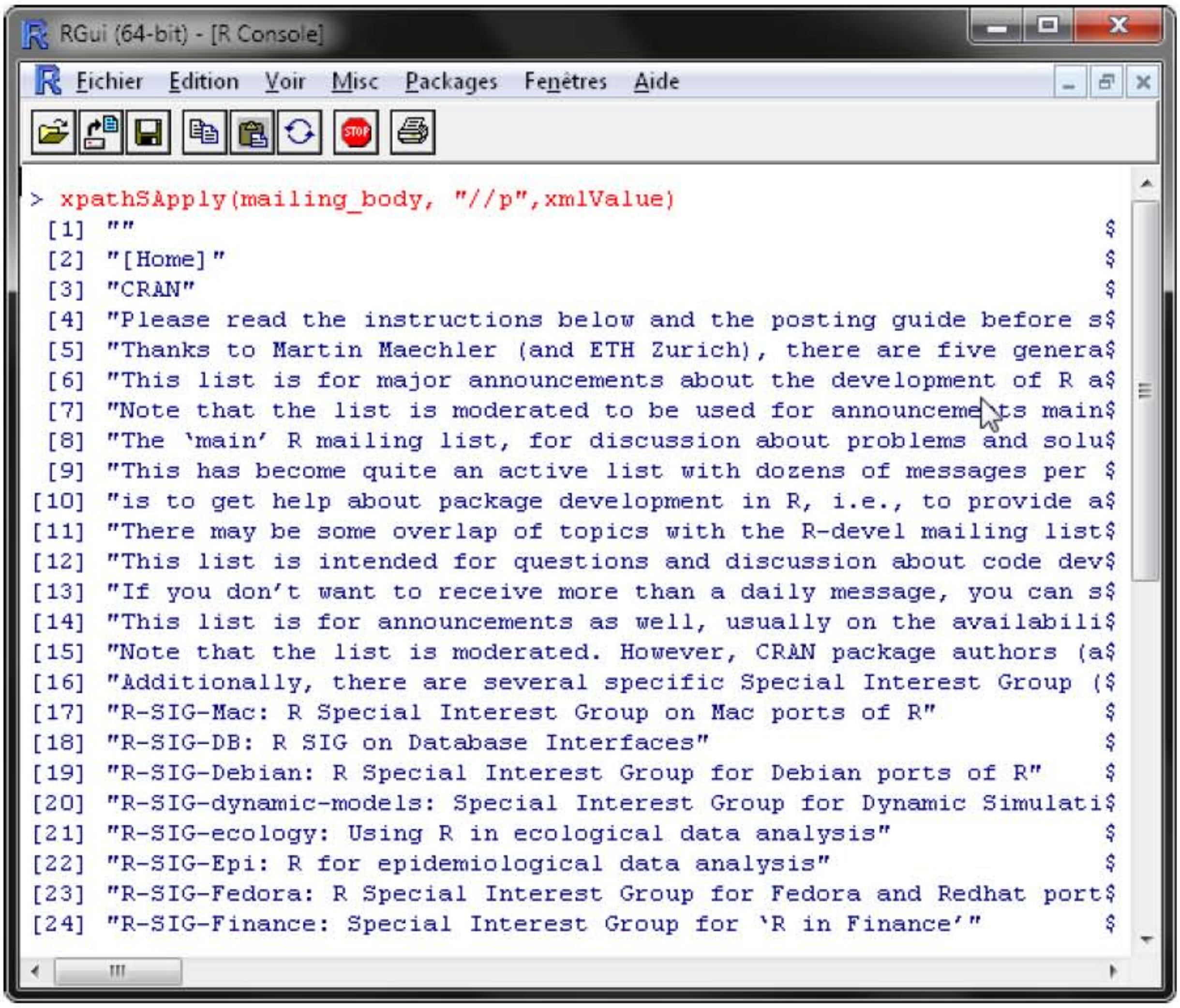
Nous utilisons alors la fonction **htmlParse( )** du package **XML**:



Pour lire ensuite les titres <h2> il nous suffit alors d'écrire:



ou les paragraphes:



Exemple 58.: Importer des données *.csv d'un fichier zip

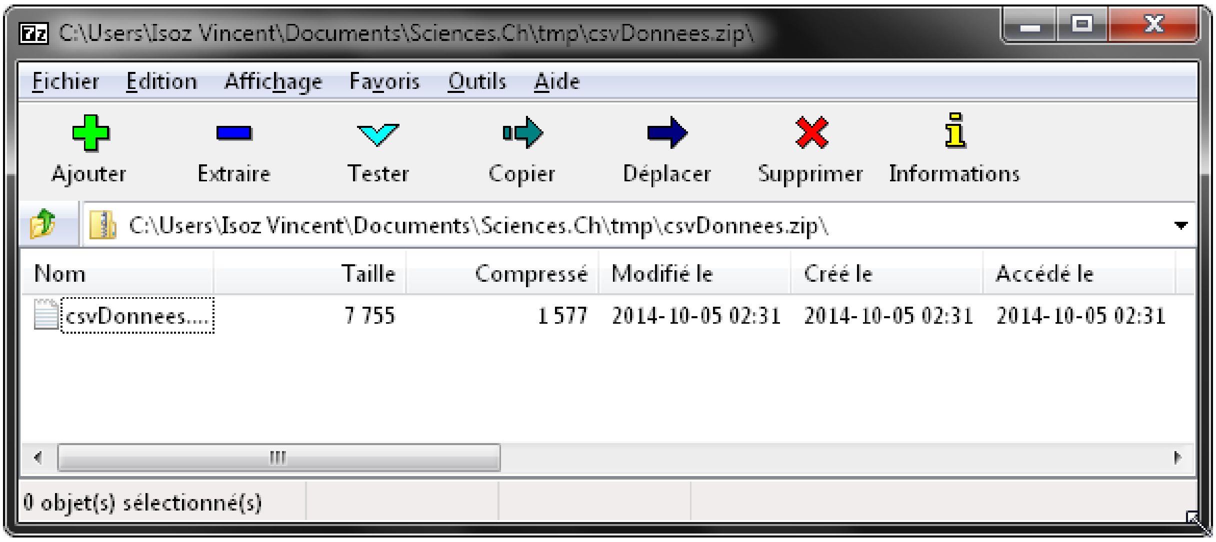
R 3.0.2

Un cas relativement courant de jeux de données se trouvant sur Internet est ceux qui sont retrouvent compressés dans des fichiers *.zip.

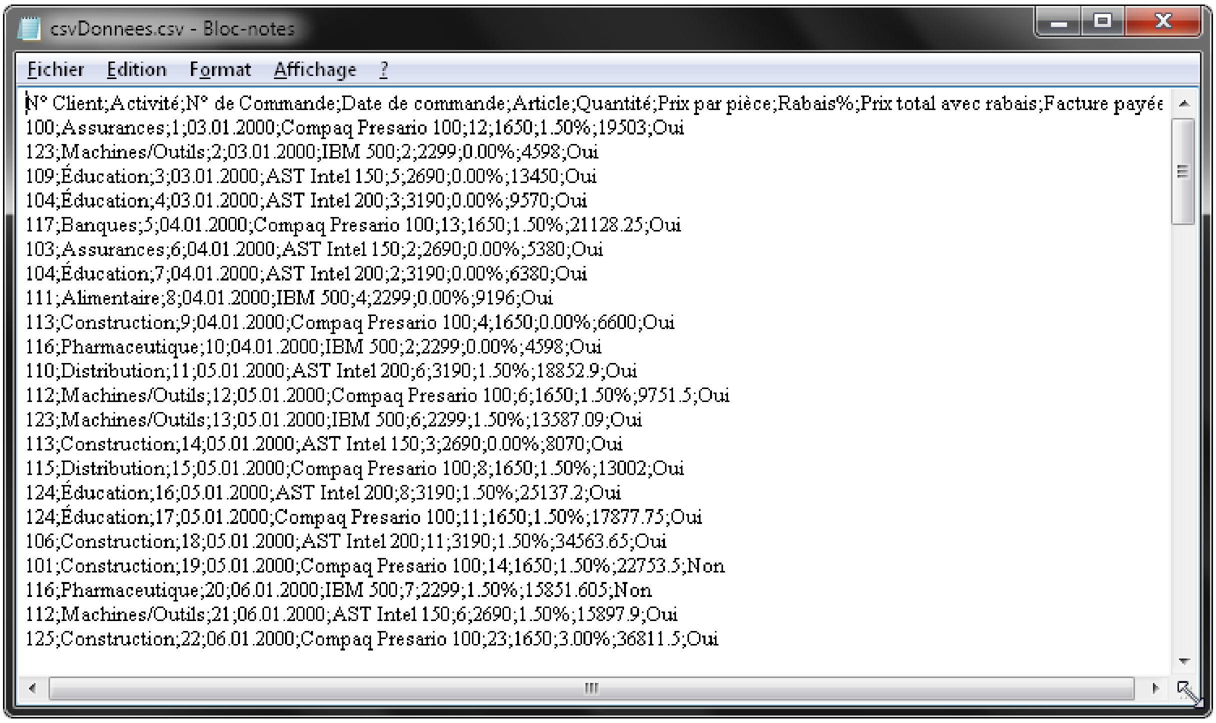
Considérons alors le fichier *.zip suivant:



Avec le contenu:

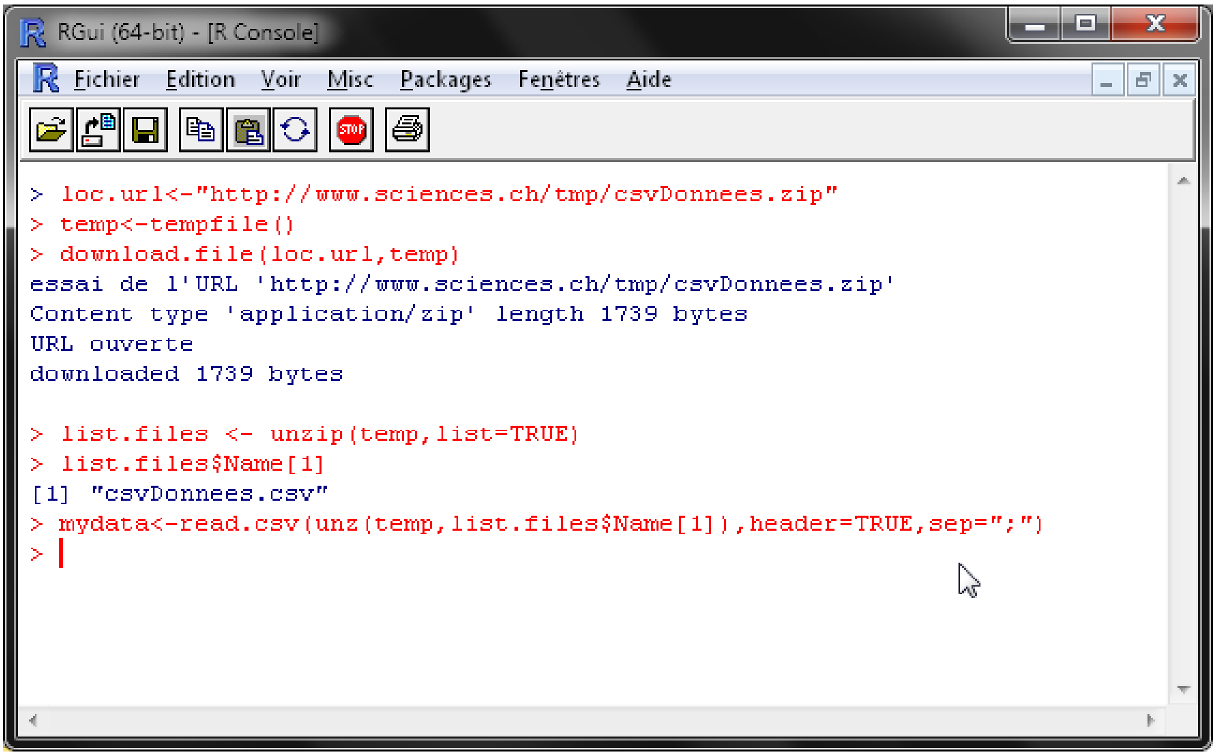


où le fichier *.csv contient:

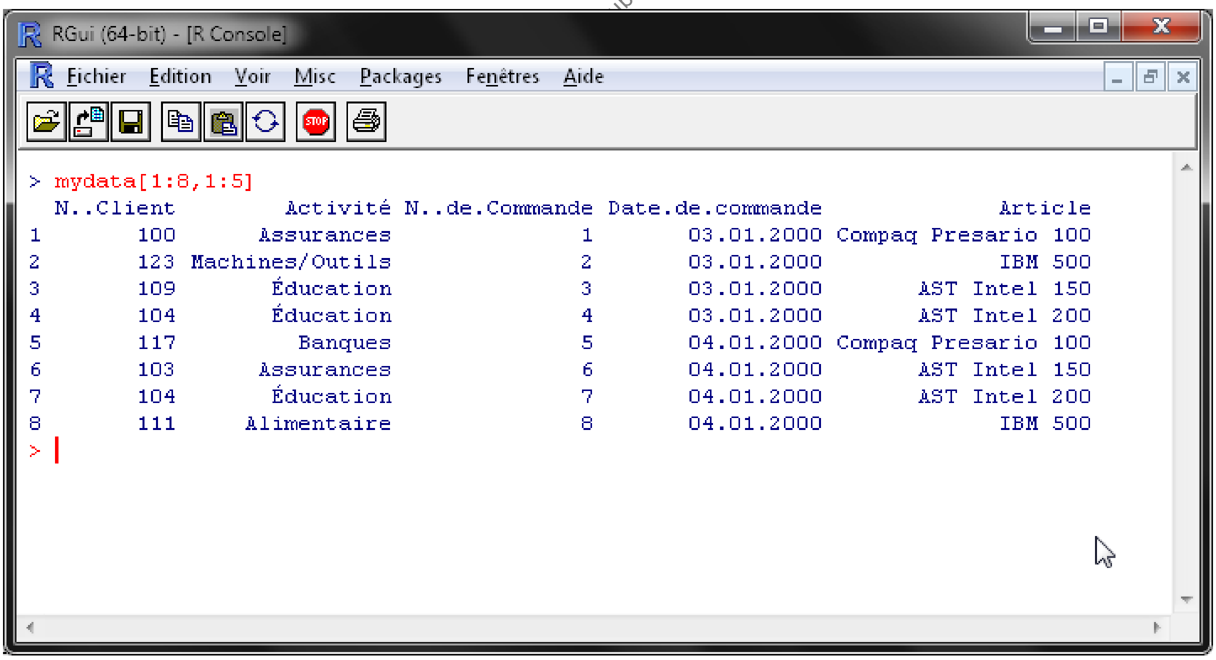



```
N° Client;Activité;N° de Commande;Date de commande;Article;Quantité;Prix par pièce;Rabais%;Prix total avec rabais;Facture payée
100;Assurances;1;03.01.2000;Compaq Presario 100;12;1650;1.50%;19503;Oui
123;Machines/Outils;2;03.01.2000;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000;Compaq Presario 100;13;1650;1.50%;21128.25;Oui
103;Assurances;6;04.01.2000;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000;Compaq Presario 100;4;1650;0.00%;6600;Oui
116;Pharmaceutique;10;04.01.2000;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000;Compaq Presario 100;6;1650;1.50%;9751.5;Oui
123;Machines/Outils;13;05.01.2000;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000;Compaq Presario 100;8;1650;1.50%;13002;Oui
124;Éducation;16;05.01.2000;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000;Compaq Presario 100;11;1650;1.50%;17877.75;Oui
106;Construction;18;05.01.2000;AST Intel 200;11;3190;1.50%;34563.65;Oui
101;Construction;19;05.01.2000;Compaq Presario 100;14;1650;1.50%;22753.5;Non
116;Pharmaceutique;20;06.01.2000;IBM 500;7;2299;1.50%;15851.605;Non
112;Machines/Outils;21;06.01.2000;AST Intel 150;6;2690;1.50%;15897.9;Oui
125;Construction;22;06.01.2000;Compaq Presario 100;23;1650;3.00%;36811.5;Oui
```

Le fichier zip est stocké à l'adresse Internet visible dans le script ci-dessous dans la fonction **download.file( )** et nous utilisons la fonction **unzip( )** pour le décompresser:



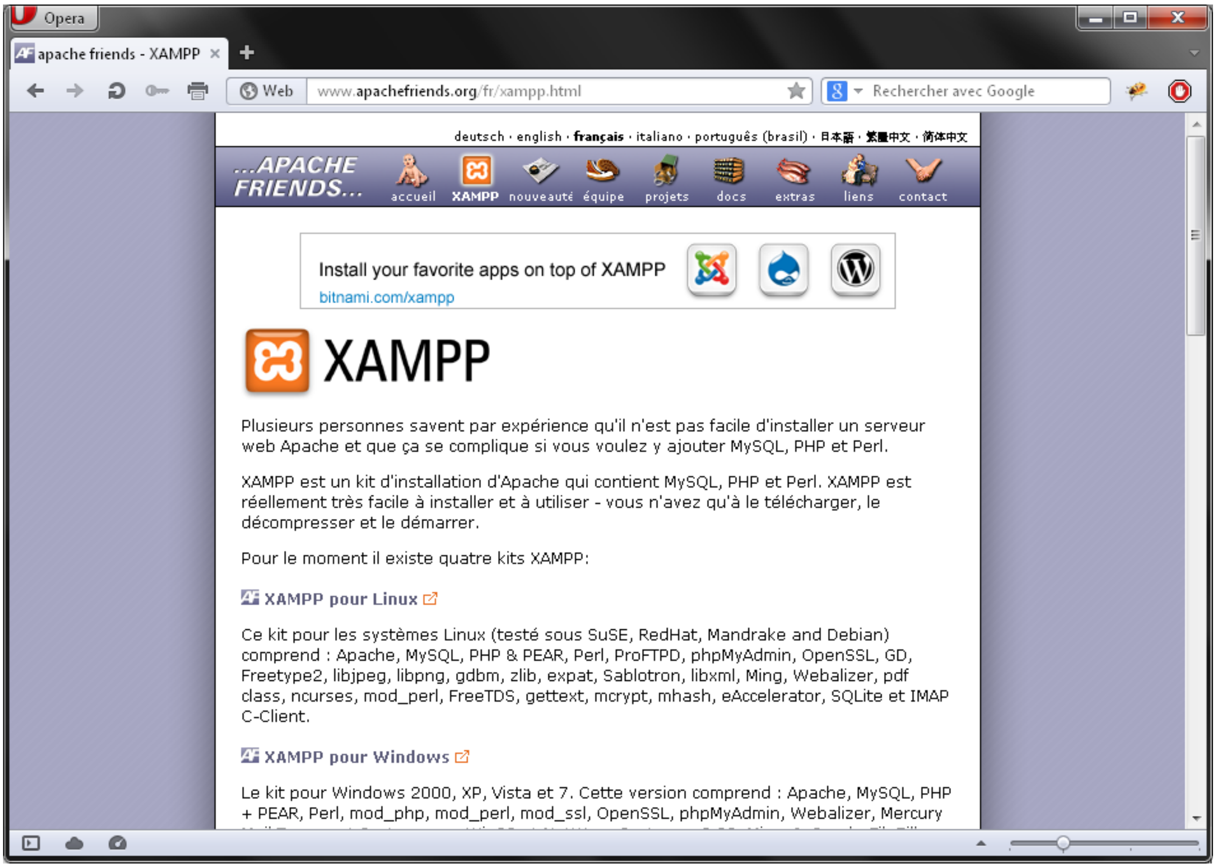
et nous avons bien récupéré les données comme le montre la commande ci-dessous:



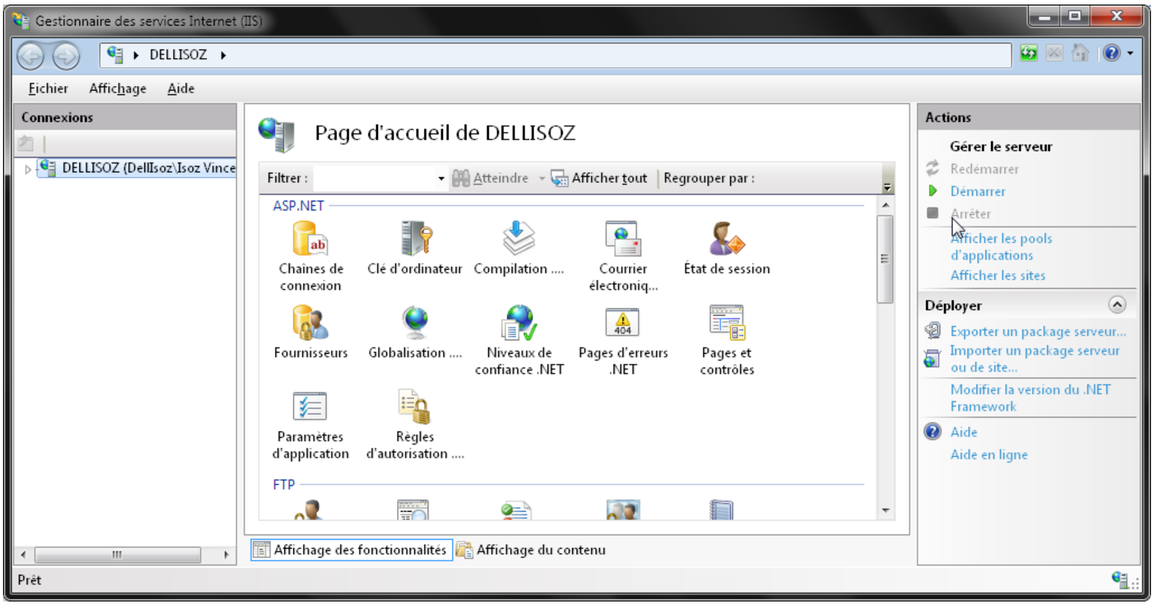
Exemple 59.: Importer des données d'un serveur MySQL

R 3.0.2

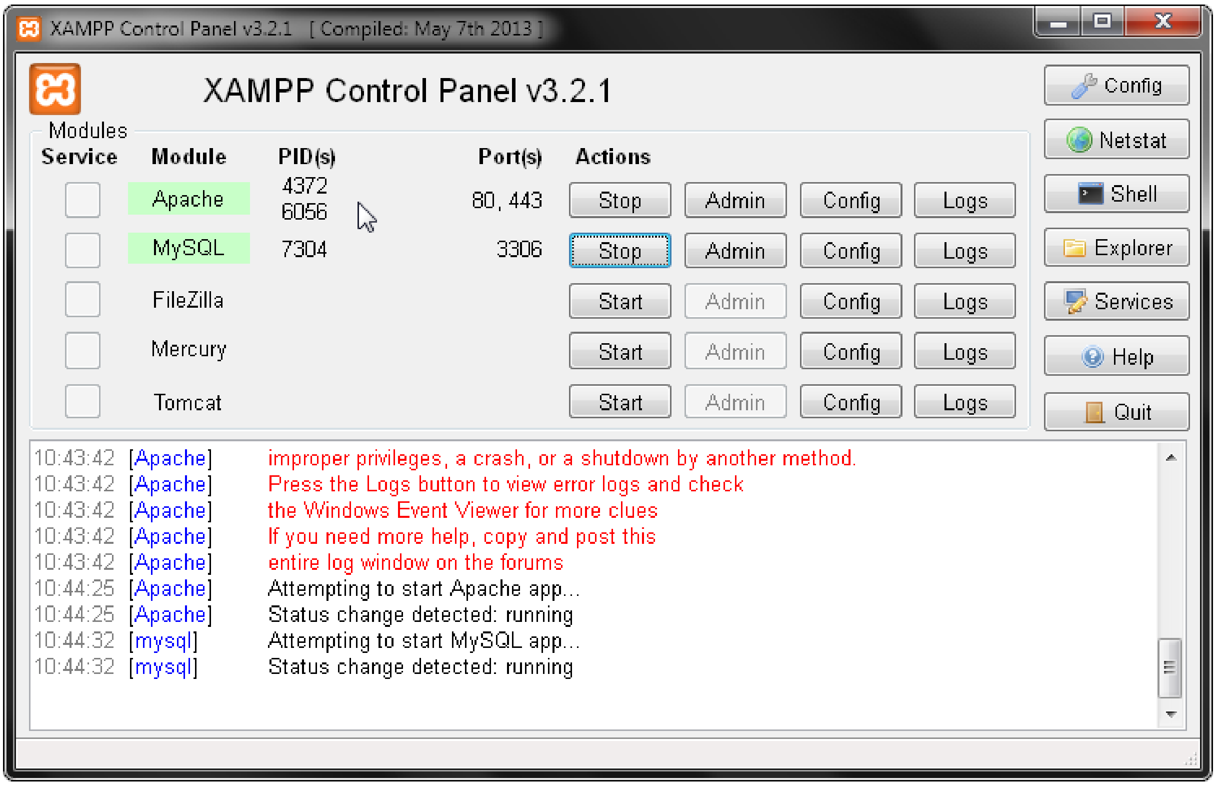
Pour l'exemple nous supposerons que le lecteur aura téléchargé et installé XAMPP:



Une fois téléchargé et installé, si vous travaillez avec une version Pro de MS Windows, il vous faudra arrêter le serveur IIS pour poursuivre:



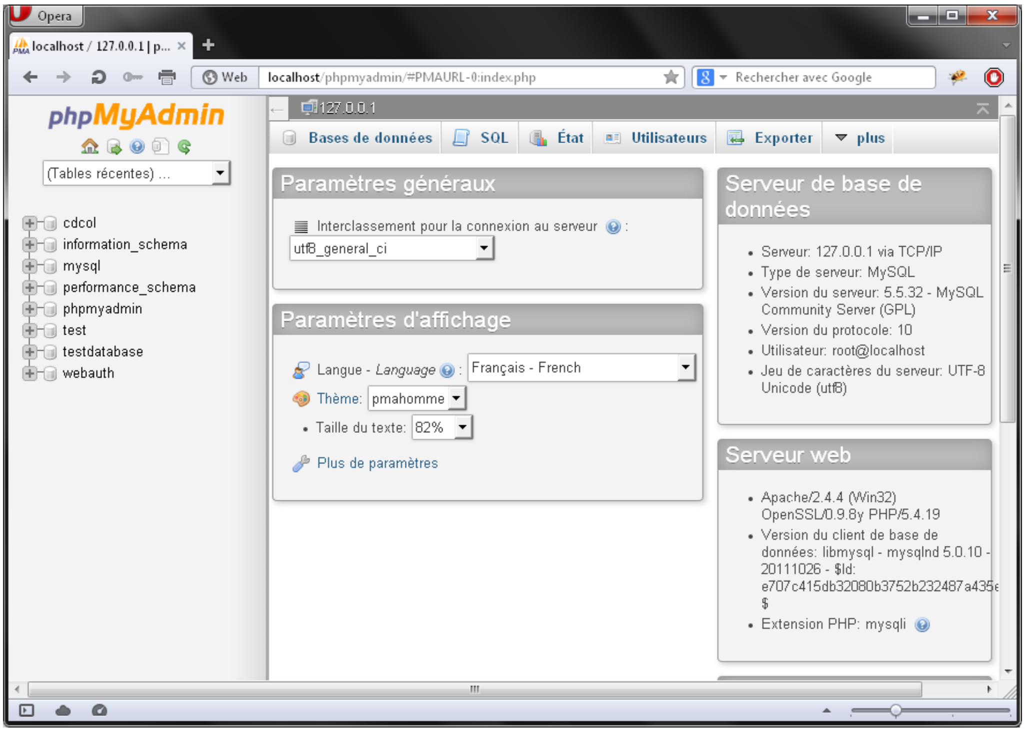
Ensuite, ouvrez XAMPP et démarrez-les servies **Apache** et **MySQL**:



Ouvrez ensuite votre navigateur préféré et tapez **http://localhost**. Vous devrez alors avoir:

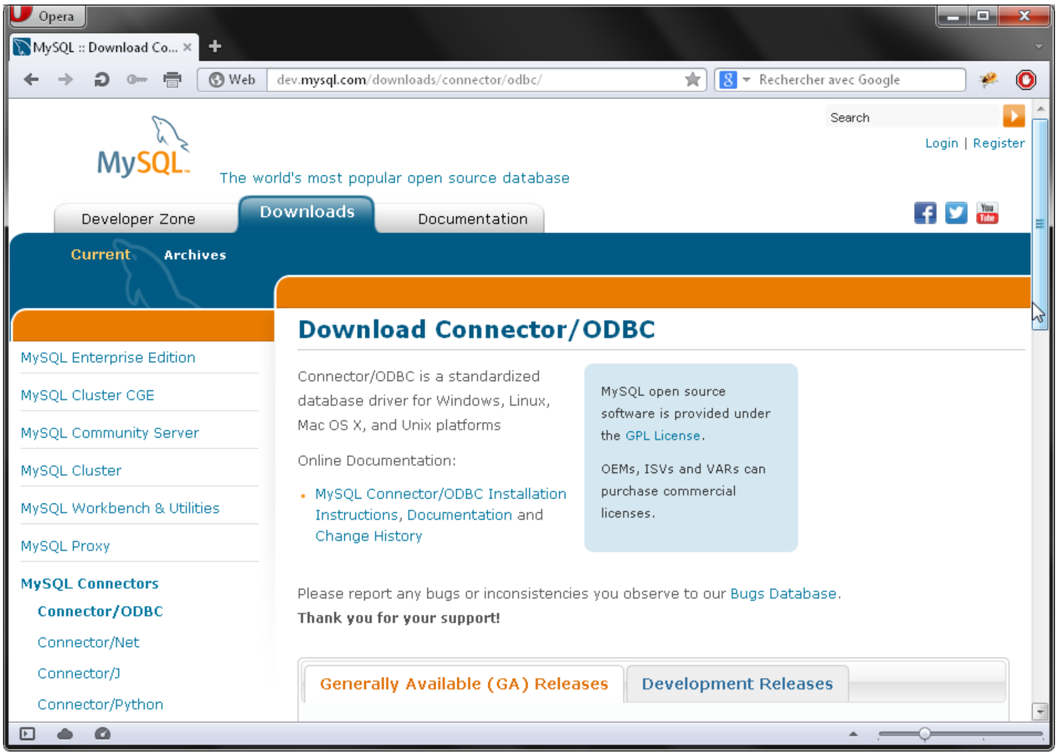


et cliquez en bas à gauche de cette même page sur **phpMyAdmin** pour accéder aux bases de données et tables de MySQL:

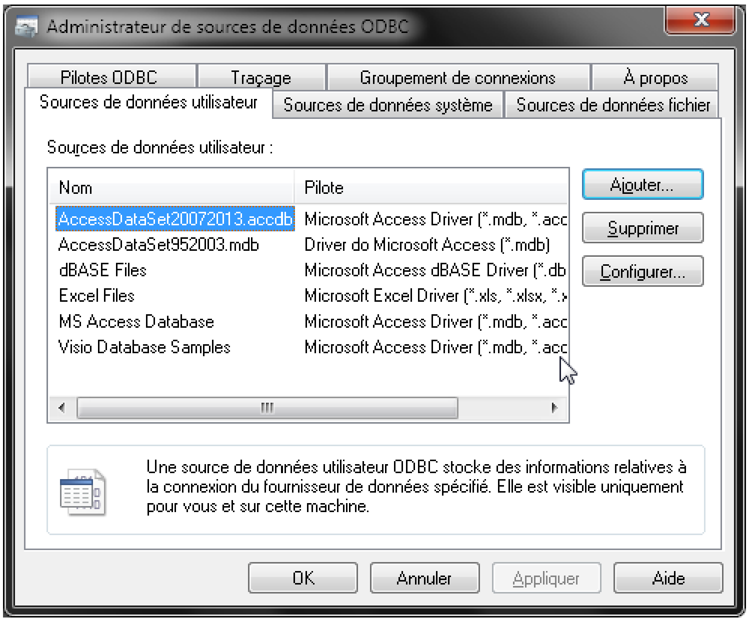


Ensuite, téléchargez le connecteur ODBC MySQL pour MS Windows à l'adresse suivante:

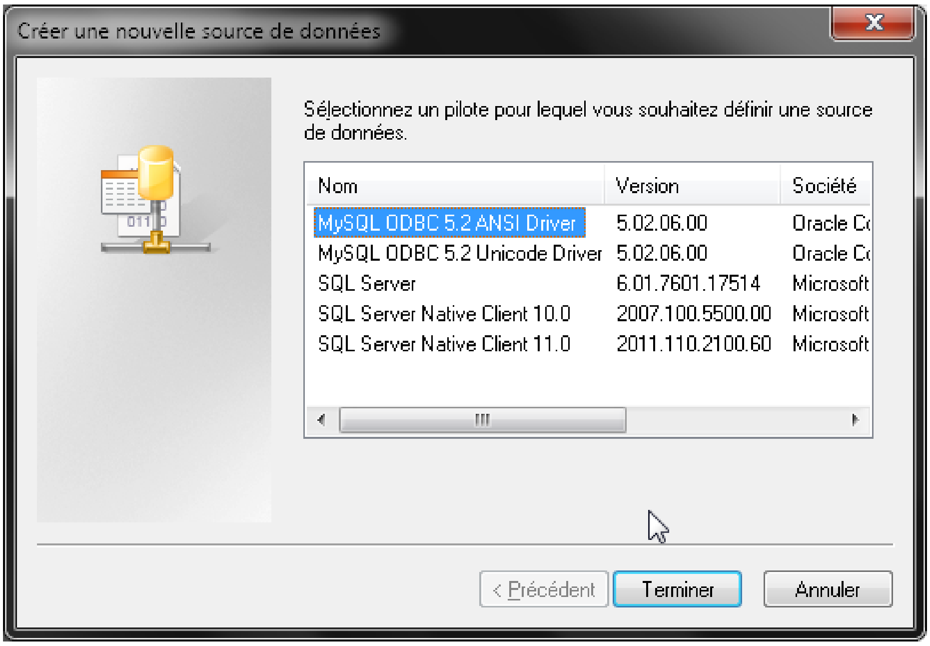
http://dev.mysql.com/downloads/connector/odbc/



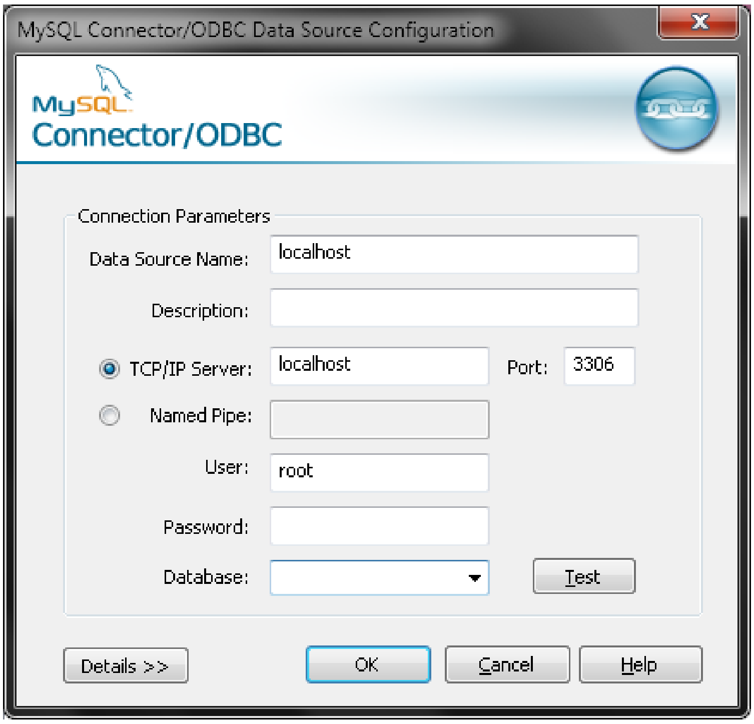
Une fois les drivers ODBC de MySQL téléchargés, ouvrez le gestionnaire ODBC de MS Windows et cliquez sur **Ajouter**:



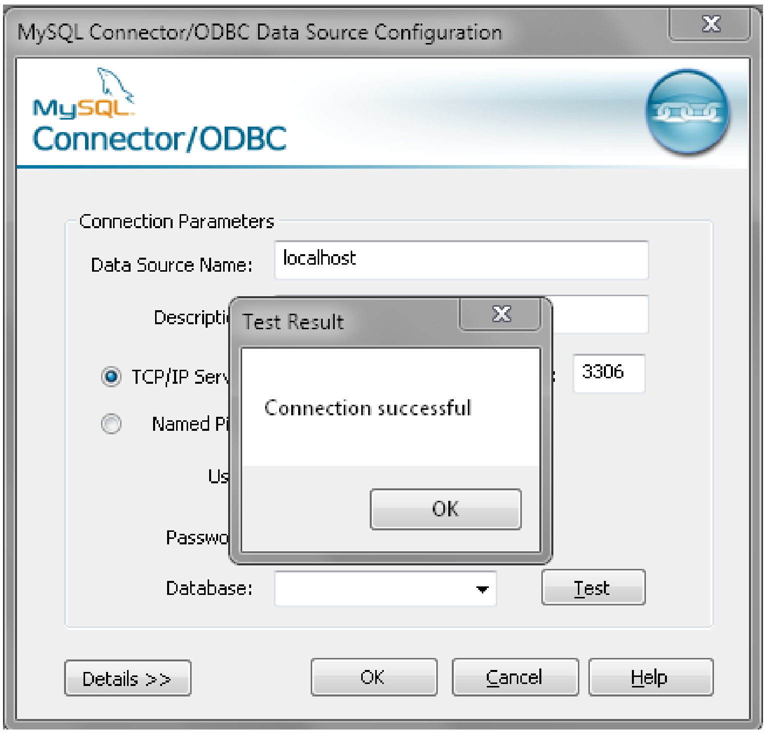
et dans la boîte de dialogue qui vient, sélectionnez le driver mis en surbrillance ci-dessous:



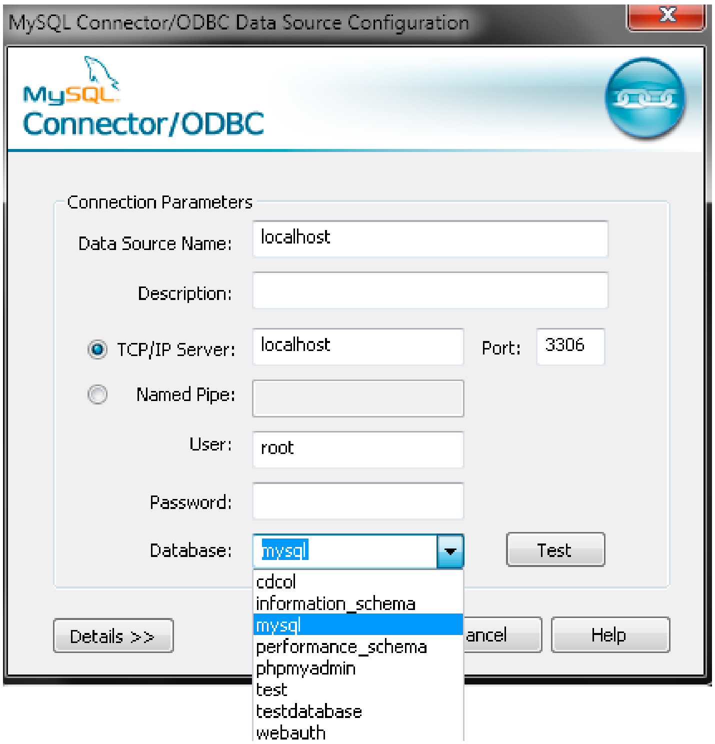
Cliquez sur **Terminer** et vous aurez alors une boîte de dialogue qu'il vous faudra remplir comme indiqué ci-dessous:



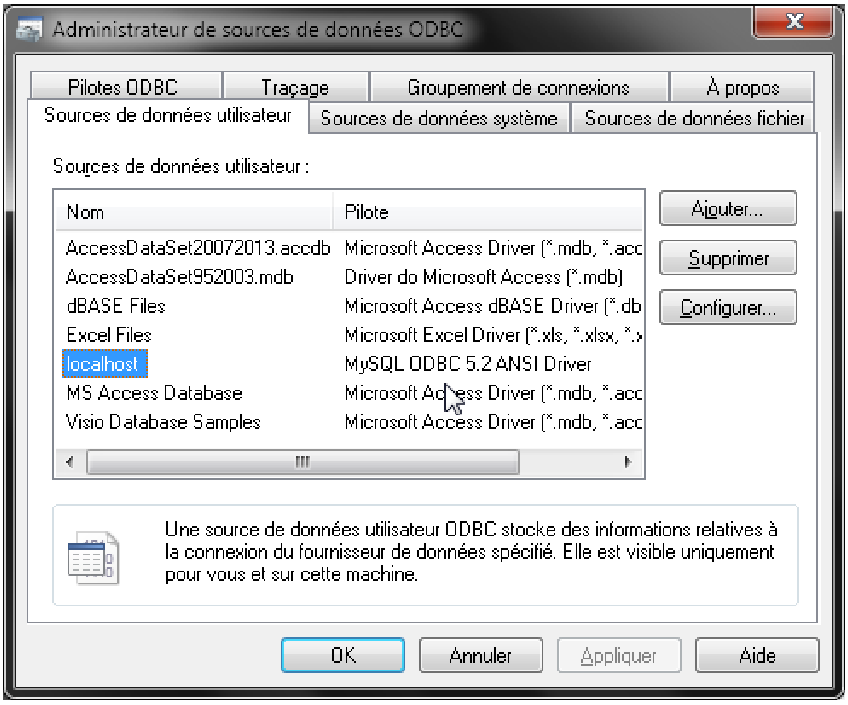
Cliquez sur le bouton **Test**:



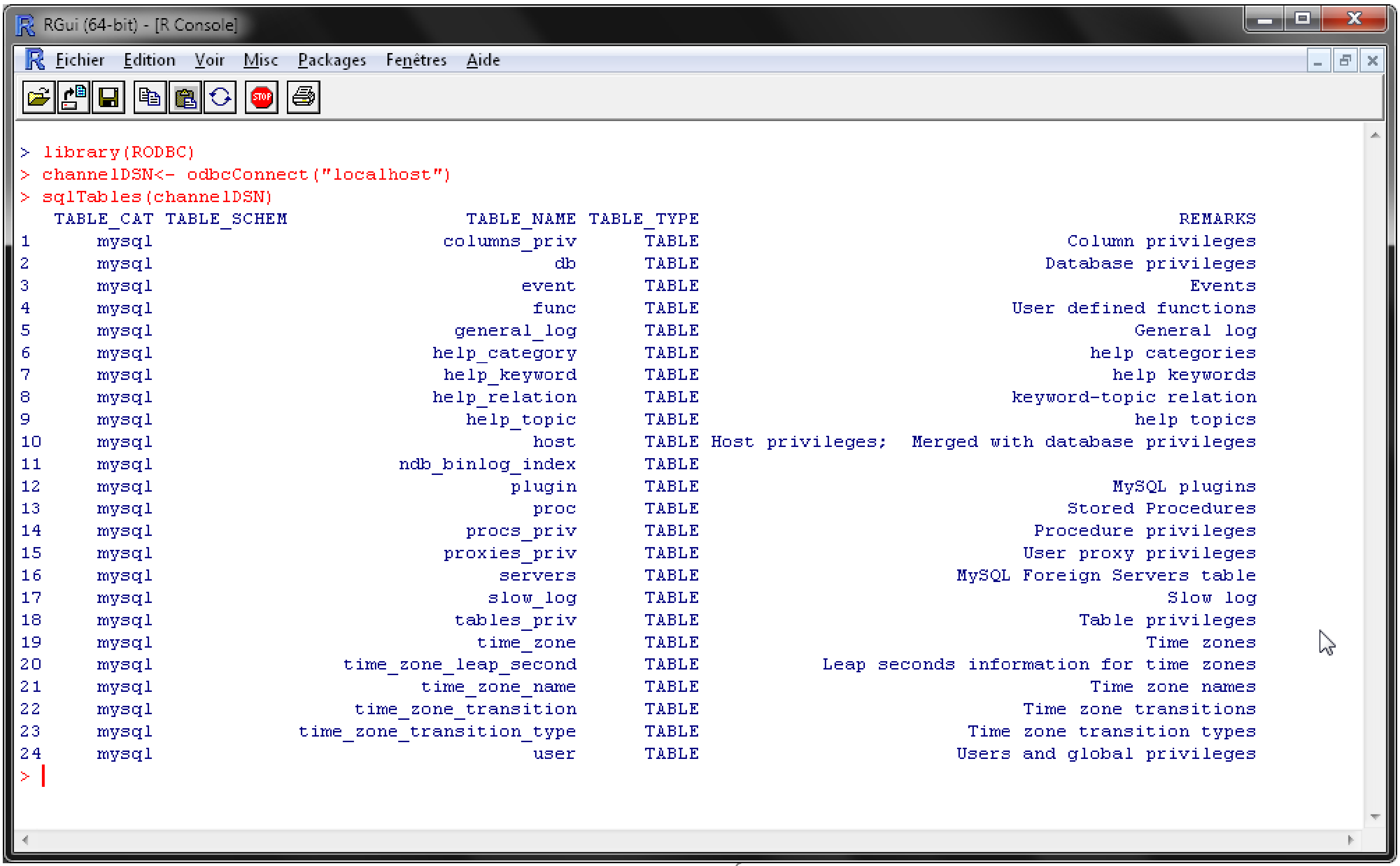
Vous pourrez alors choisir la base de données disponible dans le serveur MySQL:



Ici, nous avons pris empiriquement pour exemple la base de données installée par défaut avec MySQL et qui se nomme *mysql*. Nous validons par **OK** et revenons alors à:

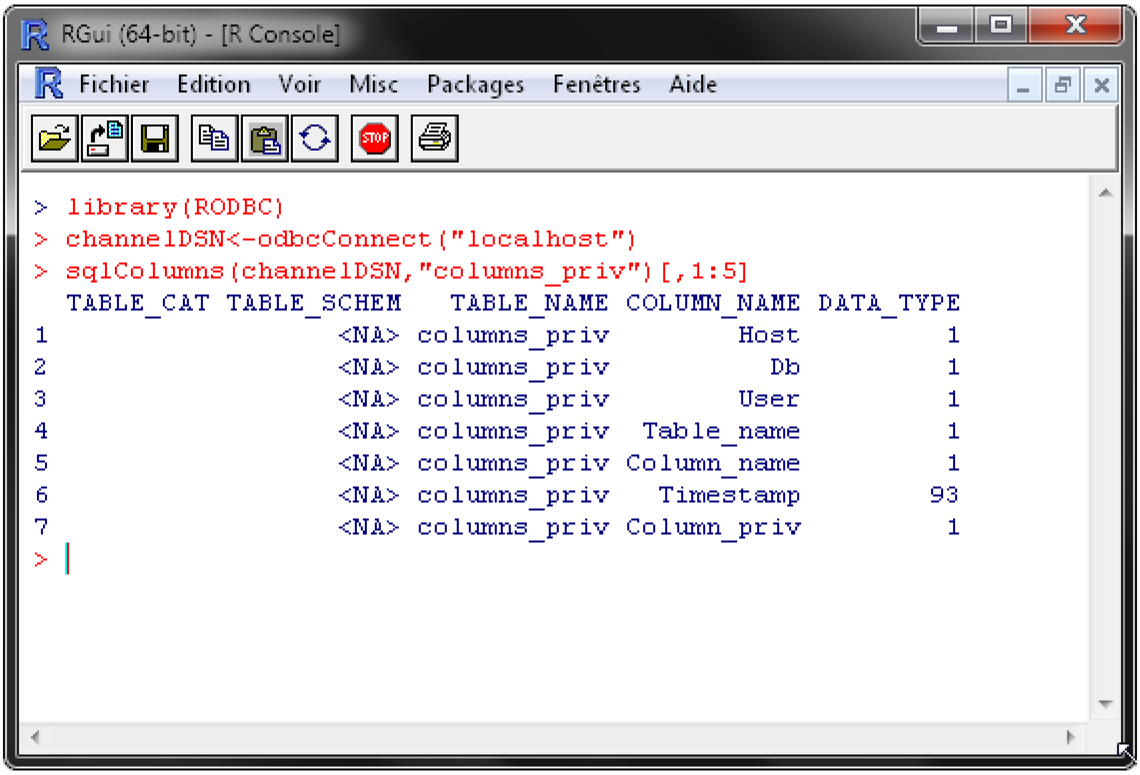


Ensuite, nous retournons dans R et pour voir la liste des tables de la base de données, nous utilisons la commande **sqlTables( )** du package **RODBC** (souvenez que ce package permet pas seulement de lire, mais aussi de modifier, créer des données dans la base!!!):



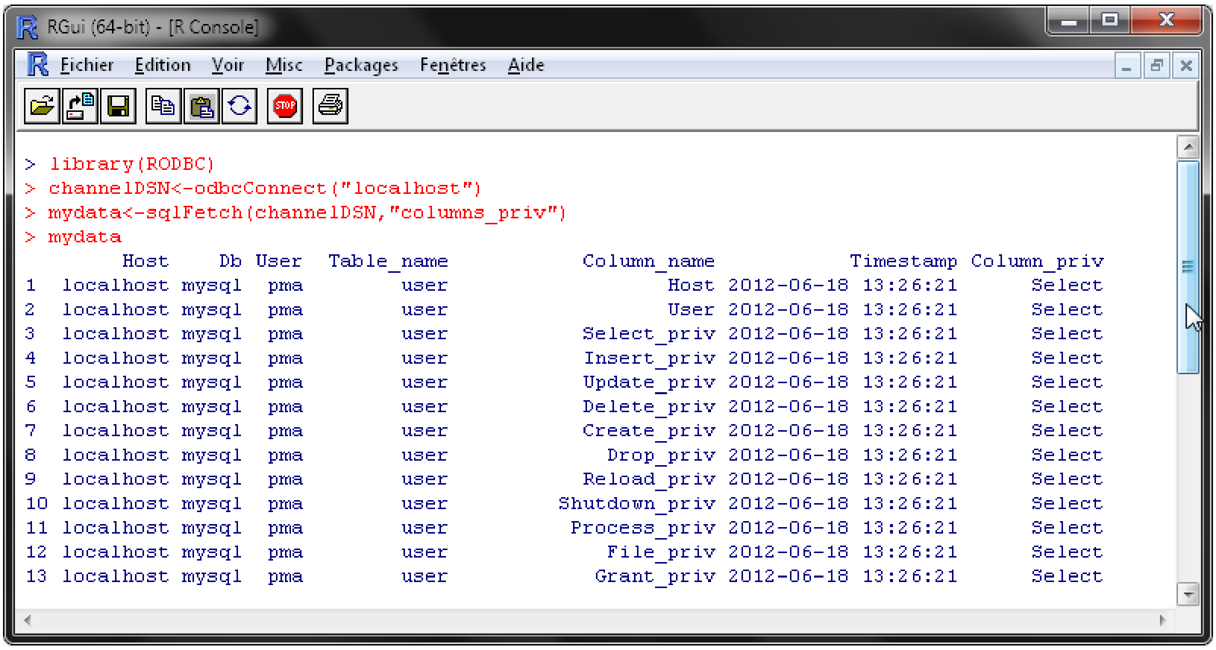
```
> library(RODBC)
> channelDSN<- odbcConnect("localhost")
> sqlTables(channelDSN)
   TABLE_CAT TABLE_SCHEM                TABLE_NAME TABLE_TYPE                                          REMARKS
1      mysql                          columns_priv      TABLE                                Column privileges
2      mysql                                    db      TABLE                              Database privileges
3      mysql                                 event      TABLE                                           Events
4      mysql                                  func      TABLE                           User defined functions
5      mysql                           general_log      TABLE                                      General log
6      mysql                         help_category      TABLE                                  help categories
7      mysql                          help_keyword      TABLE                                    help keywords
8      mysql                         help_relation      TABLE                           keyword-topic relation
9      mysql                            help_topic      TABLE                                      help topics
10     mysql                                  host      TABLE Host privileges;  Merged with database privileges
11     mysql                       ndb_binlog_index      TABLE
12     mysql                                plugin      TABLE                                    MySQL plugins
13     mysql                                  proc      TABLE                                Stored Procedures
14     mysql                            procs_priv      TABLE                             Procedure privileges
15     mysql                          proxies_priv      TABLE                            User proxy privileges
16     mysql                               servers      TABLE                       MySQL Foreign Servers table
17     mysql                              slow_log      TABLE                                         Slow log
18     mysql                           tables_priv      TABLE                                 Table privileges
19     mysql                             time_zone      TABLE                                       Time zones
20     mysql                 time_zone_leap_second      TABLE           Leap seconds information for time zones
21     mysql                        time_zone_name      TABLE                                  Time zone names
22     mysql                  time_zone_transition      TABLE                             Time zone transitions
23     mysql             time_zone_transition_type      TABLE                        Time zone transition types
24     mysql                                  user      TABLE                       Users and global privileges
>
```


Nous pouvons aussi afficher les colonnes d'une table donnée de la base. Ainsi, avec la commande **sqlColumns( )** nous demandons à R de nous montrer la structure de la table *columns_priv*:



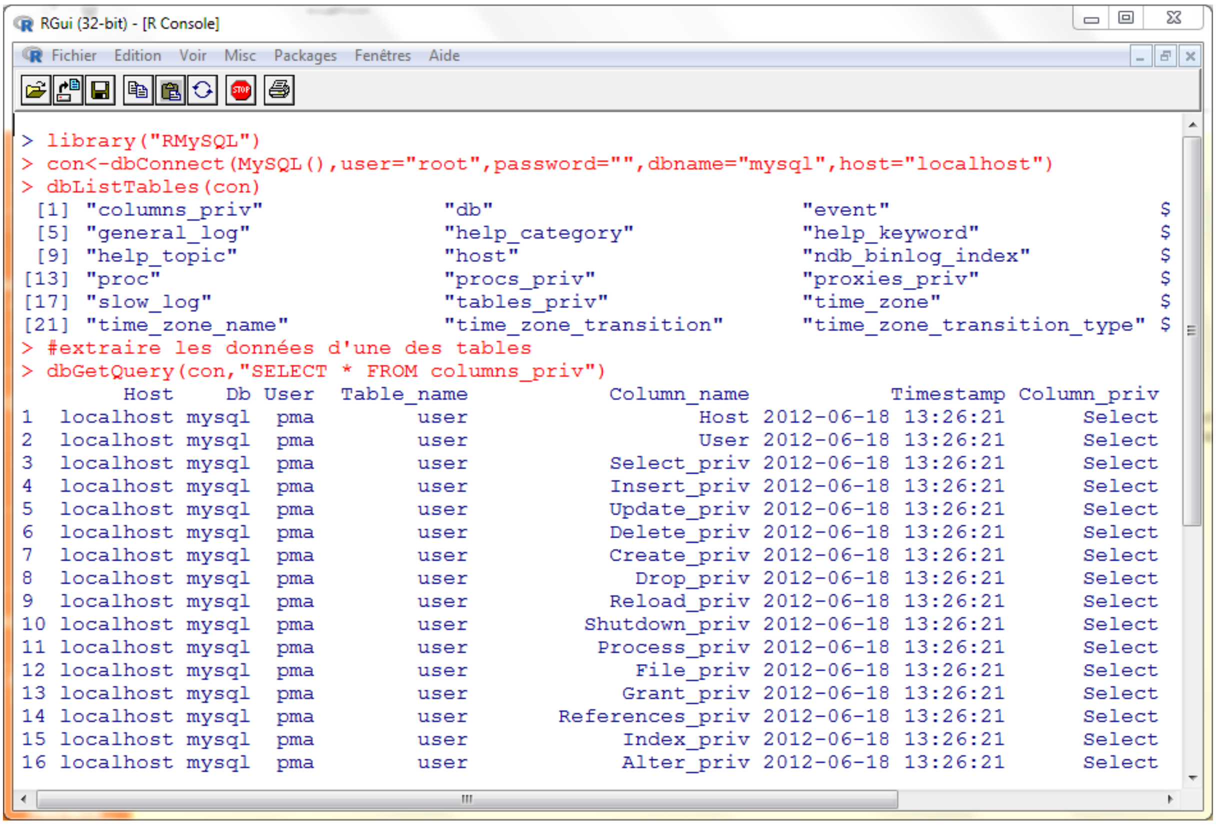
```
> library(RODBC)
> channelDSN<-odbcConnect("localhost")
> sqlColumns(channelDSN,"columns_priv")[,1:5]
  TABLE_CAT TABLE_SCHEM   TABLE_NAME COLUMN_NAME DATA_TYPE
1                  <NA> columns_priv        Host         1
2                  <NA> columns_priv          Db         1
3                  <NA> columns_priv        User         1
4                  <NA> columns_priv  Table_name         1
5                  <NA> columns_priv Column_name         1
6                  <NA> columns_priv   Timestamp        93
7                  <NA> columns_priv Column_priv         1
>
```

et pour lire le contenu de la table MySQL et la mettre dans un data frame, nous utilisons la commande **sqlFetch( )** :

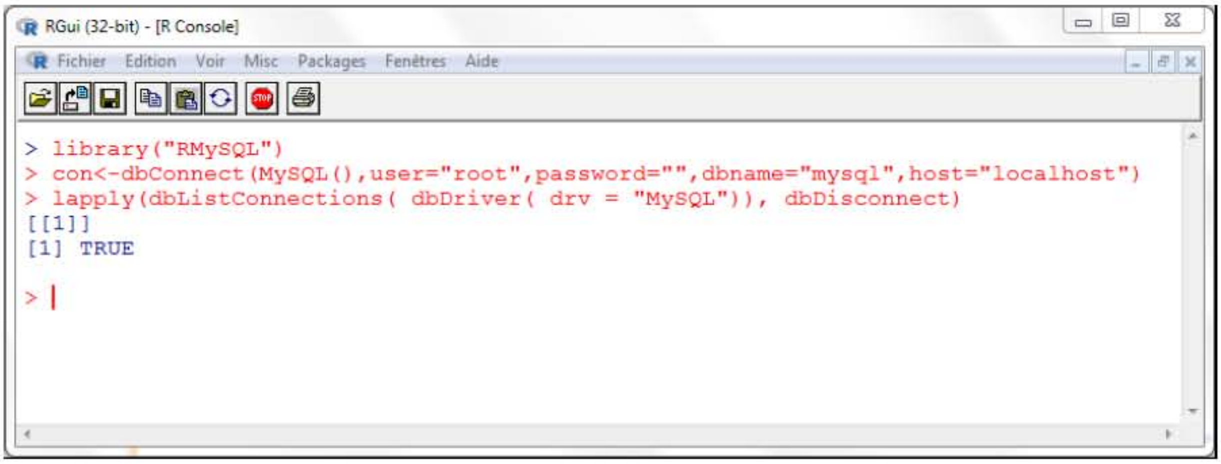


Il existe encore d'autres nombreuses possibilités comme de faire des requêtes en SQL directement sur les tables de votre choix mais pour cela vous pouvez vous référer au PDF du package RODBC qui contient des exemples très bien fait.

Sinon il y a aussi le package RMySQL comme l'illustre l'exemple ci-dessous:



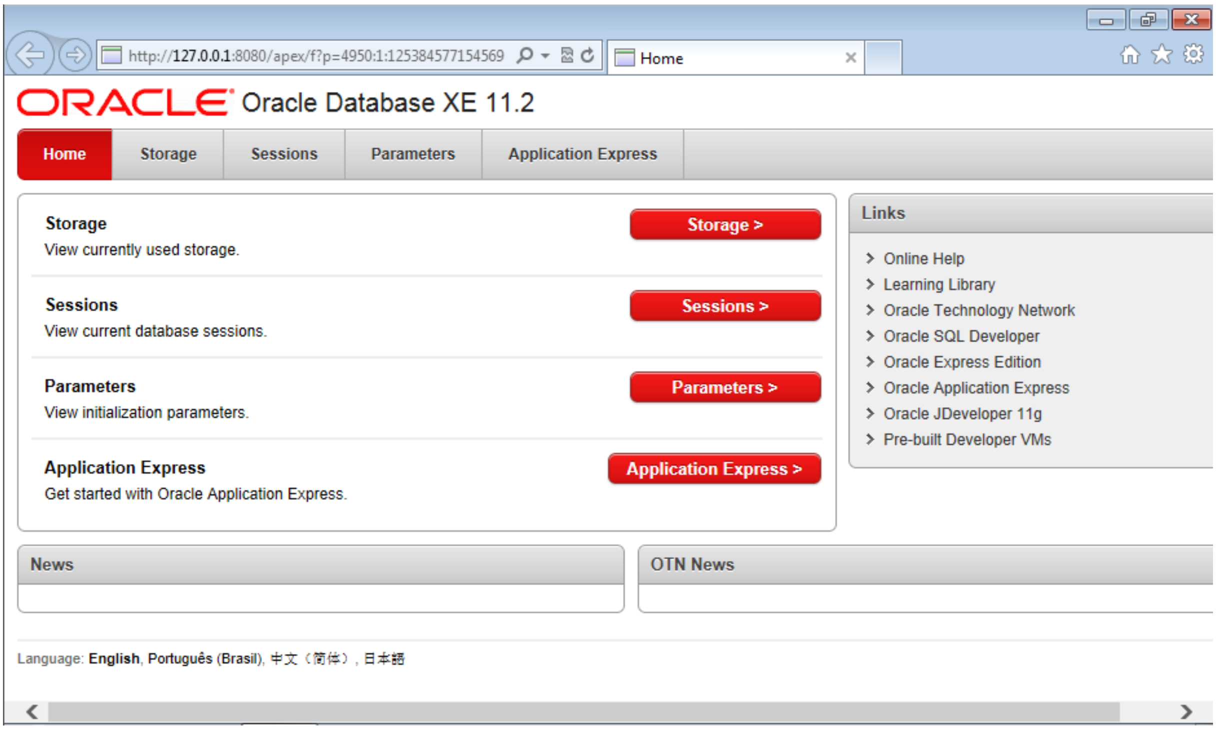
Avec ce package on fermer la connections de la manière suivante:



Exemple 60.: Importer des données d'un serveur Oracle Express

R 3.0.2

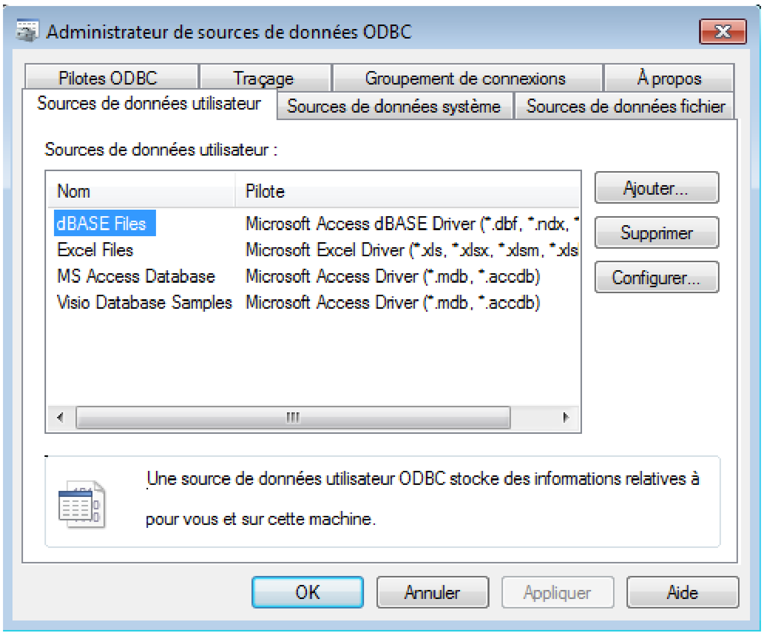
Nous allons donc supposer qu'Oracle Express 10g est installé[8]:



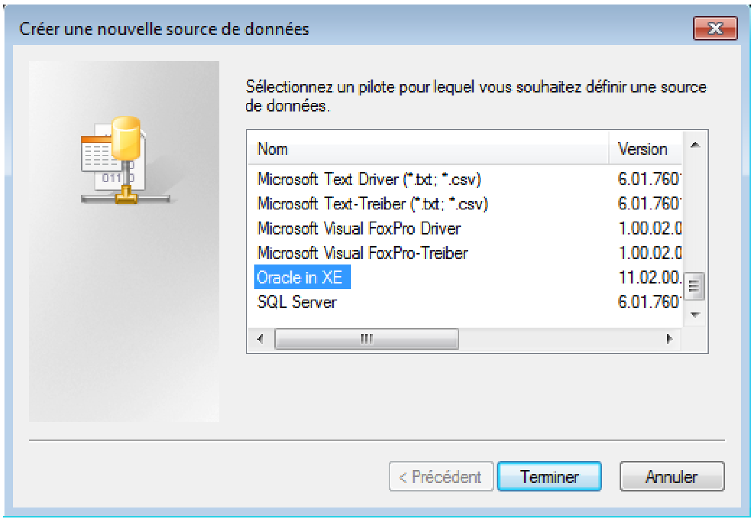
Pour la démo nous utiliserons le compte système (dont le nom est *system* et le mot de passe demandé par l'installateur d'Oracle Express lors de... l'installation).

Une fois donc constaté le démarrage d'Oracle Express, nous ouvrons l'application ODBC de MS Windows:

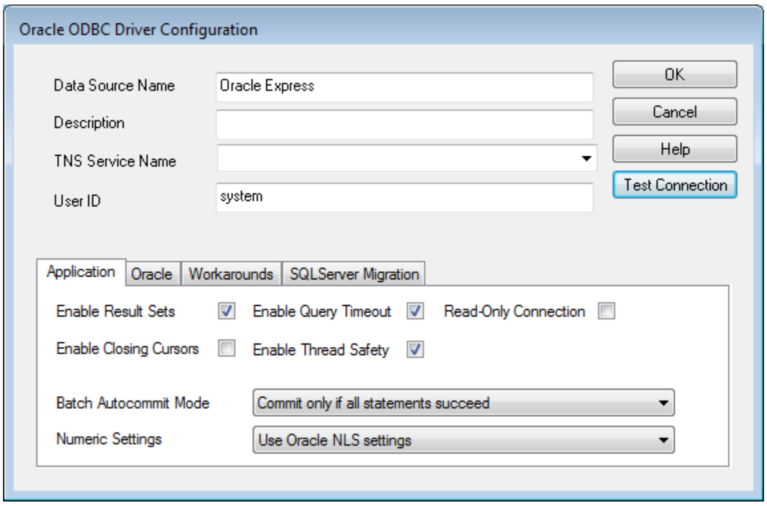
[8] Si l'icône d'Oracle Expresse ne fonctionne pas, tapez le lien suivant: http://127.0.0.1:8080/apex/f?p=4950



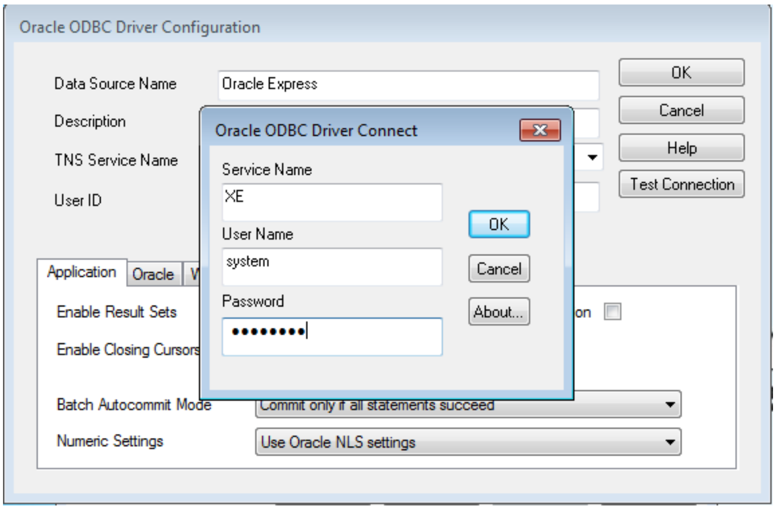
Nous cliquons sur **Ajouter...**:



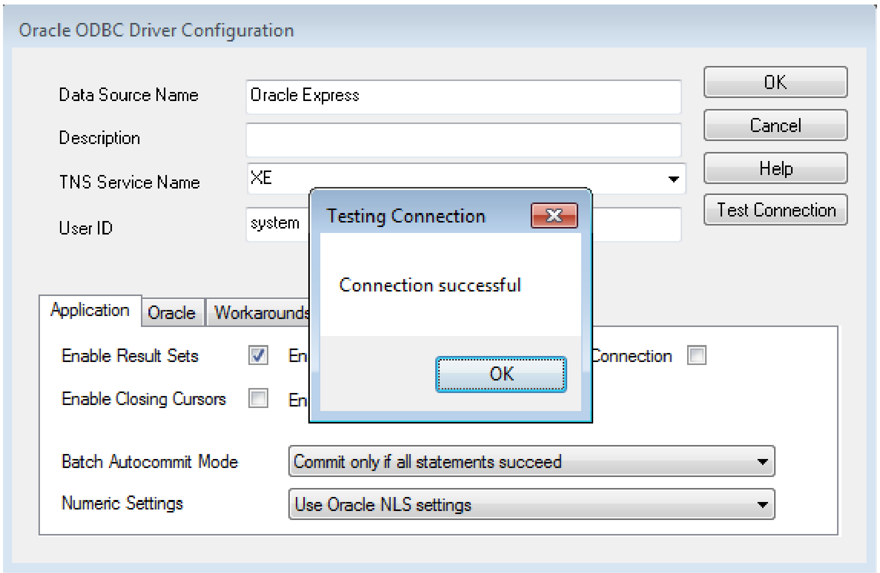
Nous choisissons le pilote **Oracle in XE** et cliquons sur **Terminer** et nous remplissons la boîte de dialogue qui apparaît comme ci-dessous:



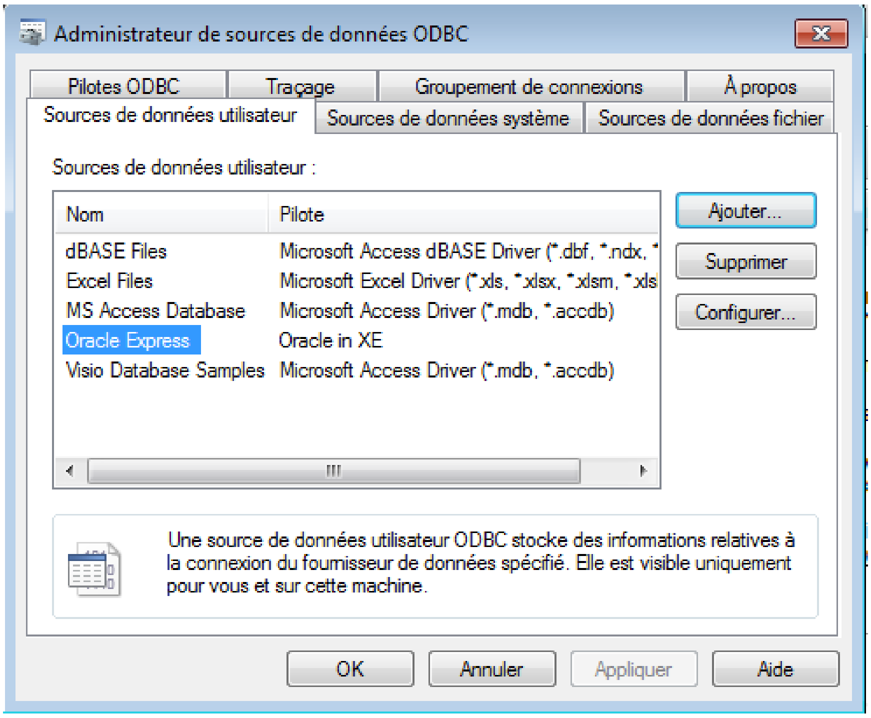
Nous cliquons ensuite sur **Test Connection**. Le nom de la base ainsi que le compte et la password d'accès ava alors nous être demandé:



Nous validons par **OK** pour obtenir un succès de la connexion:

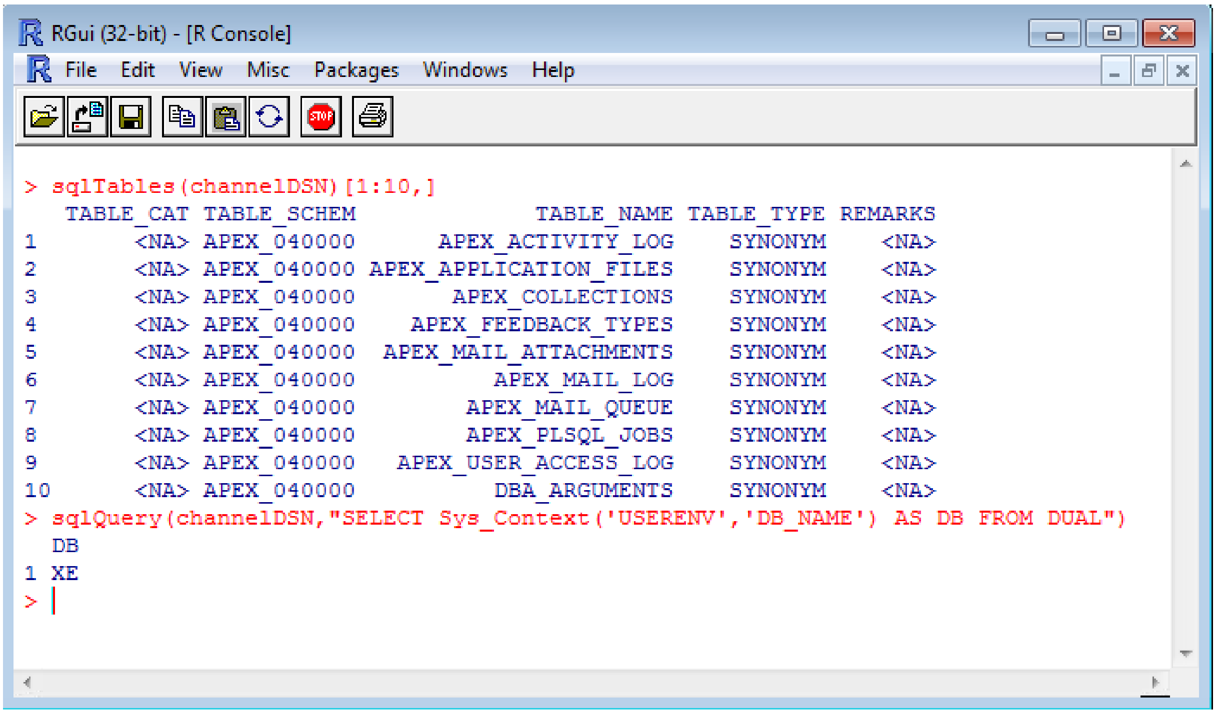


Nous validons encore par **OK** pour revenir à:



et encore une dernière fois par **OK**.

De retour dans R, nous chargeons la libraire **RODBC** (souvenez que ce package ne permet pas seulement de lire, mais aussi de modifier, créer des données dans la base!!!), regardons le nombre de tables dans la base par défaut associée au compte système (nous affichons seulement les dix premières tables) et nous faisons une requête sur une des tables:

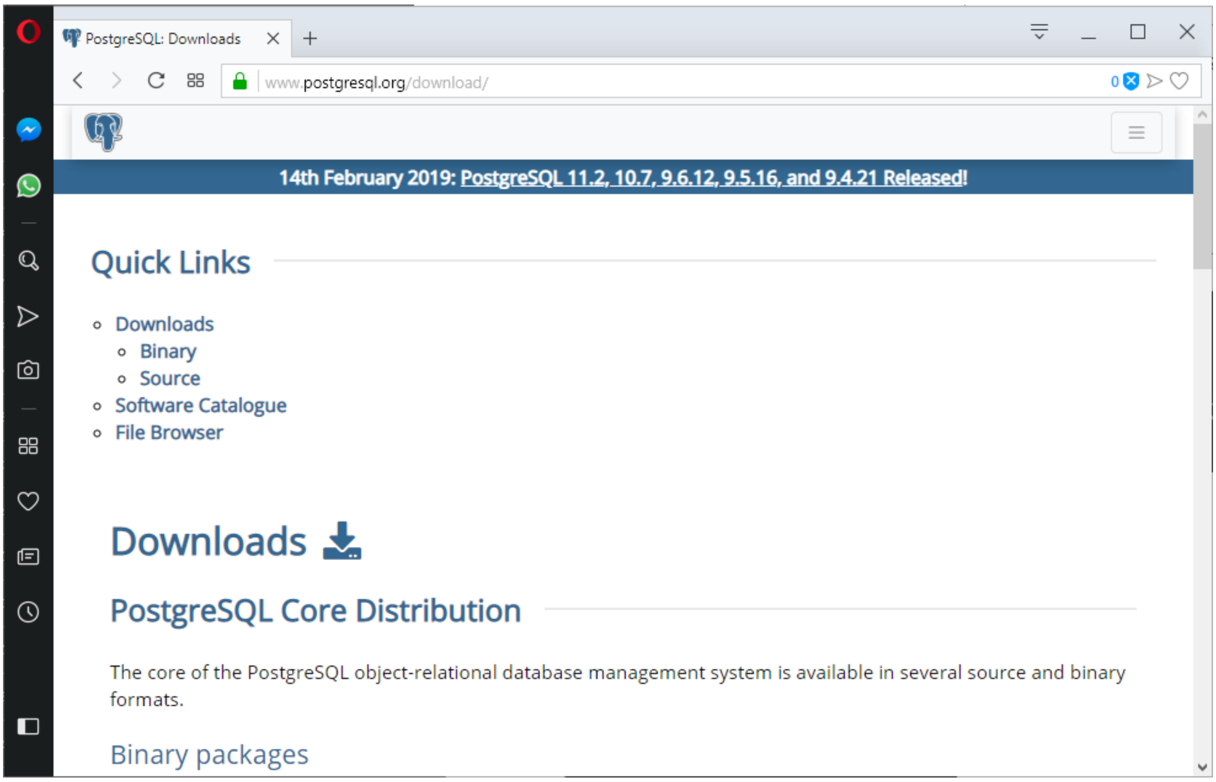


bingo!

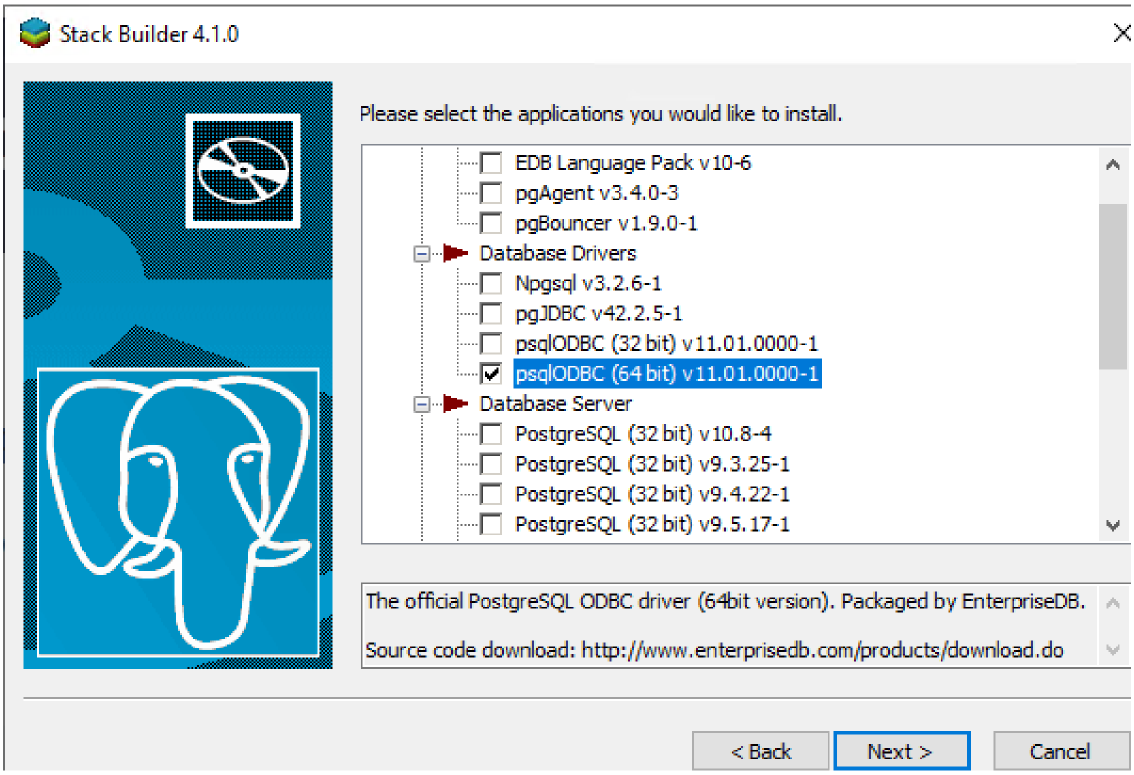
Exemple 61.: Importer des données d'un serveur Postgre SQL

R 3.6.0

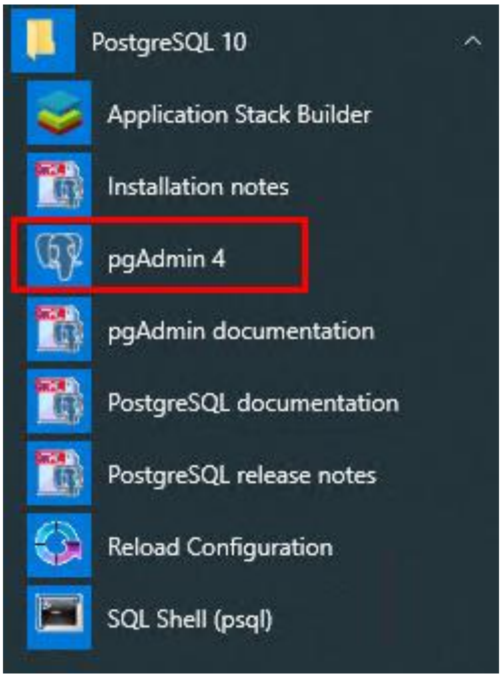
Pour l'exemple nous supposerons que le lecteur aura téléchargé et installé PostgreSQL:



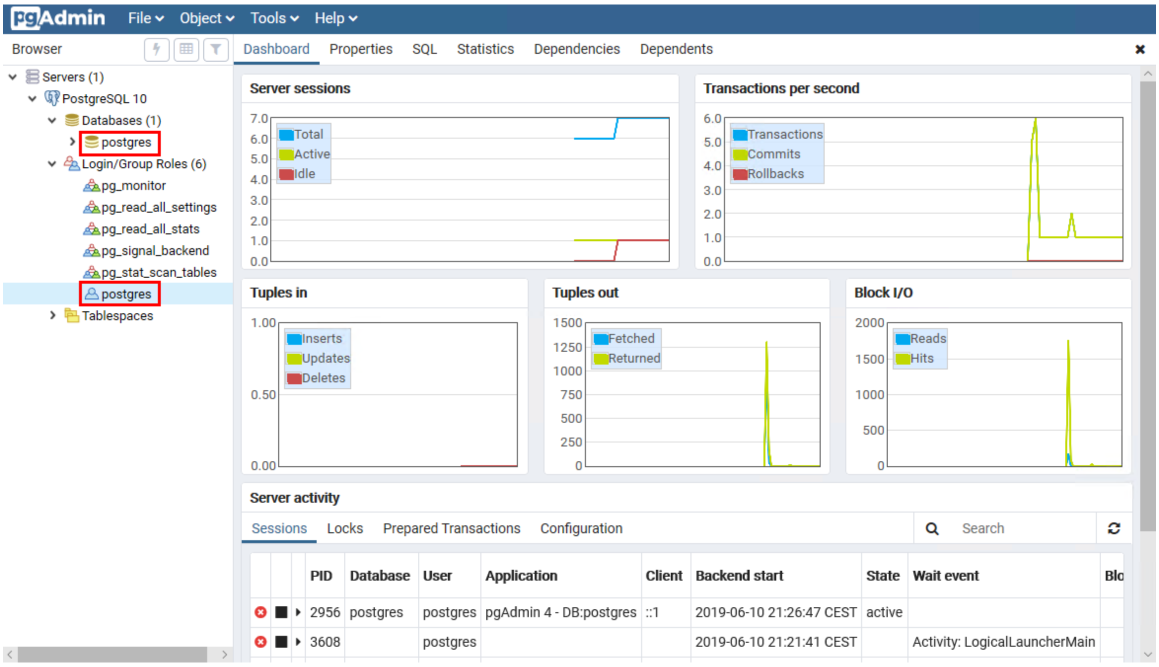
Avec le module suivant activé lors de l'assistant d'installation:



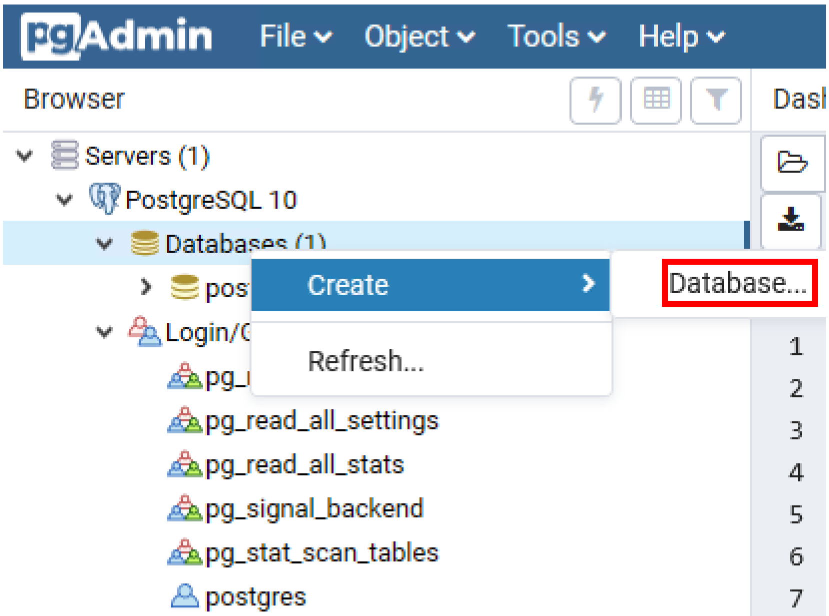
Une fois l'installation terminée, lancez pgAdmin via le menu démarrer:



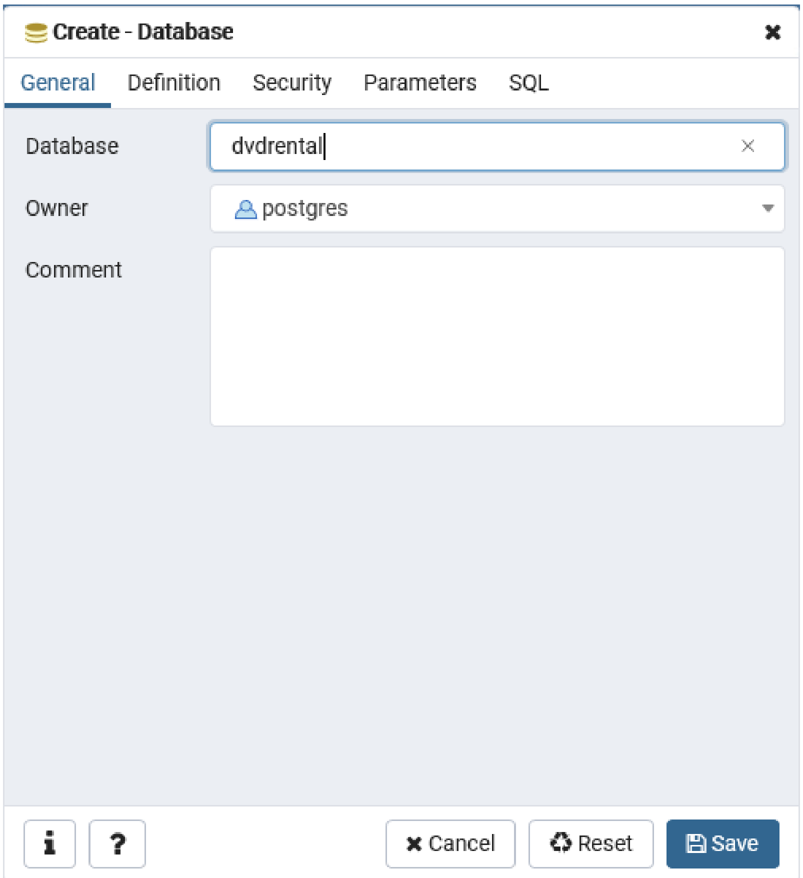
Constatez qu'il existe déjà une base de données nommée *postgres* et un utilisateur nommé *postgres* sur la page qui apparaît alors à l'écran (c'est important pour la suite!):



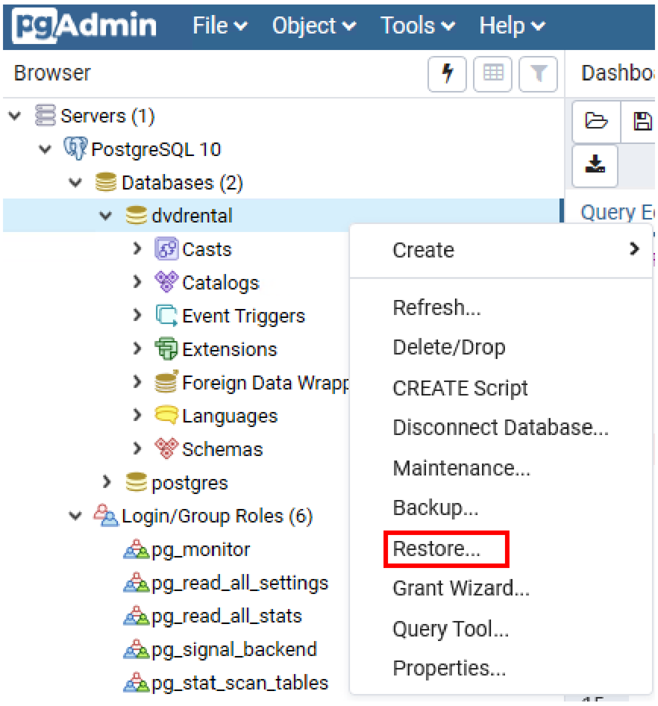
Pour les besoins, nous allons créer une nouvelle base de données avec un contenu utilisable pour R. Pour cela, nous faisons un clic droit sur *Database* et choisissons l'option *Create*/*Database*…:



Dans la fenêtre qui apparaît, mettez le nom de *Database* suivant:



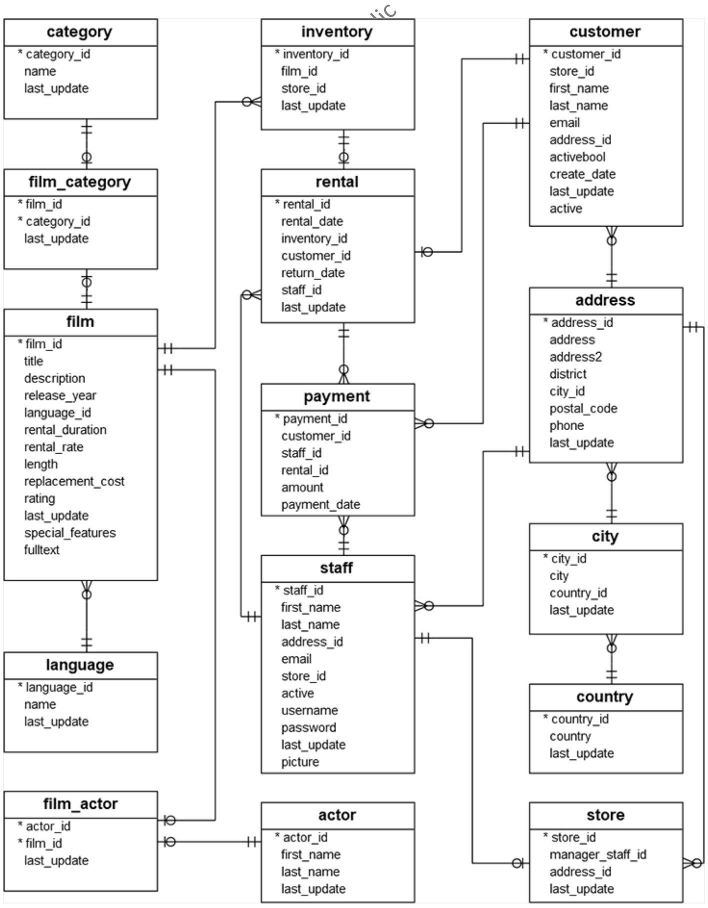
Et validez par *Save*. Une fois ceci fait, faites un clic droit sur la base de données *dvdrental* et choisissez *Restore* dans le menu contextuel:



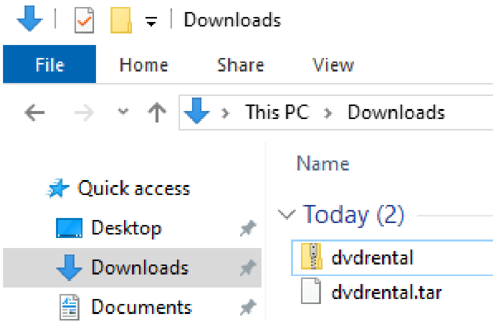
Allez télécharger la base de données d'exemple de Postegresql disponible via le lien suivant:

http://www.postgresqltutorial.com/postgresql-sample-database/

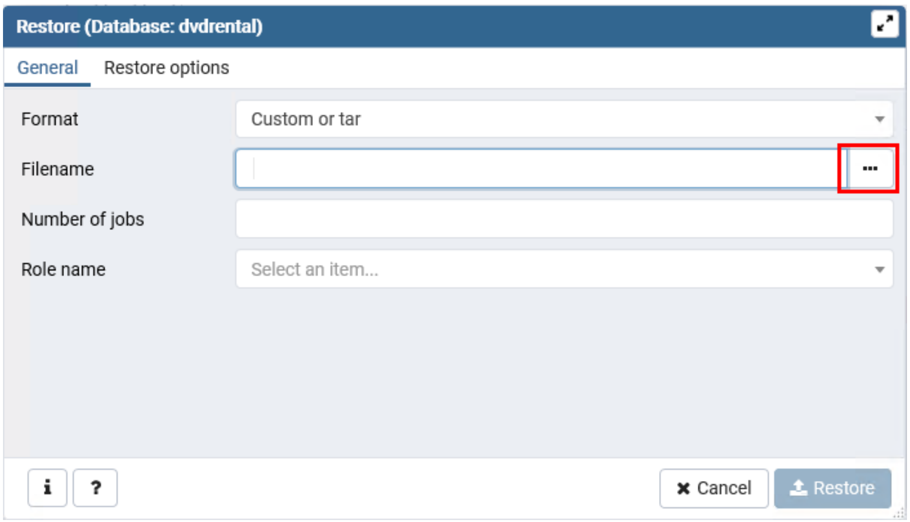
Dont le schéma relationnel est résumé ci-dessous:



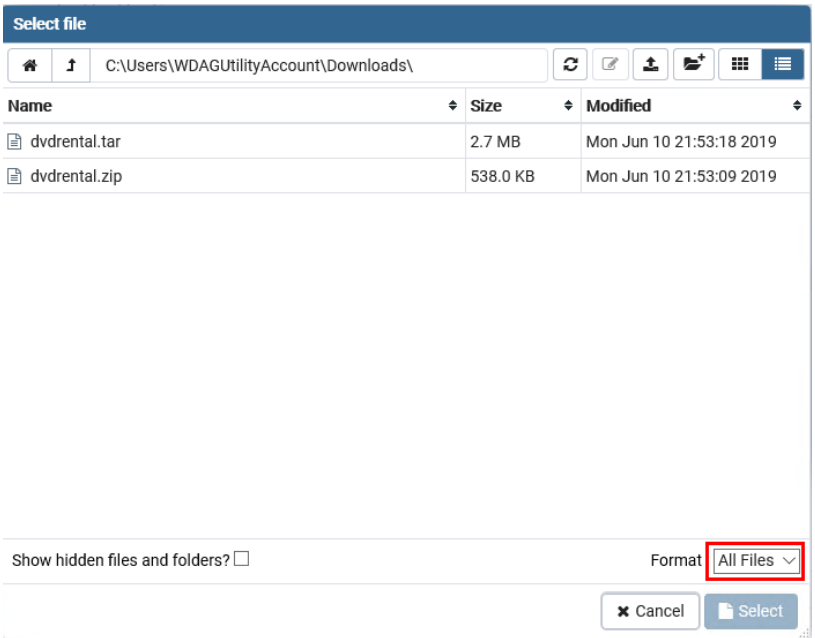
Décompressez le fichier *.zip téléchargé précédemment et conservez le fichier *.tar en résultant comme visible ci-dessous dans l'explorateur Microsoft Windows:



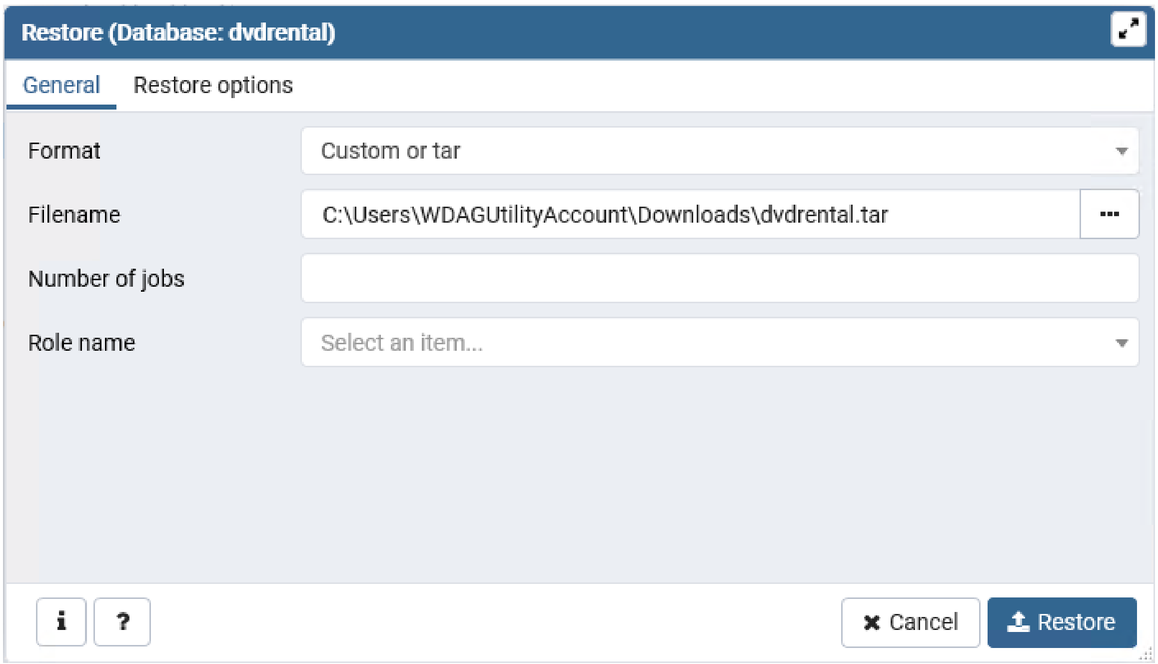
De retour dans *pgAdmin*, sélectionnez le bouton … à la hauteur du champ *Filename*:



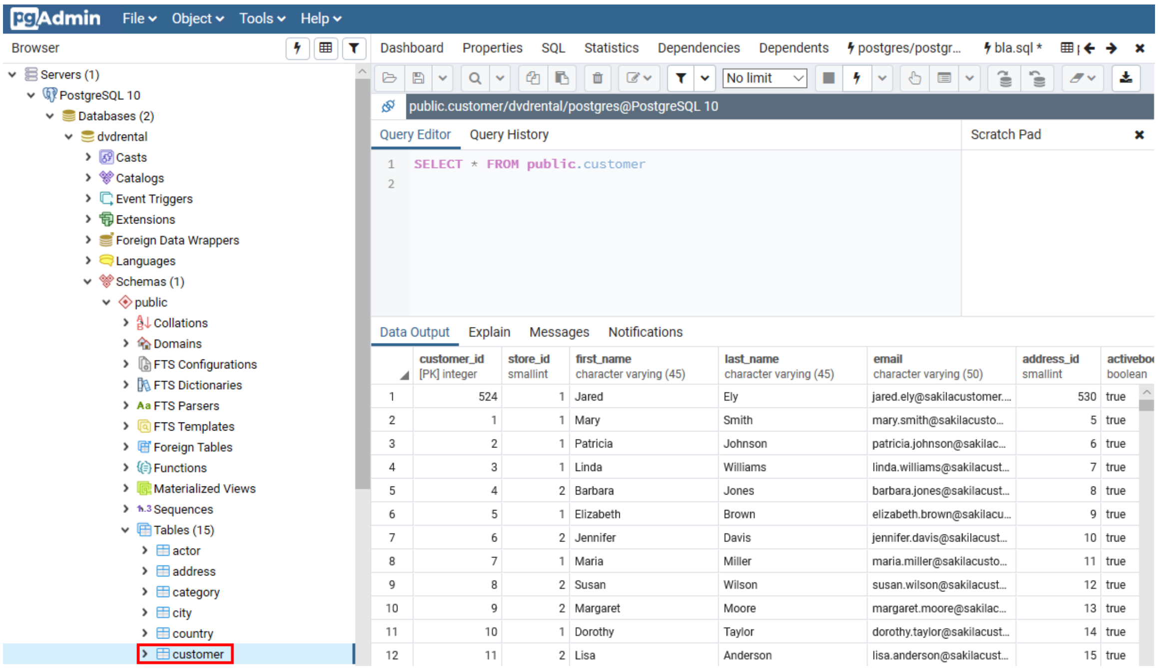
Apparaît alors la boîte de dialogue suivante, où vous devrez naviguer jusqu'au dossier où vous avez décompressé le fichier *.zip, et sélectionner l'option *All files* (sinon quoi vous ne verrez pas le fichier *.tar):



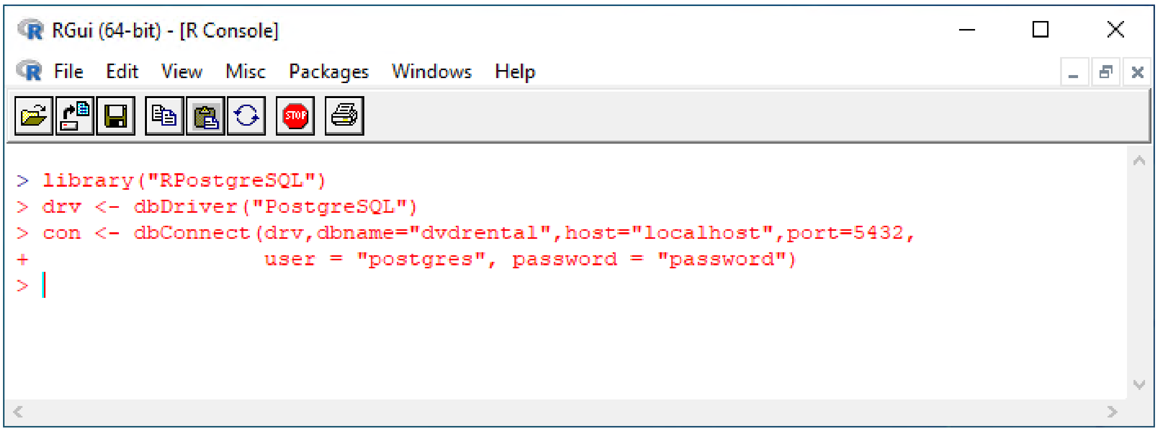
Validez par *Select* et vous revenez alors à:



Cliquez sur *Restore* et alors toutes les tables seront importées (un exemple est donné avec la table *customer* dans la capture d'écran ci-dessous):



Maintenant, nous pouvons retourner à R avec le package **RPostgresSQL** et ses fonctions **dbDriver( )**, **dbConnect( )** pour nous connecter au serveur comme visible ci-dessous:



Une fois la connexion effectuée, nous utilisons la fonction **dbExistsTable( )** pour vérifier que la table *customer* existe et enfin nous utilisons la fonction **dbGetQuery( )** pour y lancer une requête:

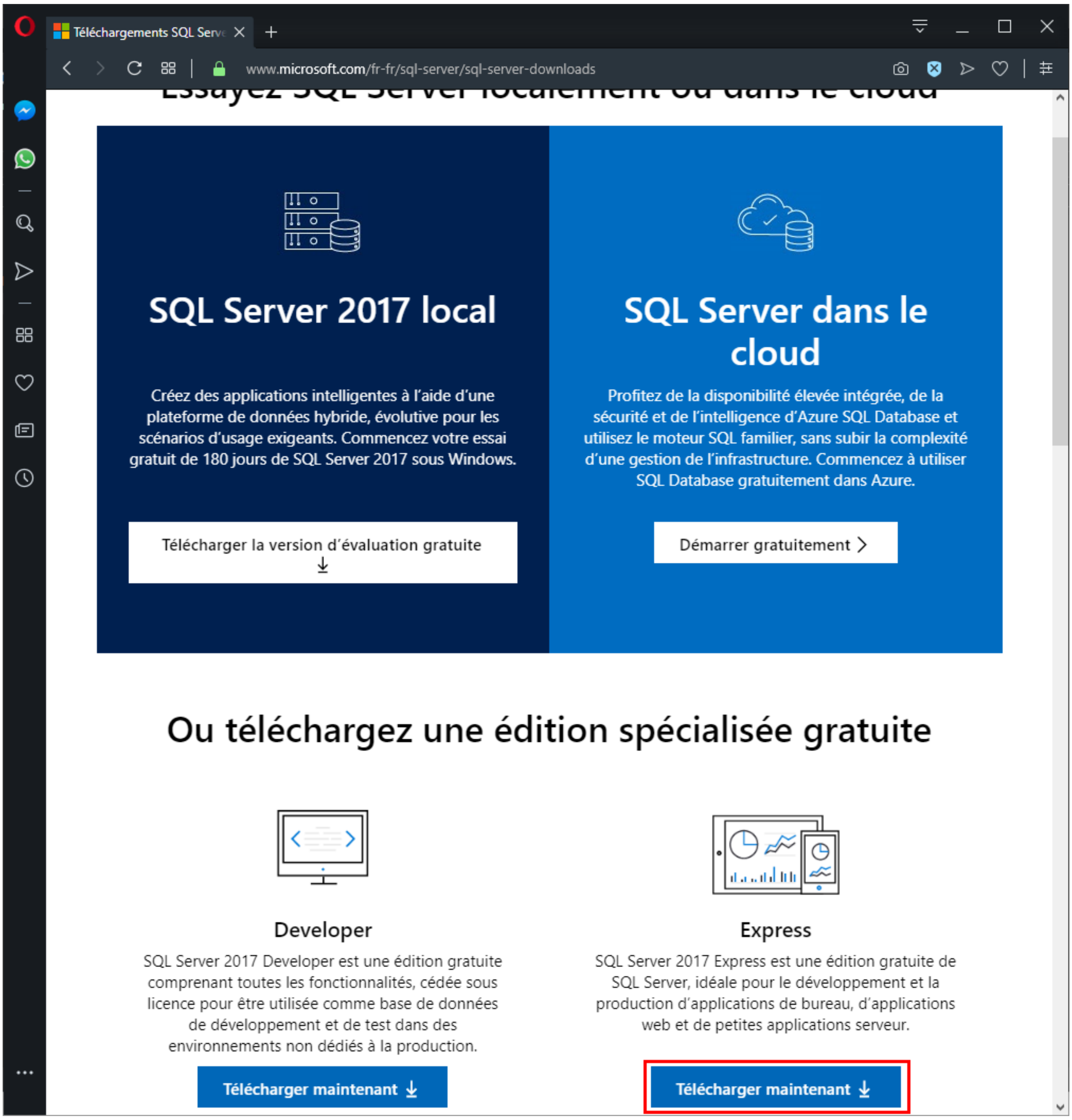


Et enfin, nous avons fermé la connexion avec la fonction **dbDisconnect( )**!

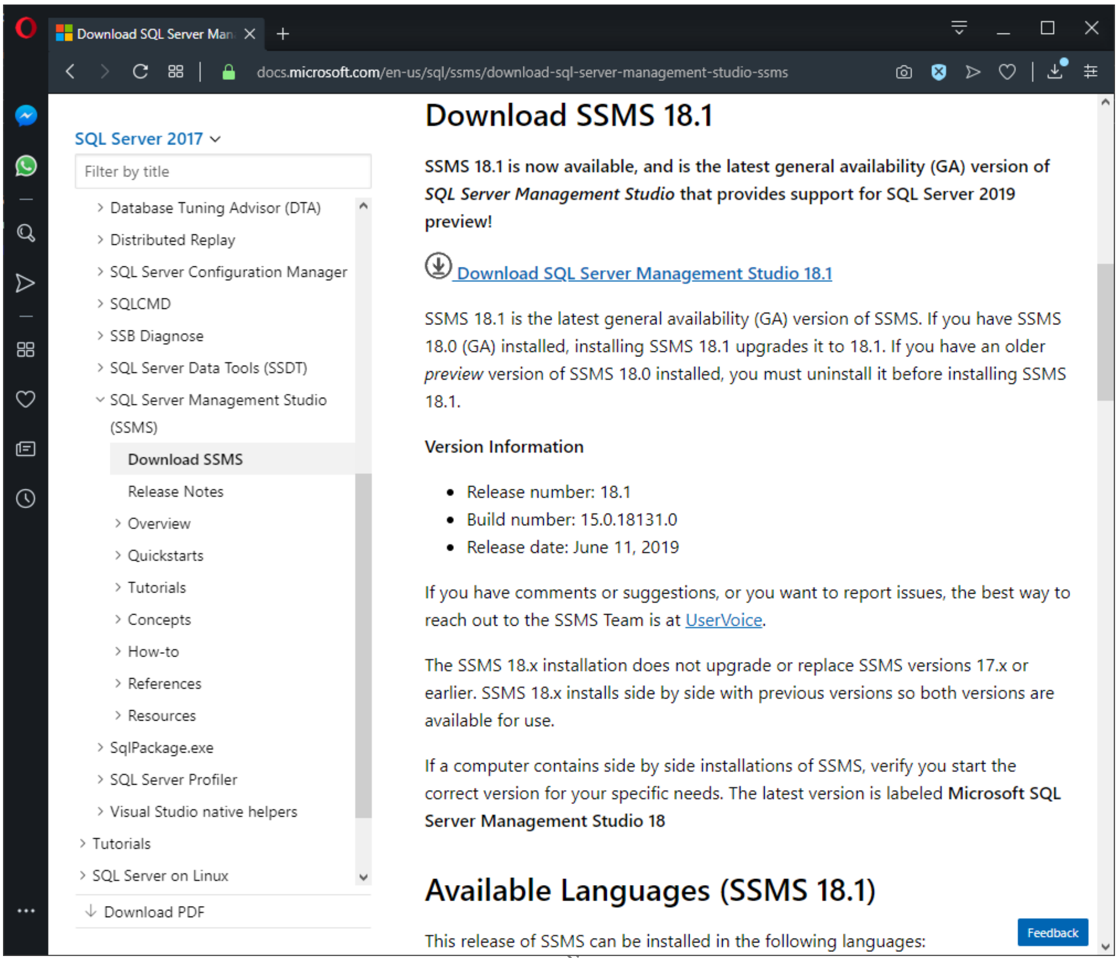
Exemple 62.: Importer des données d'un serveur Microsoft SQL Server Express

R 3.6.0

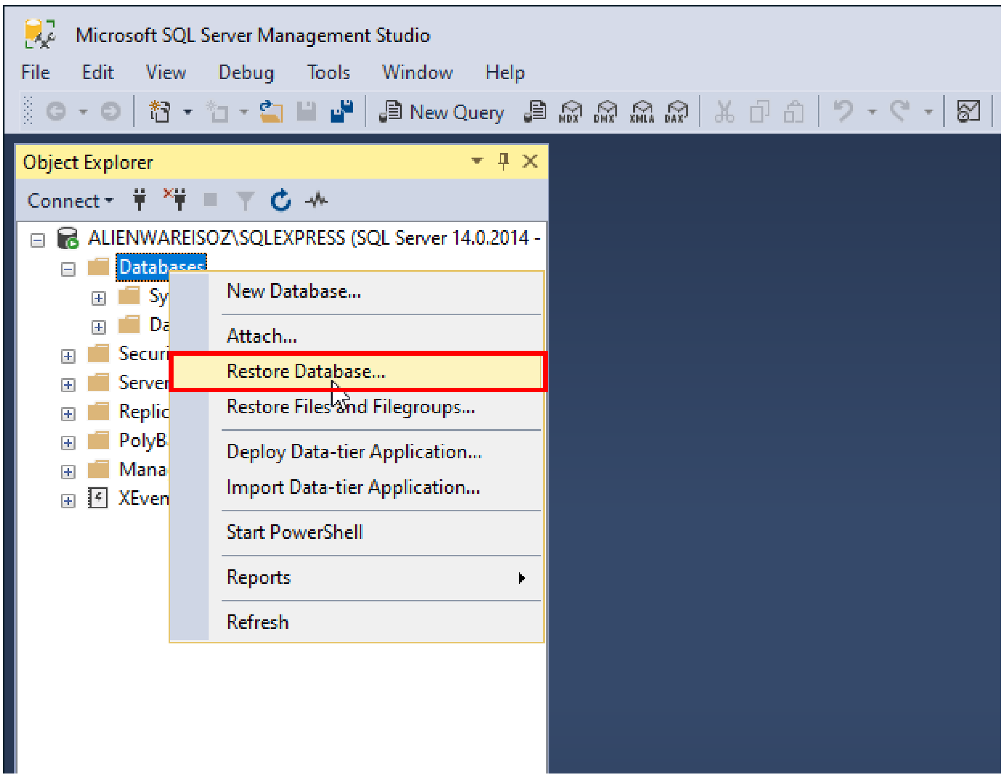
Pour l'exemple nous supposerons que le lecteur aura téléchargé et installé Microsoft SQL Server Express:



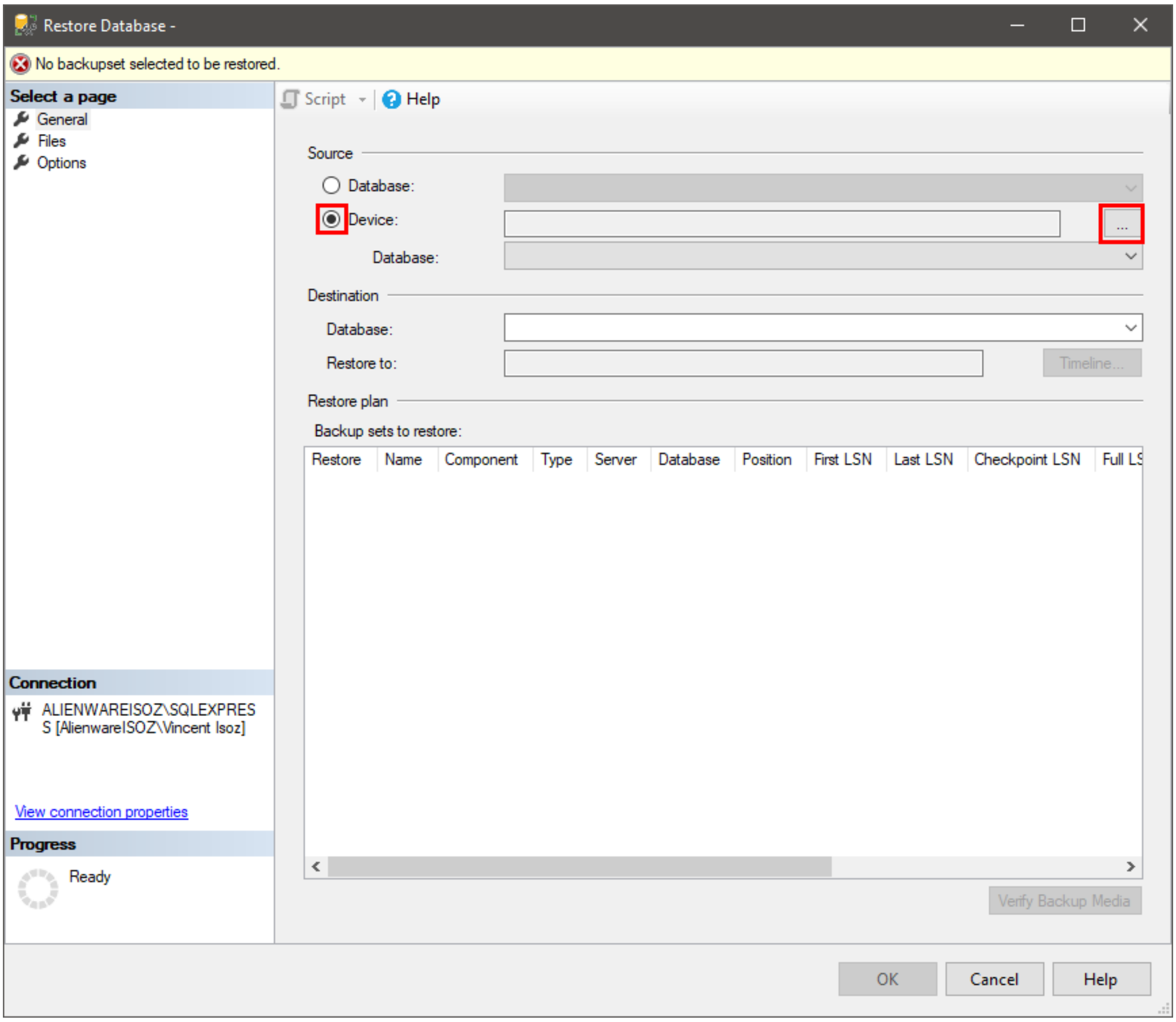
Ainsi que Microsoft SQL Server Management Studio:



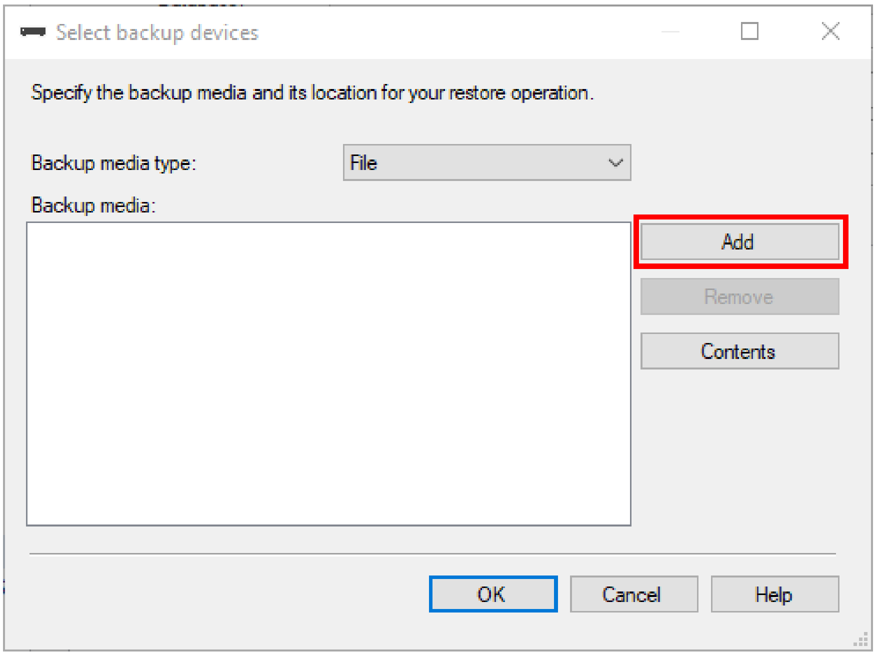
Une fois tout cela installé, ouvrez SQL Server Management Studio et faites un clic droit sur *Databases* et choisissez l'option *Restore Database* dans le menu contextuel:



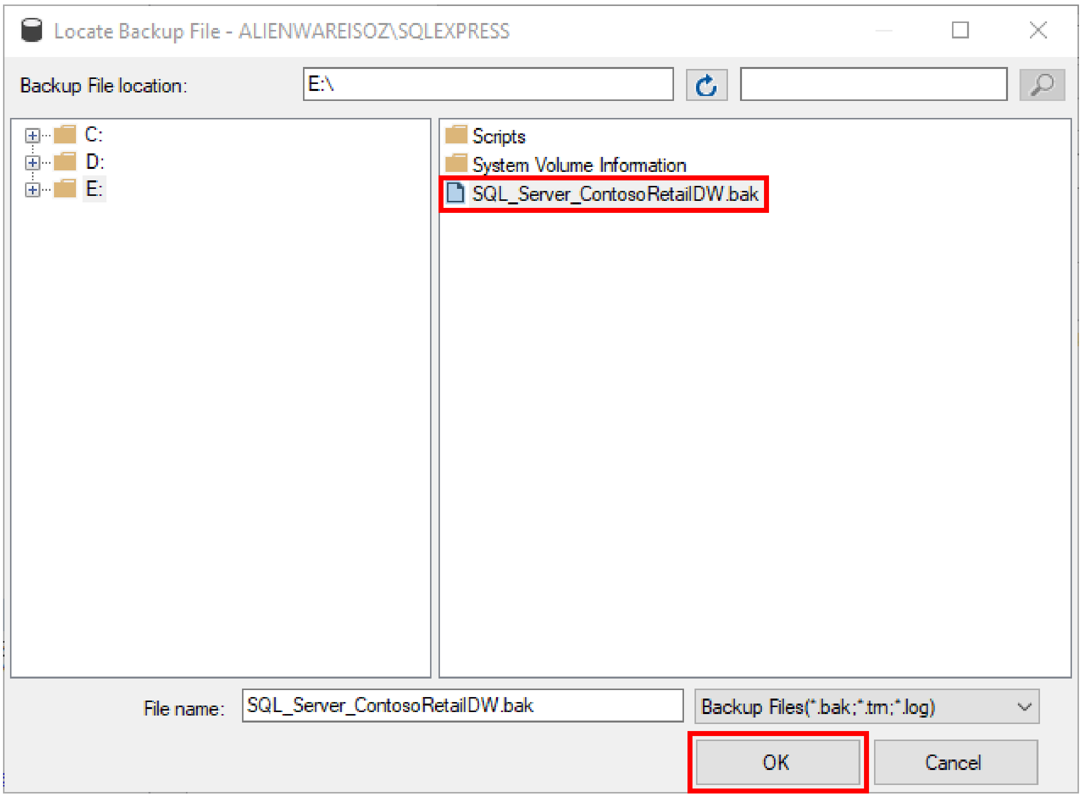
Activez le bouton radio en face de *Device* et cliquez sur le buton …:



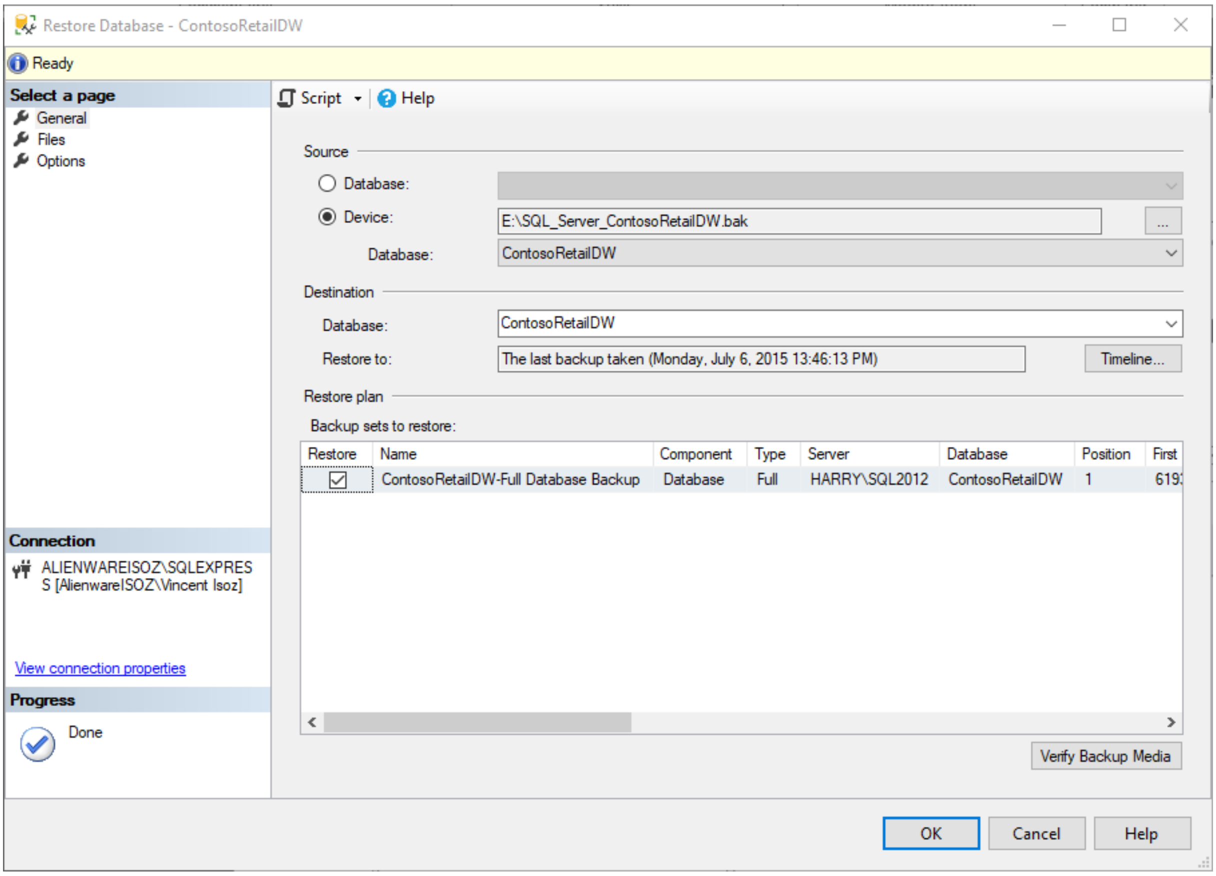
Apparaît alors la boîte de dialogue suivante où on cliquer sur *Add*:



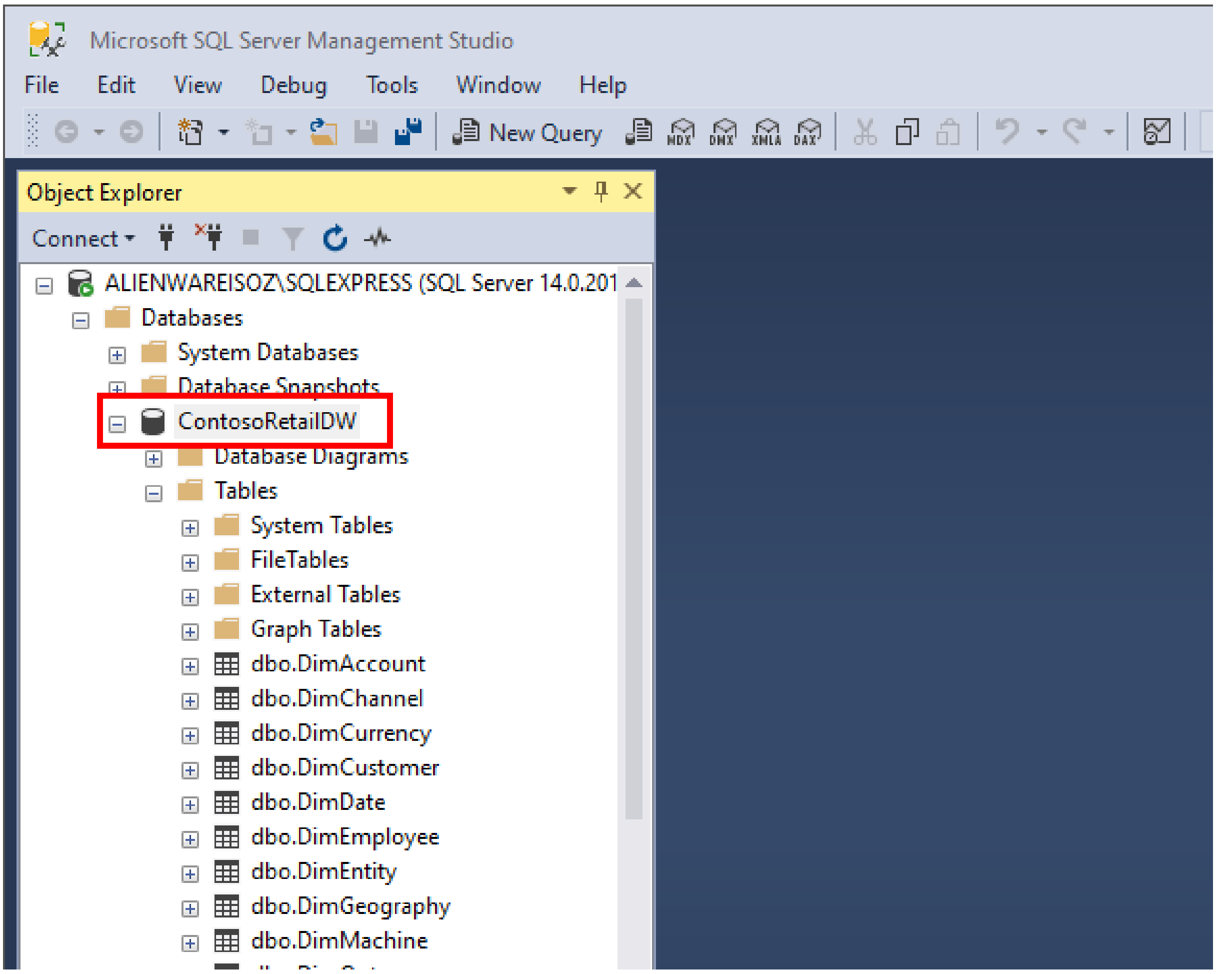
Et naviguez jusqu'au dossier où se trouve le fichier *SQL_Server_ContosoRetailDW.bak* qu'on vous donné pour le cours:



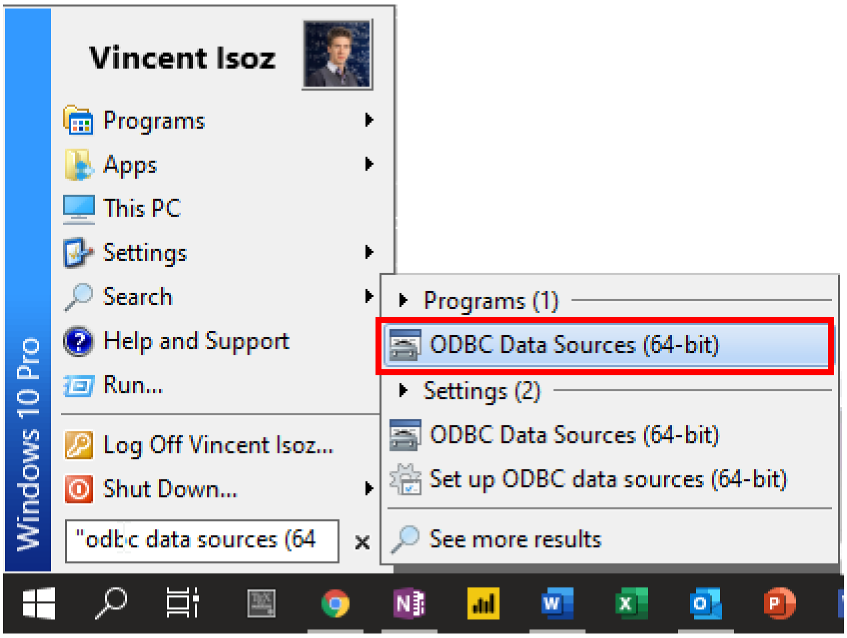
On valide deux fois par **OK** pour revenir à:



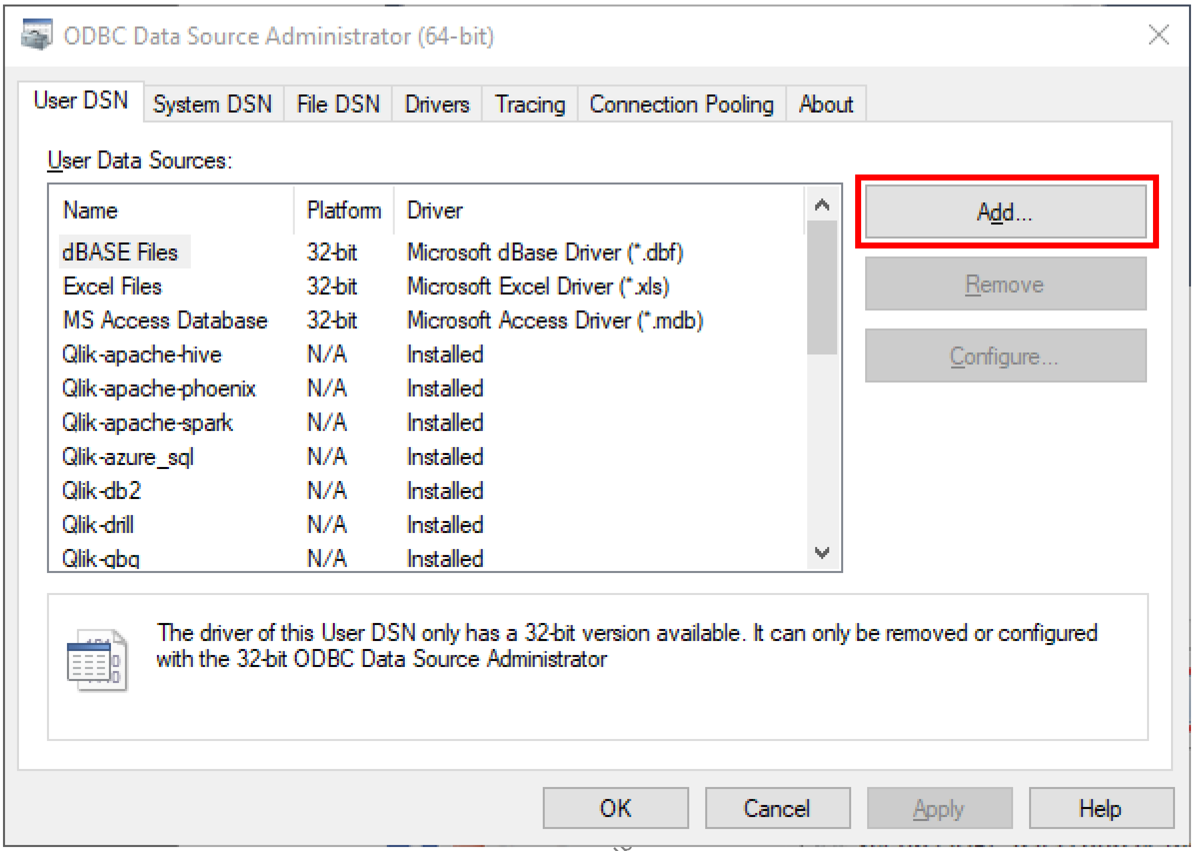
Et encore une fois on valide par **OK**, et on y voit alors:



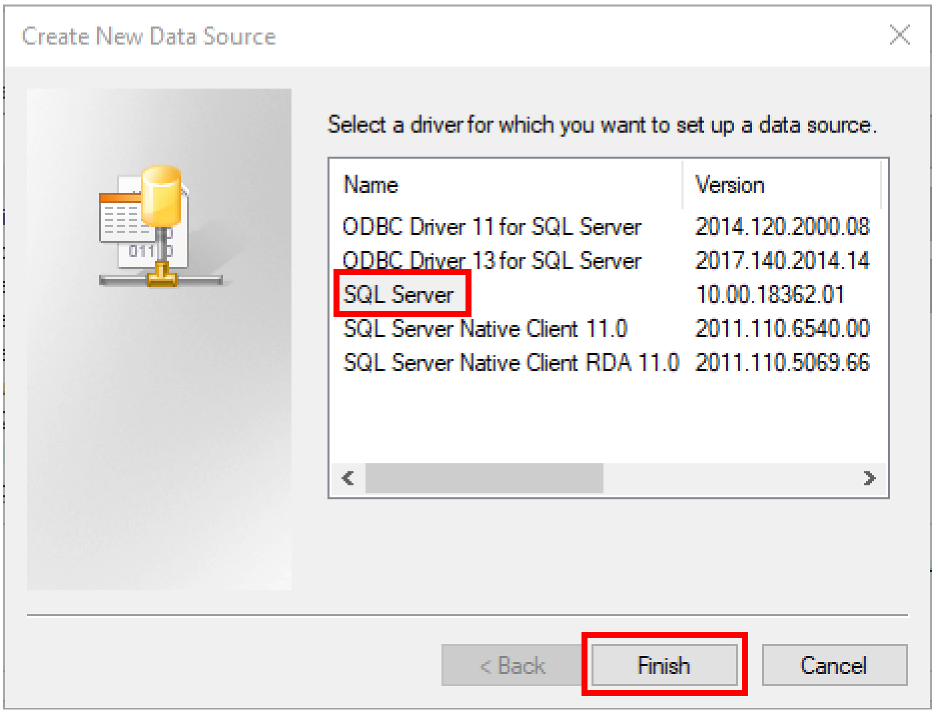
Maintenant dans Microsoft Windows, on ouvre le gestion de sources ODBC:



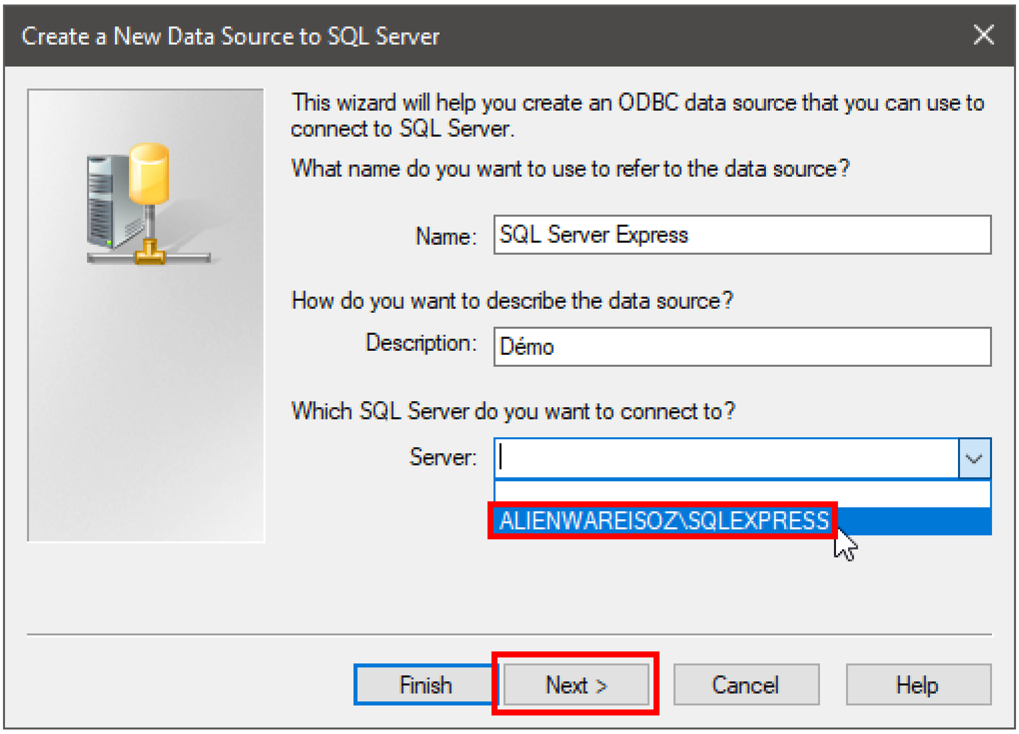
Et dans la boîte de dialogue qui apparaît, dans l'onglet *User DSN*, on clique sur le bouton *Add…*:



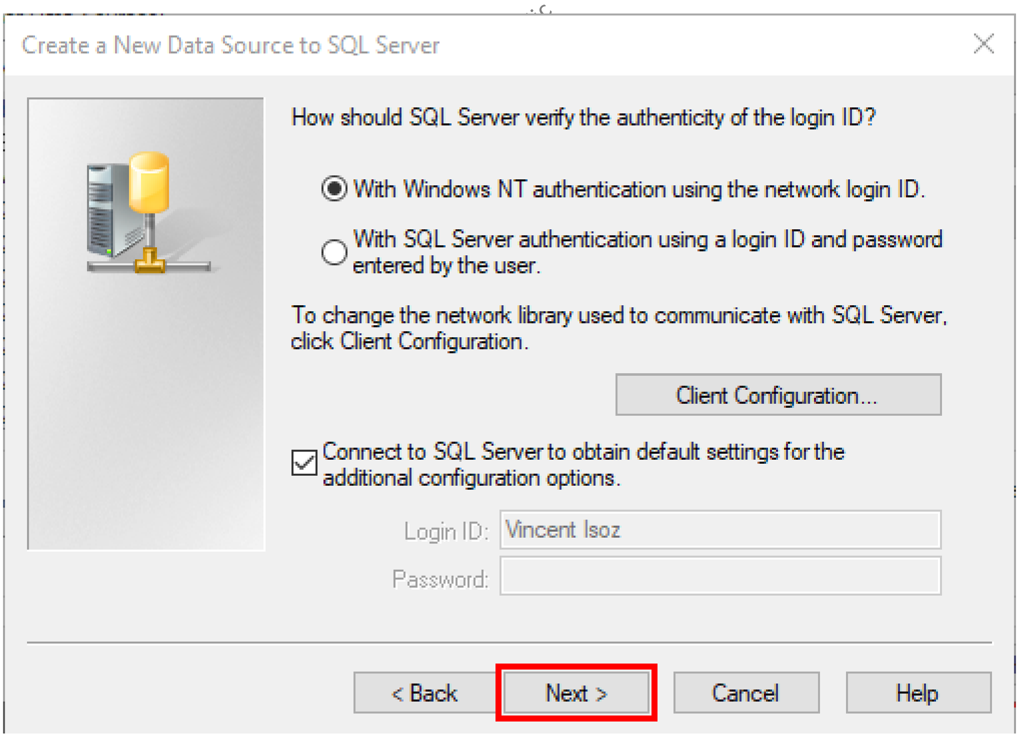
Et dans la boîte de dialogue qui apparaît alors, on sélectionner *SQL Server* et on clique sur le bouton **Finish**:



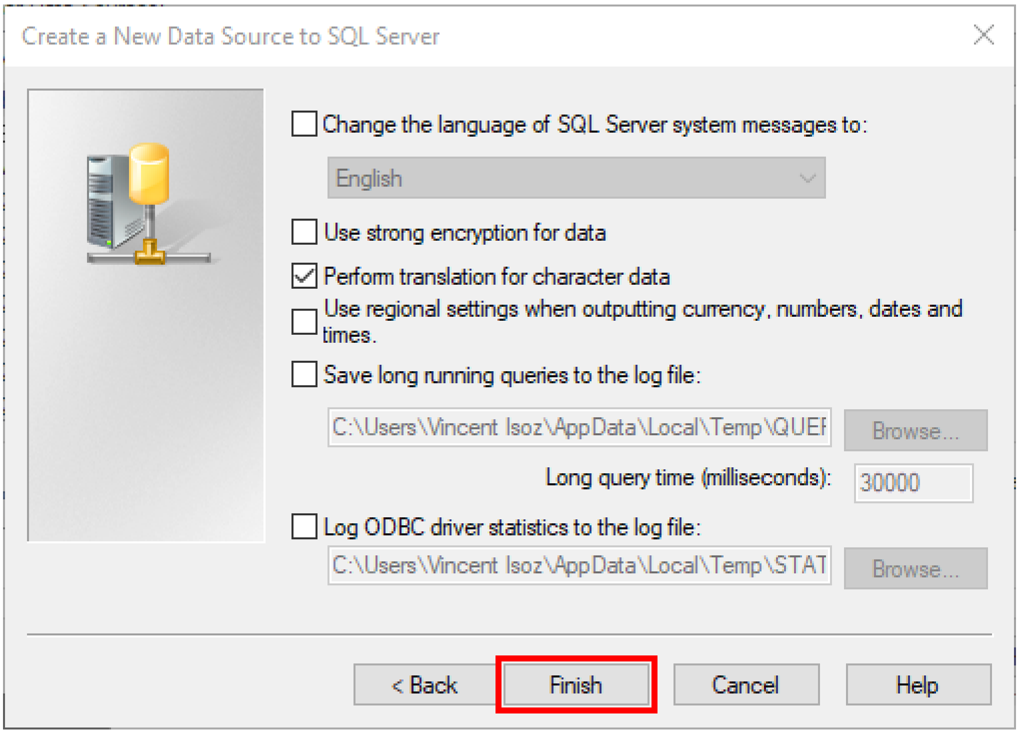
Apparaît la boîte de dialogue alors suivante que l'on remplira comme indiqué et où on sélectionnera le nom du serveur installé précédemment:



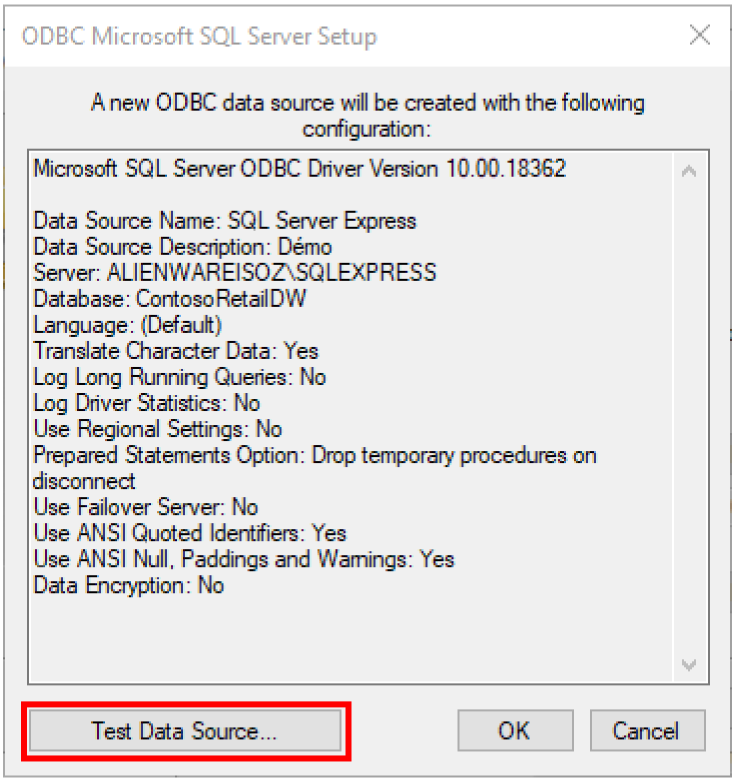
On clique alors sur **Next**:



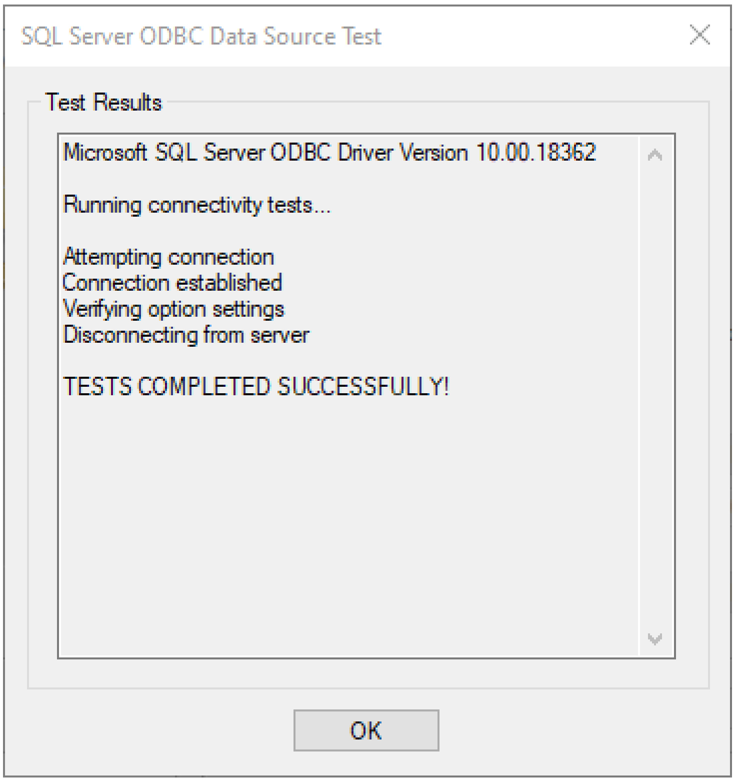
Et encore sur **Next**:



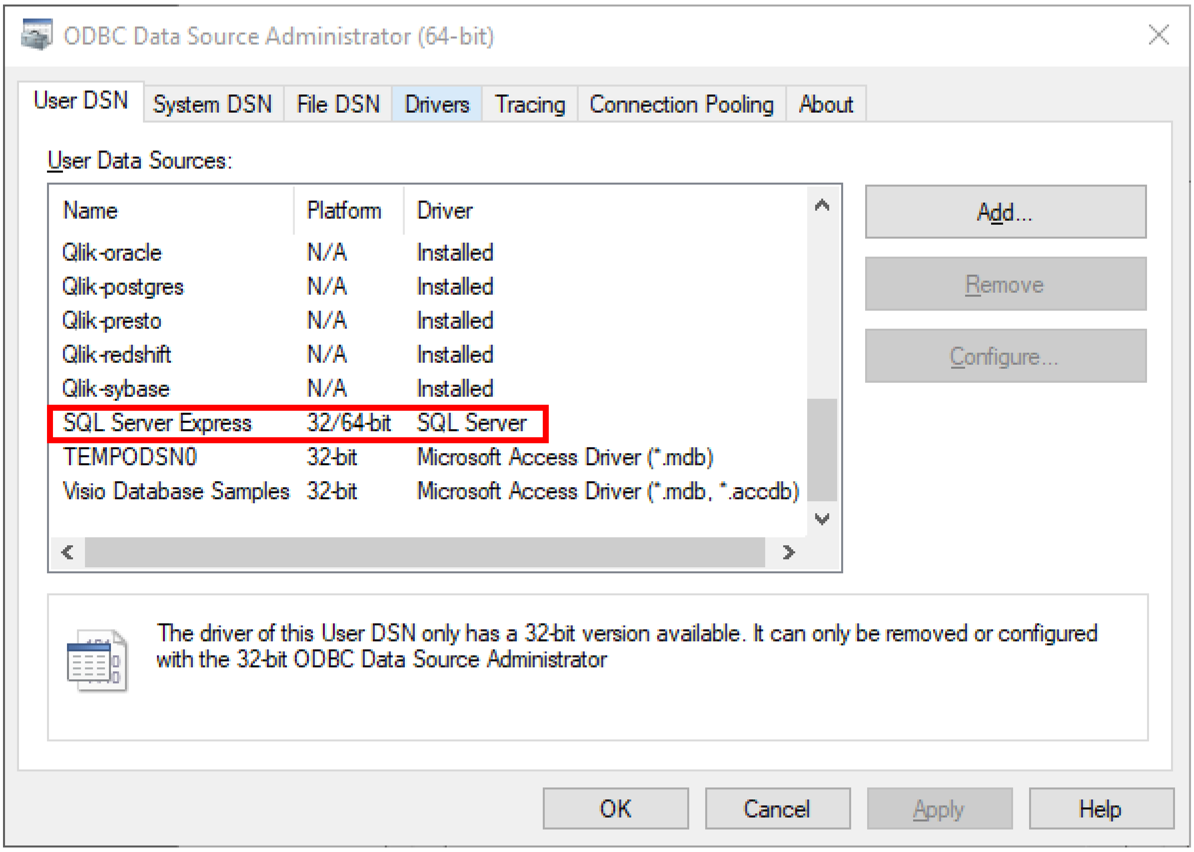
Et encore sur **Next** pour arriver à:



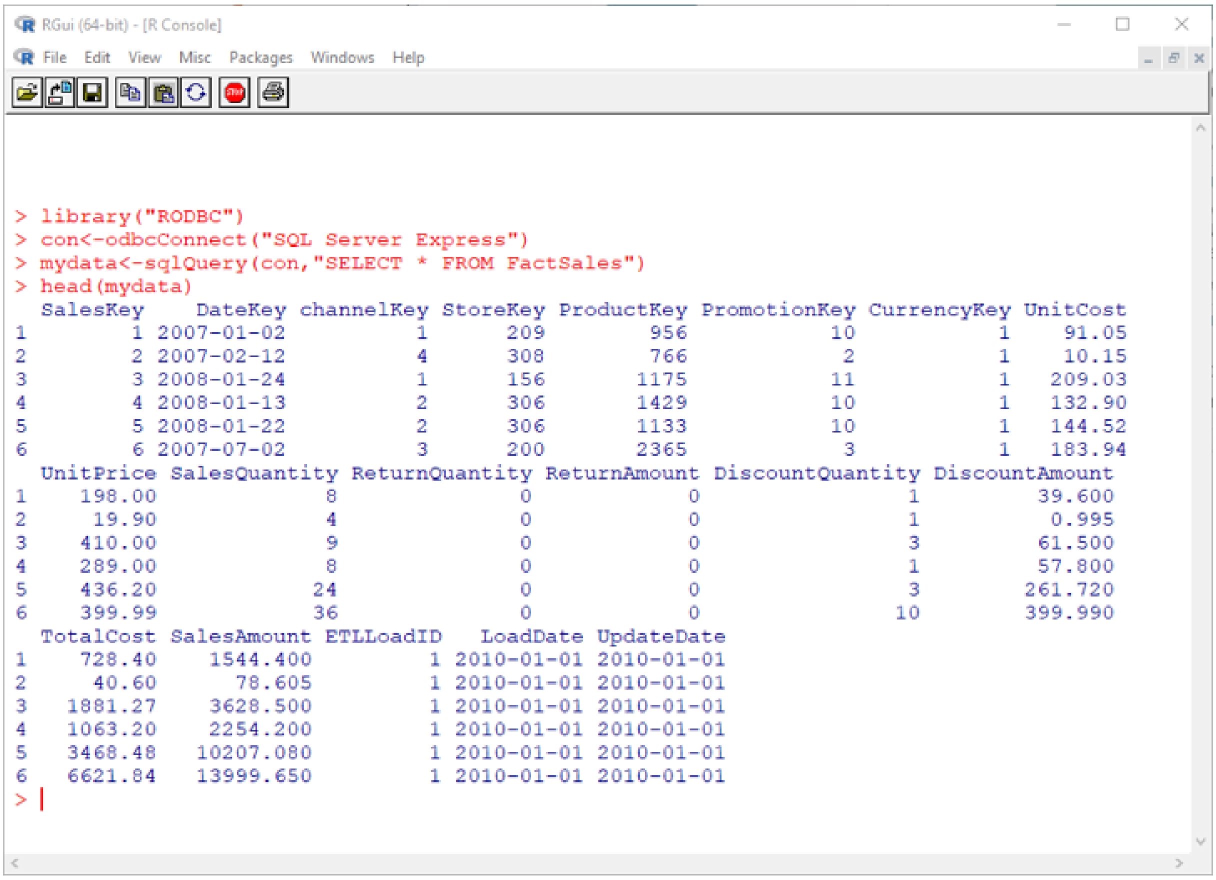
On cliquer sur *Test Data source* pour s'assurer que tout s'est bien passé:



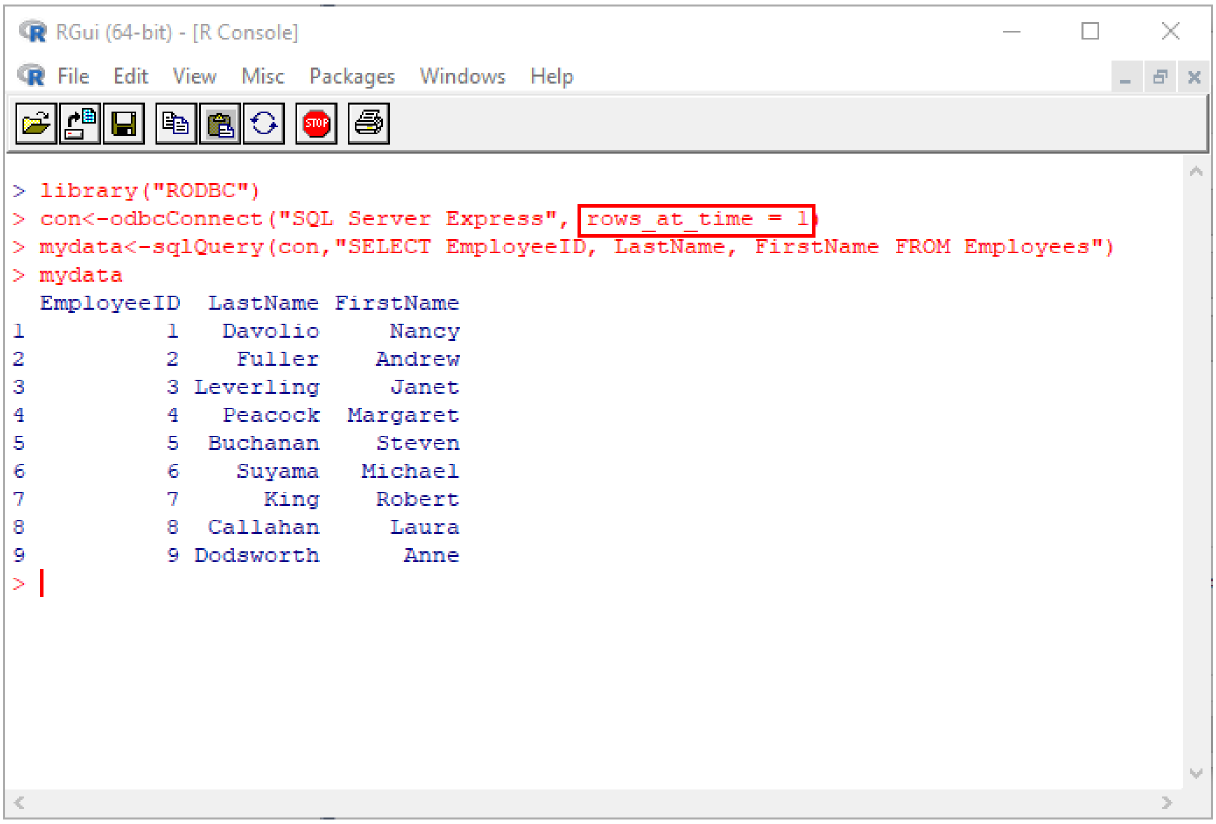
Et là on a une bonne nouvelle (ce qui n'est de loin pas toujours le cas…) Et on valide deux fois par **OK** pour revenir à:



Et on valide par **OK** pour finaliser le tout. On revient alors à R et on utilise le package **RODBC**, que l'on connaît déjà, et ce de la manière suivante:



Attention!!! Dans certains cas il faut ajouter un paramètre supplémentaire (*rows_at_time*) pour que cela marche:



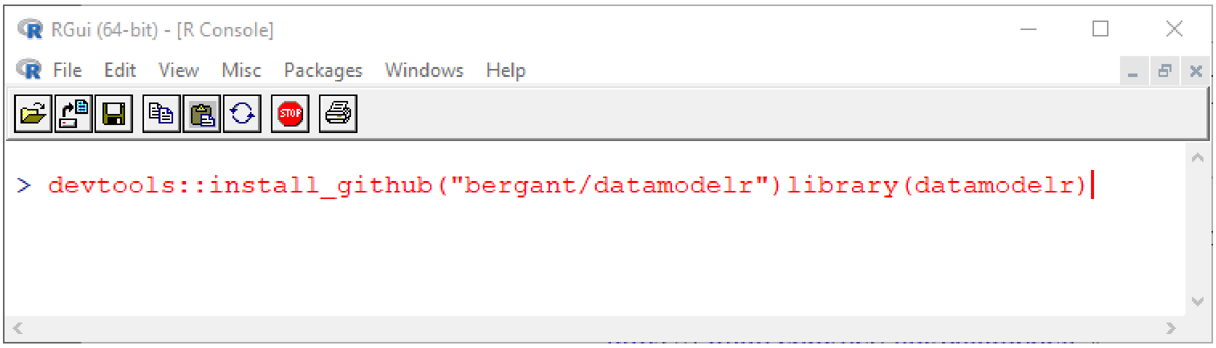
Exemple 63.: Extraire le schéma relationnel d'un BDD

R 3.0.2

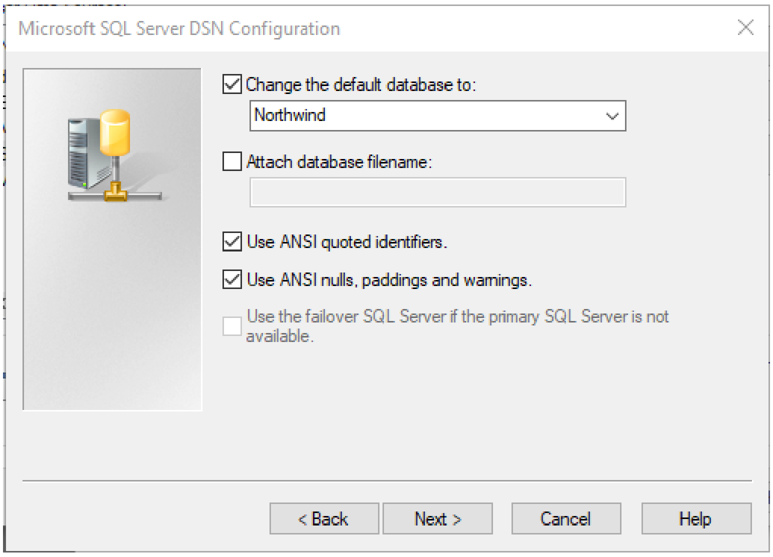
Le package que nous allons utiliser n'est pas au jour où nous écrivons ces lignes disponible sur le CRAN. Voici sa documentation:

https://github.com/bergant/datamodelr

On l'installera comme à l'habitude avec tout package sur github avec devtools:



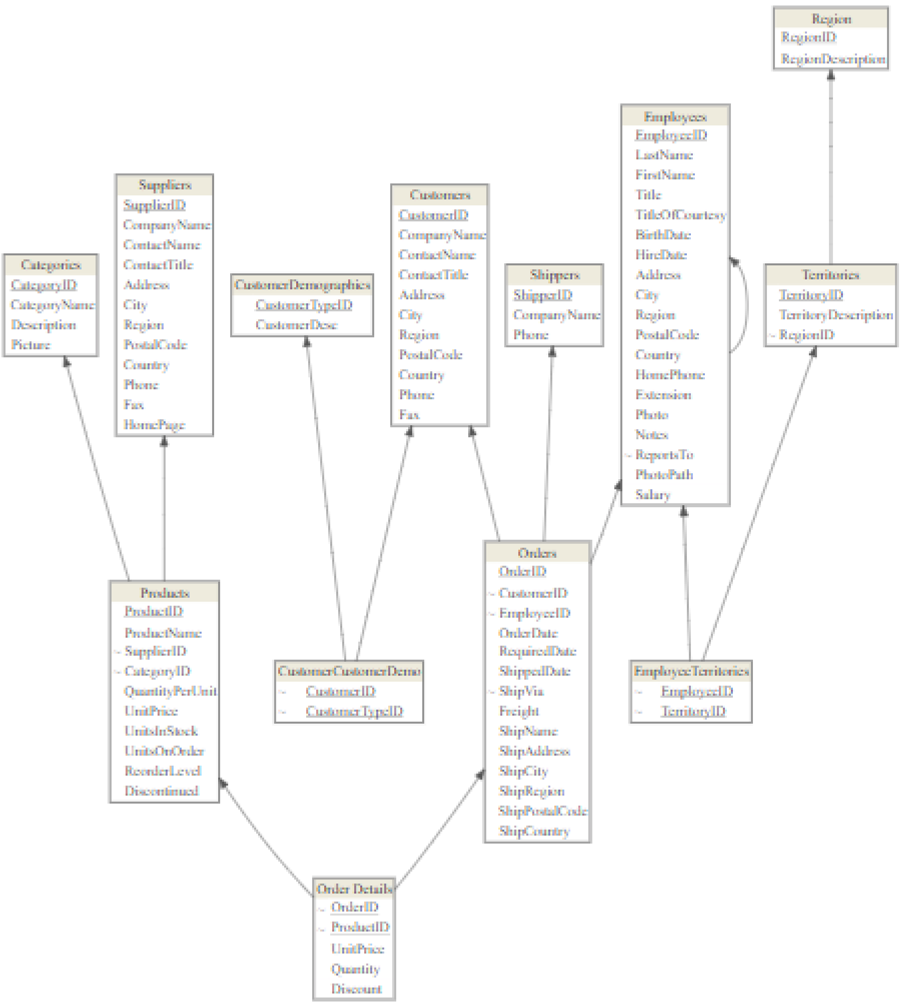
Pour la démo changez la base de données par défaut du connecteur SQL Server:



Maintenant nous faisons usage du package susmentionné:

```
> library("datamodelr")
> library("RODBC")
> con <- odbcConnect(dsn = "SQL Server Express")
> sQuery <- dm_re_query("sqlserver")
> dm_northwind <- sqlQuery(con, sQuery, stringsAsFactors = FALSE, errors=TRUE)
> odbcClose(con)
>
> #conversion en modèle de données
> dm_northwind <- as.data_model(dm_northwind)
>
> graph <- dm_create_graph(dm_northwind, rankdir = "BT")
> dm_render_graph(graph)
>
```

Ce qui donne:



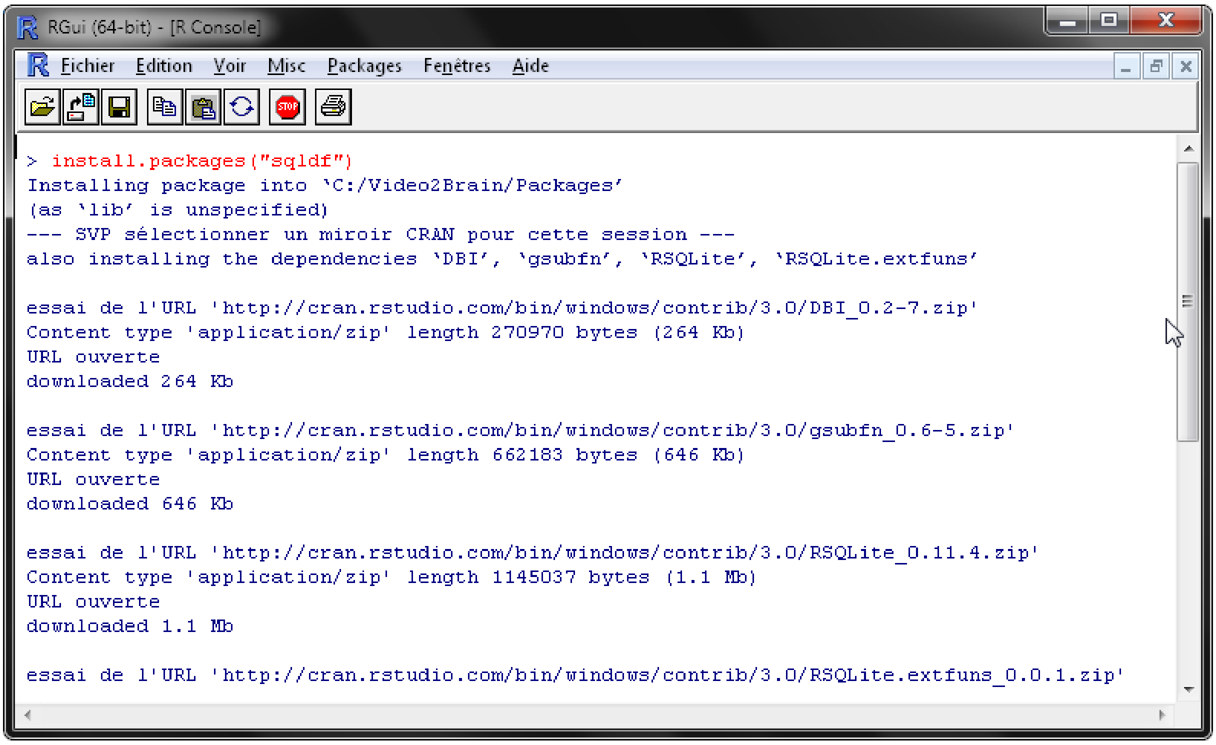
Exemple 64.: Utiliser le SQL

R 3.0.2

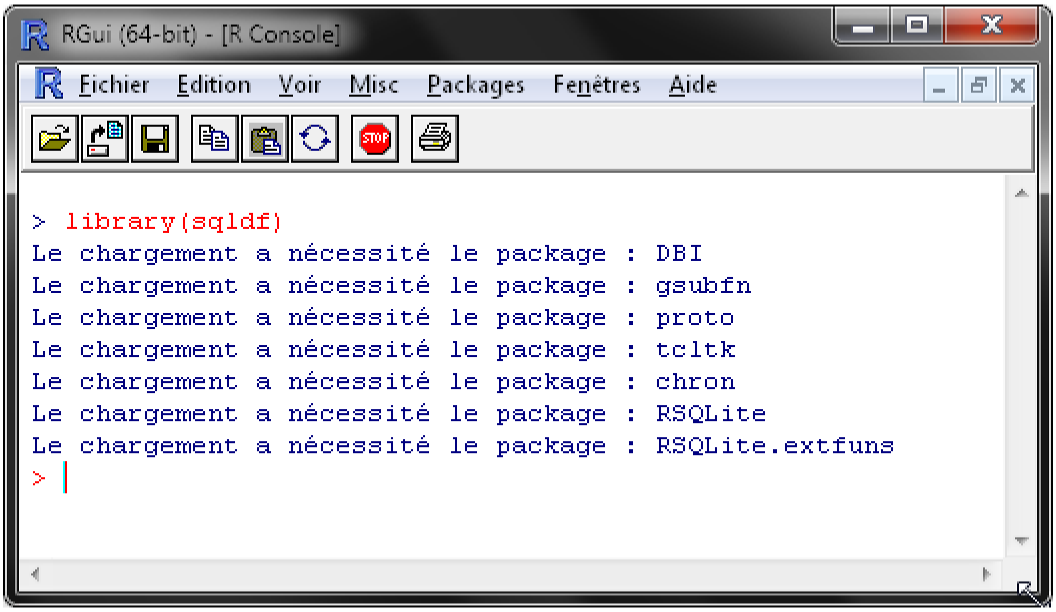
Le SQL est un langage extrêmement puissant, bien pensé et très rapide pour traiter et manipuler des données dans des bases de données relationnelles. Il existe ainsi un package dans R permettant d'utiliser le SQL sur un ou plusieurs data frames en même temps afin de pouvoir utiliser les jointures. Ce package est extrêmement utile dans la pratique du big data et plus rapide que les commandes classiques de R.

Voyons quelques exemples qui seront basés sur les tables d'Oracle Express qui ont été préalablement exportées au format *.csv.

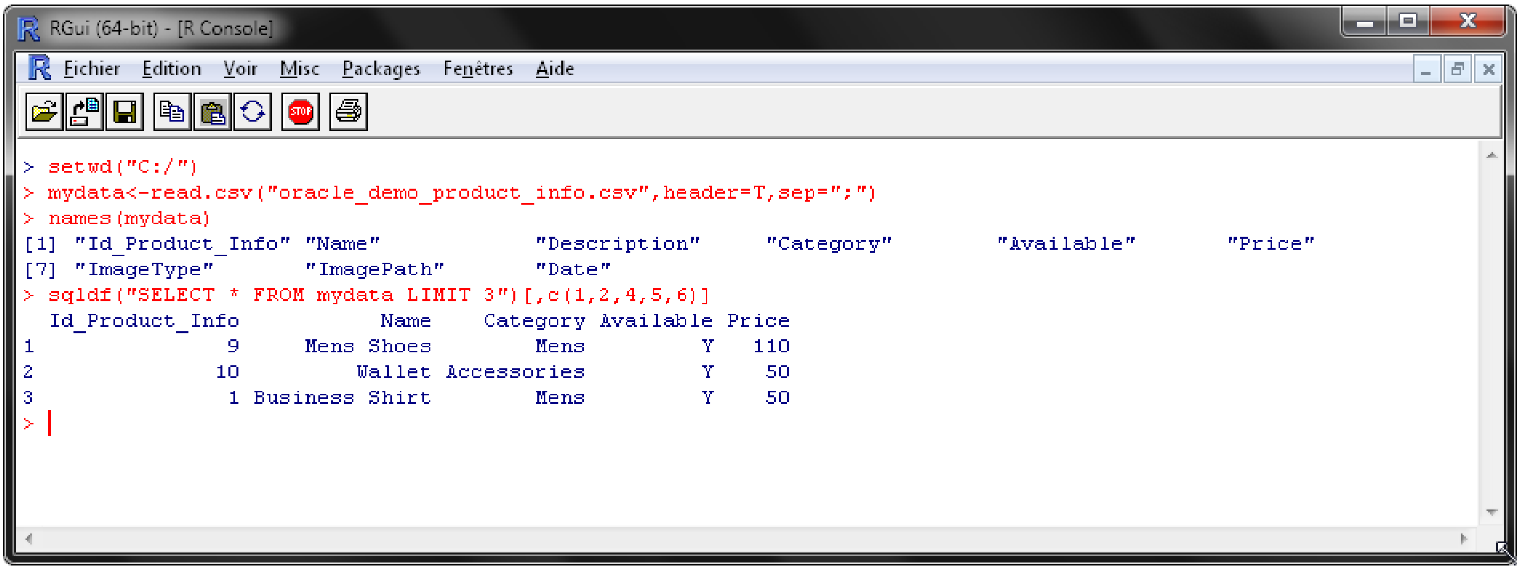
D'abord, nous installons le package **sqldf** (qui utilise SQLite en arrière plan):



Ensuite, nous le chargeons:



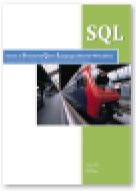
Et nous pouvons commencer à appliquer notre connaissance du SQL en utilisant la commande **sqldf( )** :



Il n'est malheureusement pas possible de faire tout ce qui est possible dans Oracle avec du pour SQL mais nous pouvons faire à ce jour:

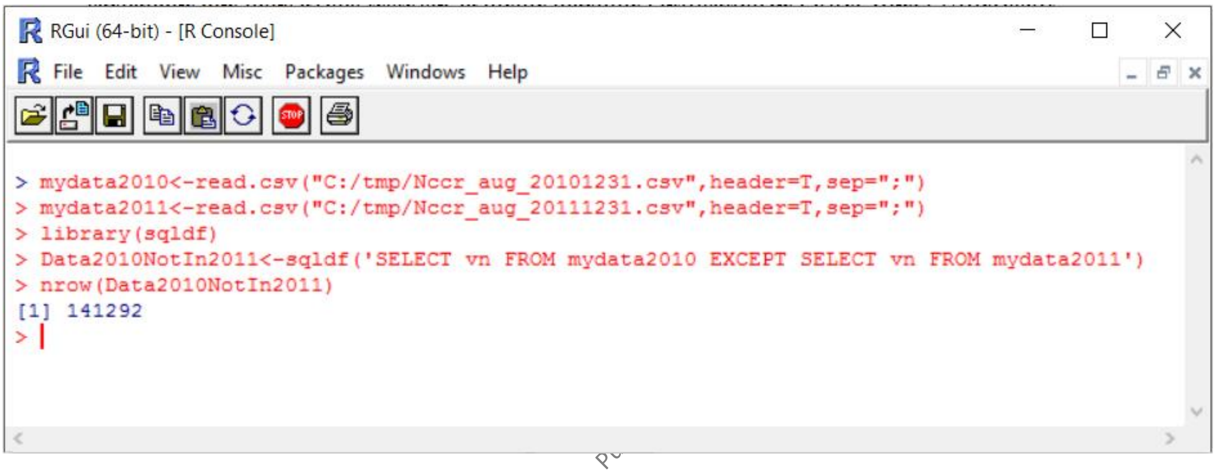
- Des requêtes de sélection (SELECT)
- Des requêtes de sélection (EXCEPT)
- Des requêtes de comptage (COUNT)
- Des requêtes de limite (LIMIT)
- Des requêtes avec filtre (WHERE)
- Des requêtes de groupement (GROUP BY)
- Des requêtes de tri (ORDER BY)
- Des requêtes avec jointure (INNER JOIN)

Pour plus d'informations avec des cas pratiques basés sur le même jeu de données, le lecteur pourra se reporter à mon e-book sur le SQL (282 pages):



www.sciences.ch/tiny?url=18

Suite à un problème soumis à par un élève qui consiste à comparer deux jeux de données afin de savoir quels sont les éléments de l'un qui sont pas dans l'autre relativement à un champ donné sans prendre en compte les doublons, voici un autre excellent exemple d'application de R sachant que nous avons testé sur la même machine l'utilisation de Power Query (Microsoft Excel) et la requête de non-correspondance dans Microsoft Access qui étaient tous deux non satisfaisants et beaucoup trop lents par rapport à R:



Exemple 65.: Créer et importe de bases SQLite

R 3.6.3

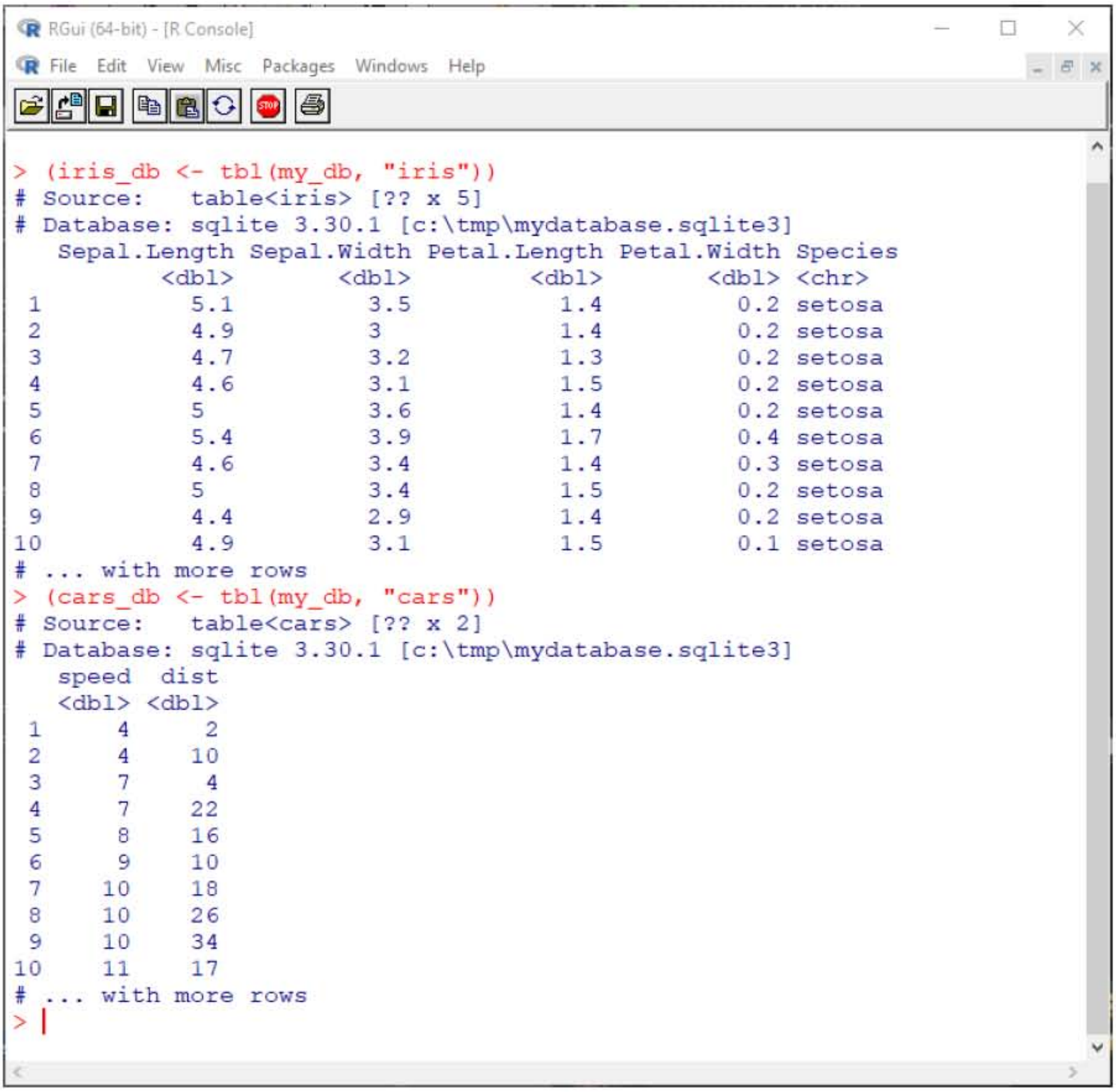
Voyons un exemple:

```
> library("dplyr")
> library("dbplyr")
> #cas avec plusieurs dataframes
> data(iris)
> data(cars)
> res <- list("iris" = iris, "cars" = cars)
> my_db <- dplyr::src_sqlite("c:/tmp/mydatabase.sqlite3",create = TRUE)
> #la base est vide
> my_db
src:  sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
tbls:
> lapply(seq_along(res), function(i, dt = res) dplyr::copy_to(my_db, dt[[i]], names(dt)[[i]]))
[[1]]
# Source:   table<iris> [?? x 5]
# Database: sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
# ... with more rows

[[2]]
# Source:   table<cars> [?? x 2]
# Database: sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
   speed  dist
   <dbl> <dbl>
 1     4     2
 2     4    10
 3     7     4
 4     7    22
 5     8    16
 6     9    10
 7    10    18
 8    10    26
 9    10    34
10    11    17
# ... with more rows

> #la base n'est plus vide
> my_db
src:  sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
tbls: cars, iris, sqlite_stat1, sqlite_stat4
> 
```

Rien d'extraordinaire. Poursuivons un peu:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> (iris_db <- tbl(my_db, "iris"))
Source: table<iris> [?? x 5]
Database: sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
<dbl> <dbl> <dbl> <dbl> <chr>
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
7 4.6 3.4 1.4 0.3 setosa
8 5 3.4 1.5 0.2 setosa
9 4.4 2.9 1.4 0.2 setosa
10 4.9 3.1 1.5 0.1 setosa
... with more rows
> (cars_db <- tbl(my_db, "cars"))
Source: table<cars> [?? x 2]
Database: sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
speed dist
<dbl> <dbl>
1 4 2
2 4 10
3 7 4
4 7 22
5 8 16
6 9 10
7 10 18
8 10 26
9 10 34
10 11 17
... with more rows
>

et enfin:

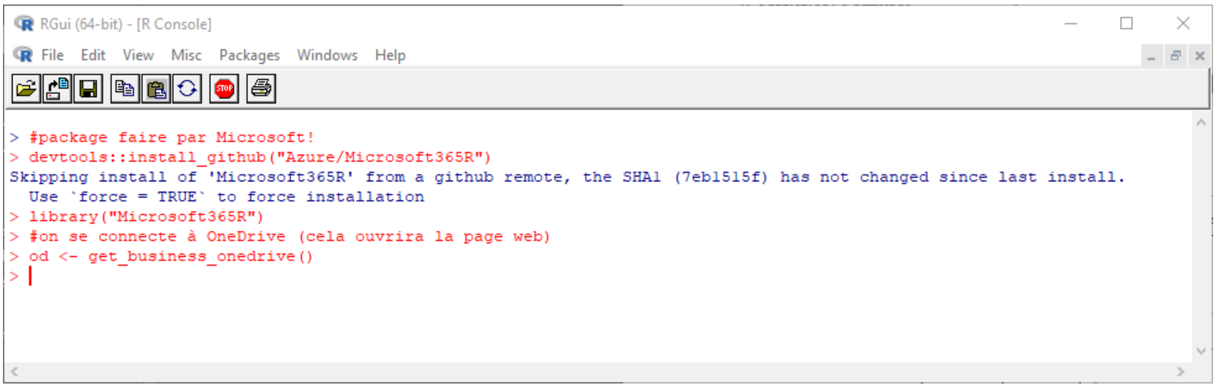
```
> #et on requête
> (myquery<-iris_db %>% select(Sepal.Width,Species))
# Source:   lazy query [?? x 2]
# Database: sqlite 3.30.1 [c:\tmp\mydatabase.sqlite3]
   Sepal.Width Species
         <dbl> <chr>
 1         3.5 setosa
 2         3   setosa
 3         3.2 setosa
 4         3.1 setosa
 5         3.6 setosa
 6         3.9 setosa
 7         3.4 setosa
 8         3.4 setosa
 9         2.9 setosa
10         3.1 setosa
# ... with more rows
> #on affiche explicitement la requête correspondante
> myquery %>% show_query()
<SQL>
SELECT `Sepal.Width`, `Species`
FROM `iris`
>
```

Exemple 66.: Importer des données de fichiers OneDrive, de bibliothèques SharePoint

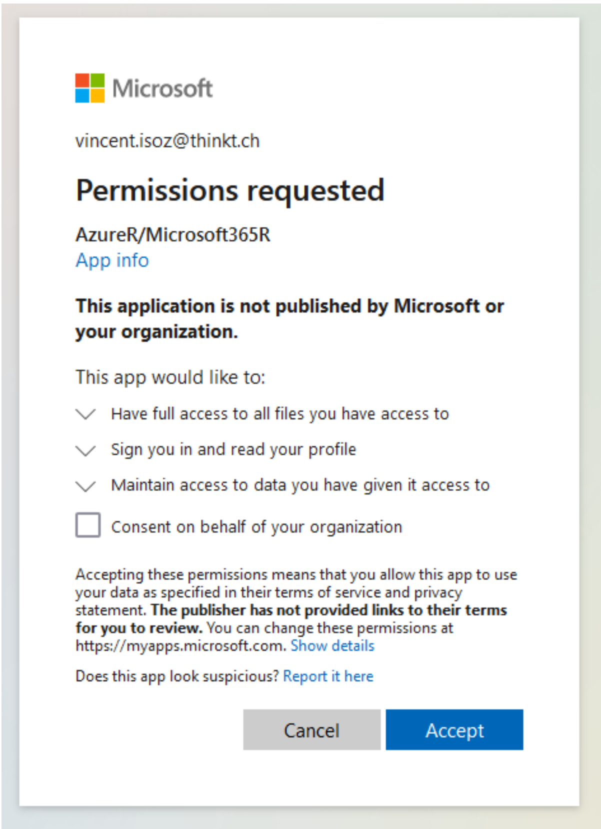
R 3.6.2

Bon je ne suis vraiment plus un fan de SharePoint et OneDrive (ayant été très déçu de l'évolution absurde du produit depuis 2010).

Donc je vais faire le minimal syndical en faisant des captures d'écran du début de la procédure



Ce qui donnera:



et ensuite pour en savoir plus le lecteur se référera au Github de Microsoft:

https://github.com/Azure/Microsoft365R

Comme par exemple:

```
> od$list_items()[1:10,]
                                    name       size isdir                                 id
1                     Approvals Document      94673  TRUE 01APZNN3LD73NJ6HQ5LFAKYAJEH4GXILXR
2                            Blocs-notes          0  TRUE 01APZNN3I2A7SXW2UN2BBJZNXE5HHNSD5D
3        Blocs-notes OneNote pour la classe  525448  TRUE 01APZNN3JK4KTWHYXMP5ALMJM43ZK3VUI5
4                               Contacts      17397  TRUE 01APZNN3LPN6VE63JFOFCYXLYJ2WPVEHPI
5                                   Docs    2095891  TRUE 01APZNN3PB72LBSV75ABAI2Y7XCXT57POC
6                        Enregistrements 2338395611  TRUE 01APZNN3KP6ZWSWWDVUNCZUHWSY4OGVIFF
7                      Exercices_OneNote  104049200  TRUE 01APZNN3NX63VRZBAHPNFZ6U3ISTFISZK7
8  Fichiers de conversation Microsoft Teams       0  TRUE 01APZNN3L5UKR5NRMZXVH2MSN4JDS4GJHD
9                                 Images    1590478  TRUE 01APZNN3KPJF4H62RNRNF2GM5HPT6GOARR
10                                JQuery        924  TRUE 01APZNN3J7EFCXBEYSEBDLZZQ46EX7ZLJZ
>
```

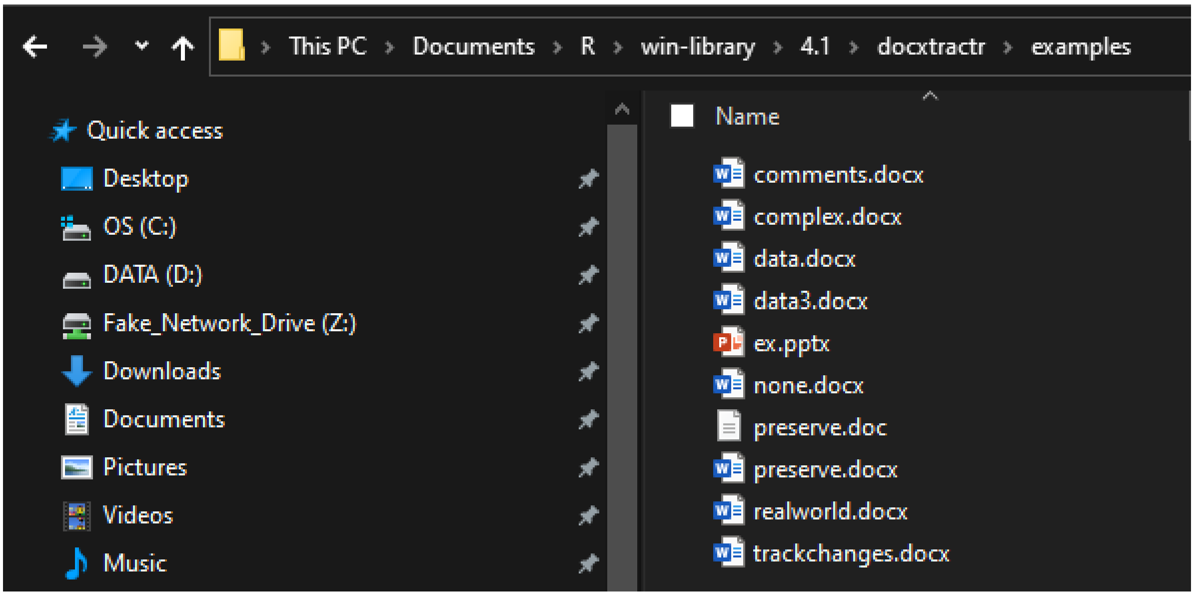
On peut se connecter à du contenu Microsoft SharePoint, Microsoft Teams et Microsoft Outlook!

Exemple 67.: Importer des données de Microsoft Word

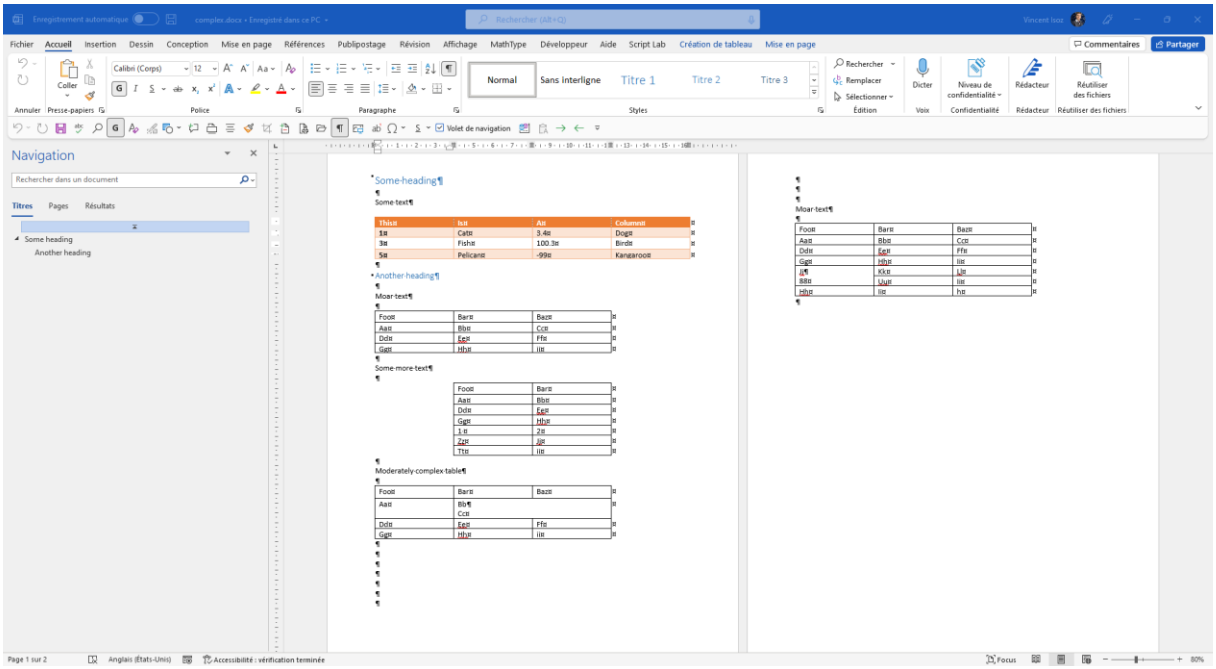
R 3.4.1

Il arrive parfois malheureusement qu'on doive même importer des tables de Microsoft Word (cas typique dans les TPE qui n'ont pas de système de gestion de bases de données professionnelle).

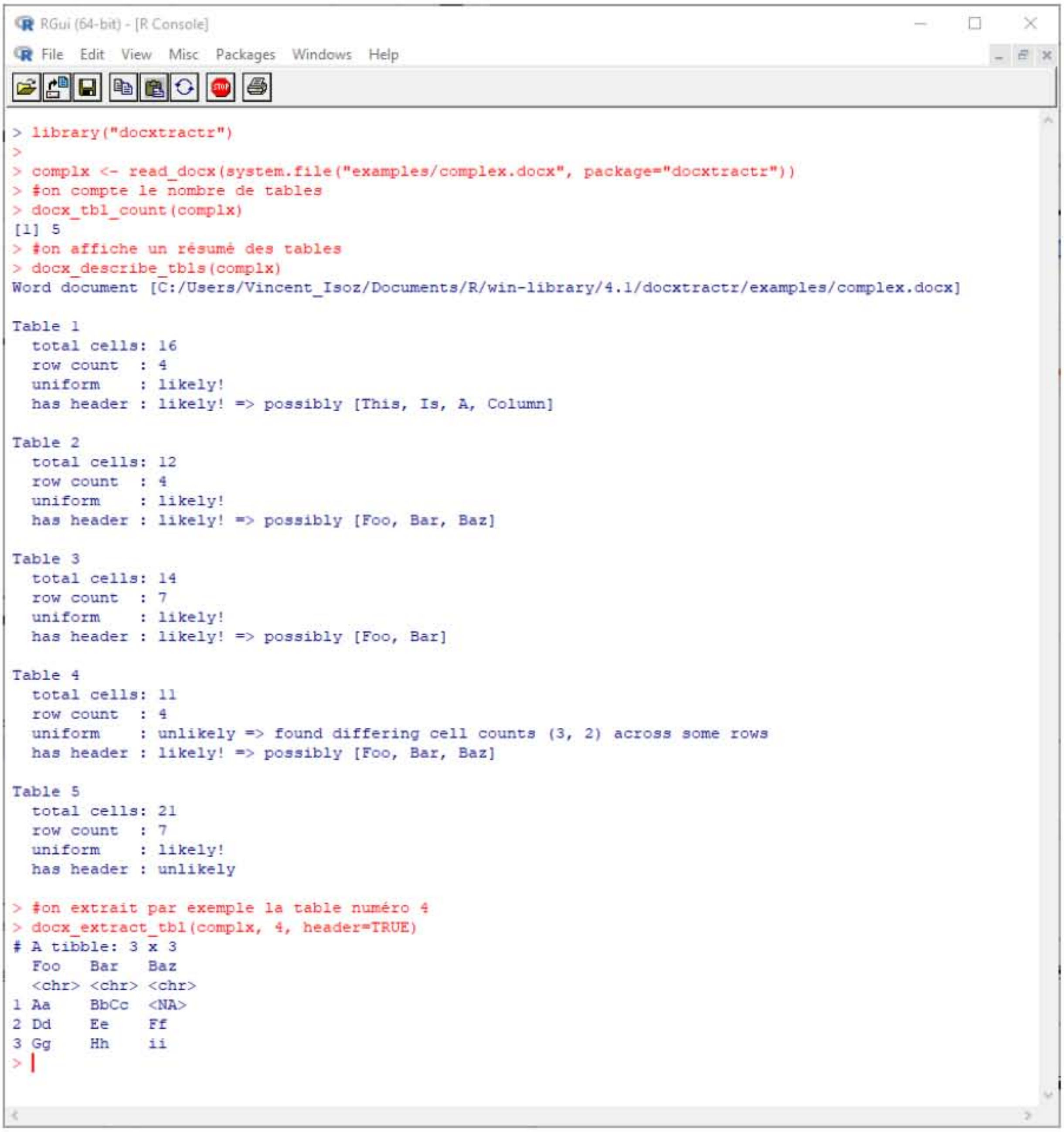
On prendra comme exemple de base un des fichiers Microsoft Word fournis dans le dossier d'installation du package **docxtractr**:



et particulièrement le fichier suivant (dans le genre pire…):



Et voyons maintenant comment utiliser les fonction **read_docx( )** et **docx_tbl_count( )** et **docx_describe_tbls( )** et **docx_extract_tbl( )** du package **docxtractr**:

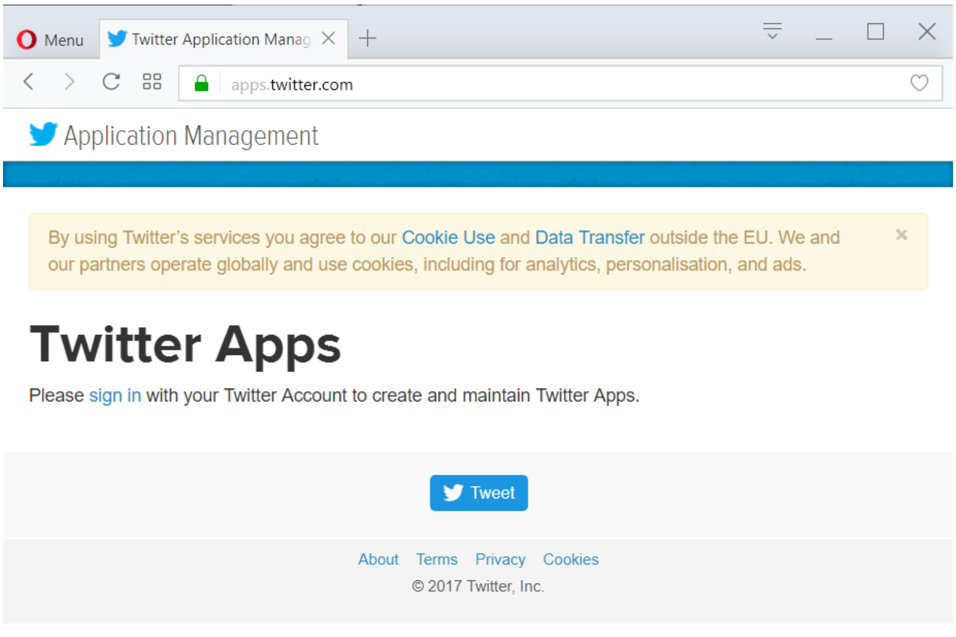

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("docxtractr")
>
> complx <- read_docx(system.file("examples/complex.docx", package="docxtractr"))
> #on compte le nombre de tables
> docx_tbl_count(complx)
[1] 5
> #on affiche un résumé des tables
> docx_describe_tbls(complx)
Word document [C:/Users/Vincent_Isoz/Documents/R/win-library/4.1/docxtractr/examples/complex.docx]
Table 1
total cells: 16
row count : 4
uniform : likely!
has header : likely! => possibly [This, Is, A, Column]
Table 2
total cells: 12
row count : 4
uniform : likely!
has header : likely! => possibly [Foo, Bar, Baz]
Table 3
total cells: 14
row count : 7
uniform : likely!
has header : likely! => possibly [Foo, Bar]
Table 4
total cells: 11
row count : 4
uniform : unlikely => found differing cell counts (3, 2) across some rows
has header : likely! => possibly [Foo, Bar, Baz]
Table 5
total cells: 21
row count : 7
uniform : likely!
has header : unlikely
> #on extrait par exemple la table numéro 4
> docx_extract_tbl(complx, 4, header=TRUE)
A tibble: 3 x 3
Foo Bar Baz
<chr> <chr> <chr>
1 Aa BbCc <NA>
2 Dd Ee Ff
3 Gg Hh ii
>

Exemple 68.: Importer des données de Twitter

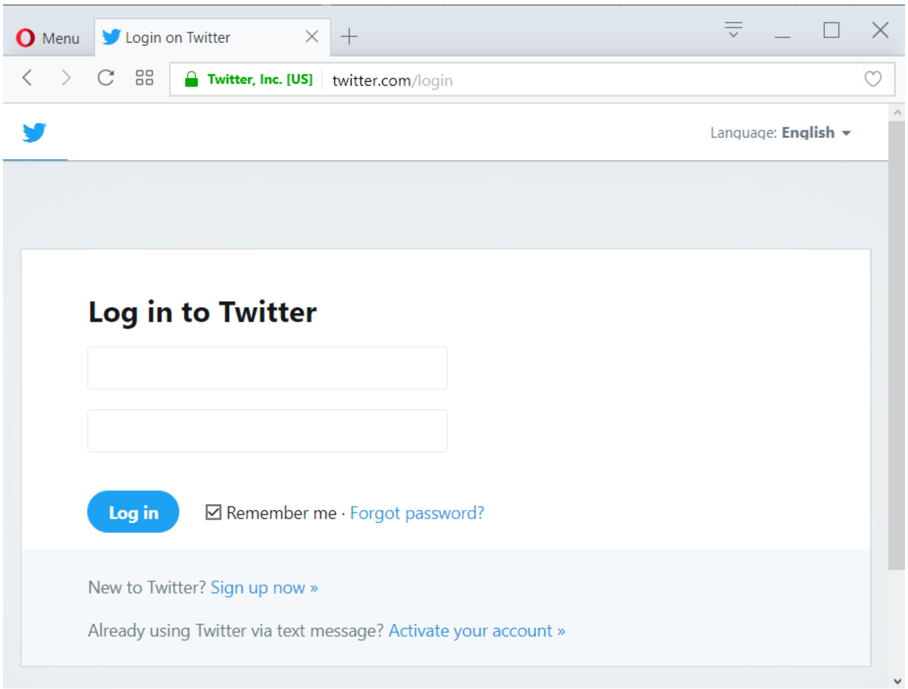
R 3.4.1

Le package que nous allons voir maintenant est loin d'être complet d'autant plus que l'équipe de Twitter change régulièrement les règles et les options de leur API, donc ne rêvez pas trop…

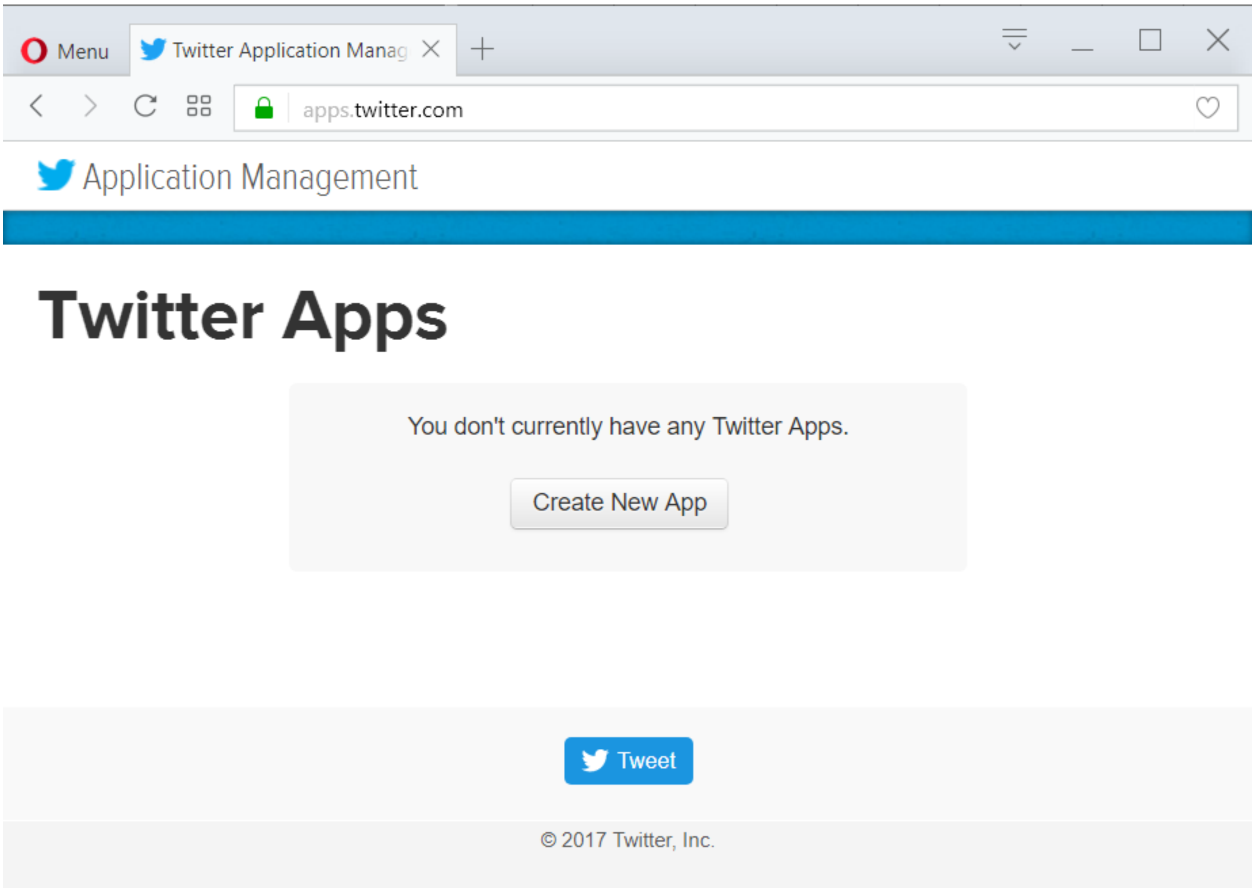
D'abord pour pouvoir utiliser le package **twitteR**, vous allez devoir vous rendre sur la page https://apps.twitter.com afin d'y obtenir des clés d'accès aux API de données:



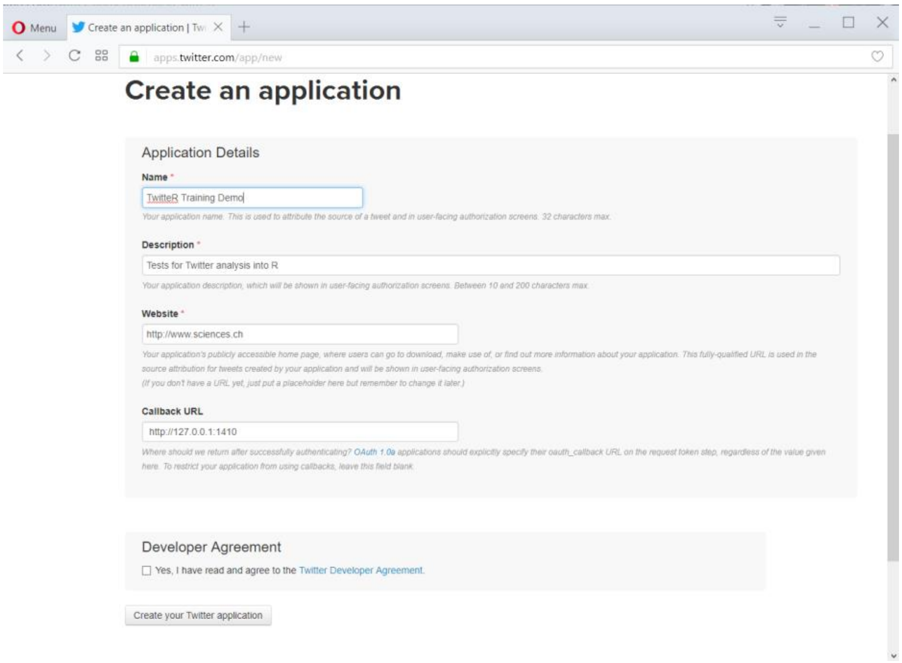
Nous cliquons sur *sign in*:



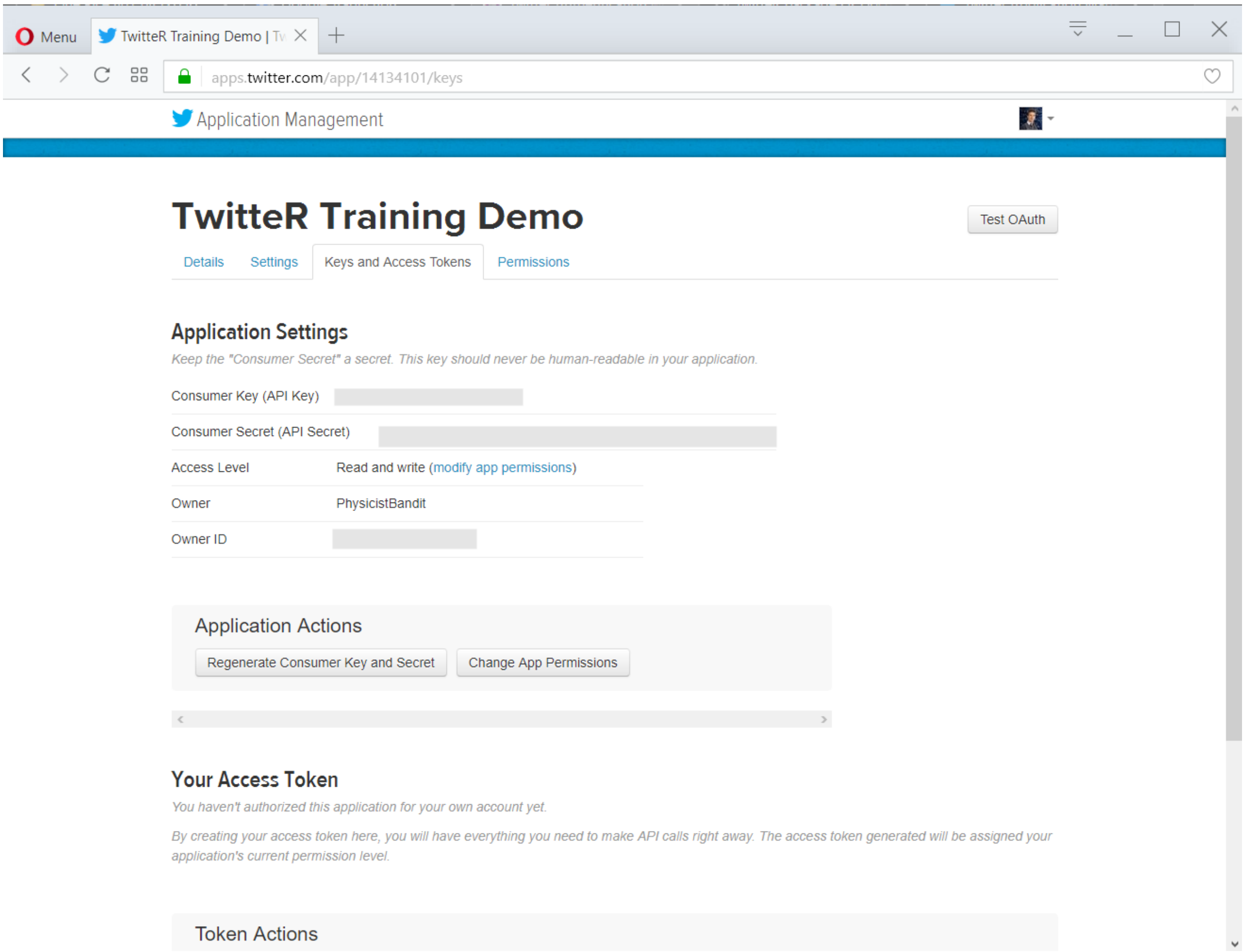
Après avoir renseigné l'e-mail et le mot de passe de votre compte Twitter vous cliquez sur *Log in* pour arriver à:



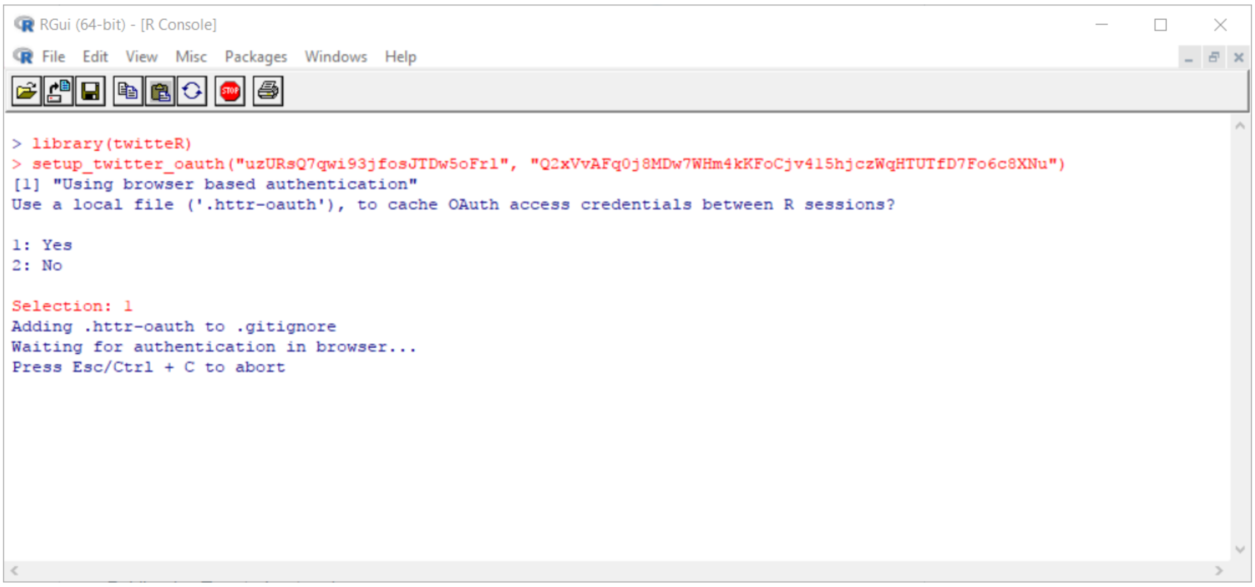
Cliquez sur *Create New App* pour obtenir:



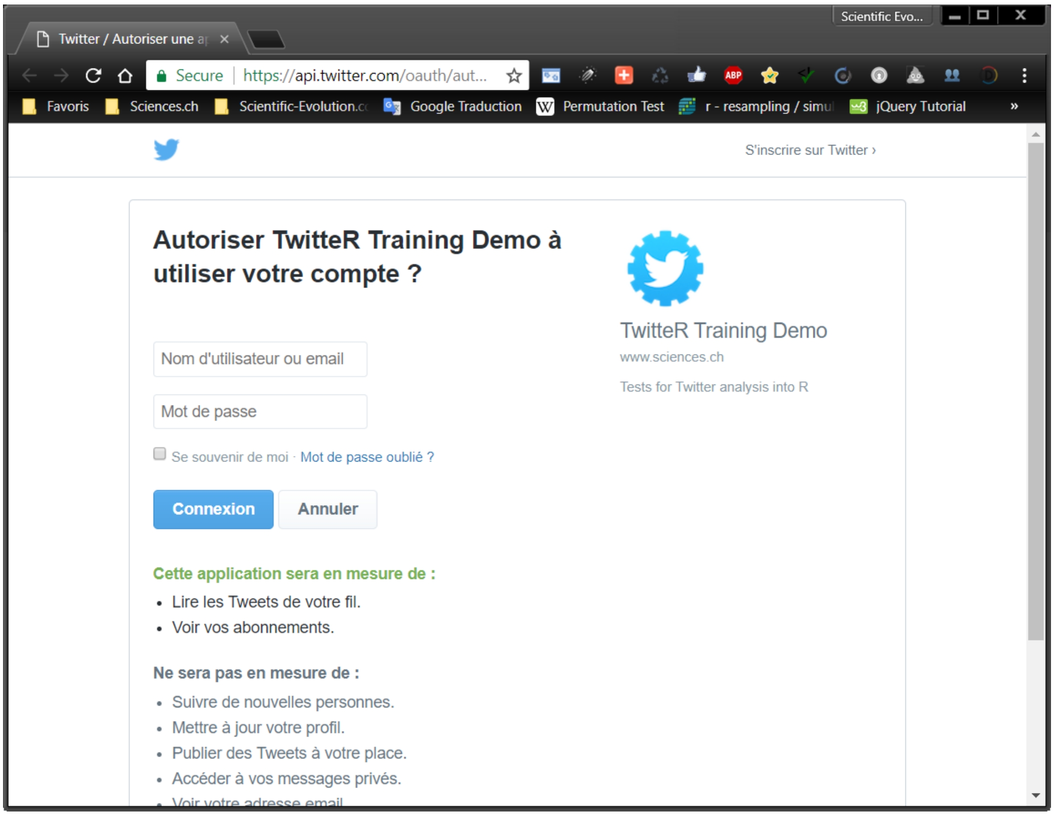
Quand vous validez par *Create your Twitter applications* vous obtiendrez la page suivante où il fadura vous rendre dans l'onglet *Keys and Access Tokens*:



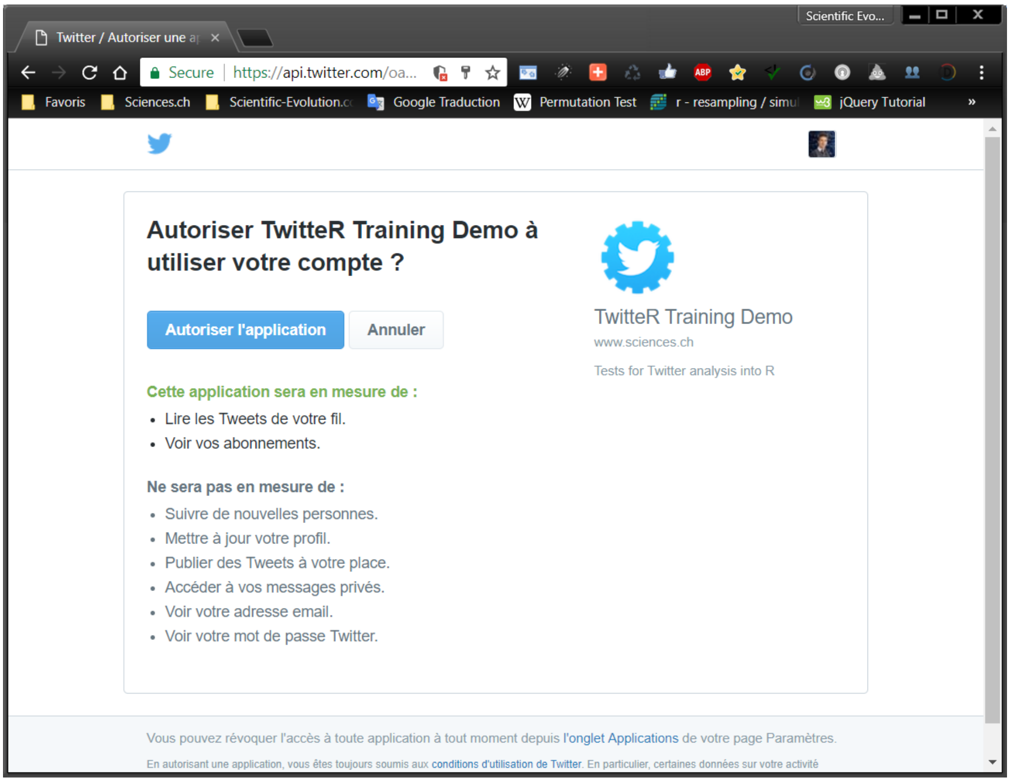
Ensuite, dans R, nous chargeons le package **twitteR** et utilisons la fonction **setup_twitter_oauth( )** pour nous connecter à l'API en utilisant respectivement la *consumer key* en premier et la *consumer secret* en second:



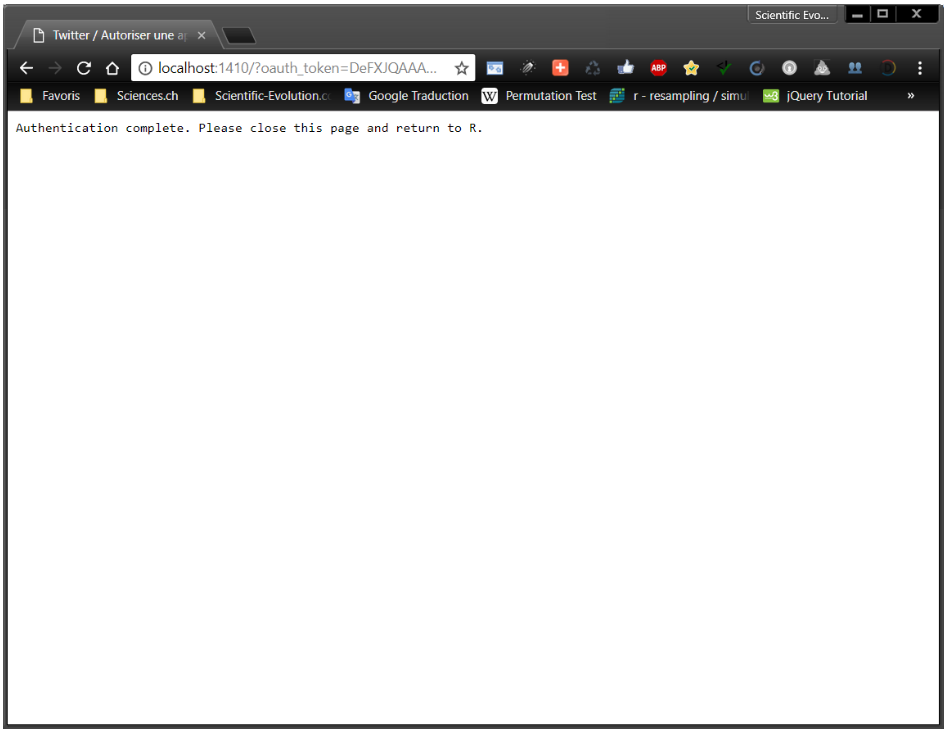
Après avoir validé en ayant tapé la valeur *1* (Yes), votre navigateur par défaut va s'ouvrir avec:



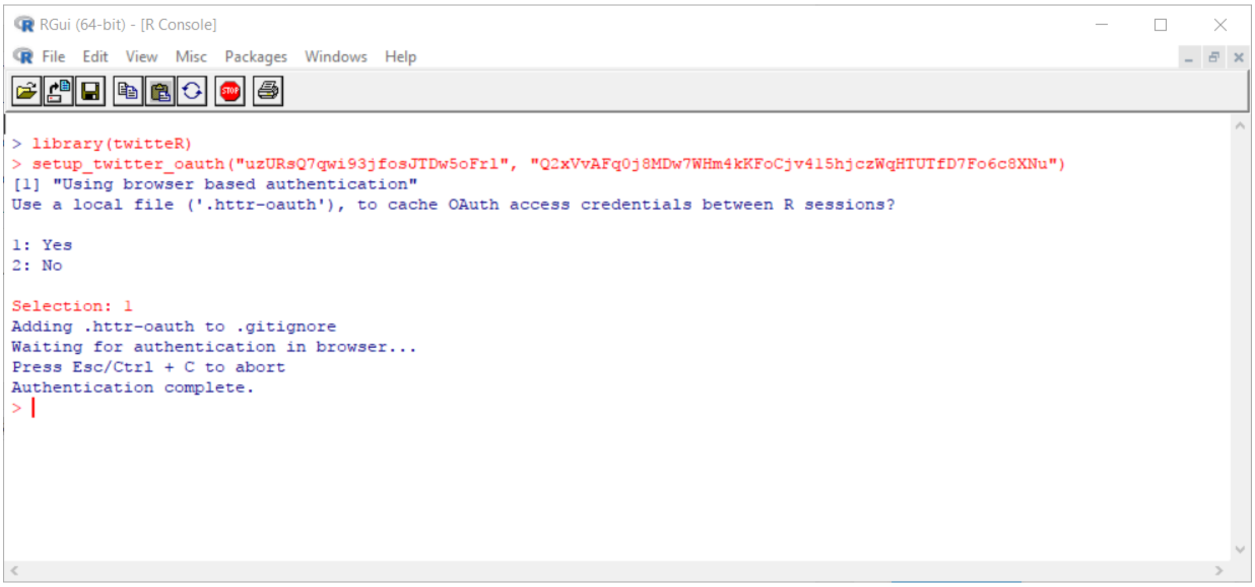
Saisissez les données de votre compte et cliquez sur *Connexion*, vous arrivez alors sur:



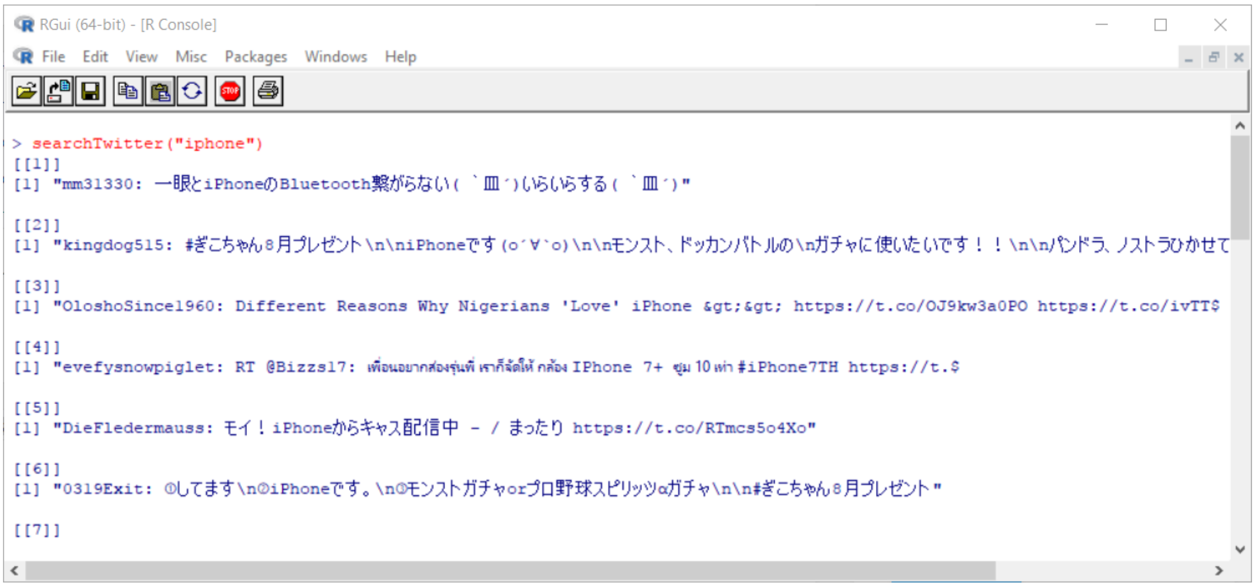
et quand nous cliquons sur *Autoriser l'application*, nous obtenons:



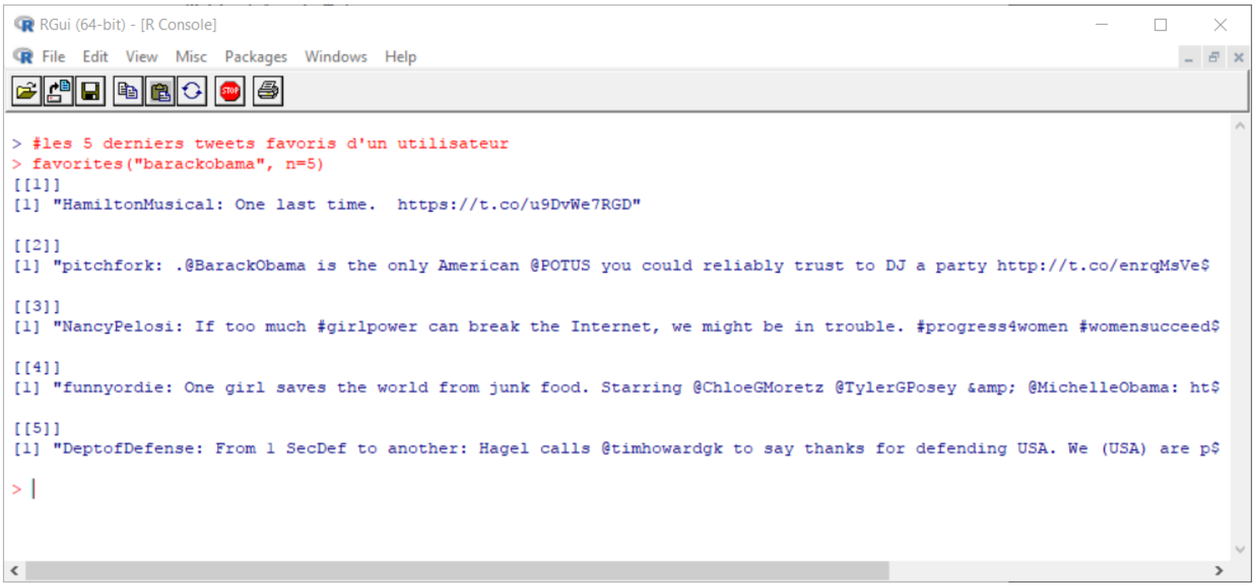
Ensuite, lorsque nous retournons dans R, nous avons:



Nous pouvons avec la fonction **searchTwitter( )** obtenir tous les tweets publics autorisés par l'API avec un certain terme:



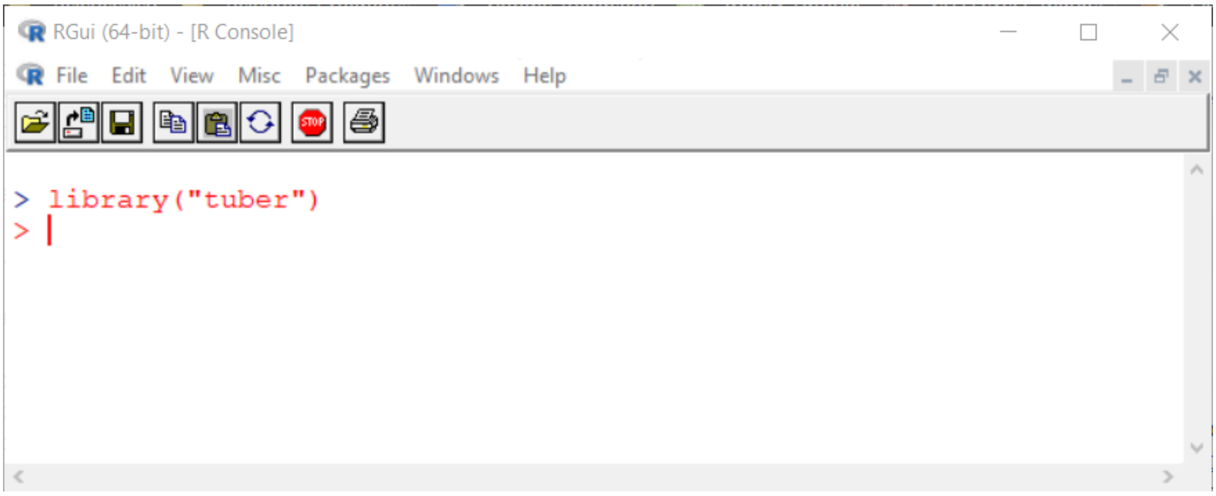
ou encore avec la fonction **favorites( )** les *n* derniers tweets favoris d'un certain utilisateur de Twitter:



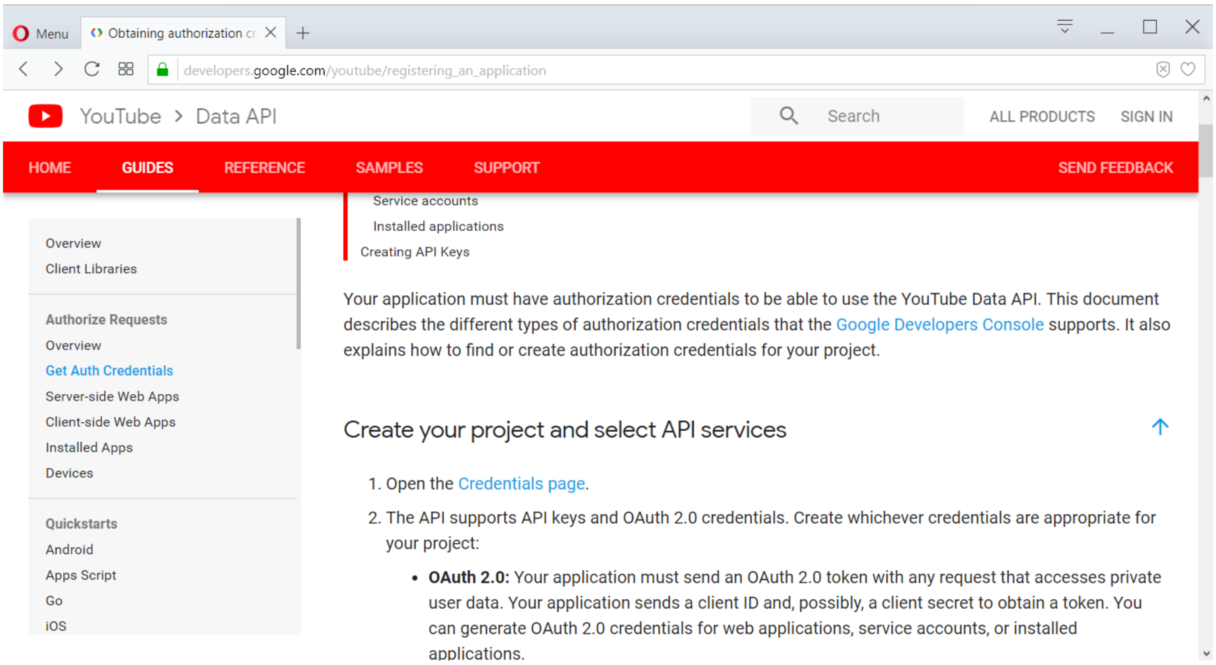
Exemple 69.: Importer des données de YouTube

R 3.5

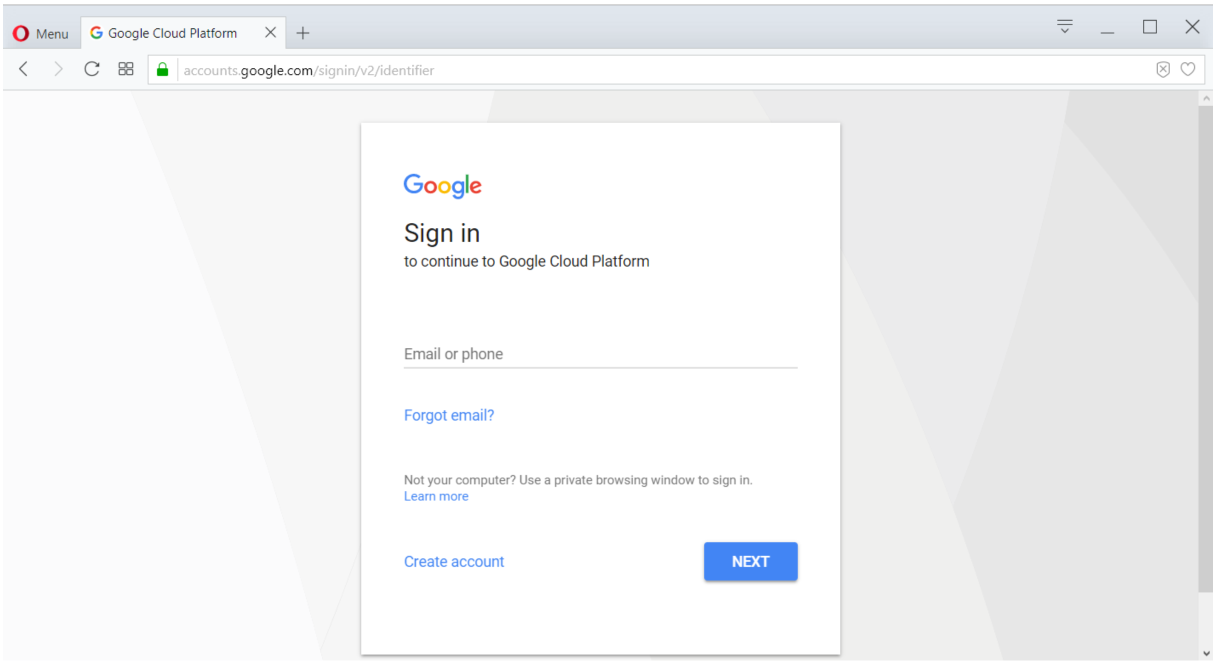
D'abord on charge le package **tuber** après l'avoir installé:



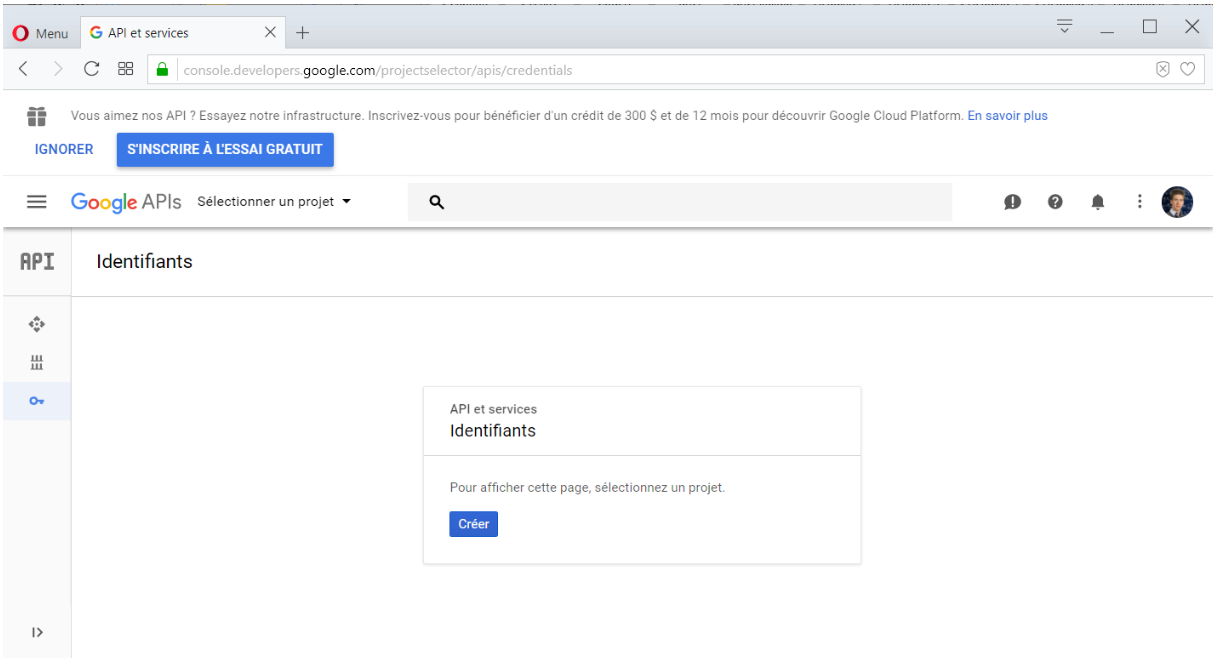
Ensuite, on va sur le site des développeurs Google:



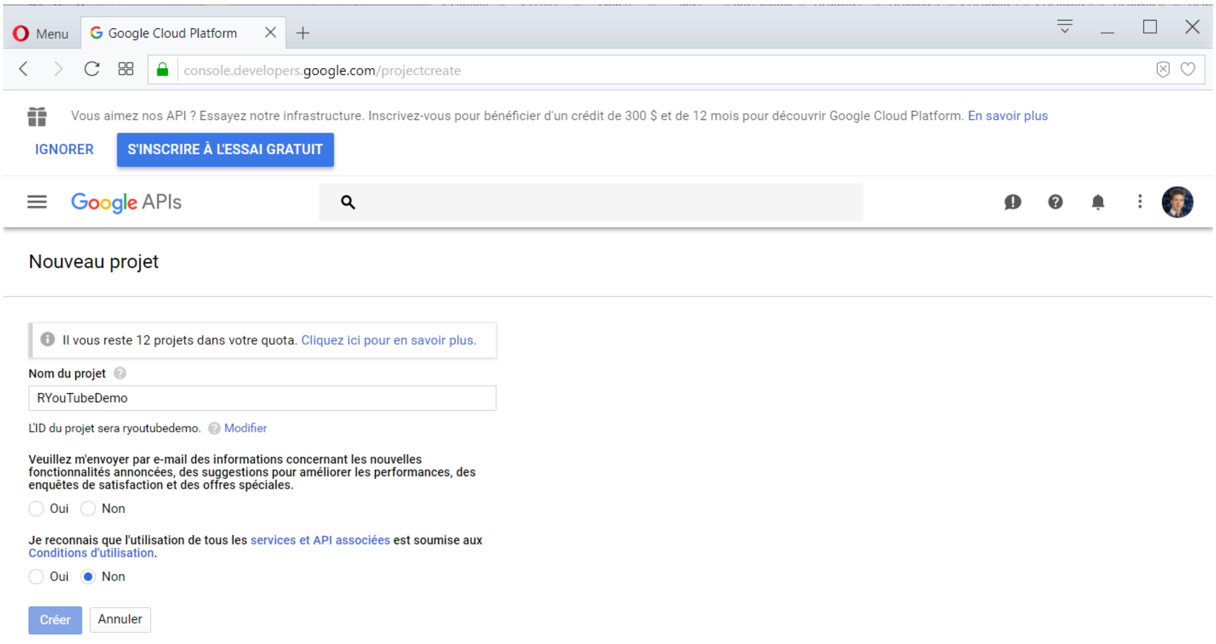
Et on clique sur *Credentiels page*:



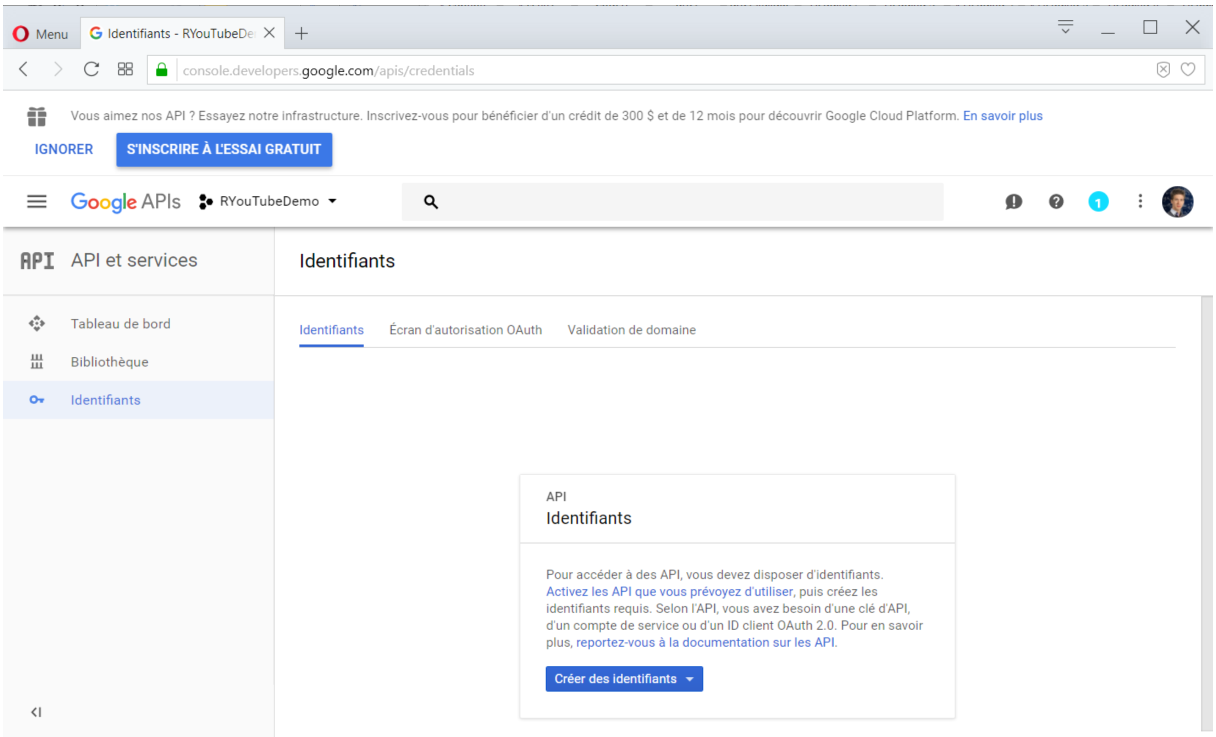
Une fois loggé, on clique sur le bouton *Créér*:



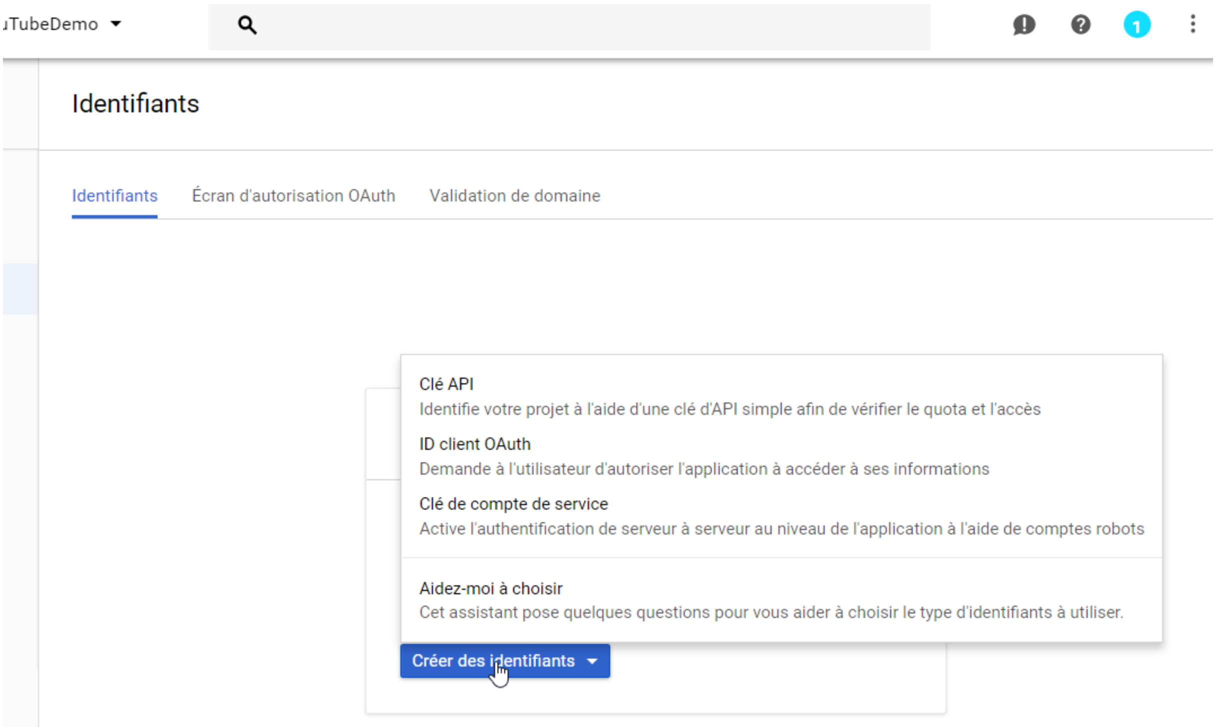
On arrive alors sur:



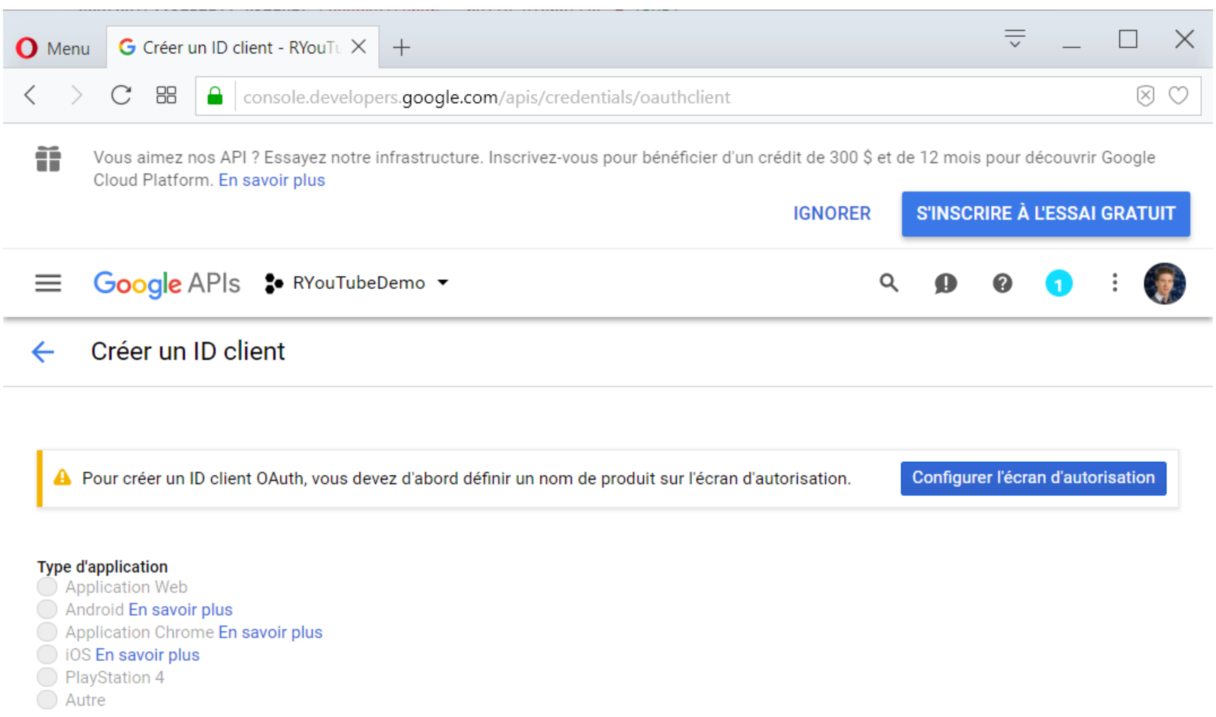
Et on valide par *Créer* encore une fois:



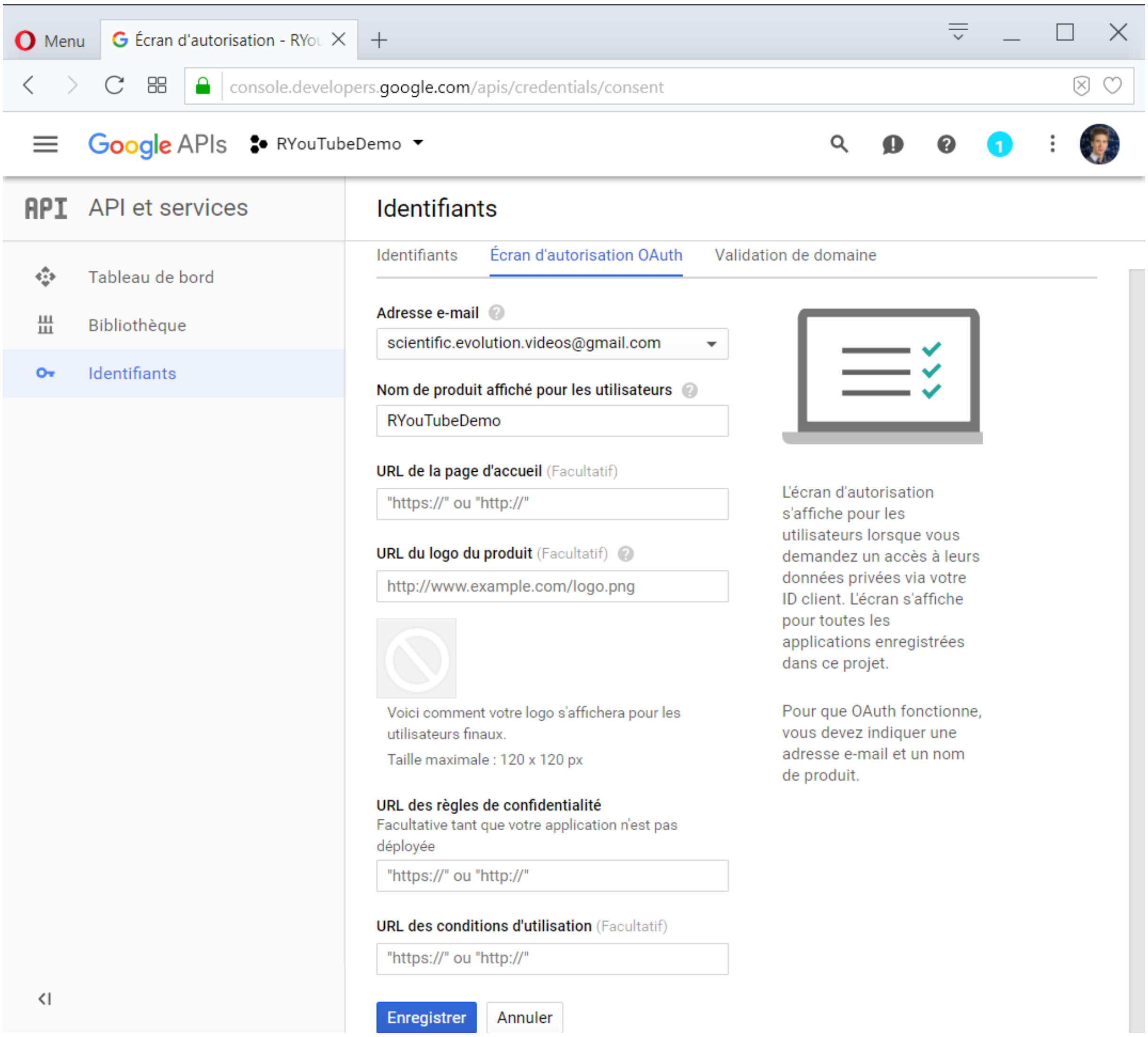
On cliquer sur Créer des identifiants… :



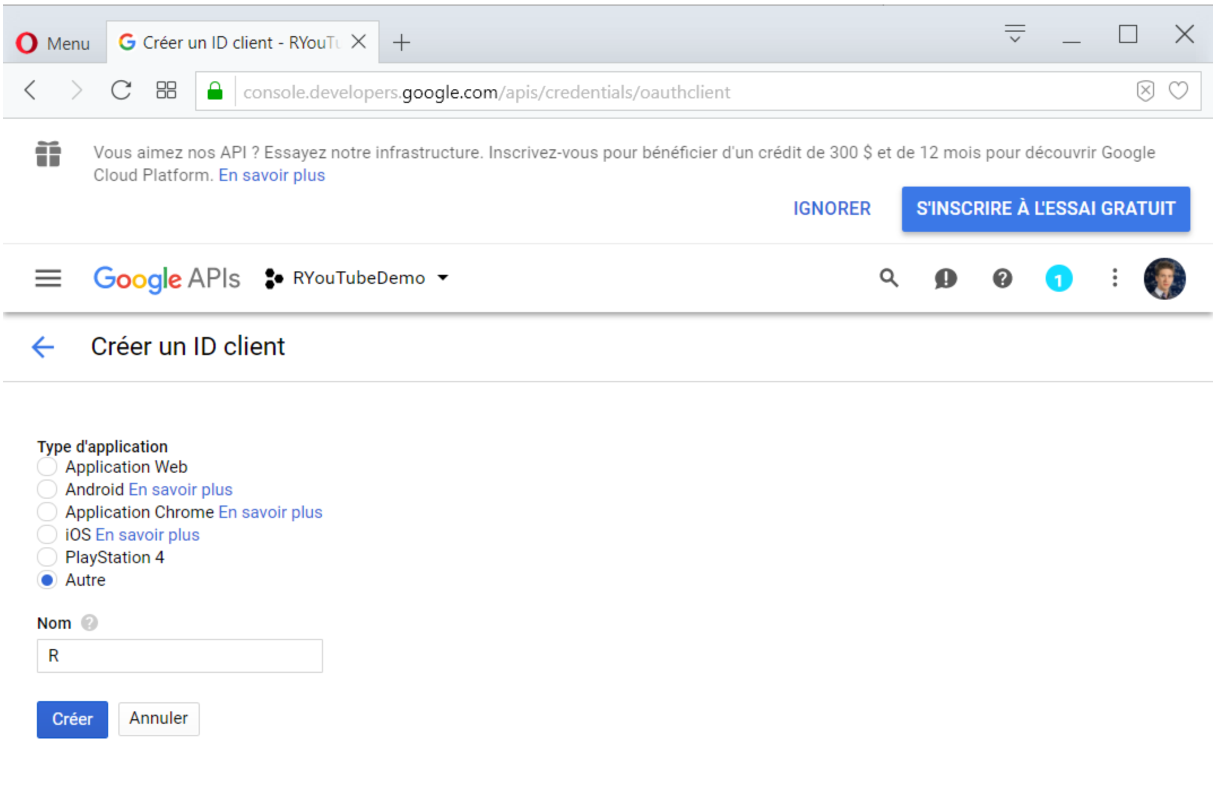
On arrive alors sur:



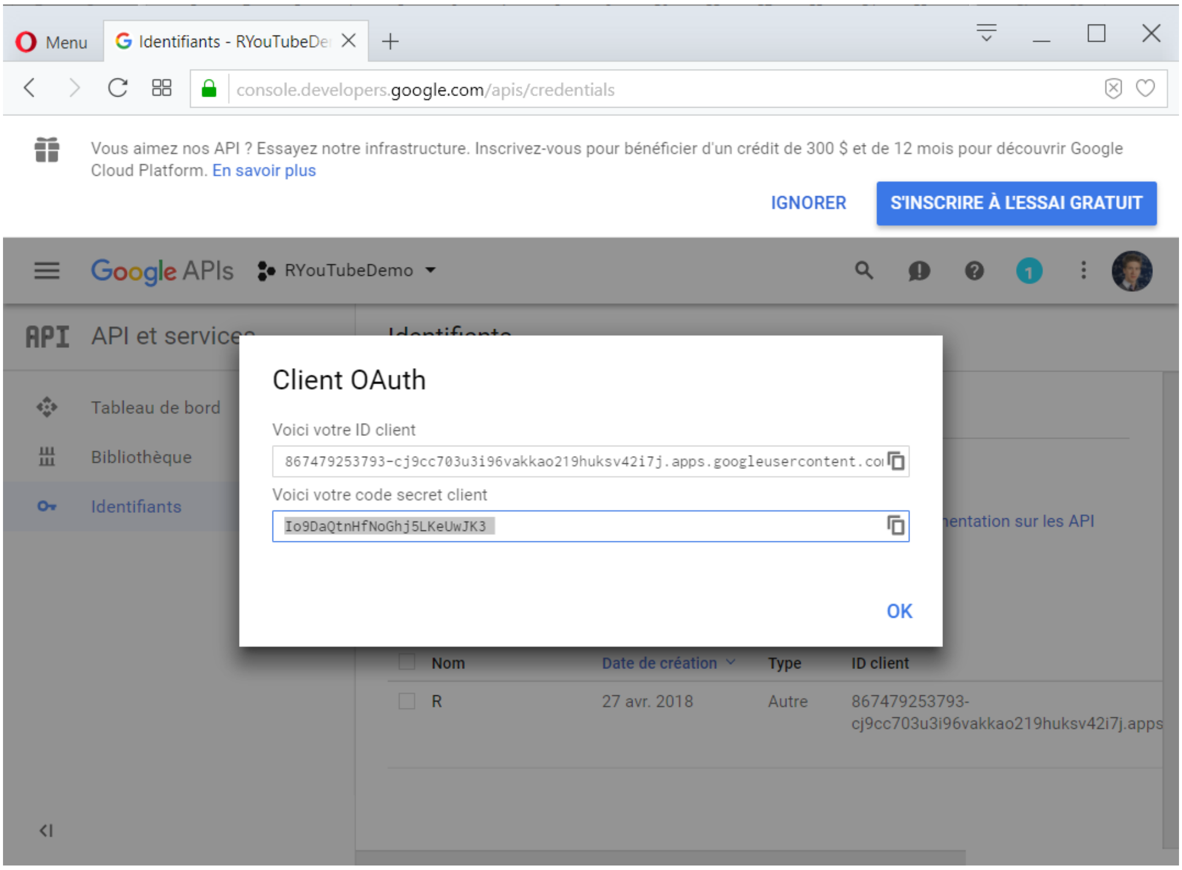
On clique donc sur Configurer l'écran d'autorisation:



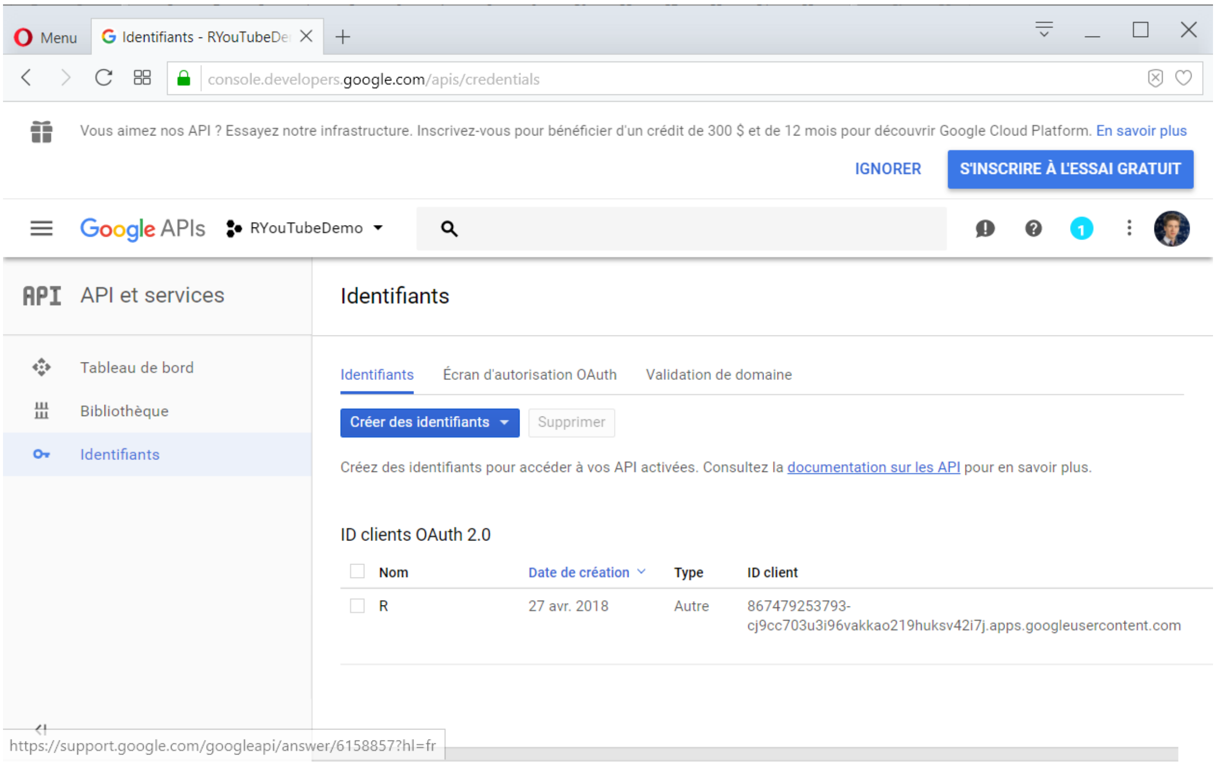
On remplit comme il faut et on valider par *Enregister*, on arrive alors sur:



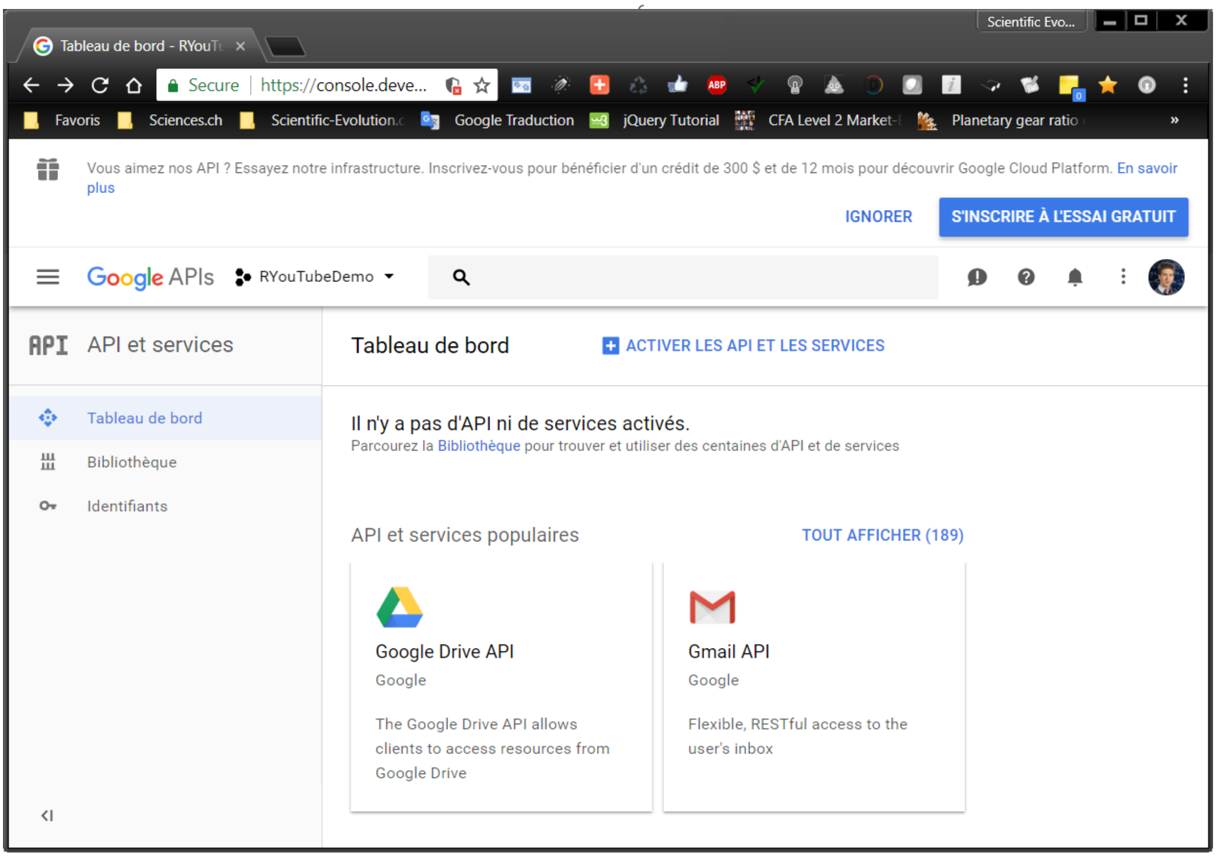
Et on valide par Créer pour obtenir:



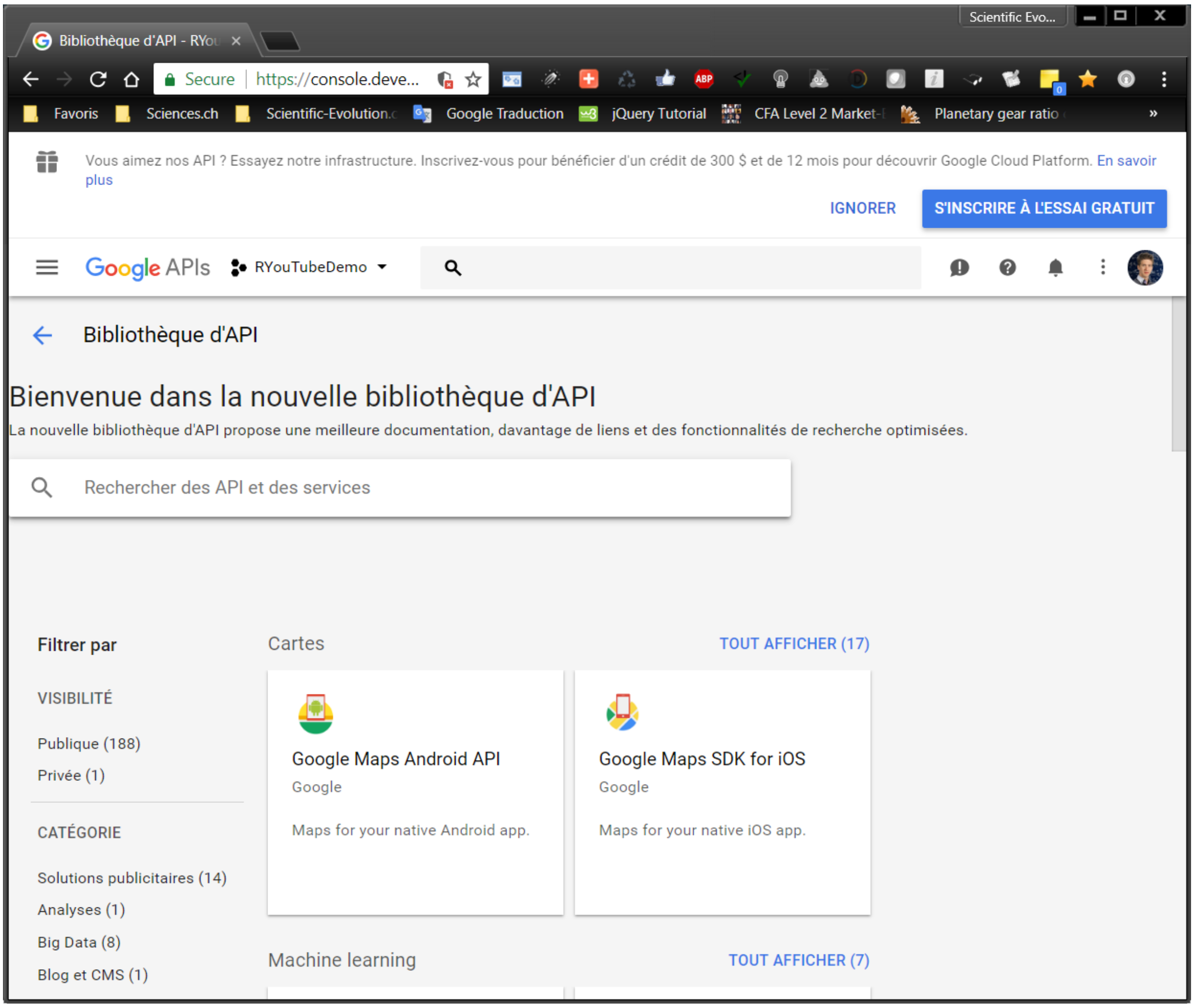
On copie le code secret client et on valide par OK:



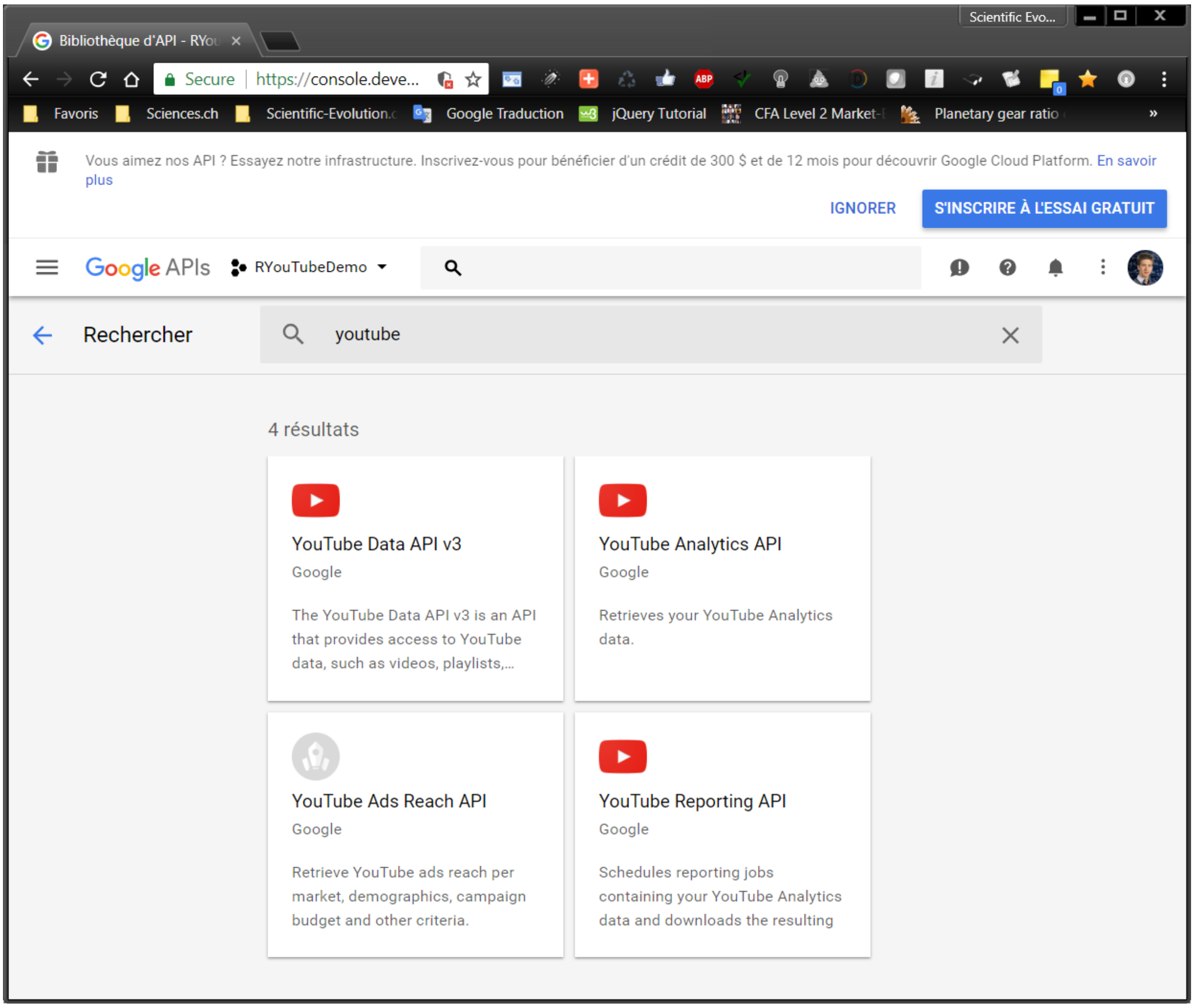
On cliquer sur Tableau de bord:



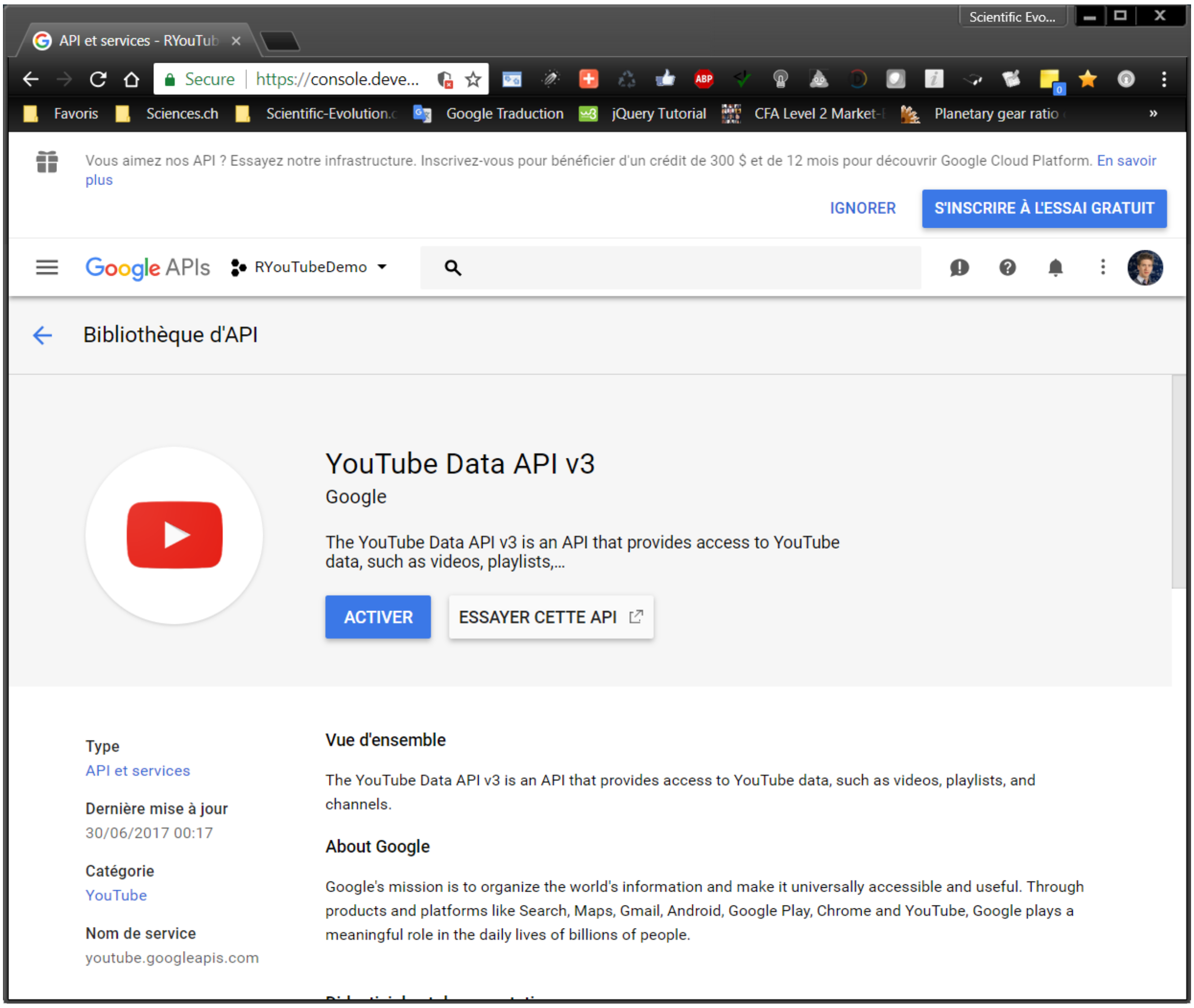
On cliquer sur TOUT AFFICHER:



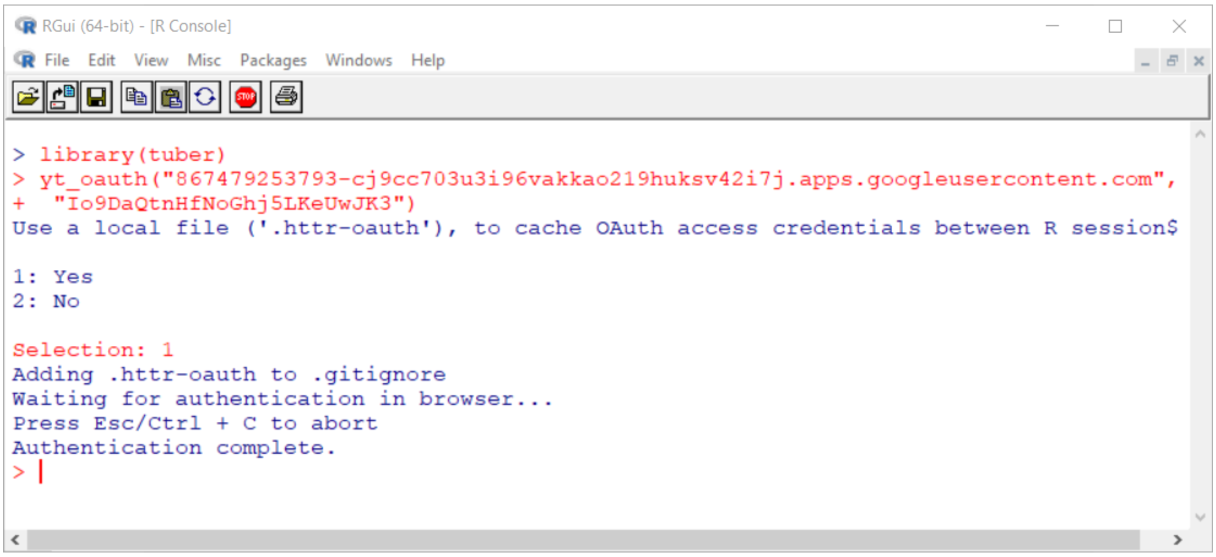
Et on fait une rechercher des API YouTube pour arriver sur:



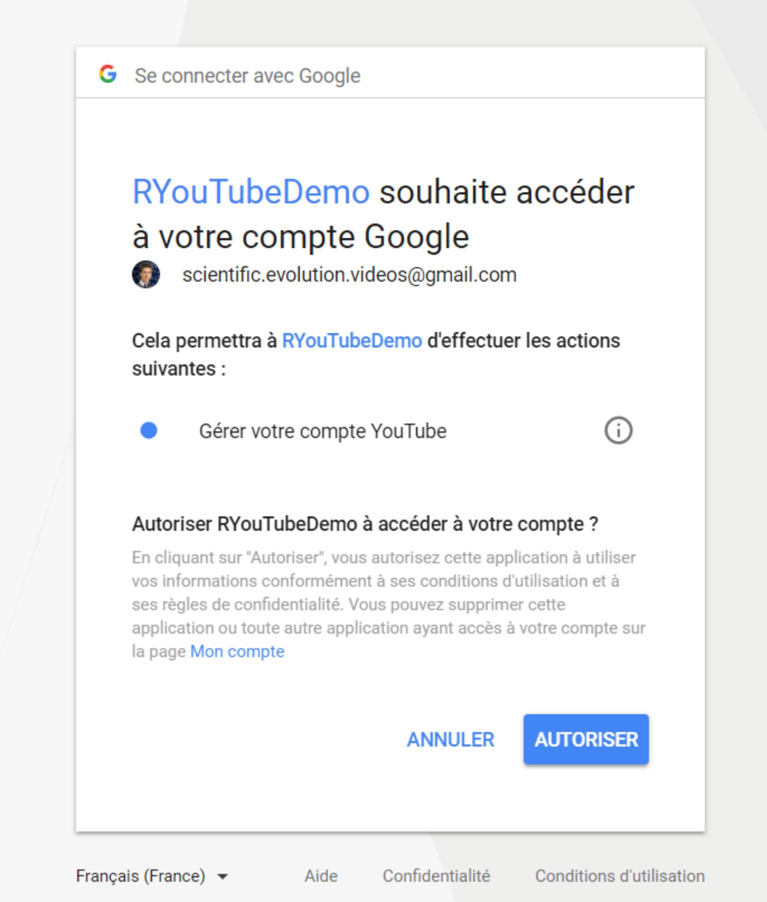
On sélectionne YouTube DATA API v3:



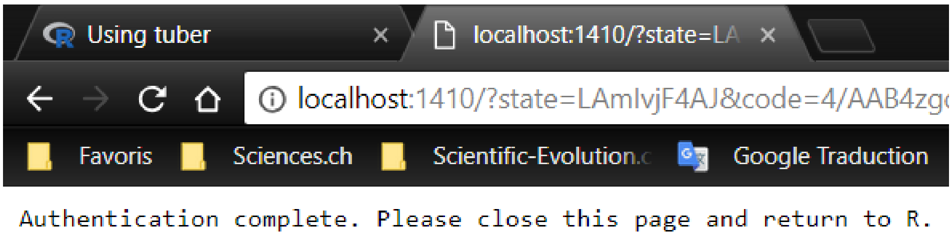
Et on clique sur ACTIVER. Une fois ceci fait on peut revenir dans R et utiliser la fonction **yt_oauth( )** pour se connecter à l'API:



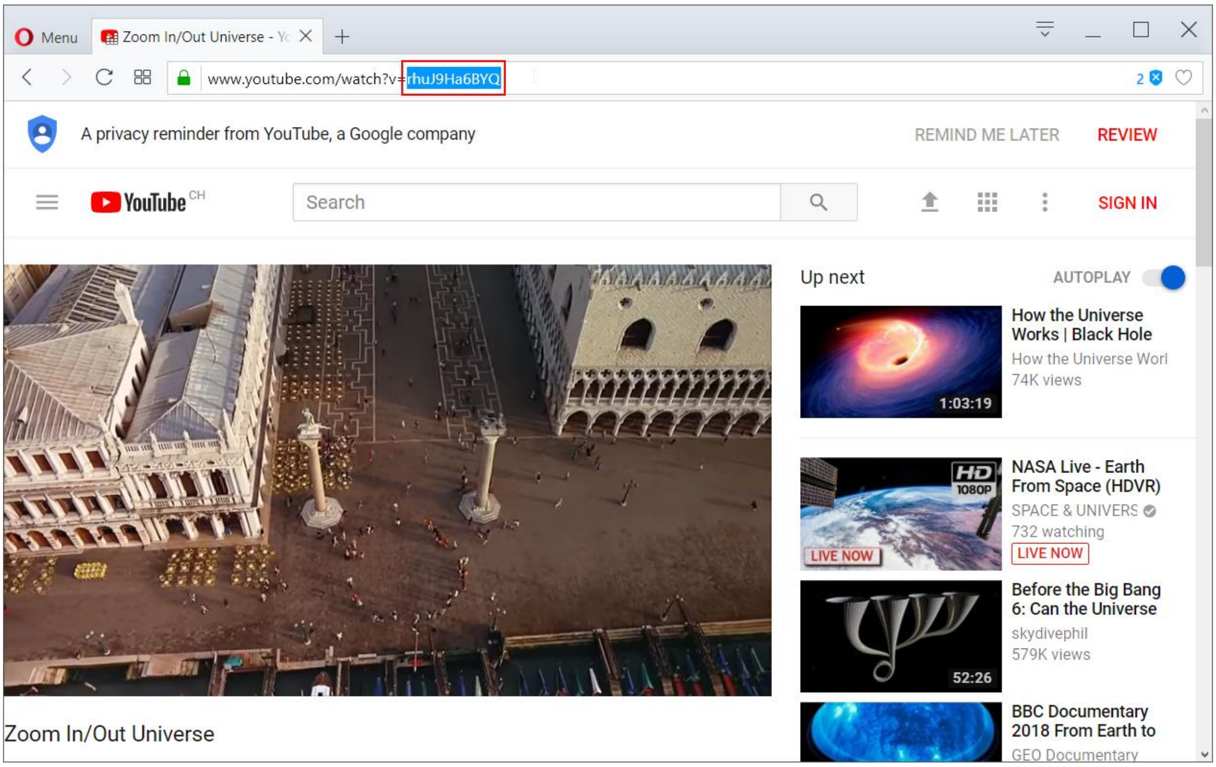
Une page va alors s'ouvrir:



On cliquer alors sur AUTORISER, nous avons alors la page suivant qui apparaît:



Ensuite, nous choisissons une vidéo et copions son code:



Et en utilisant la fonction **get_stats( )** et ce code, nous avons alors quelques statistiques:

```
> library(tuber)
> yt_oauth("867479253793-cj9cc703u3i96vakkao219huksv42i7j.apps.googleusercontent.com",
+  "Io9DaQtnHfNoGhj5LKeUwJK3")
> get_stats(video_id="rhuJ9Ha6BYQ")
$`id`
[1] "rhuJ9Ha6BYQ"

$viewCount
[1] "32277"

$likeCount
[1] "207"

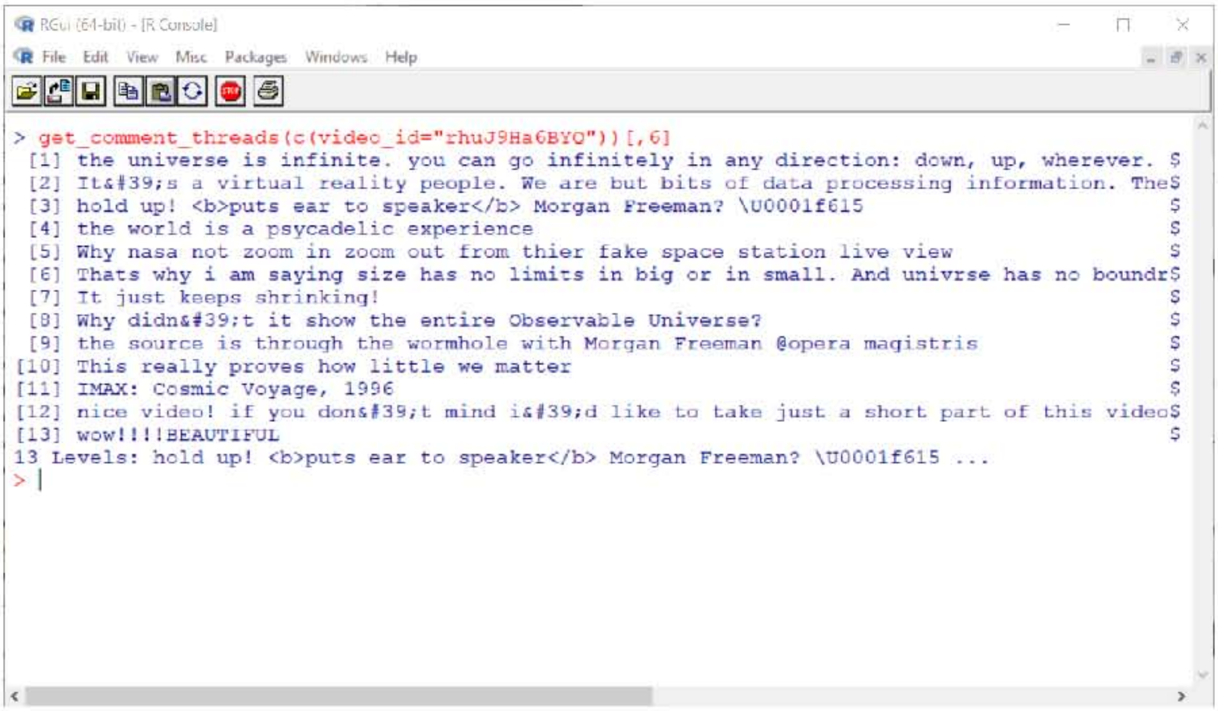
$dislikeCount
[1] "9"

$favoriteCount
[1] "0"

$commentCount
[1] "19"

>
```

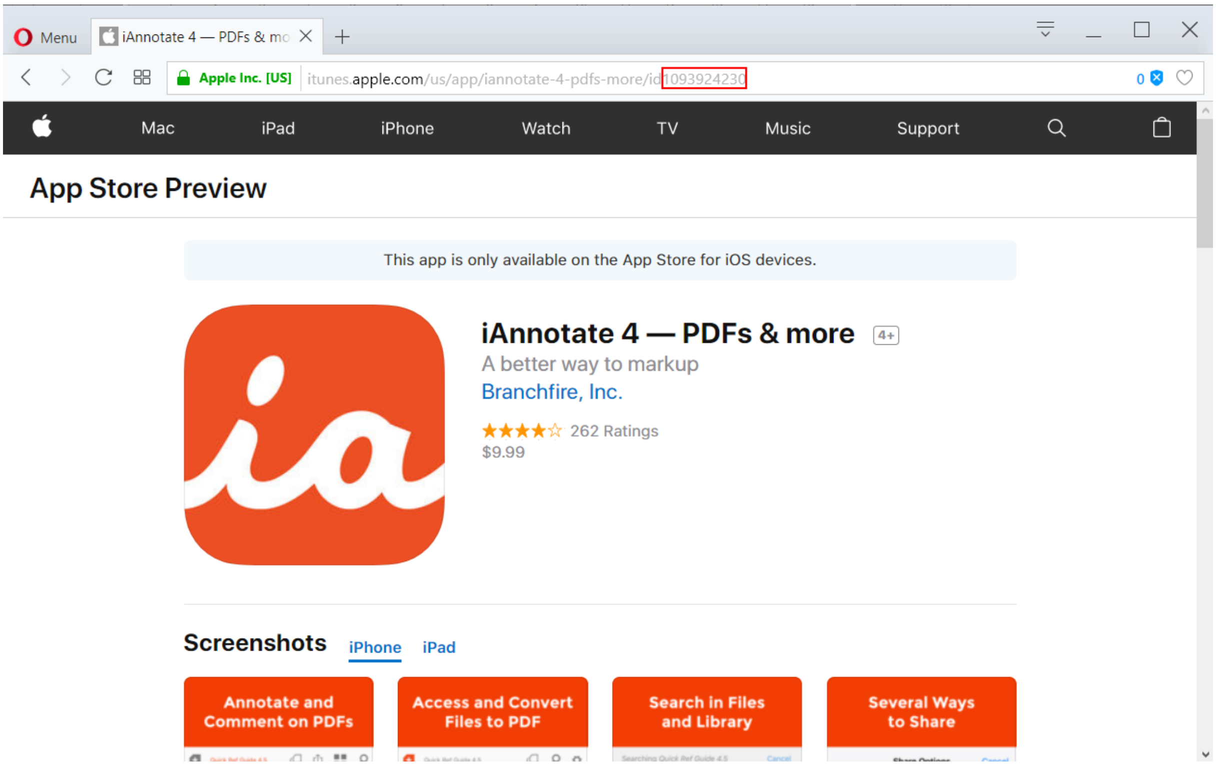
Nous pouvons aussi obtenir les commentaires de la vidéo avec la fonction **get_comment_threads( )**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> get_comment_threads(c(video_id="rhuJ9Ha6BYQ"))[,6]
[1] the universe is infinite. you can go infinitely in any direction: down, up, wherever. $
[2] It's a virtual reality people. We are but bits of data processing information. The$
[3] hold up! puts ear to speaker Morgan Freeman? \U0001f615 $
[4] the world is a psycadelic experience $
[5] Why nasa not zoom in zoom out from thier fake space station live view $
[6] Thats why i am saying size has no limits in big or in small. And univrse has no boundr$
[7] It just keeps shrinking! $
[8] Why didn't it show the entire Observable Universe? $
[9] the source is through the wormhole with Morgan Freeman @opera magistris $
[10] This really proves how little we matter $
[11] IMAX: Cosmic Voyage, 1996 $
[12] nice video! if you don't mind i'd like to take just a short part of this video$
[13] wow!!!!BEAUTIFUL $
13 Levels: hold up! puts ear to speaker Morgan Freeman? \U0001f615 ...
>

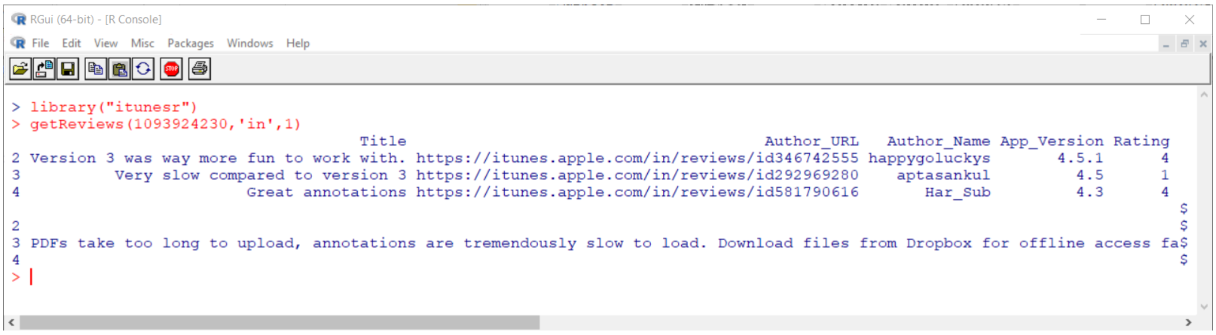
Exemple 70.: Importer des données de iTunes

R 3.5

D'abord nous choisissons un app et identifions via son URL son numéro d'identifiant:



Et nous allons dans R utiliser le package **itunesr** et sa fonction **getReviews( )** pour lequel le deuxième paramètre et le code pays d'origine des commentaires ("in" est pour India) et le "1" est pour la première page de commentaires:



```
> library("itunesr")
> getReviews(1093924230,'in',1)
                                 Title                                        Author_URL   Author_Name App_Version Rating
2 Version 3 was way more fun to work with. https://itunes.apple.com/in/reviews/id346742555 happygoluckys       4.5.1      4
3          Very slow compared to version 3 https://itunes.apple.com/in/reviews/id292969280    aptasankul         4.5      1
4                        Great annotations https://itunes.apple.com/in/reviews/id581790616       Har_Sub         4.3      4
                                                                                                                            $
2                                                                                                                           $
3 PDFs take too long to upload, annotations are tremendously slow to load. Download files from Dropbox for offline access fa$
4                                                                                                                           $
> 
```

On peut évidemment rapatrier que ce qui nous intéresse vu que l'objet de retour est heureusement un simple data frame:

```
> getReviews(1093924230,'gb',1)$Title
 [1] indispensable for me
 [2] Good app
 [3] Love the App but have a problem
 [4] Awesome
 [5] User since IPad 1
 [6] Does exactly what I need it to do
 [7] Lost some of the features that  v. 3 the best PDF annotation software
 [8] Does what it says
 [9] Great features and stable
[10] Iannotate
[11] A way to go....
[12] Step backward
[13] Solid, reliable App
[14] Step backwards
[15] Poor compatability
[16] Now better than ever
[17] Still not as good as previous version
[18] Not as i would expect
[19] Brilliant update to an already brilliant app
[20] No local files
[21] Waste of money!!!!!!!!!!!! Biggest regret ever!!!!
[22] A Backward Step
[23] Completely disagree with the negative reviews!
[24] Total waste of money
[25] DO NOT buy this
[26] Potential, but not a great start...
[27] Very disappointed without Local Saving
[28] Missing Functionality
[29] Some positives, some negatives.
[30] Disappointing
31 Levels: A Backward Step A way to go.... Awesome ... Waste of money!!!!!!!!!!!! Bigges$
>
```

ou encore:

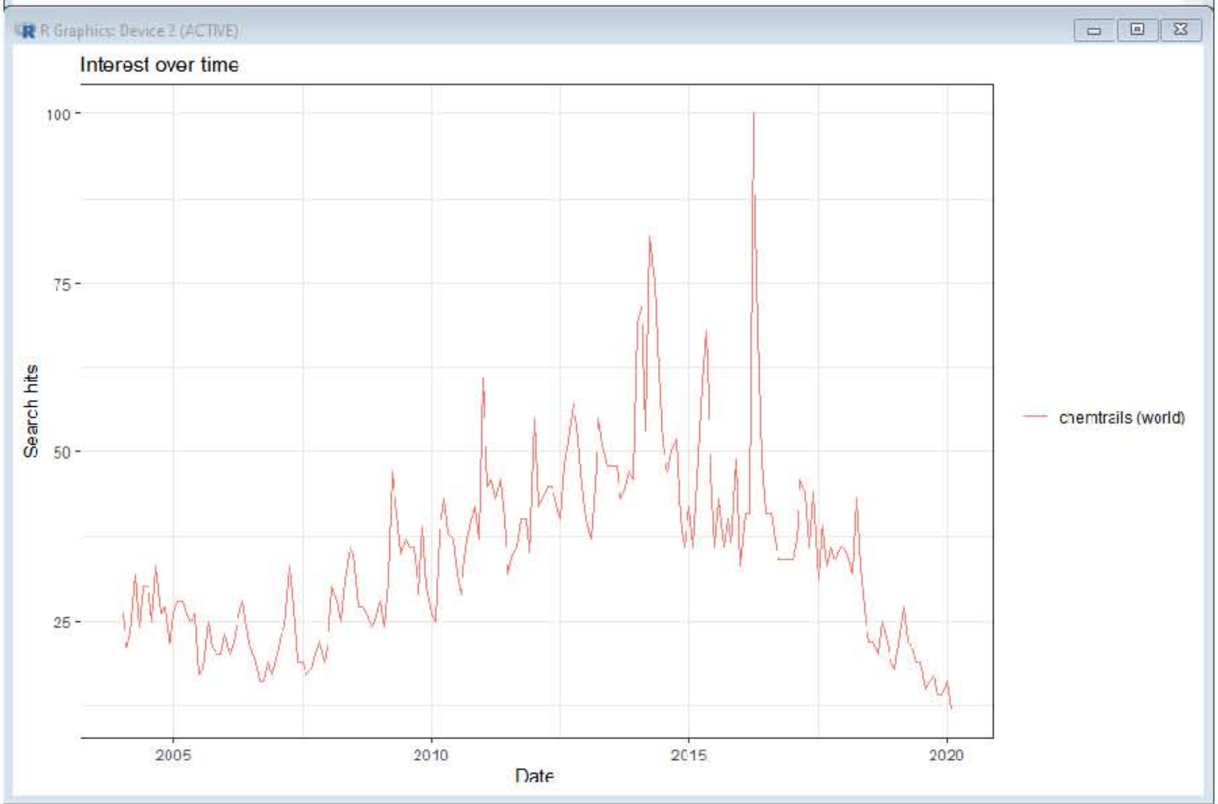
```
> getReviews(1093924230,'gb',1)$Rating
 [1] 5 5 5 5 5 5 1 5 4 4 2 1 5 2 1 5 2 1 5 1 1 1 5 1 1 1 1 1 2 2
Levels: 1 2 4 5
>
```

Exemple 71.: Importer des données de Google Trends

R 3.6

Voyons comment importer simplement des données google trends avec le package **gtrends** et sa fonction **gtrends( )**:

```
> library("gtrendsR")
>
> google_trends <- function(keyword) {
+   pres_data <- gtrends(keyword = keyword, time = "all", onlyInterest = TRUE)
+   plot(pres_data)
+   hits <- pres_data$interest_over_time$hits
+   last <- length(hits)
+   round((mean(hits[(last-10):last]) / mean(hits[1:10]) - 1) * 100) # percentage rise
+ }
>
> google_trends("chemtrails")
[1] -33
>
```



Exemple 72.: Importer des données objets 3D (*.obj)

R 3.5

Le but ici va être d'apprendre à importer des objets 3D. Particulièrement dans le cas présent, les données de topographie relevées par la sonde Rosetta de la comète 67P/Churyumov–Gerasimenko.

Pour cela nous utilisons le package **rgb** et sa fonction **readOBJ( )**, ainsi qu'au passage **shade3d( )** et **movie3d( )**:



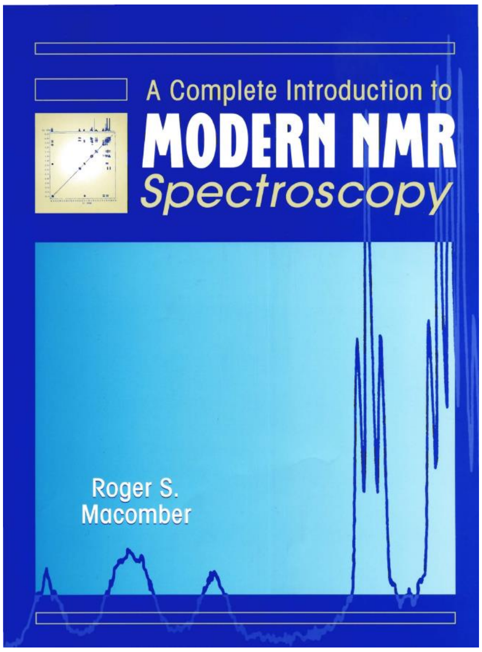
Exemple 73.: Importer des données d'un scanner NRM Bruker Advanced III 500Mz

R 3.6.3

Nous allons importer des données d'un NMR du type suivant:



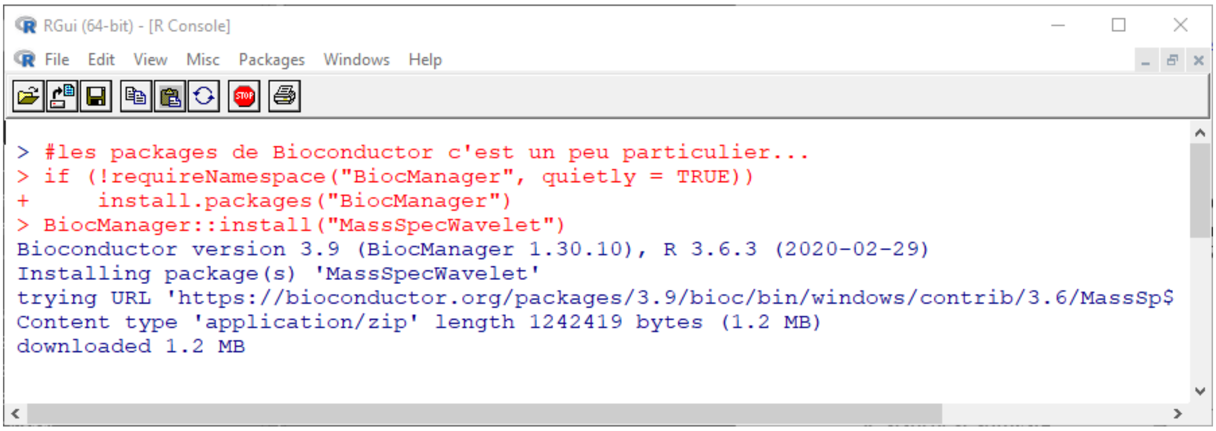
Pour ceux intéressés aux principe de fonctionnement physique de ce type de NMR, nous conseillons la lecture suivante:



Ce qui va suivre est essentiellement tiré de:

http://www.socr.umich.edu/people/dinov/courses/DSPA_notes/18_BigLongitudinalDataAnalysis_RNN.html

D'abord installons les packages:



Pour illustrer les possibilités de Rnmr1D 1.0, nous utiliserons l'ensemble de données fourni dans le package. Il s'agit d'un très petit ensemble de spectres RMN 1H[9] (6 échantillons) acquis sur un instrument Bruker Advanced III 500Mz (séquence ZG, solvant D20, pH 6), dérivé de feuilles de plantes. La conception expérimentale de l'étude s'est concentrée sur un traitement (stress vs contrôle) avec 3 répétitions pour chaque échantillon (non publié).

Nous allons donc lire les fichiers fournis dans le package **Rnmr1D** avec les fonctions natives de R pour ensuite les processer avec la fonction **doProcessing( )**:

[9] Hydrogène

```
> library("Rnmr1D")
> data_dir <- system.file("extra", package = "Rnmr1D")
> cmdfile <- file.path(data_dir, "NP_macro_cmd.txt")
> str(cmdfile)
 chr "C:/Users/Vincent Isoz/Documents/R/win-library/3.6/Rnmr1D/extra/NP_macro_cmd.txt"
> samplefile <- file.path(data_dir, "Samples.txt")
> str(samplefile)
 chr "C:/Users/Vincent Isoz/Documents/R/win-library/3.6/Rnmr1D/extra/Samples.txt"
> out <- Rnmr1D::doProcessing(data_dir, cmdfile=cmdfile,samplefile=samplefile, ncpu=detectCores())
Rnmr1D:  --- READING and CONVERTING ---
Rnmr1D:  Vendor=bruker, Type=fid, LB=0.3, GB=0, ZF=2, BLPHC=FALSE, PHC1=TRUE, FP=0, TSP=TRUE
Rnmr1D:  Generate the 'samples' & 'factors' files from the list of raw spectra
Rnmr1D:  -- Nb Spectra = 6 -- Nb Cores = 4

Rnmr1D:  Generate the final matrix of spectra...
Rnmr1D: ------------------------------------
Rnmr1D: Process the Macro-commands file
Rnmr1D: ------------------------------------
Rnmr1D:
Rnmr1D:  Baseline Correction: PPM Range = ( 4.966 , 9.348 )
Rnmr1D:     Type=airPLS, lambda= 3
Rnmr1D:  Baseline Correction: PPM Range = ( 0.396 , 4.712 )
Rnmr1D:     Type=airPLS, lambda= 4
Rnmr1D:  Baseline Correction: PPM Range = ( 0.621 , 1.522 )
Rnmr1D:     Type=airPLS, lambda= 5
Rnmr1D:  Baseline Correction: PPM Range = ( 1.227 , 1.353 )
Rnmr1D:     Type=airPLS, lambda= 6
Rnmr1D:  Normalisation of the Intensities based on the selected PPM ranges...
Rnmr1D:     Method =CSN
Rnmr1D:  Zeroing the selected PPM ranges ...
Rnmr1D:     Zone 1 = ( 4.683 , 5.015 )
Rnmr1D:  Alignment: PPM Range = ( 5.024 , 9.611 )
Rnmr1D:     CluPA - Resolution =0.01 - SNR threshold=5 - Reference=0
Rnmr1D:     --- Peak detection : nDivRange = 64
Rnmr1D:     --- Peak detection time:  9.39000000000033  sec
Rnmr1D:     --- The reference spectrum is:  3
Rnmr1D:     --- Spectra alignment to the reference: maxShift = 16
Rnmr1D:     --- Spectra alignment time:  1.59999999999991  sec
Rnmr1D:  Alignment: PPM Range = ( 2.652 , 2.742 )
Rnmr1D:     Rel. Shift Max.=0.05 - Reference=0
Rnmr1D:  Alignment: PPM Range = ( 2.651 , 2.677 )
Rnmr1D:     Rel. Shift Max.=0.05 - Reference=0
Rnmr1D:  Alignment: PPM Range = ( 0.642 , 4.699 )
Rnmr1D:     CluPA - Resolution =0.01 - SNR threshold=5 - Reference=0
Rnmr1D:     --- Peak detection : nDivRange = 64
Rnmr1D:     --- Peak detection time:  8.05999999999995  sec
Rnmr1D:     --- The reference spectrum is:  2
Rnmr1D:     --- Spectra alignment to the reference: maxShift = 16
Rnmr1D:     --- Spectra alignment time:  1.71000000000004  sec
Rnmr1D:  Bucketing the selected PPM ranges ...
Rnmr1D:     UNIF - Resolution =0.01 - SNR threshold=3 - Append=0
```

Voyons un peu le contenu des deux fichiers procéssés:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

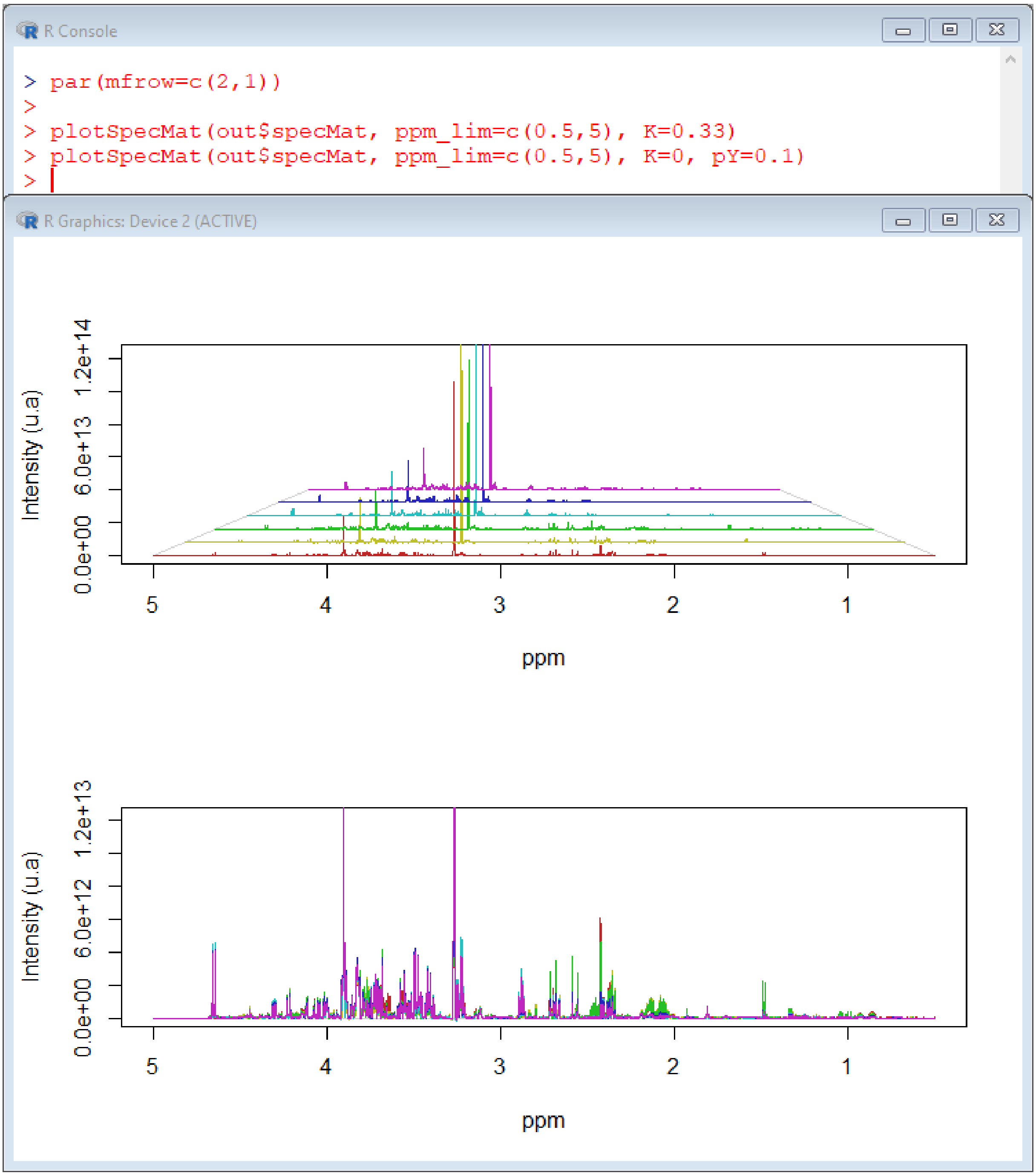
> cmdtext <- readLines(cmdfile )
> cmdtext [grep("^#$", cmdtext , invert=TRUE)]
 [1] "#%% Vendor=bruker; Type=fid; LB=0.3; GB=0; ZF=2; BLPHC=FALSE; PHC1=TRUE; FP=0; TSP=TRUE"
 [2] ""
 [3] "# Baseline Correction: PPM Range = (  4.966  ,  9.348  )"
 [4] "airpls 4.966 9.348 3 "
 [5] ""
 [6] "# Baseline Correction: PPM Range = (  0.396  ,  4.712  )"
 [7] "airpls 0.396 4.712 4 "
 [8] ""
 [9] "# Baseline Correction: PPM Range = (  0.621  ,  1.522  )"
[10] "airpls 0.621 1.522 5 "
[11] ""
[12] "# Baseline Correction: PPM Range = (  1.227  ,  1.353  )"
[13] "airpls 1.227 1.353 6"
[14] ""
[15] "# Normalisation ( CSN ) of the Intensities based on the selected PPM ranges..."
[16] "normalisation CSN"
[17] "0.90 1.005"
[18] "5.024 9.202"
[19] "1.451 4.696"
[20] "EOL"
[21] ""
[22] "# Zeroing the selected zones ..."
[23] "zero"
[24] "4.683 5.015"
[25] "EOL"
[26] ""
[27] "# Alignment of the selected zones ( 5.024 , 9.611 )"
[28] "clupa 10.2 10.5 5.024 9.611 0.01 5 0"
[29] ""
[30] "# Alignment of the selected zones ( 2.652 , 2.742 )"
[31] "align 2.652 2.742 0.05 0"
[32] ""
[33] "# Alignment of the selected zones ( 2.651 , 2.677 )"
[34] "align 2.651 2.677 0.05 0"
[35] ""
[36] "# Alignment of the selected zones ( 0.642 , 4.699 )"
[37] "clupa 10.2 10.5 0.642 4.699 0.01 5 0"
[38] ""
[39] "# Bucketing - UNIFORM"
[40] "bucket unif 10.2 10.5 0.01 3 0"
[41] "4.696 0.98"
[42] "9.389 5.008"
[43] "EOL"
>
```

Et:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> read.table(samplefile , sep="\t", header=T,stringsAsFactors=FALSE)
          Spectrum Samplecode EXPNO PROCNO Treatment
1 CD_BBI_16P02-R1          R1    10      1   control
2 CD_BBI_16P02-R2          R2    10      1   control
3 CD_BBI_16P02-R3          R3    10      1   control
4 CD_BBI_16P02-R7          R7    10      1    stress
5 CD_BBI_16P02-R8          R8    10      1    stress
6 CD_BBI_16P02-R9          R9    10      1    stress
>
```

Nous pouvons maintenant fair quelques graphes:



Bref pour plus d'informations et autres exemples très intéressants, allez voir le lien déjà mentionné plus haut:

https://cran.r-project.org/web/packages/Rnmr1D/vignettes/Rnmr1D.html

Exemple 74.: Web scraping de pages HTML avec formulaires (RSelenium)

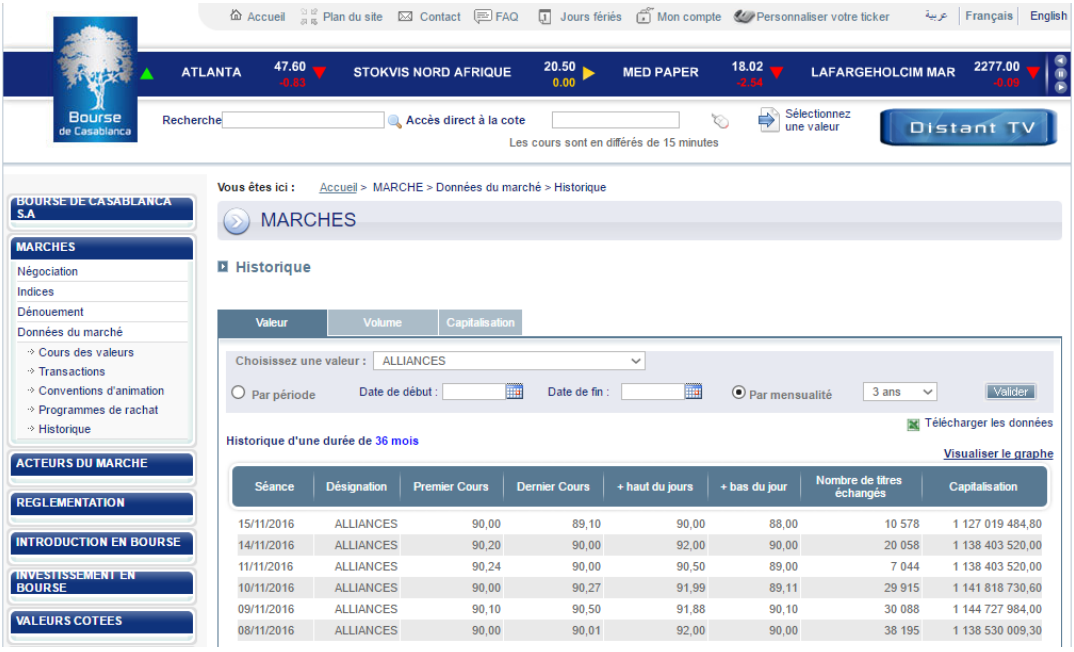
R 4.1.1

Inutile de préciser ô combien le Webscrapping est important. Par contre il est peut-être bon de rappeler qu'au niveau légal… vaut mieux faire les choses de manière juridiquement correctes si on ne veut pas de problèmes dans certaines pays…

Pour les exemples qui vont suivre, nous allons jouer avec le site internet suivant:

http://www.casablanca-bourse.com/bourseweb/Negociation-Historique.aspx?Cat=24\&IdLink=302

Qui ressemble à:



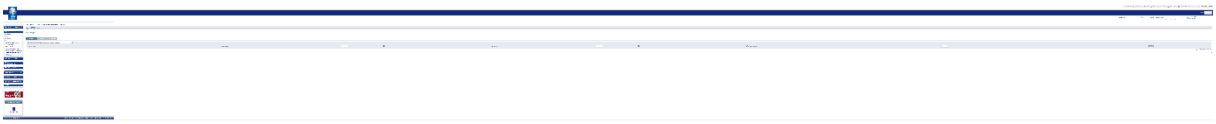
D'abord nous vérifions que le site est "up" avec le package **httr** et sa fonction **GET( )**:

```
> library("httr")
> reponse <- GET("http://www.casablanca-bourse.com")
> #la réponse '200' dans le monde du web veut dire "OK"
> reponse$status_code
[1] 200
>
```

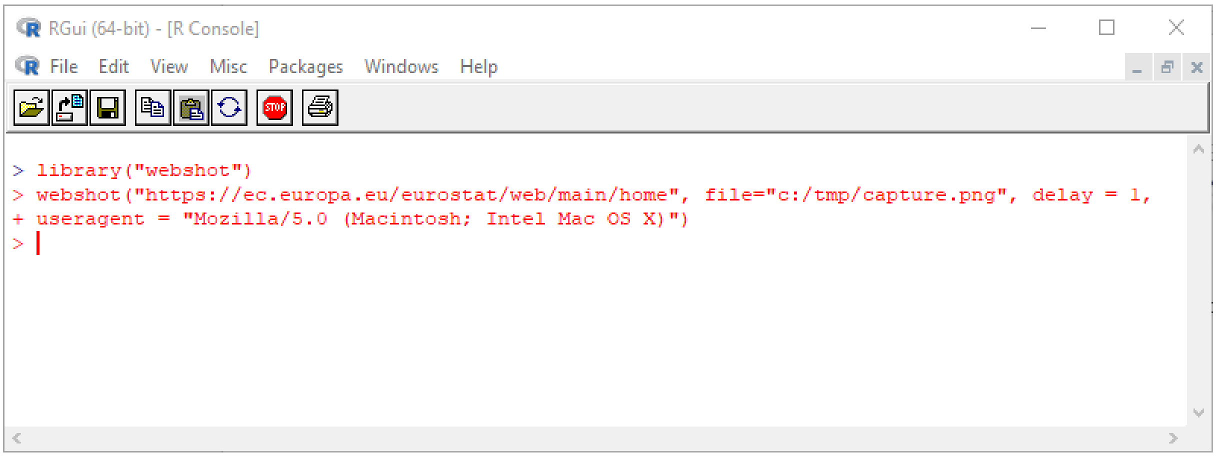
Commençons par le chose la plus élémentaire qui est de faire une capture d'écran d'un site Internet avec le package **webshot** et sa fonction correspondante **webshot( )**:

```
> library("webshot")
> URL <- "http://www.casablanca-bourse.com/bourseweb/Negociation-Historique.aspx?Cat=24&IdLink=302"
> #on est obligés d'expliciter un useragent car sinon certains sites détectent les bots
> #la capture n'est pas forcément bonne et ce surtout si la page web est mal scriptée
> webshot(URL, file="c:/tmp/capture.png", delay = 1,
+ useragent = "Mozilla/5.0 (Macintosh; Intel Mac OS X)")
>
```

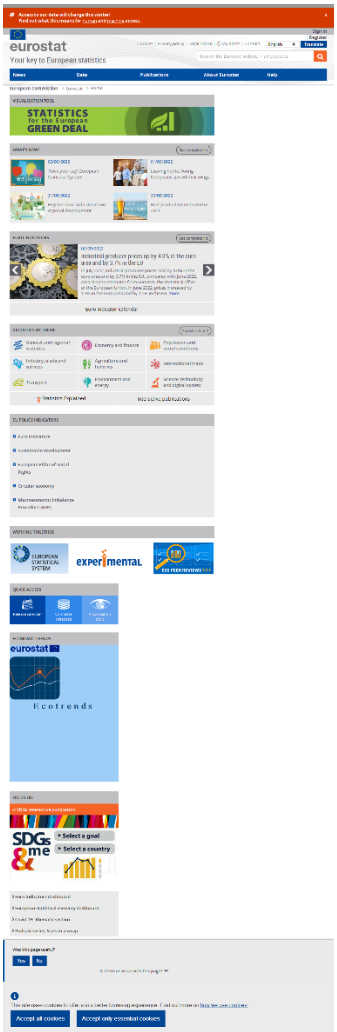
Ce qui donne:



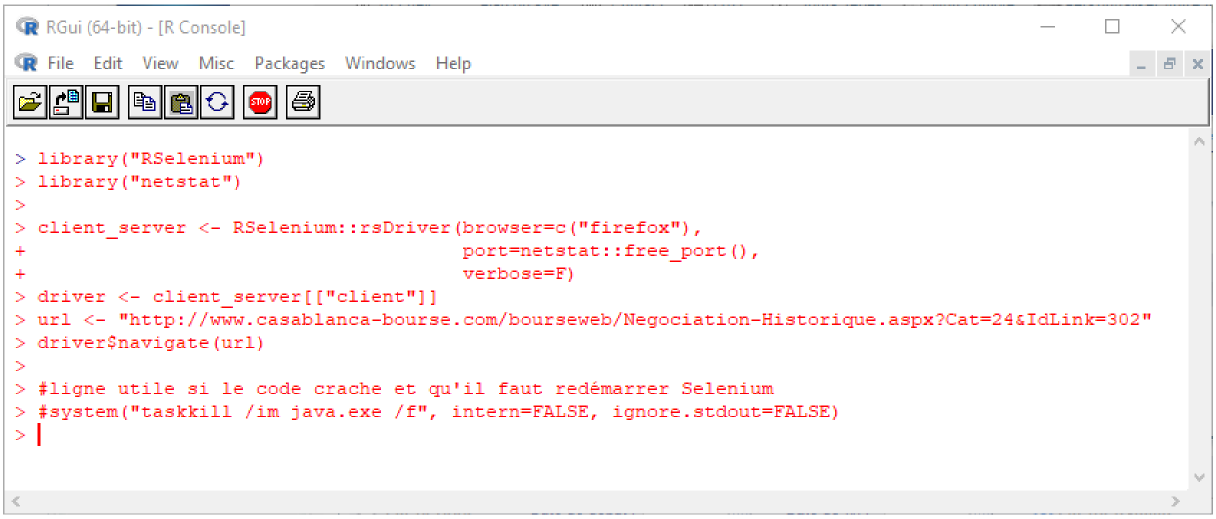
Donc on voit que contrairement à Selenium sur Python, la capture est catastrophique pour ce site. Alors que pour un autre site:



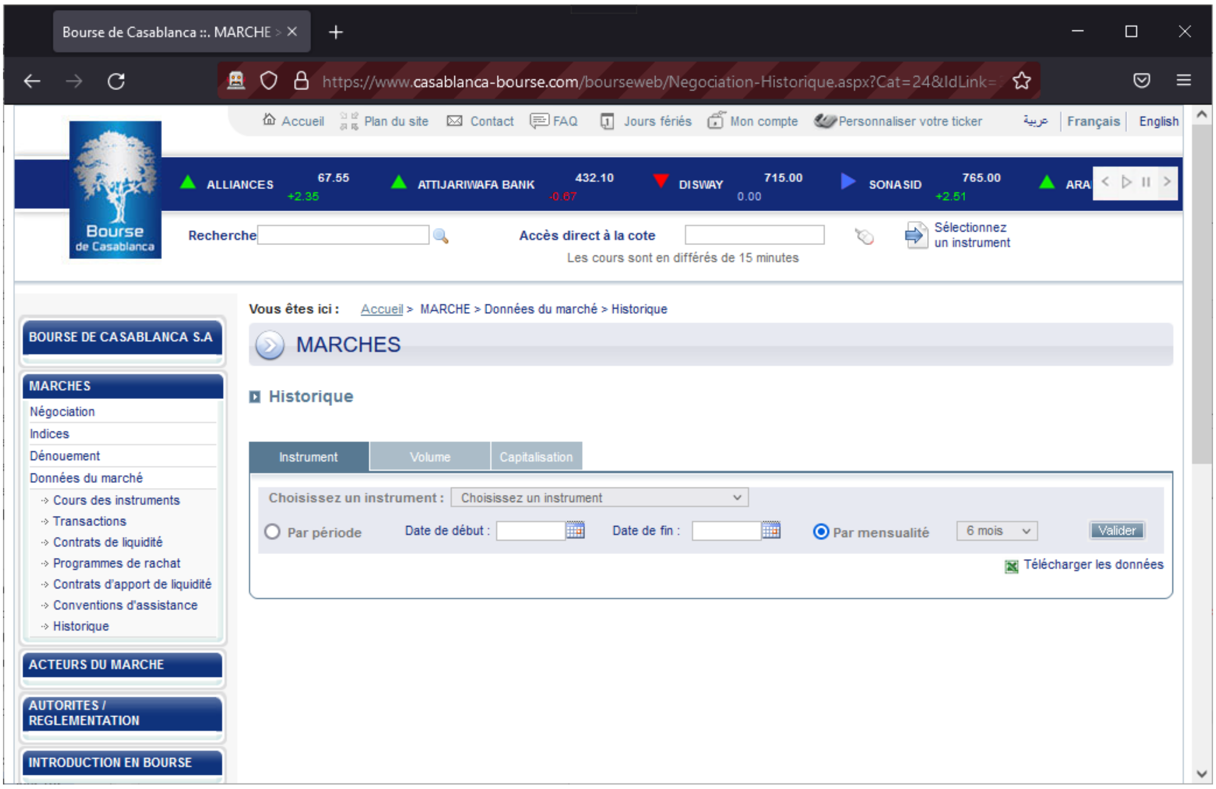
Le résultat est beaucoup plus satisfaisant:



Maintenant voyons comment ouvrir le navigateur avec RSelenium afin de par la suite y faire des manipulations sur la page Internet choisie:



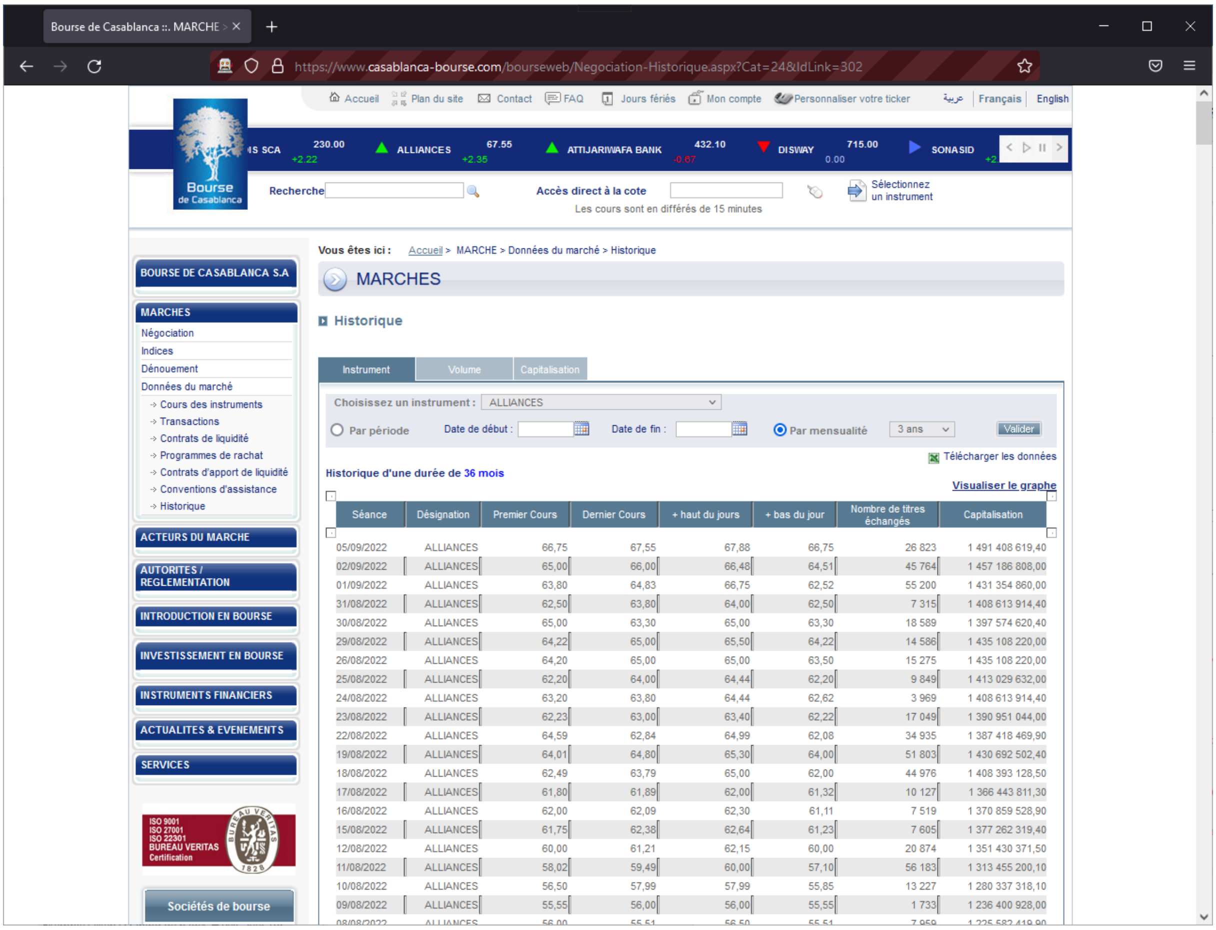
Ce qui donne:



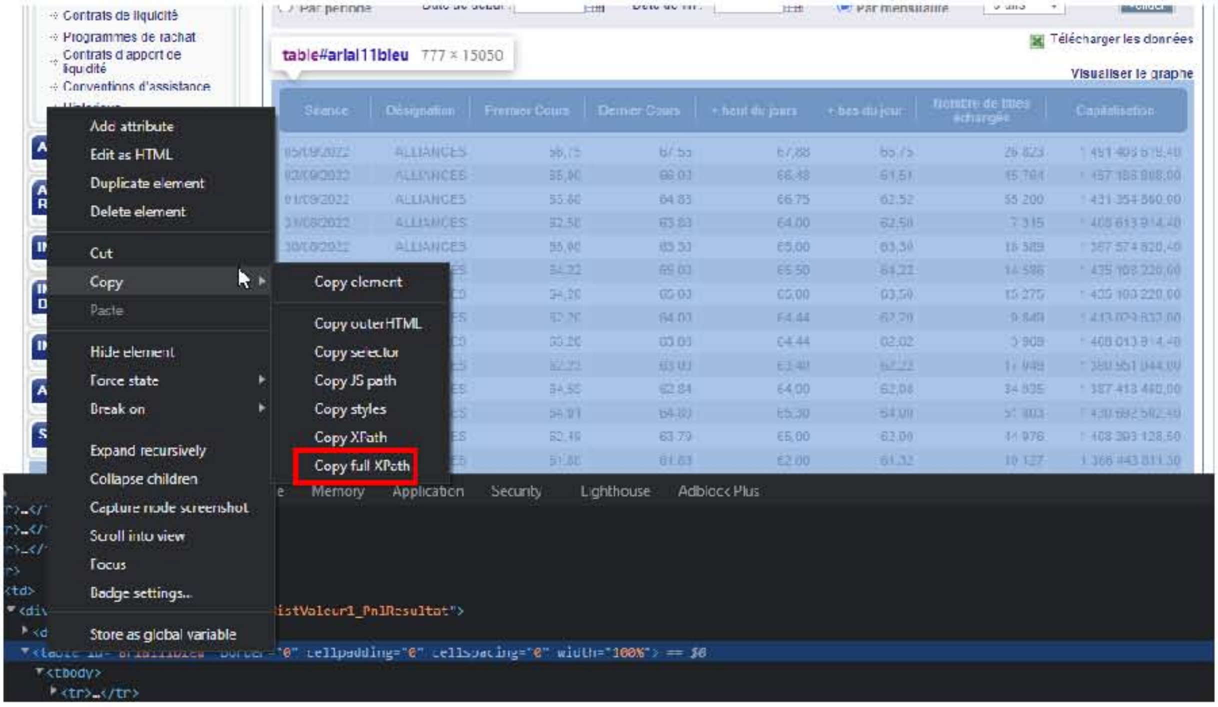
Nous voulons un code qui choisira dans la première liste déroulante comme instrument *ALLIANCES* et pour mensualité dans la deuxième liste déroulante, l'option *3* ans. Le code correspondant donne:

```
> library("RSelenium")
> library("netstat")
>
> client_server <- RSelenium::rsDriver(browser=c("firefox"),
+                                      port=netstat::free_port(),
+                                      verbose=F)
> driver <- client_server[["client"]]
> url <- "http://www.casablanca-bourse.com/bourseweb/Negociation-Historique.aspx?Cat=24&IdLink=302"
> driver$navigate(url)
>
> #on met l'instrument sur ALLIANCES
> option <- driver$findElement(using = 'xpath',
+ "//select[@id='HistoriqueNegociation1_HistValeur1_DDValeur']/option[text()='ALLIANCES']")
> option$clickElement()
>
> #on met le par mensualité sur 3 ANS
> option <- driver$findElement(using = 'xpath',
+ "//select[@id='HistoriqueNegociation1_HistValeur1_DDuree']/option[text()='3 ans']")
> option$clickElement()
>
> #on clique sur VALIDER
> driver$findElement(using = 'id', value = 'HistoriqueNegociation1_HistValeur1_ImageButton1')$clickElement()
>
> #on clos tout correctement au besoin
> #driver$closeWindow()
> #system("taskkill /im java.exe /f", intern=FALSE, ignore.stdout=FALSE)
>
```

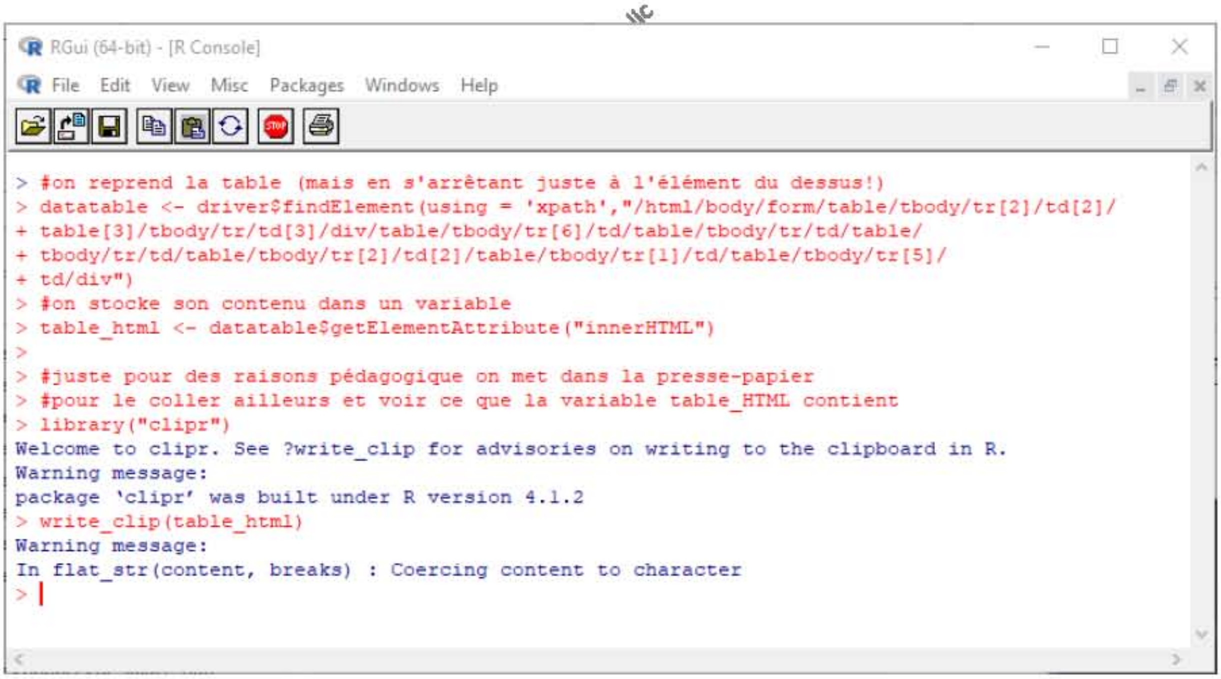
Le résultat sera:



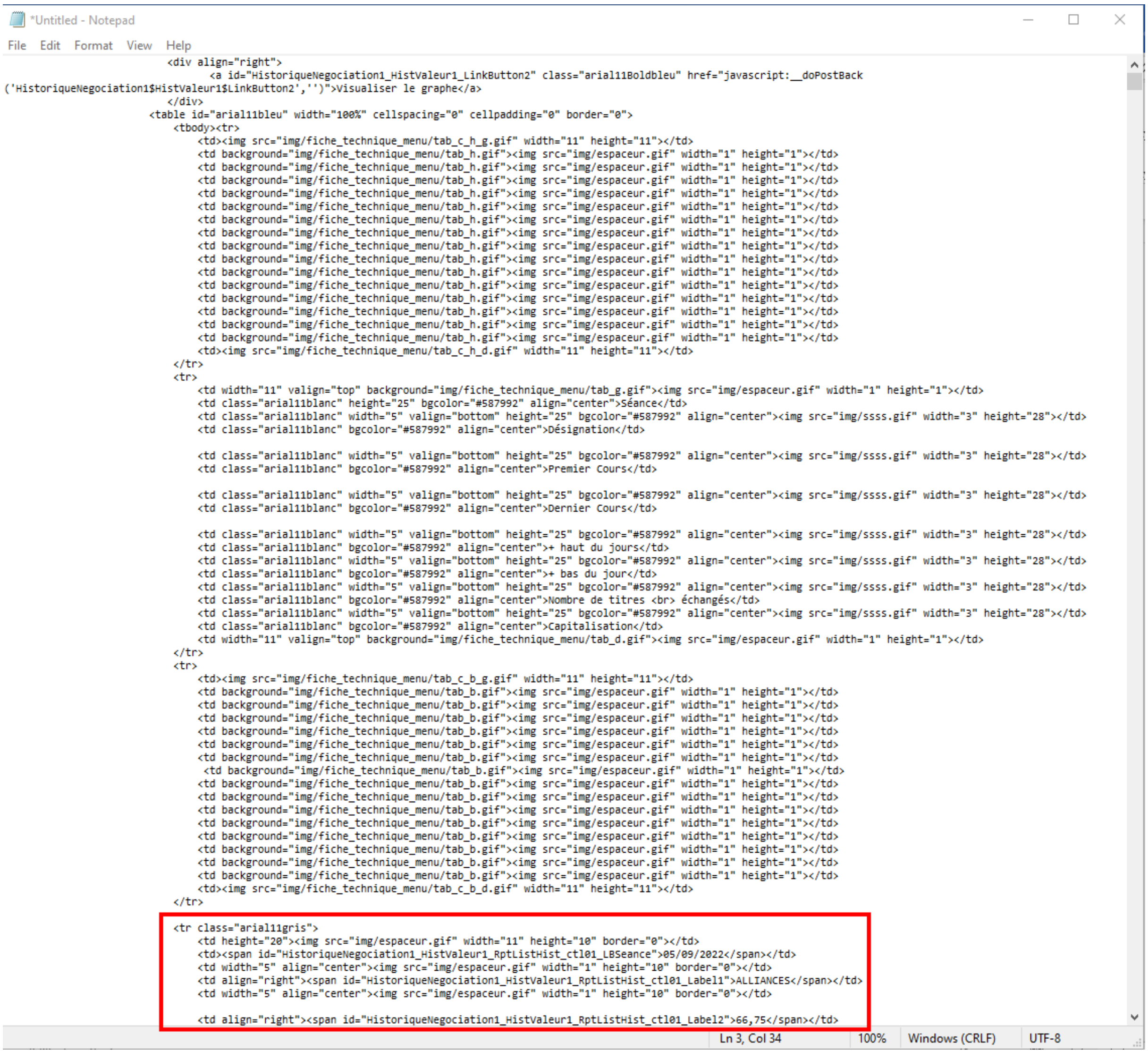
Pour la suite il faut connaître le XPath de l'élément qu'on veut lire. Le mieux pour cela consiste à utiliser les *Developper Tools* natifs de Chrome, de chercher le nœud HTML de l'élément d'intérêt et de faire un clic droit et sélectionner **Copy full XPath**:



Nous évcrivons alors:



Nous avons alors en collant le contenu du presse-papier dans le bloc-notes le résultat suivant où nous voyons bien le début des données qui nous intéressent en bas (rectangle rouge):



```
*Untitled - Notepad
File  Edit  Format  View  Help
                <div align="right">
                    <a id="HistoriqueNegociation1_HistValeur1_LinkButton2" class="arial11Boldbleu" href="javascript:__doPostBack
('HistoriqueNegociation1$HistValeur1$LinkButton2','')">Visualiser le graphe</a>
                </div>
            <table id="arial11bleu" width="100%" cellspacing="0" cellpadding="0" border="0">
                <tbody><tr>
                    <td><img src="img/fiche_technique_menu/tab_c_h_g.gif" width="11" height="11"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_h.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td><img src="img/fiche_technique_menu/tab_c_h_d.gif" width="11" height="11"></td>
                </tr>
                <tr>
                    <td width="11" valign="top" background="img/fiche_technique_menu/tab_g.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td class="arial11blanc" height="25" bgcolor="#587992" align="center">Séance</td>
                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">Désignation</td>

                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">Premier Cours</td>

                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">Dernier Cours</td>

                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">+ haut du jours</td>
                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">+ bas du jour</td>
                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">Nombre de titres <br> échangés</td>
                    <td class="arial11blanc" width="5" valign="bottom" height="25" bgcolor="#587992" align="center"><img src="img/ssss.gif" width="3" height="28"></td>
                    <td class="arial11blanc" bgcolor="#587992" align="center">Capitalisation</td>
                    <td width="11" valign="top" background="img/fiche_technique_menu/tab_d.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                </tr>
                <tr>
                    <td><img src="img/fiche_technique_menu/tab_c_b_g.gif" width="11" height="11"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                     <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td background="img/fiche_technique_menu/tab_b.gif"><img src="img/espaceur.gif" width="1" height="1"></td>
                    <td><img src="img/fiche_technique_menu/tab_c_b_d.gif" width="11" height="11"></td>
                </tr>

                <tr class="arial11gris">
                    <td height="20"><img src="img/espaceur.gif" width="11" height="10" border="0"></td>
                    <td><span id="HistoriqueNegociation1_HistValeur1_RptListHist_ctl01_LBSeance">05/09/2022</span></td>
                    <td width="5" align="center"><img src="img/espaceur.gif" width="1" height="10" border="0"></td>
                    <td align="right"><span id="HistoriqueNegociation1_HistValeur1_RptListHist_ctl01_Label1">ALLIANCES</span></td>
                    <td width="5" align="center"><img src="img/espaceur.gif" width="1" height="10" border="0"></td>

                    <td align="right"><span id="HistoriqueNegociation1_HistValeur1_RptListHist_ctl01_Label2">66,75</span></td>
Ln 3, Col 34    100%    Windows (CRLF)    UTF-8
```

Pour extraire les données de ce tableau, nous utilisons (nous remettons tout le code depuis le début et avons mis dans un rectangle rouge la nouvelle partie):

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("RSelenium")
> library("netstat")
>
> client_server <- RSelenium::rsDriver(browser=c("firefox"),
+                                      port=netstat::free_port(),
+                                      verbose=F)
> driver <- client_server[["client"]]
> url <- "http://www.casablanca-bourse.com/bourseweb/Negociation-Historique.aspx?Cat=24&IdLink=302"
> driver$navigate(url)
>
> #ligne utile si le code crache et qu'il faut redémarrer Selenium
> #system("taskkill /im java.exe /f", intern=FALSE, ignore.stdout=FALSE)
>
> #on met l'instrument sur ALLIANCES
> option <- driver$findElement(using = 'xpath',
+ "//select[@id='HistoriqueNegociation1_HistValeur1_DDValeur']/option[text()='ALLIANCES']")
> option$clickElement()
>
> #on met le par mensualité sur 3 ANS
> option <- driver$findElement(using = 'xpath',
+ "//select[@id='HistoriqueNegociation1_HistValeur1_DDuree']/option[text()='3 ans']")
> option$clickElement()
>
> #on clique sur VALIDER
> driver$findElement(using = 'id', value = 'HistoriqueNegociation1_HistValeur1_ImageButton1')$clickElement()
>
> #on reprend la table (mais en s'arrêtant juste à l'élément du dessus!)
> Sys.sleep(5)  #on attend 5 secondes
> datatable <- driver$findElement(using = 'xpath',"/html/body/form/table/tbody/tr[2]/td[2]/
+ table[3]/tbody/tr/td[3]/div/table/tbody/tr[6]/td/table/tbody/tr/td/table/
+ tbody/tr/td/table/tbody/tr[2]/td[2]/table/tbody/tr[1]/td/table/tbody/tr[5]/
+ td/div")
>
> #on stocke son contenu dans un variable
> table_html <- datatable$getElementAttribute("innerHTML")
>
> library("XML")
> elem_html<-htmlTreeParse(table_html, useInternalNodes = T, asText = TRUE)
>
> data_table <- readHTMLTable(elem_html, header = T, stringsAsFactors = FALSE)[[1]]
> head(data_table,5) #les cinq premières lignes

1    SÃƒÂ©ance  DÃƒÂ©signation  Premier Cours  Dernier Cours  + haut du jours  + bas du jour
2
3  06/09/2022       ALLIANCES          67,10          66,79            67,99          65,30
4  05/09/2022       ALLIANCES          66,75          67,55            67,88          66,75
5  02/09/2022       ALLIANCES          65,00          66,00            66,48          64,51

1 Nombre de titres  ÃƒÂ©changÃƒÂ©s       Capitalisation
2
3                         14Â 965  1Â 474Â 628Â 892,50
4                         26Â 823  1Â 491Â 408Â 619,40
5                         45Â 764  1Â 457Â 186Â 808,00
> |
```

Donc nous avons atteint notre objectif! Le reste c'est du nettoyage classique et ennuyant…

10. Manipuler, filtrer/trier des données de data frame (Data Munging)

Contenu

Pour traiter des bases des manipulations des data frames, nous allons travailler avec les données suivantes:

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> setwd("C:/")
> mydata=read.csv("VentesClean.csv",header=T,sep=";")
> mydata
   NClient        Activite NDeCommande DateDeCommande            Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1      100      Assurances           1     03.01.2000 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
2      123 Machines/Outils           2     03.01.2000            IBM 500        2         2299  0.00%             4598.00          Oui
3      109       Éducation           3     03.01.2000      AST Intel 150        5         2690  0.00%            13450.00          Oui
4      104       Éducation           4     03.01.2000      AST Intel 200        3         3190  0.00%             9570.00          Oui
5      117         Banques           5     04.01.2000 Compaq Presario 100       13         1650  1.50%            21128.25          Oui
6      103      Assurances           6     04.01.2000      AST Intel 150        2         2690  0.00%             5380.00          Oui
7      104       Éducation           7     04.01.2000      AST Intel 200        2         3190  0.00%             6380.00          Oui
8      111     Alimentaire           8     04.01.2000            IBM 500        4         2299  0.00%             9196.00          Oui
9      113    Construction           9     04.01.2000 Compaq Presario 100        4         1650  0.00%             6600.00          Oui
10     116  Pharmaceutique          10     04.01.2000            IBM 500        2         2299  0.00%             4598.00          Oui
11     110    Distribution          11     05.01.2000      AST Intel 200        6         3190  1.50%            18852.90          Oui
12     112 Machines/Outils          12     05.01.2000 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
13     123 Machines/Outils          13     05.01.2000            IBM 500        6         2299  1.50%            13587.09          Oui
14     113    Construction          14     05.01.2000      AST Intel 150        3         2690  0.00%             8070.00          Oui
15     115    Distribution          15     05.01.2000 Compaq Presario 100        8         1650  1.50%            13002.00          Oui
16     124       Éducation          16     05.01.2000      AST Intel 200        8         3190  1.50%            25137.20          Oui
17     124       Éducation          17     05.01.2000 Compaq Presario 100       11         1650  1.50%            17877.75          Oui
18     106    Construction          18     05.01.2000      AST Intel 200       11         3190  1.50%            34563.65          Oui
19     101    Construction          19     05.01.2000 Compaq Presario 100       14         1650  1.50%            22753.50          Non
20     116  Pharmaceutique          20     06.01.2000            IBM 500        7         2299  1.50%            15851.60          Non
21     112 Machines/Outils          21     06.01.2000      AST Intel 150        6         2690  1.50%            15897.90          Oui
22     125    Construction          22     06.01.2000 Compaq Presario 100       23         1650  3.00%            36811.50          Oui
23     100      Assurances          23     06.01.2000            IBM 500        3         2299  0.00%             6897.00          Oui
24     125    Construction          24     06.01.2000      AST Intel 200        2         3190  0.00%             6380.00          Oui
25     104       Éducation          25     07.01.2000      AST Intel 150       12         2690  1.50%            31795.80          Oui
26     126 Machines/Outils          26     07.01.2000      AST Intel 150       24         2690  3.00%            62623.20          Oui
27     121  Pharmaceutique          27     07.01.2000            IBM 500        8         2299  1.50%            18116.12          Non
28     114    Distribution          28     07.01.2000      AST Intel 200        9         3190  1.50%            28279.35          Oui
29     103      Assurances          29     07.01.2000 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
30     125    Construction          30     07.01.2000      AST Intel 200        4         3190  0.00%            12760.00          Oui
31     120         Banques          31     10.01.2000      AST Intel 150        2         2690  0.00%             5380.00          Oui
32     111     Alimentaire          32     10.01.2000 Compaq Presario 100       16         1650  3.00%            25608.00          Oui
33     118       Éducation          33     10.01.2000            IBM 500        3         2299  0.00%             6897.00          Oui
```

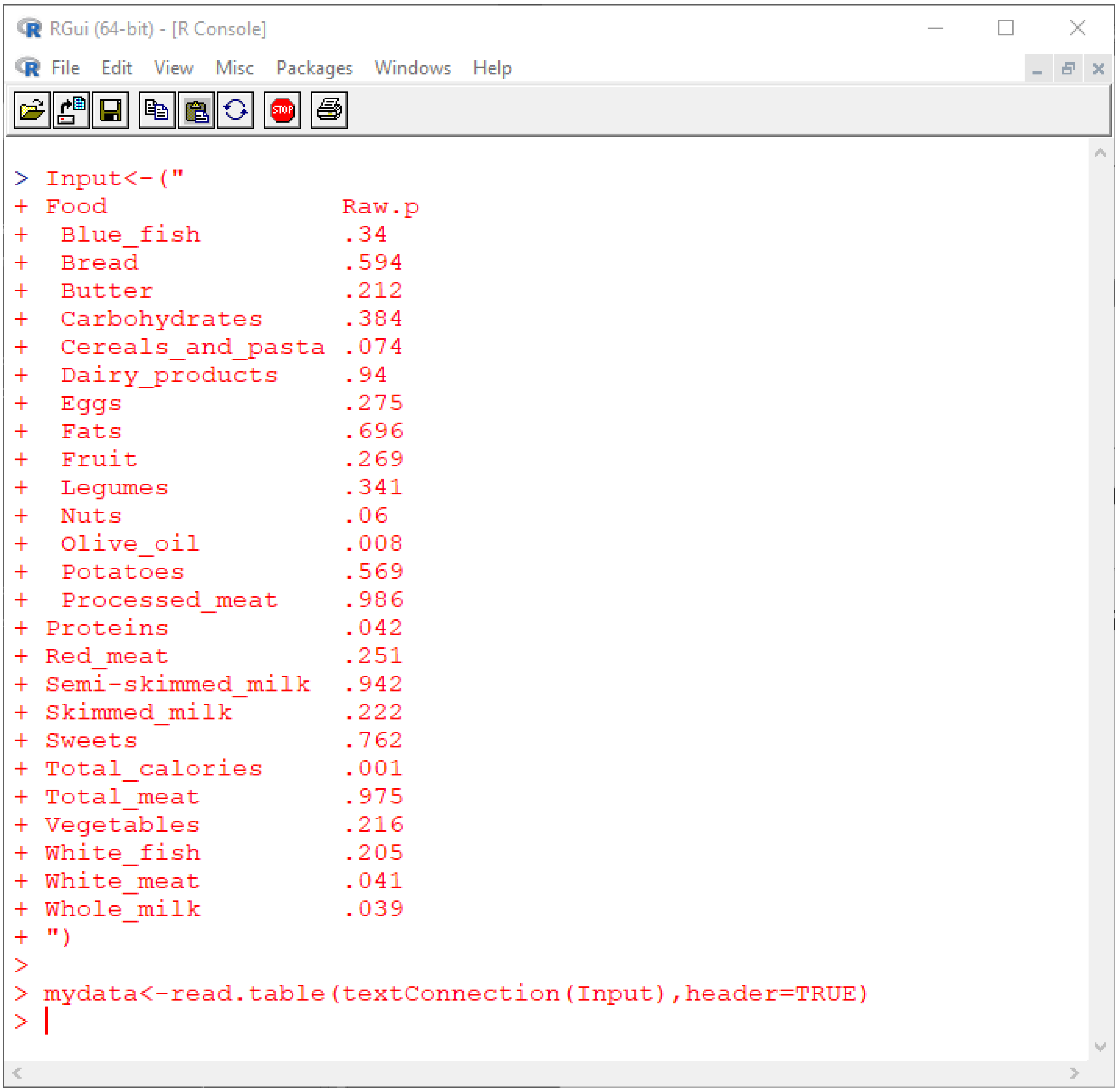
Par rapport à ce que nous allons voir il faut savoir qu'il existe toujours plusieurs manières d'arriver au résultat. Le but de cet exercice est surtout d'étudier des commandes que vous retrouverez dans d'autres contextes plus importants, sur de nombreux forums ou livres. Bref c'est une culture générale qu'il faut acquérir pour être à l'aise.

Exemple 75.: Créer un data frame à partir de saisie simple

R 3.6.1

Bon voilà une situation bien "bourrin" quand on veut faire des tests à la va-vite ou que l'on est pressé par le temps.

On peut saisir les données directement "à l'arrache" dans la console et ensuite grâce aux fonction **read.table( )** et **textConnection( )** transformer le résultat en data frame:



Exemple 76.: Créer un data frame à partir de fonctions

R 3.0.2

Voici un cas très utile pour tous les domaines de la science: Comment créer un data frame à partir d'une séquence ou de fonctions mathématique connues?

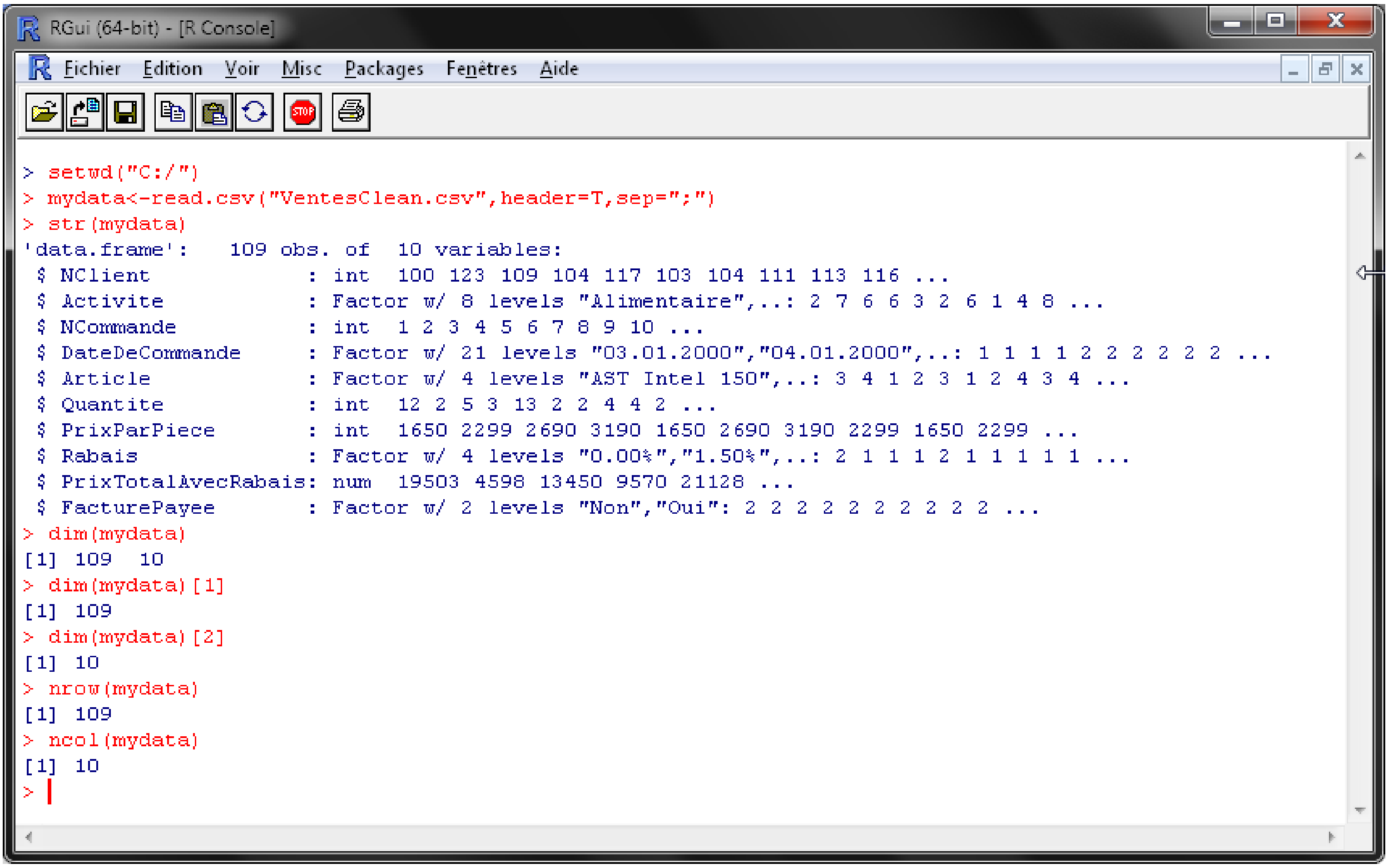
Pour ce faire on va considérer ce petit exemple prix de la micro-économie qui utilise "bien évidemment" la fonction **data.frame( )** pour arriver à nos fins (!):

```
> linear <- function(p, alpha, beta){alpha*p + beta}
> constant_elast <- function(p, alpha, beta){exp(alpha*log(p)+beta)}
> logistic <- function(p, c, alpha, p0){c/(1+exp(-alpha*(p-p0)))}
>
> p <- seq(1, 100)
> y1 <- linear(p, -1, 100)
> y2 <- constant_elast(p, -.5, 4.5)
> y3 <- logistic(p, 100, -.2, 50)
>
> df <- data.frame('Prices' = p, 'Linear' = y1, 'Constant_elast' = y2, 'Logistic' = y3)
> head(df)
  Prices Linear Constant_elast Logistic
1      1     99       90.01713 99.99446
2      2     98       63.65172 99.99323
3      3     97       51.97141 99.99173
4      4     96       45.00857 99.98990
5      5     95       40.25688 99.98766
6      6     94       36.74934 99.98493
>
```

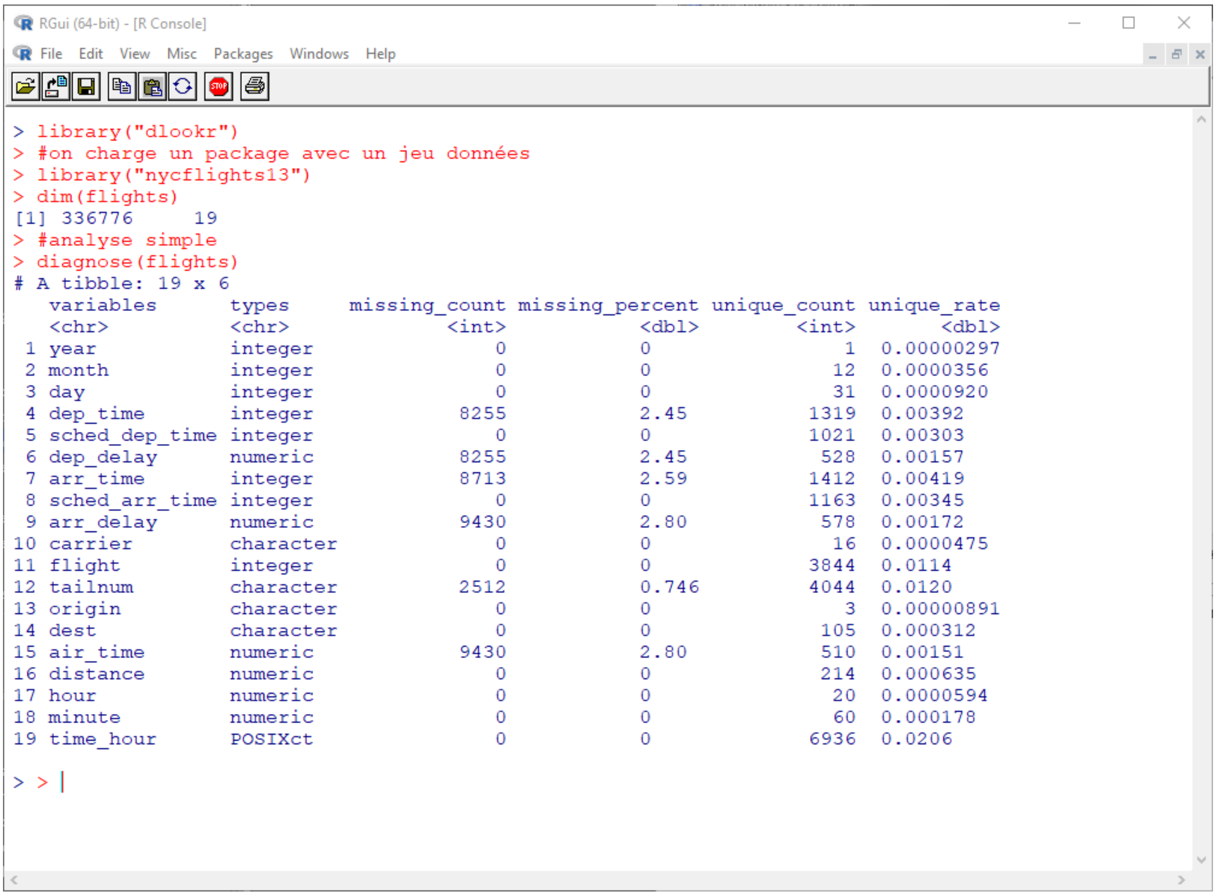
Exemple 77.: Résumé structurel d'un date frame et Data Quality

R 3.0.2

Les commandes **str( )** , **dim( )** ou encore **nrow( )** et **ncolumn( )** appliqués à des data frame sont bien utiles pour avoir un résumé de ce que l'on a:



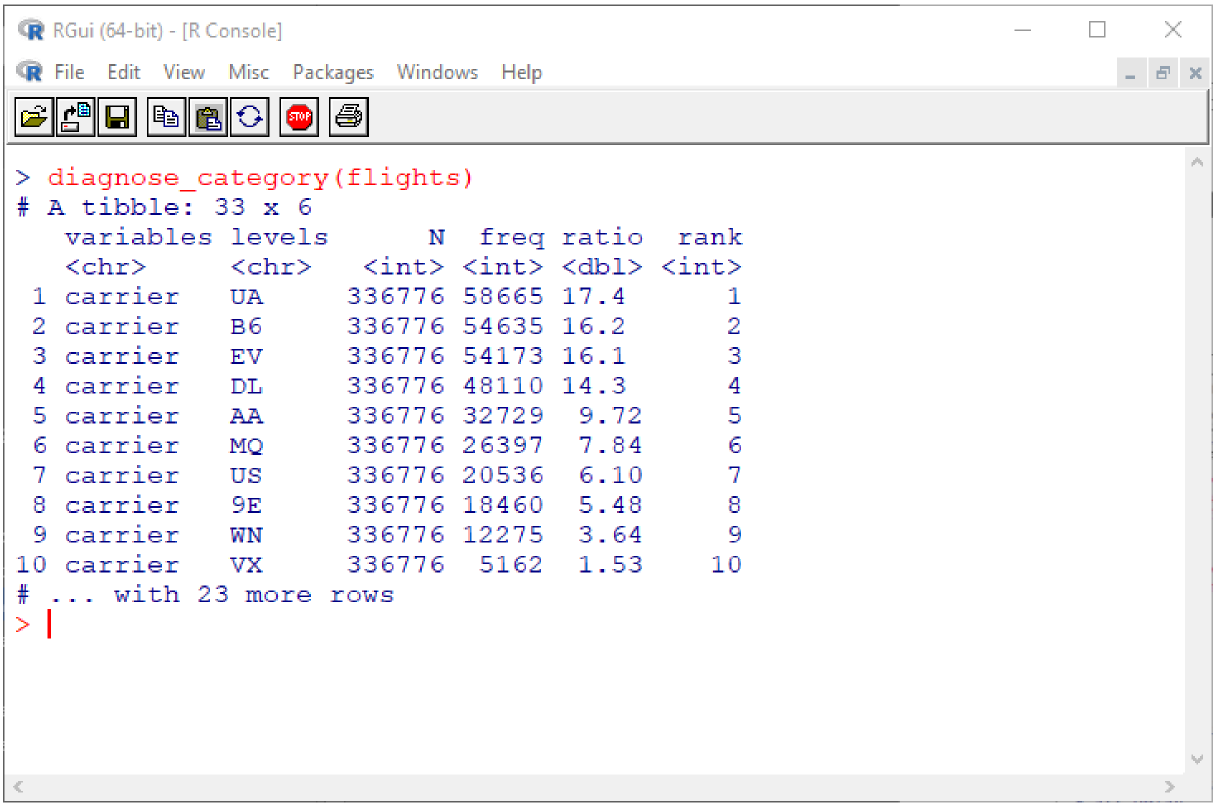
Nous avons dans le cours théorique, mentionné la norme ISO 8000 de qualité des données. Voyons un exemple de package **dlookr** permettant de faire une petite analyse rapide avec la fonction **diagnose( )**:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("dlookr")
> #on charge un package avec un jeu données
> library("nycflights13")
> dim(flights)
[1] 336776     19
> #analyse simple
> diagnose(flights)
# A tibble: 19 x 6
   variables      types     missing_count missing_percent unique_count unique_rate
   <chr>          <chr>             <int>           <dbl>        <int>       <dbl>
 1 year           integer               0           0                1 0.00000297
 2 month          integer               0           0               12 0.0000356
 3 day            integer               0           0               31 0.0000920
 4 dep_time       integer            8255           2.45          1319 0.00392
 5 sched_dep_time integer               0           0             1021 0.00303
 6 dep_delay      numeric            8255           2.45           528 0.00157
 7 arr_time       integer            8713           2.59          1412 0.00419
 8 sched_arr_time integer               0           0             1163 0.00345
 9 arr_delay      numeric            9430           2.80           578 0.00172
10 carrier        character             0           0               16 0.0000475
11 flight         integer               0           0             3844 0.0114
12 tailnum        character          2512           0.746         4044 0.0120
13 origin         character             0           0                3 0.00000891
14 dest           character             0           0              105 0.000312
15 air_time       numeric            9430           2.80           510 0.00151
16 distance       numeric               0           0              214 0.000635
17 hour           numeric               0           0               20 0.0000594
18 minute         numeric               0           0               60 0.000178
19 time_hour      POSIXct               0           0             6936 0.0206

> > |
```

Il y a aussi un rapport pour les variables catégorielles avec la fonction **diagnose_category( )**:



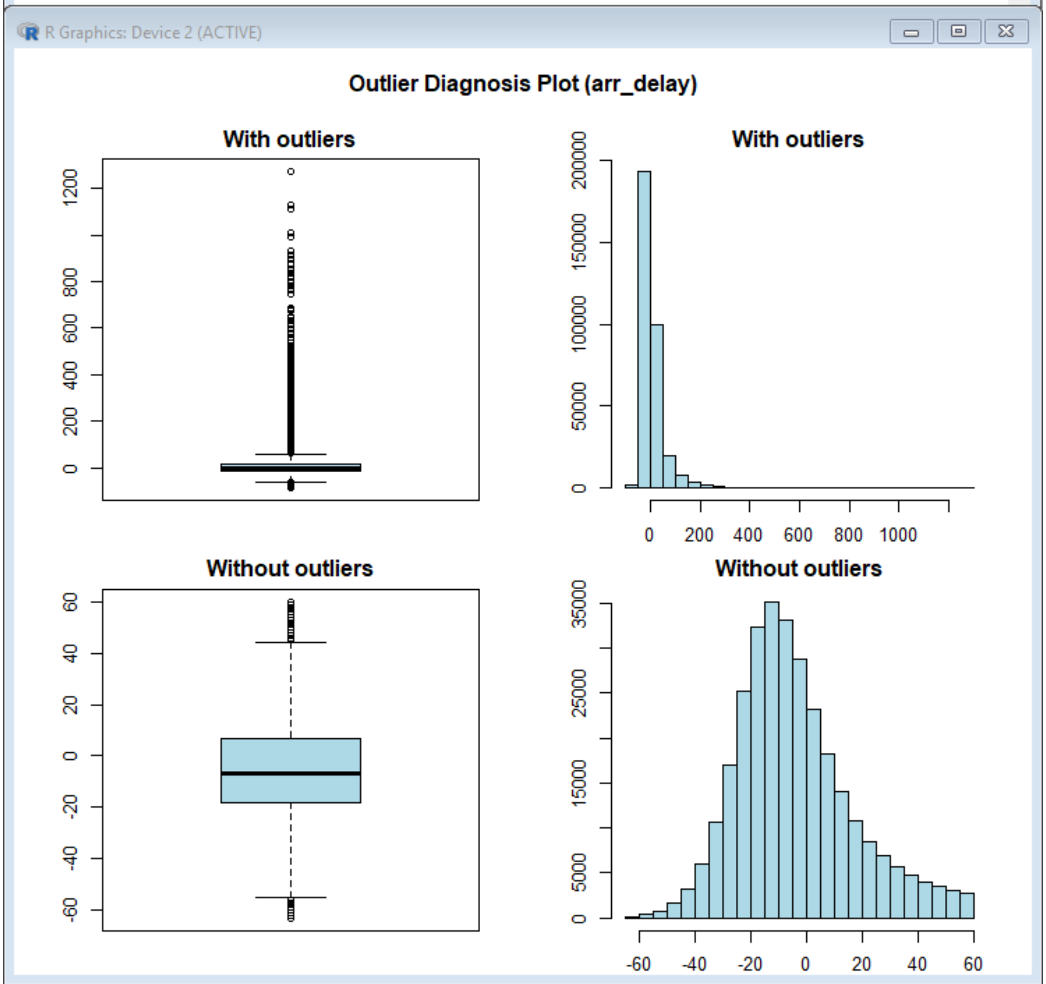
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> diagnose_category(flights)
# A tibble: 33 x 6
   variables levels      N  freq ratio  rank
   <chr>     <chr>   <int> <int> <dbl> <int>
 1 carrier   UA     336776 58665 17.4      1
 2 carrier   B6     336776 54635 16.2      2
 3 carrier   EV     336776 54173 16.1      3
 4 carrier   DL     336776 48110 14.3      4
 5 carrier   AA     336776 32729  9.72     5
 6 carrier   MQ     336776 26397  7.84     6
 7 carrier   US     336776 20536  6.10     7
 8 carrier   9E     336776 18460  5.48     8
 9 carrier   WN     336776 12275  3.64     9
10 carrier   VX     336776  5162  1.53    10
# ... with 23 more rows
> |
```

Et aussi un rapport pour les outliers avec la fonction **diagnose_outlier( )**:

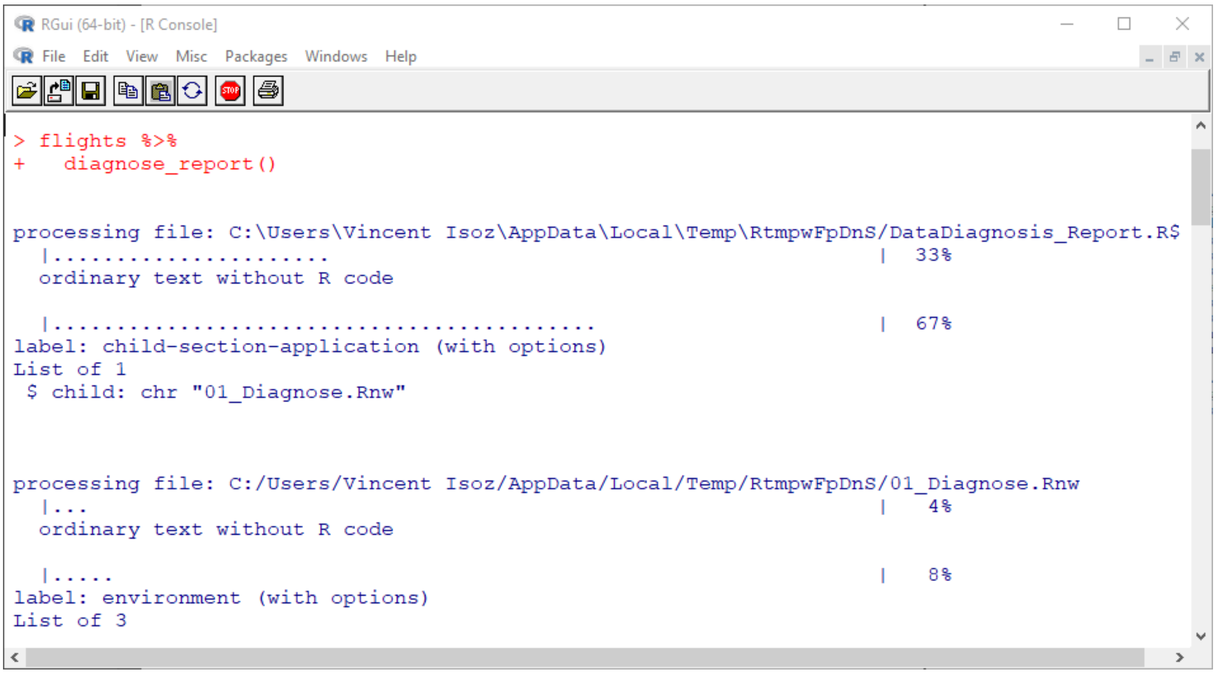
```
> diagnose_outlier(flights)
# A tibble: 14 x 6
   variables      outliers_cnt outliers_ratio outliers_mean with_mean without_mean
   <chr>                 <int>          <dbl>         <dbl>     <dbl>        <dbl>
 1 year                      0       0                  NaN    2013         2013
 2 month                     0       0                  NaN       6.55         6.55
 3 day                       0       0                  NaN      15.7         15.7
 4 dep_time                  0       0                  NaN    1349.        1349.
 5 sched_dep_time            0       0                  NaN    1344.        1344.
 6 dep_delay             43216      12.8                 93.1    12.6          0.444
 7 arr_time                  0       0                  NaN    1502.        1502.
 8 sched_arr_time            0       0                  NaN    1536.        1536.
 9 arr_delay             27880       8.28               121.      6.90        -3.69
10 flight                    1       0.000297          8500    1972.        1972.
11 air_time               5448       1.62               400.    151.         146.
12 distance                715       0.212             4955.   1040.        1032.
13 hour                      0       0                  NaN      13.2         13.2
14 minute                    0       0                  NaN      26.2         26.2
> 
```

Il y a aussi une fonction **plot_outliers( )**:

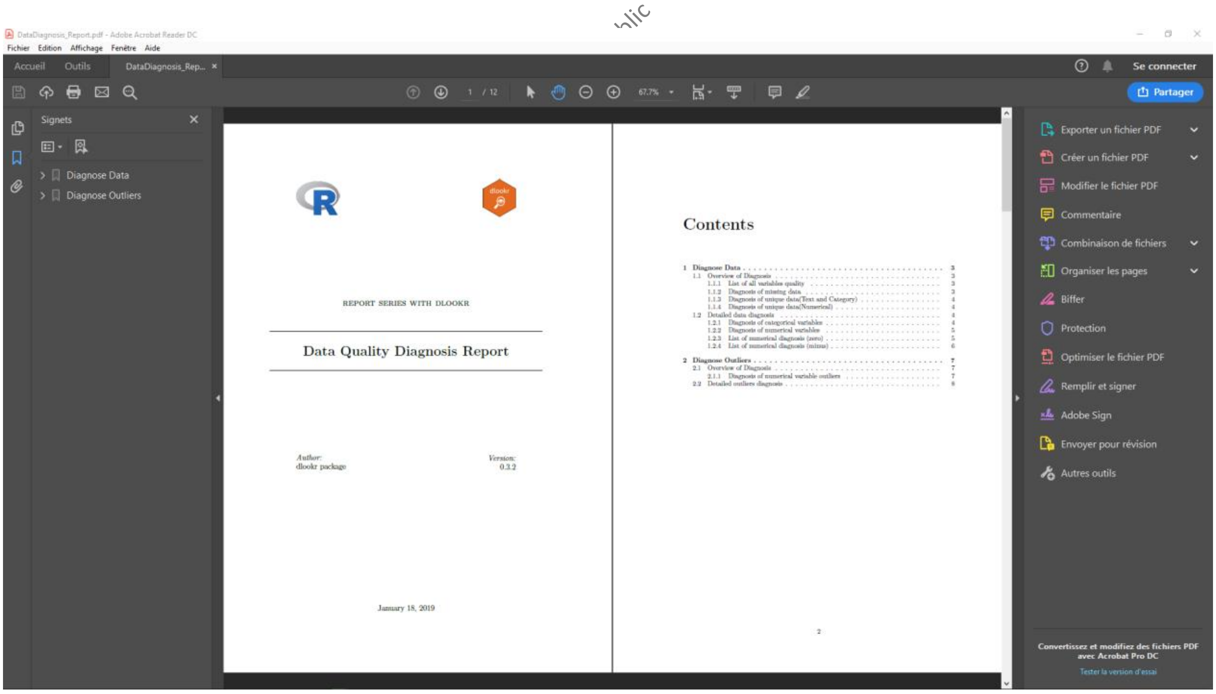
```
> library("dplyr")
> flights %>%
+   plot_outlier(arr_delay)
> 
```



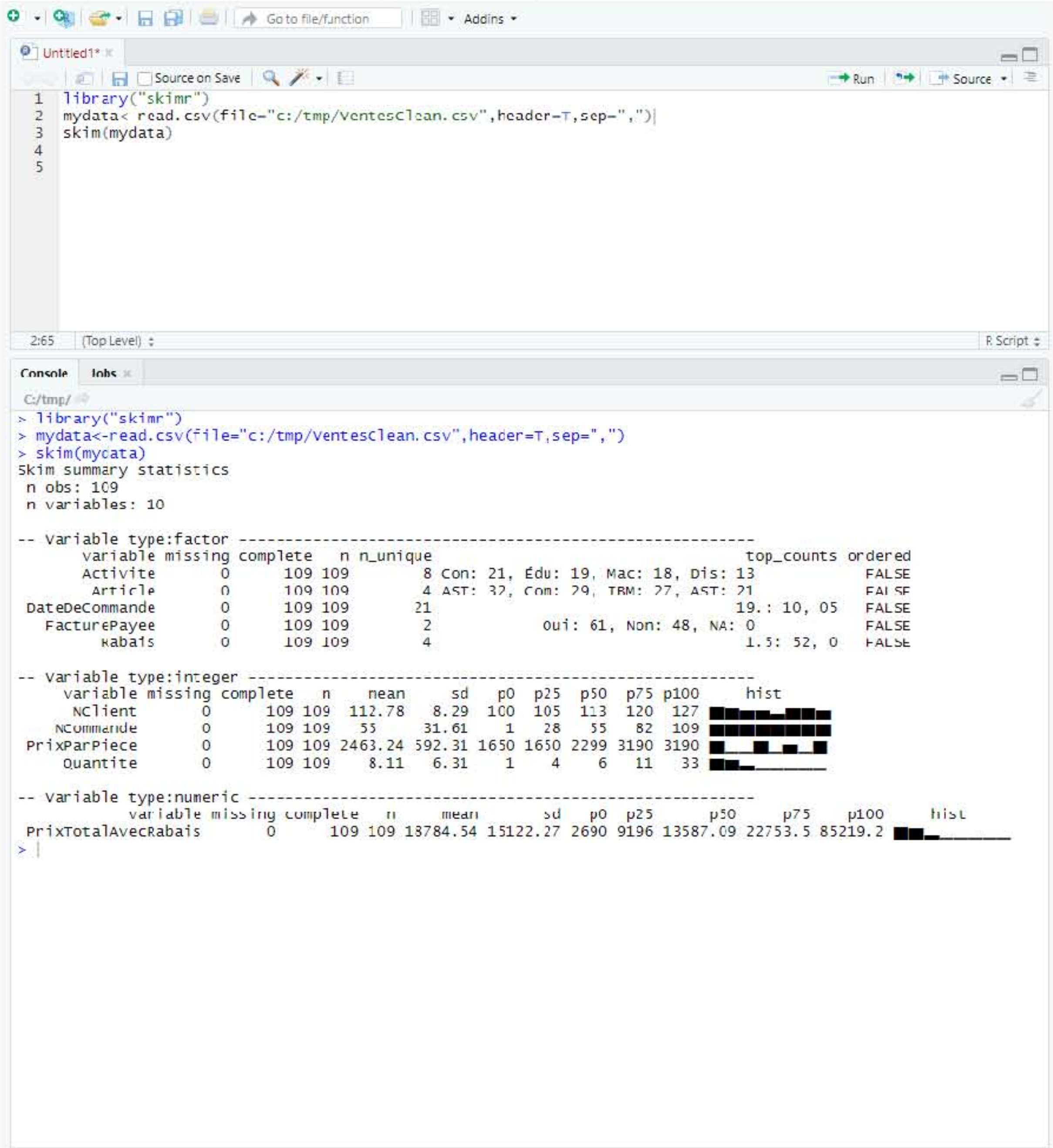
Il y a également une fonction de reporting nommée **diagnose_report( )** qui si MikTeX est installé (avec déjà de nombreux package installés sinon quoi vous risquez d'avoir des problèmes sous Microsoft Windows), va générer un PDF de plusieurs dizaines de page de très bonne facture:



Ce qui donne:



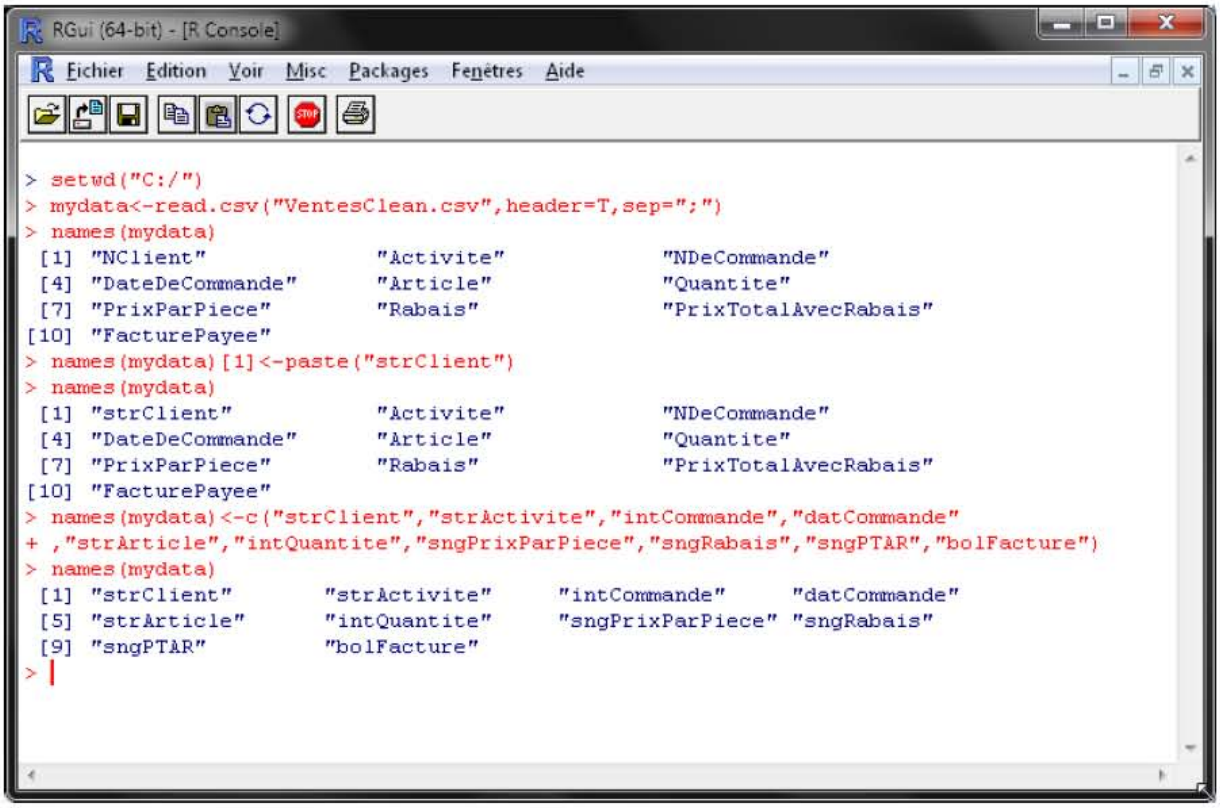
Sinon pour ceux qui ont R Studio, il y a le package **skimr** et sa fonction **skim( )** qui fait aussi un résumé d'un jeu de données sous forme particulièrement élégante (sous d'autres IDE les petits histogrammes visibles sur la capture d'écran peuvent ne pas apparaître!):



Exemple 78.: Renommer des intitulés de colonnes

R 3.0.2

Quelques commandes et options pour changer individuellement les intitulés de certaines colonnes et de plusieurs d'un coup:



Exemple 79.: Renommer des intitulés de lignes

R 3.0.2

Voici un exemple très qui nous sera utile (entre autres) beaucoup plus loin quand nous traiterons du problème du voyageur de commerce utilisant la commande native **row.names( )** :

```
R Console
> lat<-c(27.685,29.343,23.495)
> lon<-c(46.344,46.233,46.23)
> coord<-data.frame(lat,lon)
> row.names(coord)<-c("Genève","Berne","Lausanne")
> coord
            lat    lon
Genève   27.685 46.344
Berne    29.343 46.233
Lausanne 23.495 46.230
> #on calcule les distances euclidiennes entre chaque objet
> dist(coord)
           Genève    Berne
Berne    1.661711
Lausanne 4.191551 5.848001
> 
```

Pour supprimer les noms lignes, on utilisera **rownames(coord)<-NULL**.

Exemple 80.: Ajouter une colonne avec des identifiants uniques

R 3.4.2

Pour l'export ou l'import vers d'autres applications (en particulier FileMaker, Microsoft Access et Microsoft Power BI), il peut être utile de savoir rajouter une colonne d'identifiants unique à un data frame.

Voyons cela:

```
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=",")
> head(mydata)
  NClient        Activite NCommande DateDeCommande             Article Quantite
1     100       Assurances         1     03.01.2000 Compaq Presario 100       12
2     123 Machines/Outils          2     03.01.2000             IBM 500        2
3     109        Éducation         3     03.01.2000       AST Intel 150        5
4     104        Éducation         4     03.01.2000       AST Intel 200        3
5     117          Banques         5     04.01.2000 Compaq Presario 100       13
6     103       Assurances         6     04.01.2000       AST Intel 150        2
  PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1         1650  1.50%            19503.00          Oui
2         2299  0.00%             4598.00          Oui
3         2690  0.00%            13450.00          Oui
4         3190  0.00%             9570.00          Oui
5         1650  1.50%            21128.25          Oui
6         2690  0.00%             5380.00          Oui
> mydata$pk_id<-1:nrow(mydata)
> head(mydata)
  NClient        Activite NCommande DateDeCommande             Article Quantite
1     100       Assurances         1     03.01.2000 Compaq Presario 100       12
2     123 Machines/Outils          2     03.01.2000             IBM 500        2
3     109        Éducation         3     03.01.2000       AST Intel 150        5
4     104        Éducation         4     03.01.2000       AST Intel 200        3
5     117          Banques         5     04.01.2000 Compaq Presario 100       13
6     103       Assurances         6     04.01.2000       AST Intel 150        2
  PrixParPiece Rabais PrixTotalAvecRabais FacturePayee pk_id
1         1650  1.50%            19503.00          Oui     1
2         2299  0.00%             4598.00          Oui     2
3         2690  0.00%            13450.00          Oui     3
4         3190  0.00%             9570.00          Oui     4
5         1650  1.50%            21128.25          Oui     5
6         2690  0.00%             5380.00          Oui     6
> 
```

Exemple 81.: Créer une colonne de rangs

R 3.6.3

Considérons le petit jeu de données mtcars:

```
> (mydata<-mtcars)
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

Considérons la requête évidente (mais dont la solution est non-triviale) consistant à ajouter une colonne des rangs des horse power (*hp*). Pour cela, nous utilisons la fonction **transform( )** avec la fonction **ave( )** qui fait un groupement par sous-ensemble:

```
> transform(mydata,rank=ave(hp,FUN=function(x) rank(-x,ties.method="first")))
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb rank
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4   19
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4   20
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1   26
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1   21
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2   11
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1   23
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4    3
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2   31
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2   25
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4   16
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4   17
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3    8
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3    9
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3   10
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4    7
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4    6
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4    5
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1   28
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2   32
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1   30
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1   24
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2   14
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2   15
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2   12
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1   29
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2   27
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2   18
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4    2
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6   13
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8    1
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2   22
>
```

Exemple 82.: Prendre un sous-ensemble (subset)

R 3.0.2

Un sujet dont la simplicité se passe de tout commentaire:

```
> setwd("c:/tmp/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=",")
> mydata[1:5,2:3]
         Activite NCommande
1      Assurances         1
2 Machines/Outils         2
3       Éducation         3
4       Éducation         4
5         Banques         5
> colonnes<-c("Activite","NCommande")
> mydata[1:5,colonnes]
         Activite NCommande
1      Assurances         1
2 Machines/Outils         2
3       Éducation         3
4       Éducation         4
5         Banques         5
> 
```

Exemple 83.: Appliquer un filtre

R 3.0.2

Donc là évidemment on peut jouer avec énormément de combinaisons mais dans l'idée voici un exemple de base avec filtres multiples avec des conditions ET ainsi que OU en utilisant la commande **which( )** :

```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide

> setwd("C:/")
> mydata=read.csv("VentesClean.csv",header=T,sep=";")
> mydata[ which((mydata$NClient>100 & mydata$NClient<110)
+ & (mydata$Article=="AST Intel 200" | mydata$Article=="IBM 500")), ]
    NClient         Activite NDeCommande DateDeCommande       Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
4       104        Éducation           4     03.01.2000 AST Intel 200        3         3190  0.00%             9570.00          Oui
7       104        Éducation           7     04.01.2000 AST Intel 200        2         3190  0.00%             6380.00          Oui
18      106     Construction          18     05.01.2000 AST Intel 200       11         3190  1.50%            34563.65          Oui
35      101     Construction          35     11.01.2000 AST Intel 200        6         3190  1.50%            18852.90          Oui
38      106     Construction          38     11.01.2000       IBM 500        4         2299  0.00%             9196.00          Oui
41      101     Construction          41     12.01.2000 AST Intel 200        1         3190  0.00%             3190.00          Oui
55      103       Assurances          55     18.01.2000       IBM 500        7         2299  1.50%            15851.60          Oui
59      105          Banques          59     18.01.2000       IBM 500        9         2299  1.50%            20380.63          Oui
64      104        Éducation          64     19.01.2000 AST Intel 200        3         3190  0.00%             9570.00          Oui
71      101     Construction          71     19.01.2000       IBM 500        5         2299  0.00%            11495.00          Non
73      108   Pharmaceutique          73     20.01.2000 AST Intel 200       13         3190  1.50%            40847.95          Non
76      101     Construction          76     21.01.2000 AST Intel 200       10         3190  1.50%            31421.50          Non
84      108   Pharmaceutique          84     24.01.2000 AST Intel 200       22         3190  3.00%            68074.60          Non
87      105          Banques          87     25.01.2000       IBM 500        6         2299  1.50%            13587.09          Non
90      102  Machines/Outils          90     26.01.2000       IBM 500        7         2299  1.50%            15851.60          Non
94      106     Construction          94     27.01.2000 AST Intel 200        5         3190  0.00%            15950.00          Non
96      103       Assurances          96     27.01.2000       IBM 500        5         2299  0.00%            11495.00          Non
101     104        Éducation         101     28.01.2000       IBM 500        2         2299  0.00%             4598.00          Non
107     102  Machines/Outils         107     31.01.2000 AST Intel 200        4         3190  0.00%            12760.00          Non
> |
```

Attention!!!!! Notez bien que les index de ligne ne sont pas renumérotés!!!! Si vous souhaitez réinitialiser la numérotation, vous devrez utiliser le type de commande suivante:

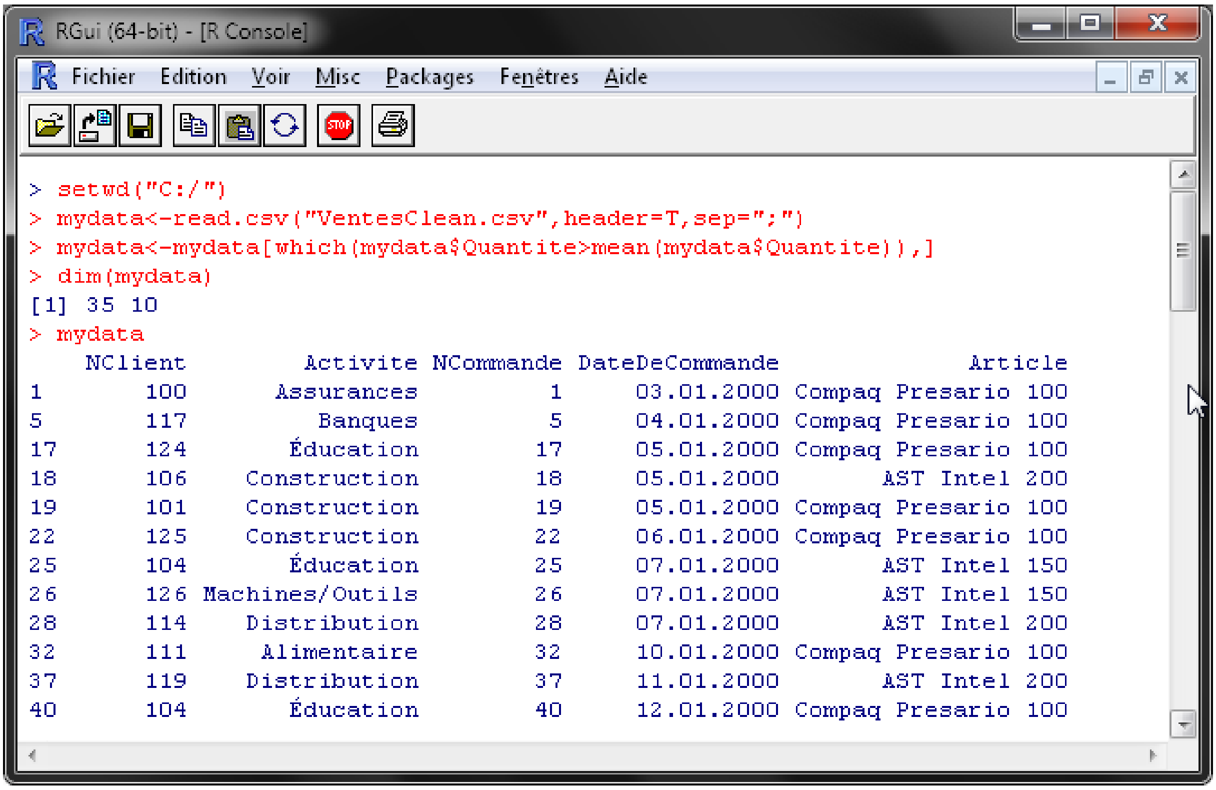
```
rownames(mydata) <- NULL
```

La commande **which( )** permet aussi plus simplement de savoir les numéros des lignes où pour une colonne donnée, certaines valeurs sont satisfaites:

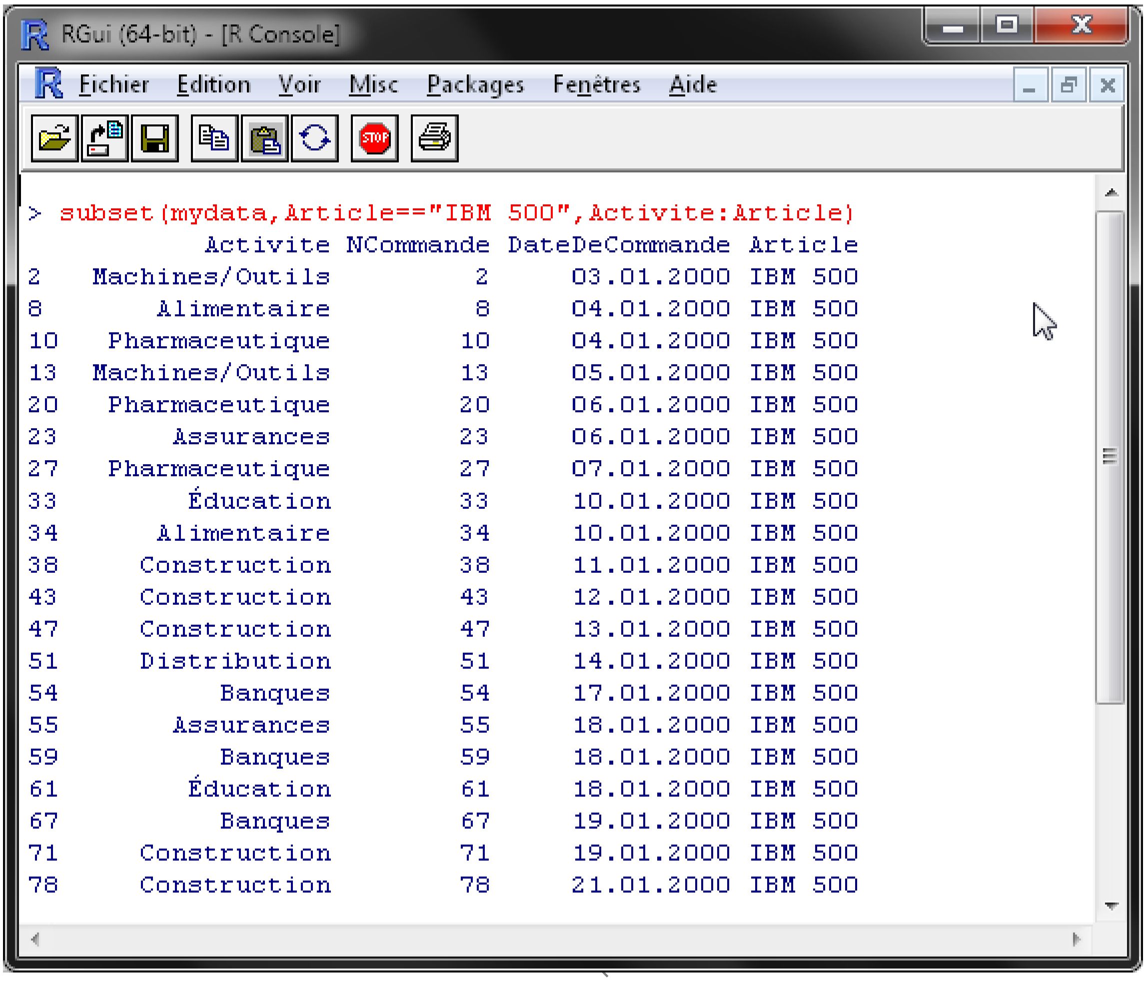
```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide

> setwd("C:/")
> mydata=read.csv("VentesClean.csv",header=T,sep=";")
> which(mydata$Quantite==3)
[1]   4  14  23  33  53  64 104 105 108
> |
```

Nous pouvons utiliser aussi **which( )** conjointement avec des fonctions:



Indiquons aussi la commande **subset( )** qui permet de filtrer aussi et que l'on retrouve dans de nombreux scripts R de data mining:



Ou encore la commande **%in%** que l'on retrouve dans certains scripts de Finance:

```
> mydata[mydata$Article %in% c("IBM 500"),1:5]
   NClient          Activite NCommande DateDeCommande Article
2      123 Machines/Outils         2     03.01.2000 IBM 500
8      111     Alimentaire         8     04.01.2000 IBM 500
10     116   Pharmaceutique       10     04.01.2000 IBM 500
13     123 Machines/Outils        13     05.01.2000 IBM 500
20     116   Pharmaceutique       20     06.01.2000 IBM 500
23     100       Assurances       23     06.01.2000 IBM 500
27     121   Pharmaceutique       27     07.01.2000 IBM 500
33     118        Éducation       33     10.01.2000 IBM 500
34     127      Alimentaire       34     10.01.2000 IBM 500
38     106     Construction       38     11.01.2000 IBM 500
43     113     Construction       43     12.01.2000 IBM 500
47     113     Construction       47     13.01.2000 IBM 500
51     119     Distribution       51     14.01.2000 IBM 500
54     117          Banques       54     17.01.2000 IBM 500
55     103       Assurances       55     18.01.2000 IBM 500
59     105          Banques       59     18.01.2000 IBM 500
61     118        Éducation       61     18.01.2000 IBM 500
67     120          Banques       67     19.01.2000 IBM 500
```

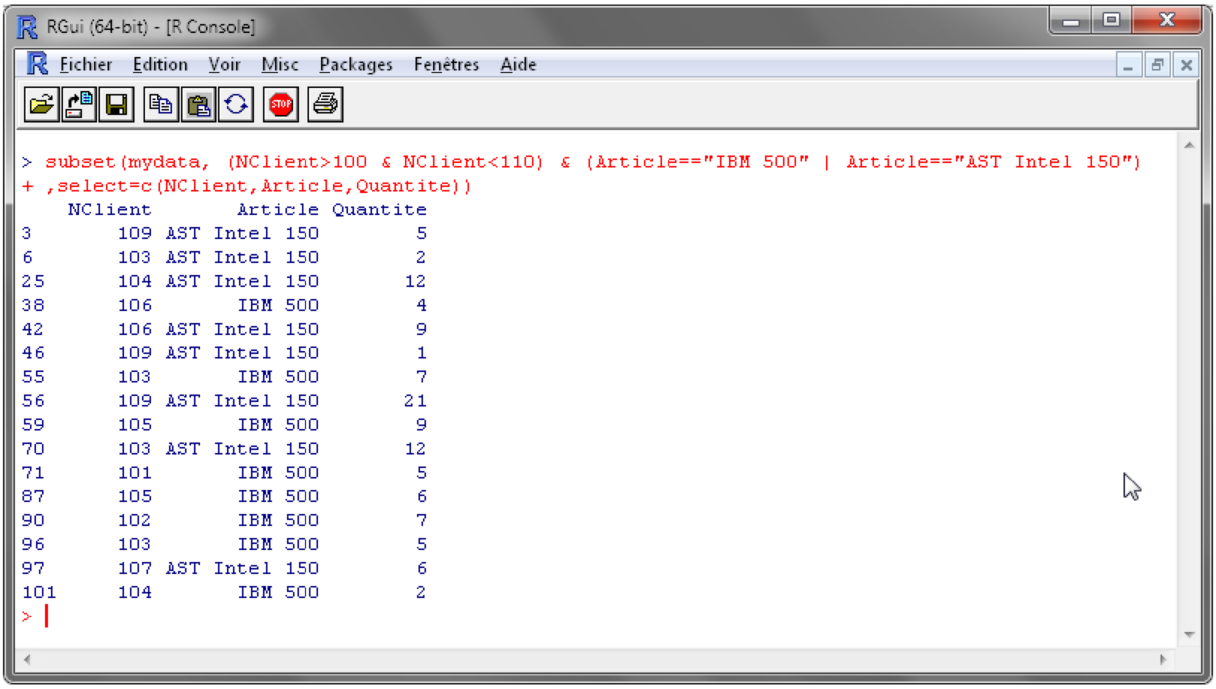
ou lorsque l'on veut utiliser un pattern de filtrage en utilisant la commande **grep( )** :

```
> grep("IBM",mydata$Article)
 [1]   2   8  10  13  20  23  27  33  34  38  43  47  51  54  55
[16]  59  61  67  71  78  87  90  95  96 101 104 108
> mydata[grep("IBM",mydata$Article),1:5]
   NClient          Activite NCommande DateDeCommande Article
2      123 Machines/Outils         2     03.01.2000 IBM 500
8      111     Alimentaire         8     04.01.2000 IBM 500
10     116   Pharmaceutique       10     04.01.2000 IBM 500
13     123 Machines/Outils        13     05.01.2000 IBM 500
20     116   Pharmaceutique       20     06.01.2000 IBM 500
23     100       Assurances       23     06.01.2000 IBM 500
27     121   Pharmaceutique       27     07.01.2000 IBM 500
33     118        Éducation       33     10.01.2000 IBM 500
34     127      Alimentaire       34     10.01.2000 IBM 500
38     106     Construction       38     11.01.2000 IBM 500
43     113     Construction       43     12.01.2000 IBM 500
47     113     Construction       47     13.01.2000 IBM 500
51     119     Distribution       51     14.01.2000 IBM 500
54     117          Banques       54     17.01.2000 IBM 500
55     103       Assurances       55     18.01.2000 IBM 500
```

Exemple 84.: Prendre un sous ensemble filtré

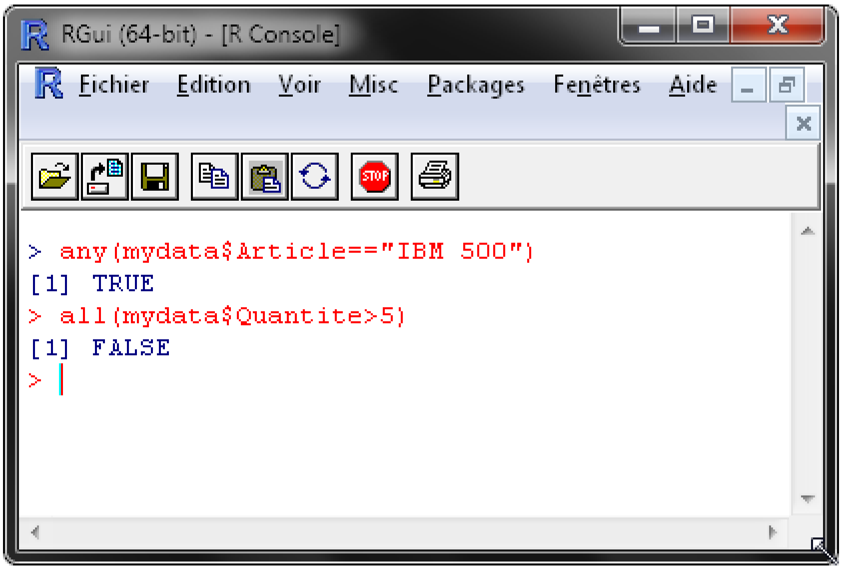
R 3.0.2

La manip est plutôt simple et ce passe probablement de commentaires:



Toujours au niveau des commandes élémentaires de filtrage, voyons les commandes **all( )** et **any( )** :

- **any( )** renvoie une valeur booléenne pour signaler si au moins une valeur correspond au critère dans le vecteur choisi.
- **all( )** renvoie une valeur booléenne pour signaler si toutes les valeurs sans aucune exception satisfont un certain critère ou non.



Exemple 85.: Utiliser les techniques de filtrage pour trouver la valeur proche

R 3.0.2

Toujours avec notre data frame habituel, nous souhaitons savoir quelle est le prix total avec rabais le plus proche de 75'000.

La technique est maligne... il fallait y penser:



c'est pas beau ça? Simple et efficace!

Exemple 86.: Échantillonnages

R 3.0.2

Avant que l'on s'attaque sérieusement au sujet, considérons la sympathique et ludique animation suivante pour un échantillonnage de 15 points parmi 100:

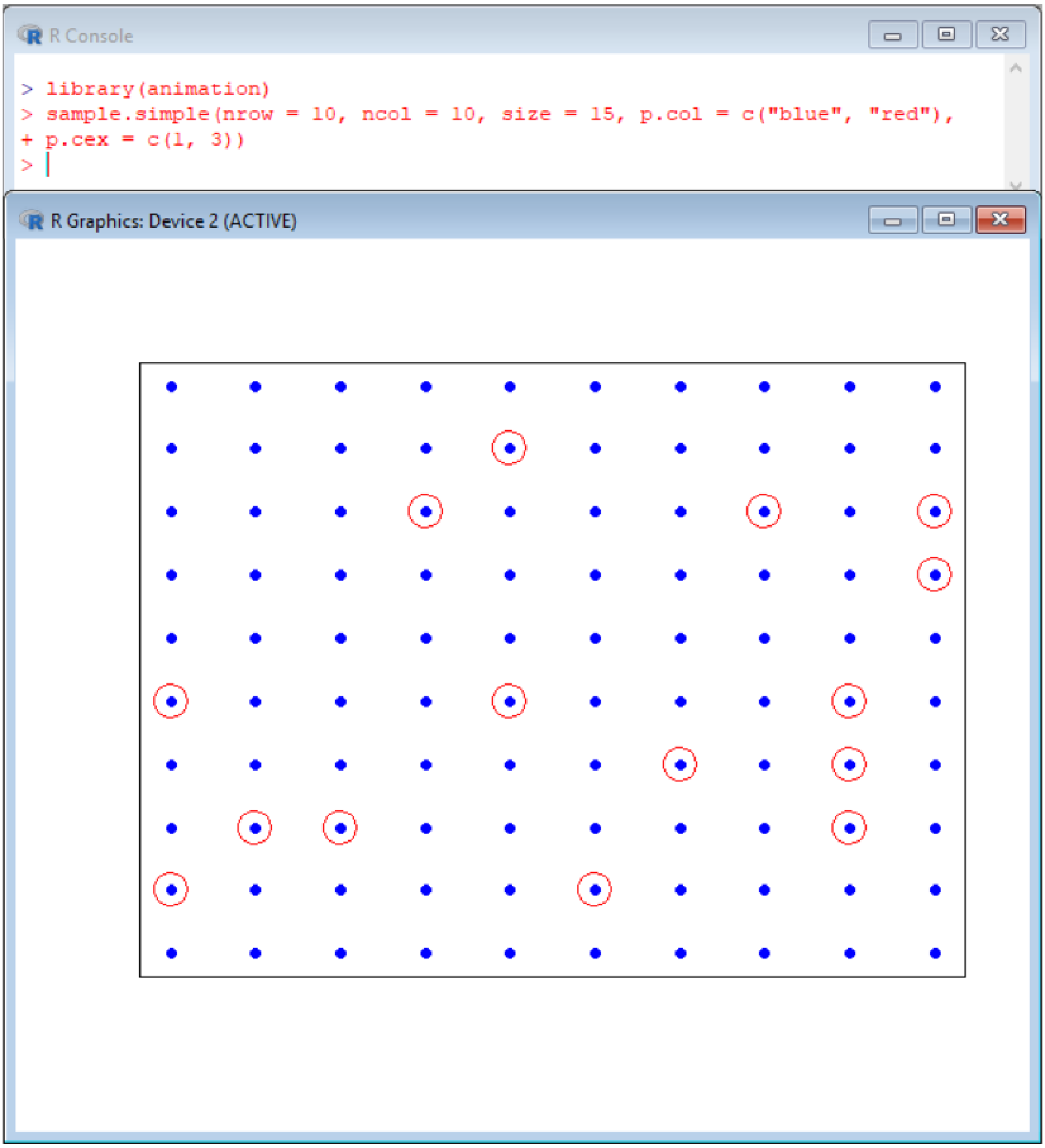


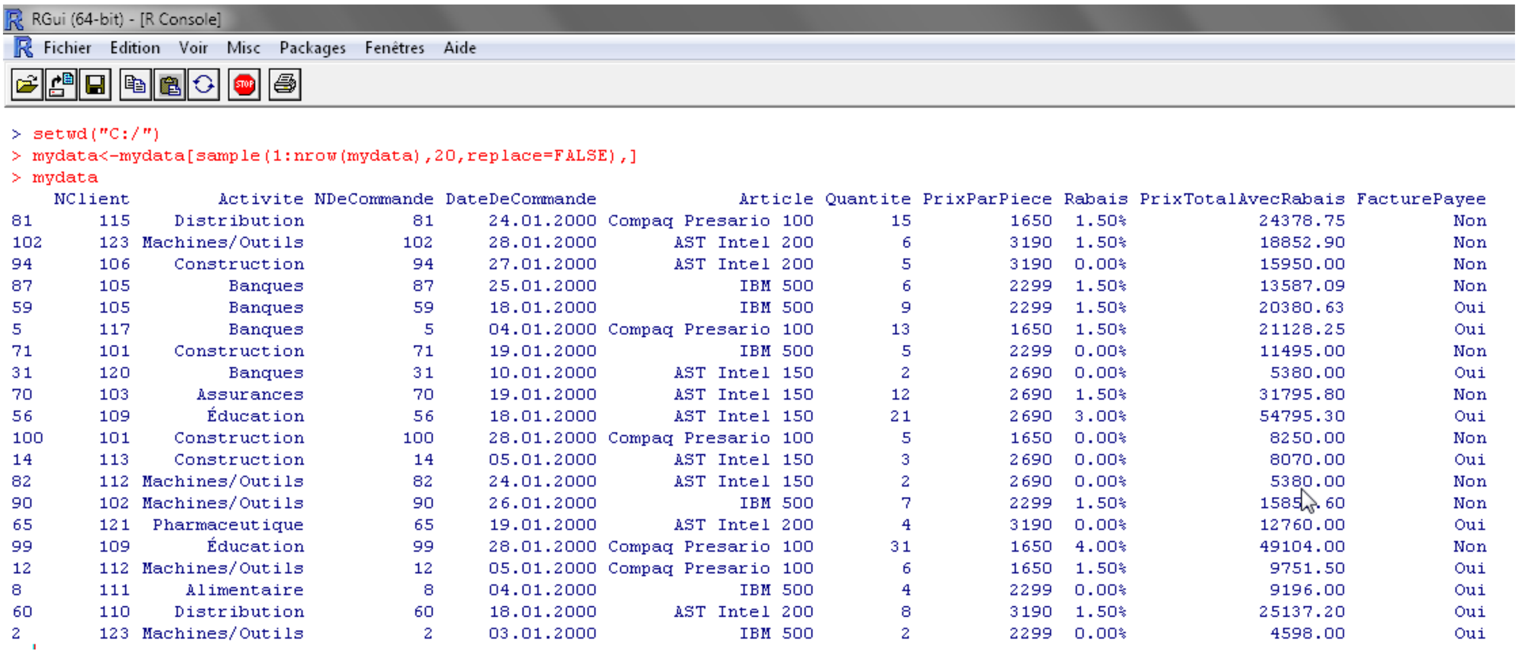
Figure 27 Graphe d'échantillonnage aléatoire

Ce fait, passons à des choses plus sérieuses…

Échantillonnage aléatoire simple

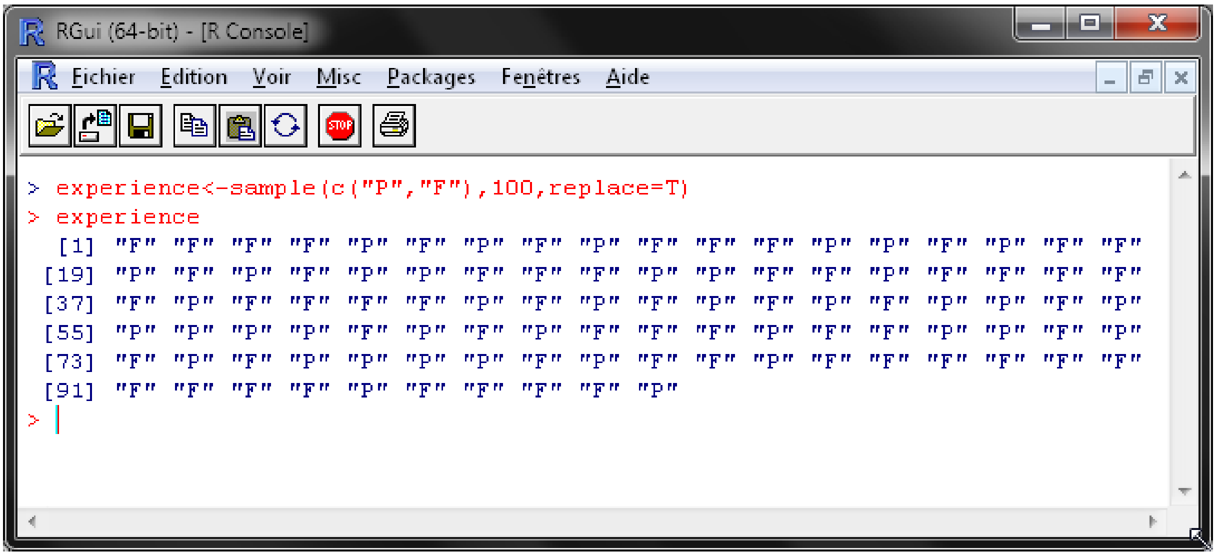
C'est un cas très important dans la pratique et surtout dans le cas industriel où il y a rarement des strates dans une ligne de production! Il s'agit simplement pour rappel de choisir des individus de telle sorte que chaque membre de la population a une chance égale de figurer dans l'échantillon.

Voyons cela en utilisant les commandes **sample( )** et les paramètres *nrow* et *replace* (dans le cas ci-dessous nous interdisons à un même individu d'apparaître deux fois dans le tirage):



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> setwd("C:/")
> mydata<-mydata[sample(1:nrow(mydata),20,replace=FALSE),]
> mydata
    NClient        Activite NDeCommande DateDeCommande          Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
81      115    Distribution          81     24.01.2000 Compaq Presario 100       15         1650  1.50%            24378.75          Non
102     123 Machines/Outils         102     28.01.2000    AST Intel 200        6         3190  1.50%            18852.90          Non
94      106    Construction          94     27.01.2000    AST Intel 200        5         3190  0.00%            15950.00          Non
87      105         Banques          87     25.01.2000          IBM 500        6         2299  1.50%            13587.09          Non
59      105         Banques          59     18.01.2000          IBM 500        9         2299  1.50%            20380.63          Oui
5       117         Banques           5     04.01.2000 Compaq Presario 100       13         1650  1.50%            21128.25          Oui
71      101    Construction          71     19.01.2000          IBM 500        5         2299  0.00%            11495.00          Non
31      120         Banques          31     10.01.2000    AST Intel 150        2         2690  0.00%             5380.00          Oui
70      103      Assurances          70     19.01.2000    AST Intel 150       12         2690  1.50%            31795.80          Non
56      109       Éducation          56     18.01.2000    AST Intel 150       21         2690  3.00%            54795.30          Oui
100     101    Construction         100     28.01.2000 Compaq Presario 100        5         1650  0.00%             8250.00          Non
14      113    Construction          14     05.01.2000    AST Intel 150        3         2690  0.00%             8070.00          Oui
82      112 Machines/Outils          82     24.01.2000    AST Intel 150        2         2690  0.00%             5380.00          Non
90      102 Machines/Outils          90     26.01.2000          IBM 500        7         2299  1.50%            1585[illegible].60          Non
65      121  Pharmaceutique          65     19.01.2000    AST Intel 200        4         3190  0.00%            12760.00          Oui
99      109       Éducation          99     28.01.2000 Compaq Presario 100       31         1650  4.00%            49104.00          Non
12      112 Machines/Outils          12     05.01.2000 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
8       111     Alimentaire           8     04.01.2000          IBM 500        4         2299  0.00%             9196.00          Oui
60      110    Distribution          60     18.01.2000    AST Intel 200        8         3190  1.50%            25137.20          Oui
2       123 Machines/Outils           2     03.01.2000          IBM 500        2         2299  0.00%             4598.00          Oui
```

La commande **sample( )** peut être aussi utilisée pour générer des vecteurs de textes (utile pour les arbres binomiaux par exemple):



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> experience<-sample(c("P","F"),100,replace=T)
> experience
  [1] "F" "F" "F" "F" "P" "F" "P" "F" "P" "F" "F" "F" "P" "P" "F" "P" "F" "F"
 [19] "P" "F" "P" "F" "P" "P" "F" "F" "F" "P" "P" "F" "F" "P" "F" "F" "F" "F"
 [37] "F" "P" "F" "F" "F" "F" "P" "F" "P" "F" "P" "F" "P" "F" "P" "P" "F" "P"
 [55] "P" "P" "P" "P" "F" "P" "F" "P" "F" "F" "F" "P" "F" "F" "P" "P" "F" "P"
 [73] "F" "P" "F" "P" "P" "P" "P" "F" "P" "F" "F" "P" "F" "F" "F" "F" "F" "F"
 [91] "F" "F" "F" "F" "P" "F" "F" "F" "F" "P"
>
```

Nous avons également du package **sampling** la fonction **srswor( )** qui correspond à ce que nous avons vu dans le cours théorique:

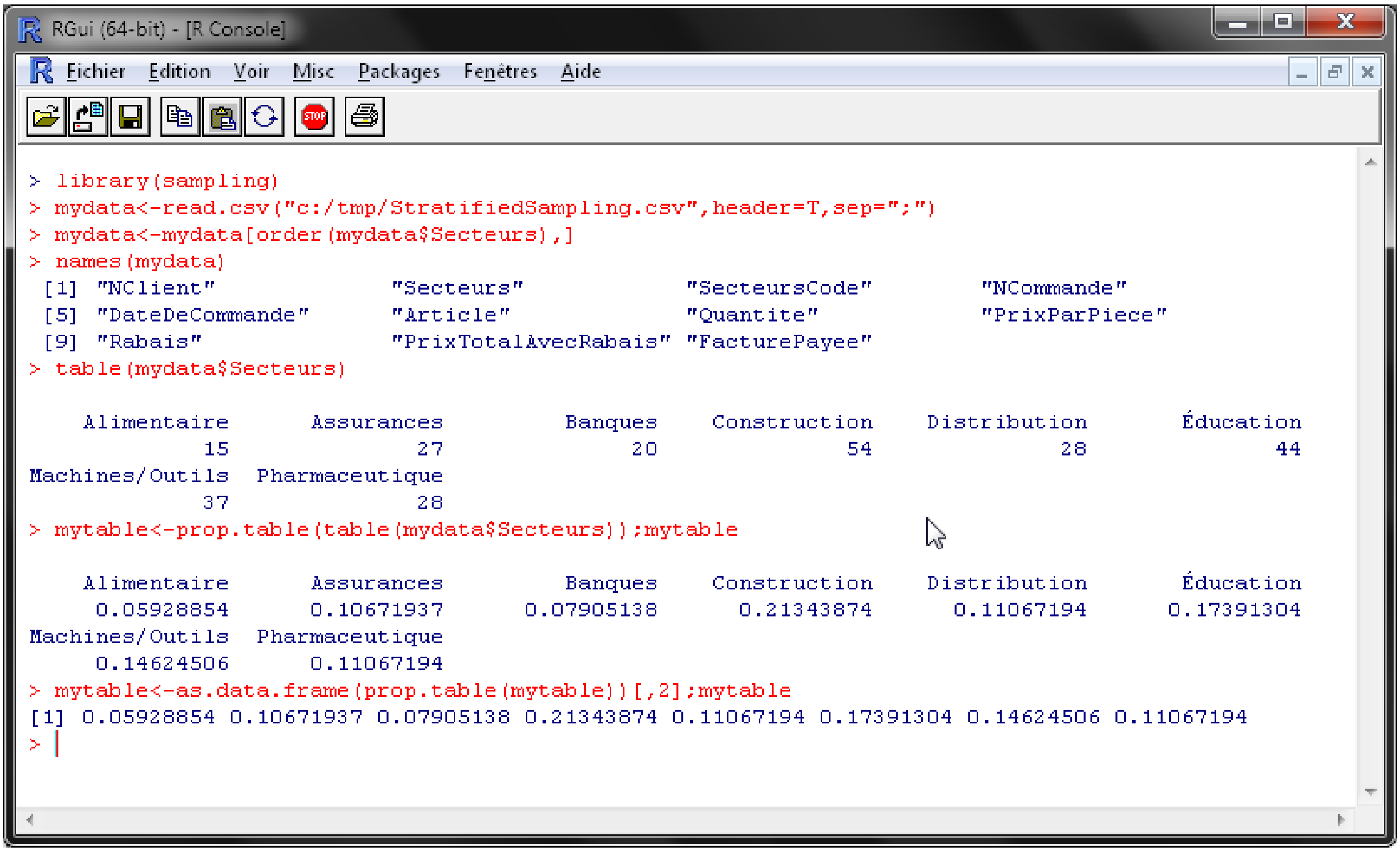
```
> library(sampling)
> mydata<-read.csv("C:/tmp/VentesClean.csv",header=T,sep=";")
> #srswor signifie Simple Random Sampling WithOut Replacement
> sample<-srswor(20, nrow(mydata))
> mydata[as.logical(sample),]
    NClient          Activite NCommande DateDeCommande           Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
4       104         Éducation         4     03.01.2000     AST Intel 200        3         3190  0.00%             9570.00          Oui
8       111       Alimentaire         8     04.01.2000           IBM 500        4         2299  0.00%             9196.00          Oui
12      112 Machines/Outils          12     05.01.2000 Compaq Presario 100      6         1650  1.50%             9751.50          Oui
15      115      Distribution        15     05.01.2000 Compaq Presario 100      8         1650  1.50%            13002.00          Oui
16      124         Éducation        16     05.01.2000     AST Intel 200        8         3190  1.50%            25137.20          Oui
33      118         Éducation        33     10.01.2000           IBM 500        3         2299  0.00%             6897.00          Oui
35      101      Construction        35     11.01.2000     AST Intel 200        6         3190  1.50%            18852.90          Oui
45      100        Assurances        45     12.01.2000     AST Intel 200        4         3190  0.00%            12760.00          Oui
60      110      Distribution        60     18.01.2000     AST Intel 200        8         3190  1.50%            25137.20          Oui
73      108    Pharmaceutique        73     20.01.2000     AST Intel 200       13         3190  1.50%            40847.95          Non
76      101      Construction        76     21.01.2000     AST Intel 200       10         3190  1.50%            31421.50          Non
82      112 Machines/Outils          82     24.01.2000     AST Intel 150        2         2690  0.00%             5380.00          Non
89      122      Distribution        89     26.01.2000     AST Intel 200        8         3190  1.50%            25137.20          Non
94      106      Construction        94     27.01.2000     AST Intel 200        5         3190  0.00%            15950.00          Non
96      103        Assurances        96     27.01.2000           IBM 500        5         2299  0.00%            11495.00          Non
100     101      Construction       100     28.01.2000 Compaq Presario 100      5         1650  0.00%             8250.00          Non
102     123 Machines/Outils         102     28.01.2000     AST Intel 200        6         3190  1.50%            18852.90          Non
103     106      Construction       103     28.01.2000 Compaq Presario 100     11         1650  1.50%            17877.75          Non
108     118         Éducation       108     31.01.2000           IBM 500        3         2299  0.00%             6897.00          Non
109     119         Éducation       109     31.01.2000     AST Intel 150        2         2690  0.00%             5380.00          Non
>
```

Échantillonnage stratifié (probabilités proportionnelles)

Pour cet exemple nous allons utiliser un autre fichier mais contenant exactement le même type de données.

Commençons d'abord par un échantillonnage stratifié qui n'est pas à probabilité proportionnelles afin de vérifier le système de fonctionnement de la fonction **strata( )** du package **sampling**.

D'abord nous faisons un résumé simple des secteurs d'activité du jeu de données avec lequel nous allons travailler et ce en n'utilisant aucune commande du package **sampling**:



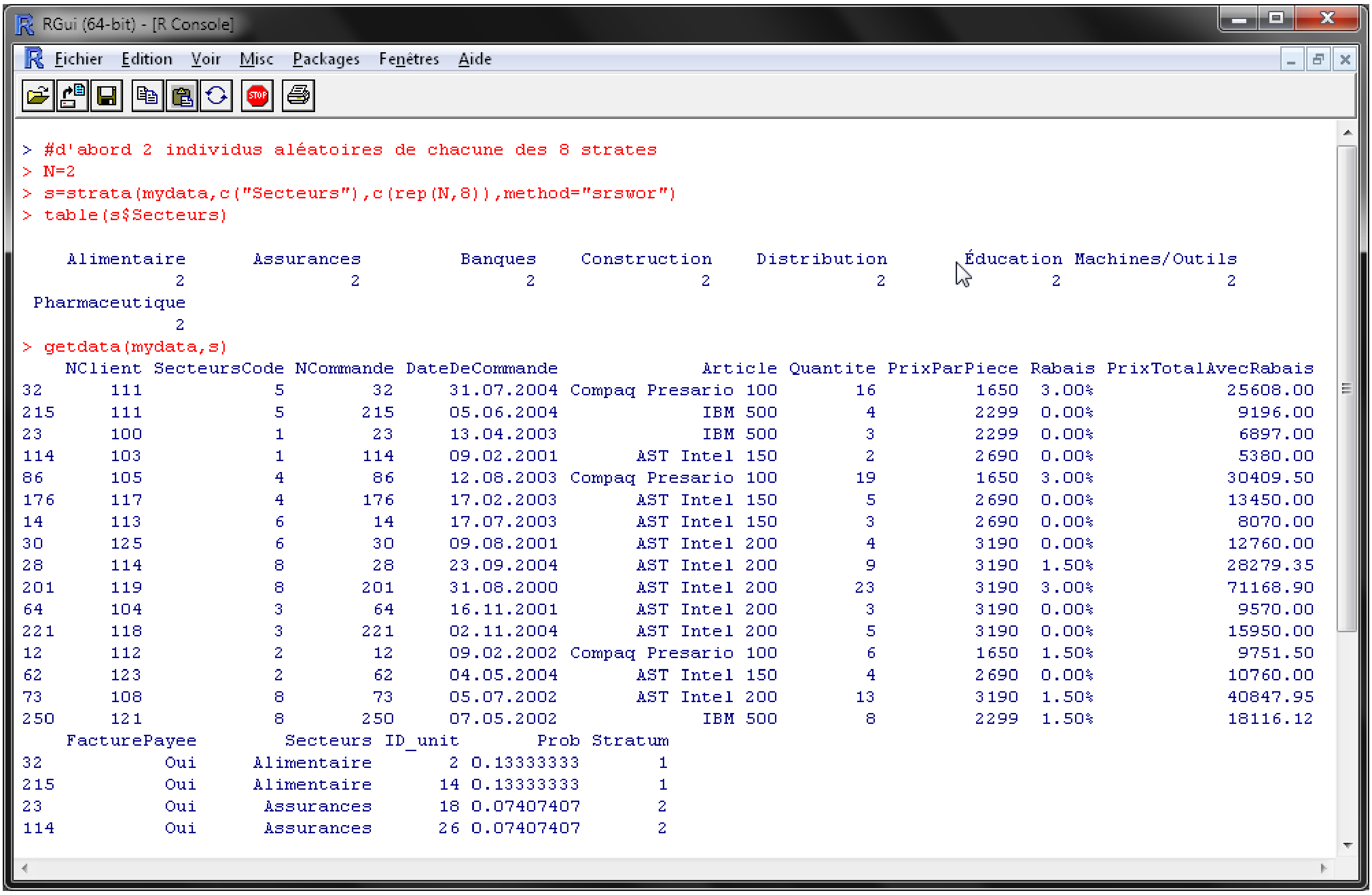
```
> library(sampling)
> mydata<-read.csv("c:/tmp/StratifiedSampling.csv",header=T,sep=";")
> mydata<-mydata[order(mydata$Secteurs),]
> names(mydata)
 [1] "NClient"          "Secteurs"             "SecteursCode"     "NCommande"
 [5] "DateDeCommande"   "Article"              "Quantite"         "PrixParPiece"
 [9] "Rabais"           "PrixTotalAvecRabais" "FacturePayee"
> table(mydata$Secteurs)

    Alimentaire      Assurances         Banques    Construction    Distribution       Éducation
             15              27              20              54              28              44
Machines/Outils  Pharmaceutique
             37              28
> mytable<-prop.table(table(mydata$Secteurs));mytable

    Alimentaire      Assurances         Banques    Construction    Distribution       Éducation
     0.05928854      0.10671937      0.07905138      0.21343874      0.11067194      0.17391304
Machines/Outils  Pharmaceutique
     0.14624506      0.11067194
> mytable<-as.data.frame(prop.table(mytable))[,2];mytable
[1] 0.05928854 0.10671937 0.07905138 0.21343874 0.11067194 0.17391304 0.14624506 0.11067194
>
```

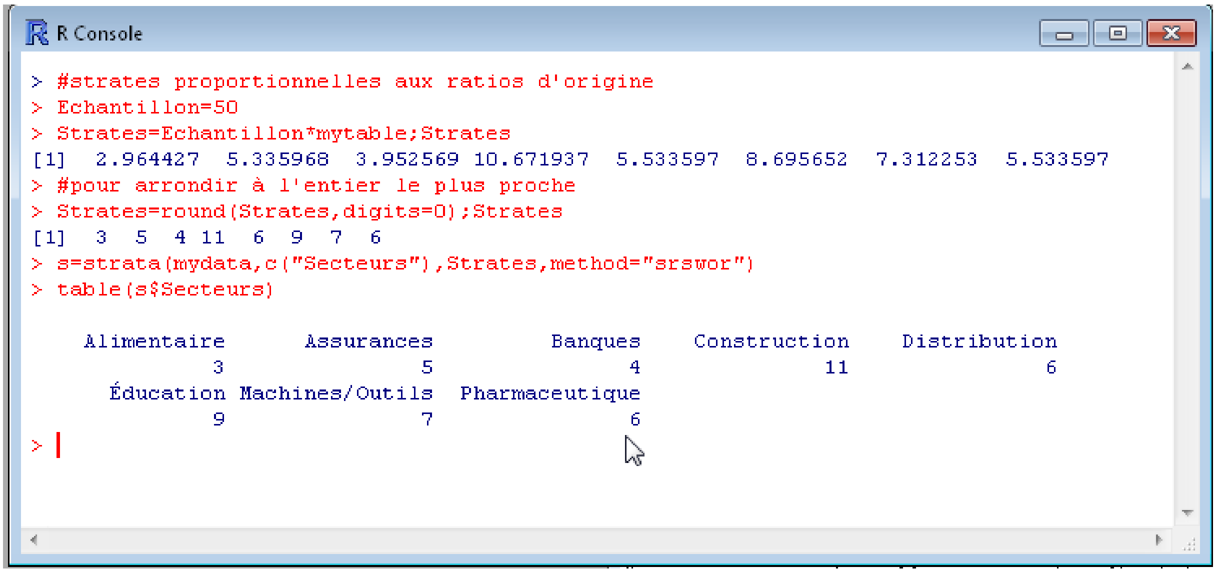
Nous avons donc les proportions représentatives de chaque strate des *Secteurs* sur l'ensemble de la population.

Prenons maintenant un même nombre d'individus de chacune des strates (il ne s'agit donc pas là d'un échantillonnage stratifié à probabilités proportionnelles mais d'un échantillonnage balancé):



Nous entrevoyons ainsi bien le concept de syntaxe de la fonction **strata( )**.

Maintenant faisons vraiment un échantillonnage stratifié à probabilités égales:



Et nous avons donc notre échantillon stratifié à probabilité égales:

R Console

```
> getdata(mydata,s)
    NClient SecteursCode NCommande DateDeCommande              Article Quantite PrixParPiece Rabais
8       111            5         8     05.06.2004              IBM 500        4         2299  0.00%
80      111            5        80     06.04.2000        AST Intel 200        2         3190  0.00%
208     127            5       208     27.08.2002        AST Intel 200        2         3190  0.00%
23      100            1        23     13.04.2003              IBM 500        3         2299  0.00%
55      103            1        55     19.02.2002              IBM 500        7         2299  1.50%
137     103            1       137     10.12.2004 Compaq Presario 100        6         1650  1.50%
178     103            1       178     06.10.2001        AST Intel 150       12         2690  1.50%
252     103            1       252     10.12.2004 Compaq Presario 100        6         1650  1.50%
67      120            4        67     06.12.2003              IBM 500       12         2299  1.50%
68      117            4        68     17.02.2003        AST Intel 150        5         2690  0.00%
162     117            4       162     18.06.2001              IBM 500        5         2299  0.00%
175     120            4       175     06.12.2003              IBM 500       12         2299  1.50%
9       113            6         9     26.12.2001 Compaq Presario 100        4         1650  0.00%
14      113            6        14     17.07.2003        AST Intel 150        3         2690  0.00%
18      106            6        18     06.06.2001        AST Intel 200       11         3190  1.50%
43      113            6        43     23.03.2000              IBM 500        6         2299  1.50%
122     113            6       122     17.07.2003        AST Intel 150        3         2690  0.00%
127     101            6       127     24.08.2004 Compaq Presario 100       14         1650  1.50%
143     101            6       143     16.11.2000        AST Intel 200        6         3190  1.50%
150     106            6       150     29.01.2001        AST Intel 150        9         2690  1.50%
205     101            6       205     05.08.2002        AST Intel 200        1         3190  0.00%
207     113            6       207     23.03.2000              IBM 500        6         2299  1.50%
216     113            6       216     26.12.2001 Compaq Presario 100        4         1650  0.00%
51      119            8        51     29.07.2004              IBM 500        2         2299  0.00%
89      122            8        89     11.10.2000        AST Intel 200        8         3190  1.50%
```

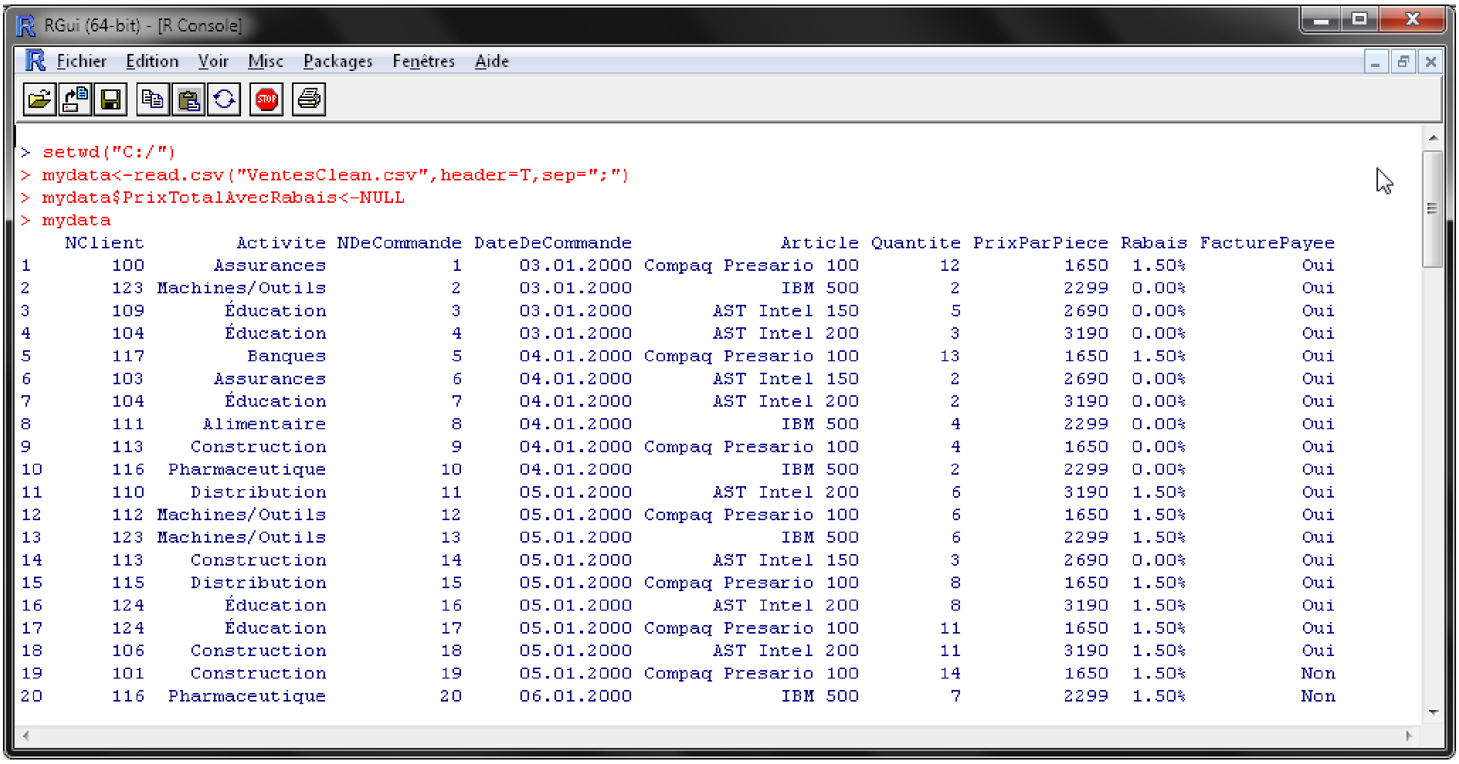
Sur et sous-échantillonnage

Exemple 87.: Éliminer des colonnes

R 3.0.2

Afin d'accélérer le traitement et l'analyse de data frame dans R il peut être judicieux de savoir supprimer des colonnes.

Voici une première méthode:



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata$PrixTotalAvecRabais<-NULL
> mydata
   NClient       Activite NDeCommande DateDeCommande           Article Quantite PrixParPiece Rabais FacturePayee
1      100     Assurances           1     03.01.2000 Compaq Presario 100       12         1650  1.50%          Oui
2      123 Machines/Outils          2     03.01.2000           IBM 500        2         2299  0.00%          Oui
3      109      Éducation           3     03.01.2000     AST Intel 150        5         2690  0.00%          Oui
4      104      Éducation           4     03.01.2000     AST Intel 200        3         3190  0.00%          Oui
5      117        Banques           5     04.01.2000 Compaq Presario 100       13         1650  1.50%          Oui
6      103     Assurances           6     04.01.2000     AST Intel 150        2         2690  0.00%          Oui
7      104      Éducation           7     04.01.2000     AST Intel 200        2         3190  0.00%          Oui
8      111    Alimentaire           8     04.01.2000           IBM 500        4         2299  0.00%          Oui
9      113   Construction           9     04.01.2000 Compaq Presario 100        4         1650  0.00%          Oui
10     116 Pharmaceutique          10     04.01.2000           IBM 500        2         2299  0.00%          Oui
11     110   Distribution          11     05.01.2000     AST Intel 200        6         3190  1.50%          Oui
12     112 Machines/Outils         12     05.01.2000 Compaq Presario 100        6         1650  1.50%          Oui
13     123 Machines/Outils         13     05.01.2000           IBM 500        6         2299  1.50%          Oui
14     113   Construction          14     05.01.2000     AST Intel 150        3         2690  0.00%          Oui
15     115   Distribution          15     05.01.2000 Compaq Presario 100        8         1650  1.50%          Oui
16     124      Éducation          16     05.01.2000     AST Intel 200        8         3190  1.50%          Oui
17     124      Éducation          17     05.01.2000 Compaq Presario 100       11         1650  1.50%          Oui
18     106   Construction          18     05.01.2000     AST Intel 200       11         3190  1.50%          Oui
19     101   Construction          19     05.01.2000 Compaq Presario 100       14         1650  1.50%          Non
20     116 Pharmaceutique          20     06.01.2000           IBM 500        7         2299  1.50%          Non
```

Une deuxième méthode:

```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata[9]<-NULL
> mydata
   NClient        Activite NDeCommande DateDeCommande             Article Quantite PrixParPiece Rabais FacturePayee
1      100      Assurances           1     03.01.2000 Compaq Presario 100       12         1650  1.50%          Oui
2      123 Machines/Outils           2     03.01.2000             IBM 500        2         2299  0.00%          Oui
3      109       Éducation           3     03.01.2000       AST Intel 150        5         2690  0.00%          Oui
4      104       Éducation           4     03.01.2000       AST Intel 200        3         3190  0.00%          Oui
5      117         Banques           5     04.01.2000 Compaq Presario 100       13         1650  1.50%          Oui
6      103      Assurances           6     04.01.2000       AST Intel 150        2         2690  0.00%          Oui
7      104       Éducation           7     04.01.2000       AST Intel 200        2         3190  0.00%          Oui
8      111     Alimentaire           8     04.01.2000             IBM 500        4         2299  0.00%          Oui
9      113    Construction           9     04.01.2000 Compaq Presario 100        4         1650  0.00%          Oui
10     116  Pharmaceutique          10     04.01.2000             IBM 500        2         2299  0.00%          Oui
11     110    Distribution          11     05.01.2000       AST Intel 200        6         3190  1.50%          Oui
12     112 Machines/Outils          12     05.01.2000 Compaq Presario 100        6         1650  1.50%          Oui
13     123 Machines/Outils          13     05.01.2000             IBM 500        6         2299  1.50%          Oui
14     113    Construction          14     05.01.2000       AST Intel 150        3         2690  0.00%          Oui
15     115    Distribution          15     05.01.2000 Compaq Presario 100        8         1650  1.50%          Oui
16     124       Éducation          16     05.01.2000       AST Intel 200        8         3190  1.50%          Oui
17     124       Éducation          17     05.01.2000 Compaq Presario 100       11         1650  1.50%          Oui
18     106    Construction          18     05.01.2000       AST Intel 200       11         3190  1.50%          Oui
19     101    Construction          19     05.01.2000 Compaq Presario 100       14         1650  1.50%          Non
20     116  Pharmaceutique          20     06.01.2000             IBM 500        7         2299  1.50%          Non
```

Et une méthode pour enlever plusieurs colonnes d'un coup:

```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata[9:10]<-list(NULL)
> mydata
   NClient        Activite NDeCommande DateDeCommande             Article Quantite PrixParPiece Rabais
1      100      Assurances           1     03.01.2000 Compaq Presario 100       12         1650  1.50%
2      123 Machines/Outils           2     03.01.2000             IBM 500        2         2299  0.00%
3      109       Éducation           3     03.01.2000       AST Intel 150        5         2690  0.00%
4      104       Éducation           4     03.01.2000       AST Intel 200        3         3190  0.00%
5      117         Banques           5     04.01.2000 Compaq Presario 100       13         1650  1.50%
6      103      Assurances           6     04.01.2000       AST Intel 150        2         2690  0.00%
7      104       Éducation           7     04.01.2000       AST Intel 200        2         3190  0.00%
8      111     Alimentaire           8     04.01.2000             IBM 500        4         2299  0.00%
9      113    Construction           9     04.01.2000 Compaq Presario 100        4         1650  0.00%
10     116  Pharmaceutique          10     04.01.2000             IBM 500        2         2299  0.00%
11     110    Distribution          11     05.01.2000       AST Intel 200        6         3190  1.50%
12     112 Machines/Outils          12     05.01.2000 Compaq Presario 100        6         1650  1.50%
13     123 Machines/Outils          13     05.01.2000             IBM 500        6         2299  1.50%
14     113    Construction          14     05.01.2000       AST Intel 150        3         2690  0.00%
15     115    Distribution          15     05.01.2000 Compaq Presario 100        8         1650  1.50%
16     124       Éducation          16     05.01.2000       AST Intel 200        8         3190  1.50%
17     124       Éducation          17     05.01.2000 Compaq Presario 100       11         1650  1.50%
18     106    Construction          18     05.01.2000       AST Intel 200       11         3190  1.50%
19     101    Construction          19     05.01.2000 Compaq Presario 100       14         1650  1.50%
20     116  Pharmaceutique          20     06.01.2000             IBM 500        7         2299  1.50%
```

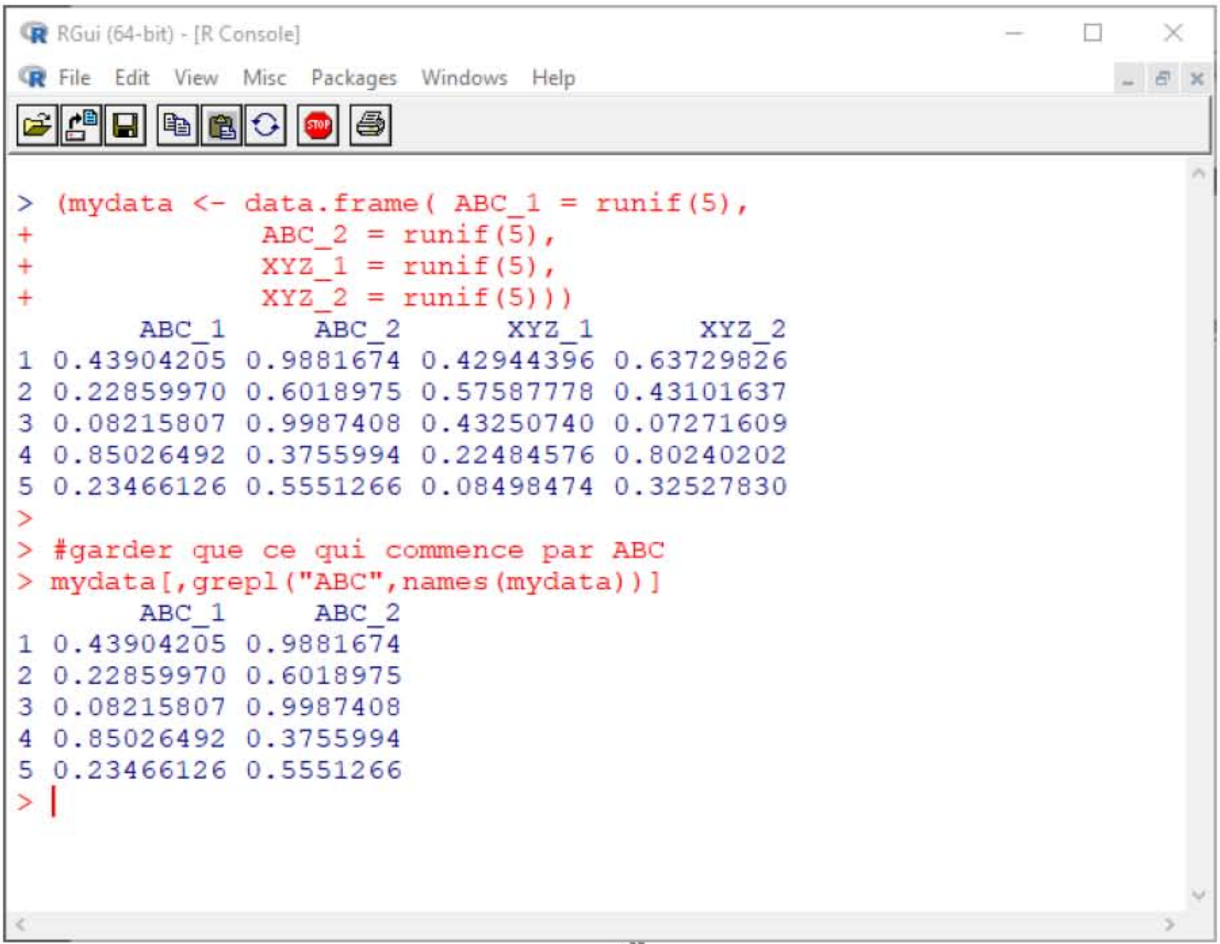
Et encore une dernière pour la route...:

```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata[,-(9:10)]
   NClient        Activite NCommande DateDeCommande             Article Quantite PrixParPiece Rabais
1      100      Assurances         1     03.01.2000 Compaq Presario 100       12         1650  1.50%
2      123 Machines/Outils         2     03.01.2000             IBM 500        2         2299  0.00%
3      109       Éducation         3     03.01.2000       AST Intel 150        5         2690  0.00%
4      104       Éducation         4     03.01.2000       AST Intel 200        3         3190  0.00%
5      117         Banques         5     04.01.2000 Compaq Presario 100       13         1650  1.50%
6      103      Assurances         6     04.01.2000       AST Intel 150        2         2690  0.00%
7      104       Éducation         7     04.01.2000       AST Intel 200        2         3190  0.00%
8      111     Alimentaire         8     04.01.2000             IBM 500        4         2299  0.00%
9      113    Construction         9     04.01.2000 Compaq Presario 100        4         1650  0.00%
10     116  Pharmaceutique        10     04.01.2000             IBM 500        2         2299  0.00%
11     110    Distribution        11     05.01.2000       AST Intel 200        6         3190  1.50%
12     112 Machines/Outils        12     05.01.2000 Compaq Presario 100        6         1650  1.50%
13     123 Machines/Outils        13     05.01.2000             IBM 500        6         2299  1.50%
14     113    Construction        14     05.01.2000       AST Intel 150        3         2690  0.00%
15     115    Distribution        15     05.01.2000 Compaq Presario 100        8         1650  1.50%
```

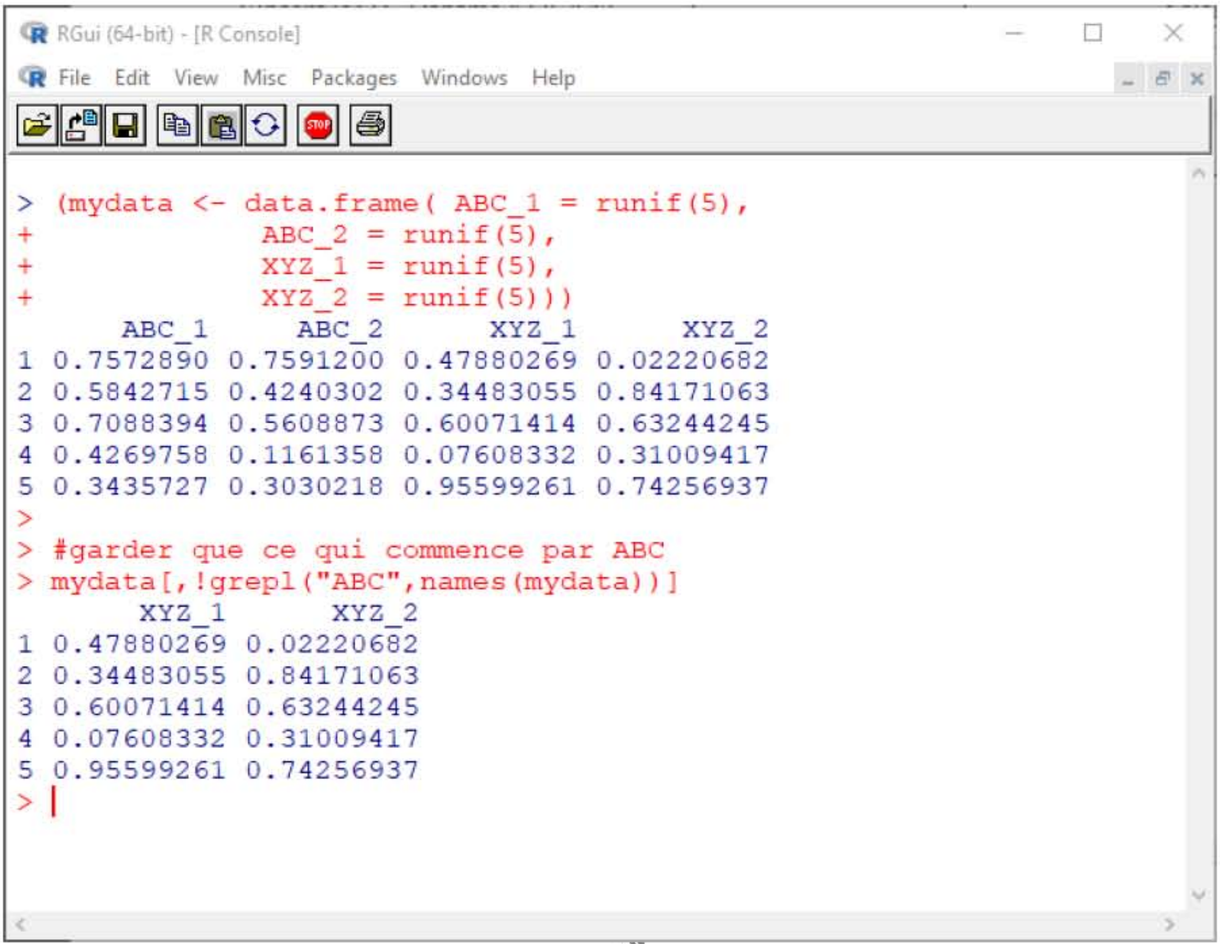
Et un cas classique consistant à éliminer toutes les colonnes qui ne sont pas numériques à l'aide de la fonction **Filter( )** du package natif **Base**:

```
> Filter(is.numeric, mydata)
   NClient NCommande Quantite PrixParPiece PrixTotalAvecRabais
1      100         1       12         1650            19503.00
2      123         2        2         2299             4598.00
3      109         3        5         2690            13450.00
4      104         4        3         3190             9570.00
5      117         5       13         1650            21128.25
6      103         6        2         2690             5380.00
7      104         7        2         3190             6380.00
8      111         8        4         2299             9196.00
9      113         9        4         1650             6600.00
10     116        10        2         2299             4598.00
11     110        11        6         3190            18852.90
12     112        12        6         1650             9751.50
13     123        13        6         2299            13587.09
14     113        14        3         2690             8070.00
15     115        15        8         1650            13002.00
16     124        16        8         3190            25137.20
17     124        17       11         1650            17877.75
18     106        18       11         3190            34563.65
19     101        19       14         1650            22753.50
20     116        20        7         2299            15851.60
21     112        21        6         2690            15897.90
22     125        22       23         1650            36811.50
23     100        23        3         2299             6897.00
24     125        24        2         3190             6380.00
25     104        25       12         2690            31795.80
26     126        26       24         2690            62623.20
27     121        27        8         2299            18116.12
28     114        28        9         3190            28279.35
29     103        29        6         1650             9751.50
30     125        30        4         3190            12760.00
31     120        31        2         2690             5380.00
32     111        32       16         1650            25608.00
33     118        33        3         2299             6897.00
34     127        34        7         2299            15851.60
```

Sinon pour garder que les colonnes qui commencent par une certaine chaîne de caractère, on utilisera la fonction **grepl( )**:



Inversement:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> (mydata <- data.frame(ABC_1 = runif(5),
+ ABC_2 = runif(5),
+ XYZ_1 = runif(5),
+ XYZ_2 = runif(5)))
ABC_1 ABC_2 XYZ_1 XYZ_2
1 0.7572890 0.7591200 0.47880269 0.02220682
2 0.5842715 0.4240302 0.34483055 0.84171063
3 0.7088394 0.5608873 0.60071414 0.63244245
4 0.4269758 0.1161358 0.07608332 0.31009417
5 0.3435727 0.3030218 0.95599261 0.74256937
>
> #garder que ce qui commence par ABC
> mydata[,!grepl("ABC",names(mydata))]
XYZ_1 XYZ_2
1 0.47880269 0.02220682
2 0.34483055 0.84171063
3 0.60071414 0.63244245
4 0.07608332 0.31009417
5 0.95599261 0.74256937
>

Exemple 88.: Tri simple et multiple (trier)

R 3.0.2

Et pour le tri des lignes voici comment faire en utilisant la commande **order( )** :

```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata[order(mydata$NClient,mydata$Article),]
    NClient        Activite NDeCommande DateDeCommande           Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
50      100      Assurances          50     13.01.2000     AST Intel 150       13         2690  1.50%            34445.45          Oui
45      100      Assurances          45     12.01.2000     AST Intel 200        4         3190  0.00%            12760.00          Oui
1       100      Assurances           1     03.01.2000 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
23      100      Assurances          23     06.01.2000           IBM 500        3         2299  0.00%             6897.00          Oui
35      101    Construction          35     11.01.2000     AST Intel 200        6         3190  1.50%            18852.90          Oui
41      101    Construction          41     12.01.2000     AST Intel 200        1         3190  0.00%             3190.00          Oui
76      101    Construction          76     21.01.2000     AST Intel 200       10         3190  1.50%            31421.50          Non
19      101    Construction          19     05.01.2000 Compaq Presario 100       14         1650  1.50%            22753.50          Non
69      101    Construction          69     19.01.2000 Compaq Presario 100       13         1650  1.50%            21128.25          Non
100     101    Construction         100     28.01.2000 Compaq Presario 100        5         1650  0.00%             8250.00          Non
71      101    Construction          71     19.01.2000           IBM 500        5         2299  0.00%            11495.00          Non
107     102 Machines/Outils         107     31.01.2000     AST Intel 200        4         3190  0.00%            12760.00          Non
52      102 Machines/Outils          52     14.01.2000 Compaq Presario 100        5         1650  0.00%             8250.00          Oui
90      102 Machines/Outils          90     26.01.2000           IBM 500        7         2299  1.50%            15851.60          Non
6       103      Assurances           6     04.01.2000     AST Intel 150        2         2690  0.00%             5380.00          Oui
70      103      Assurances          70     19.01.2000     AST Intel 150       12         2690  1.50%            31795.80          Non
29      103      Assurances          29     07.01.2000 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
74      103      Assurances          74     20.01.2000 Compaq Presario 100        8         1650  1.50%            13002.00          Non
55      103      Assurances          55     18.01.2000           IBM 500        7         2299  1.50%            15851.60          Oui
96      103      Assurances          96     27.01.2000           IBM 500        5         2299  0.00%            11495.00          Non
```

Il suffit d'ajouter le paramètre:

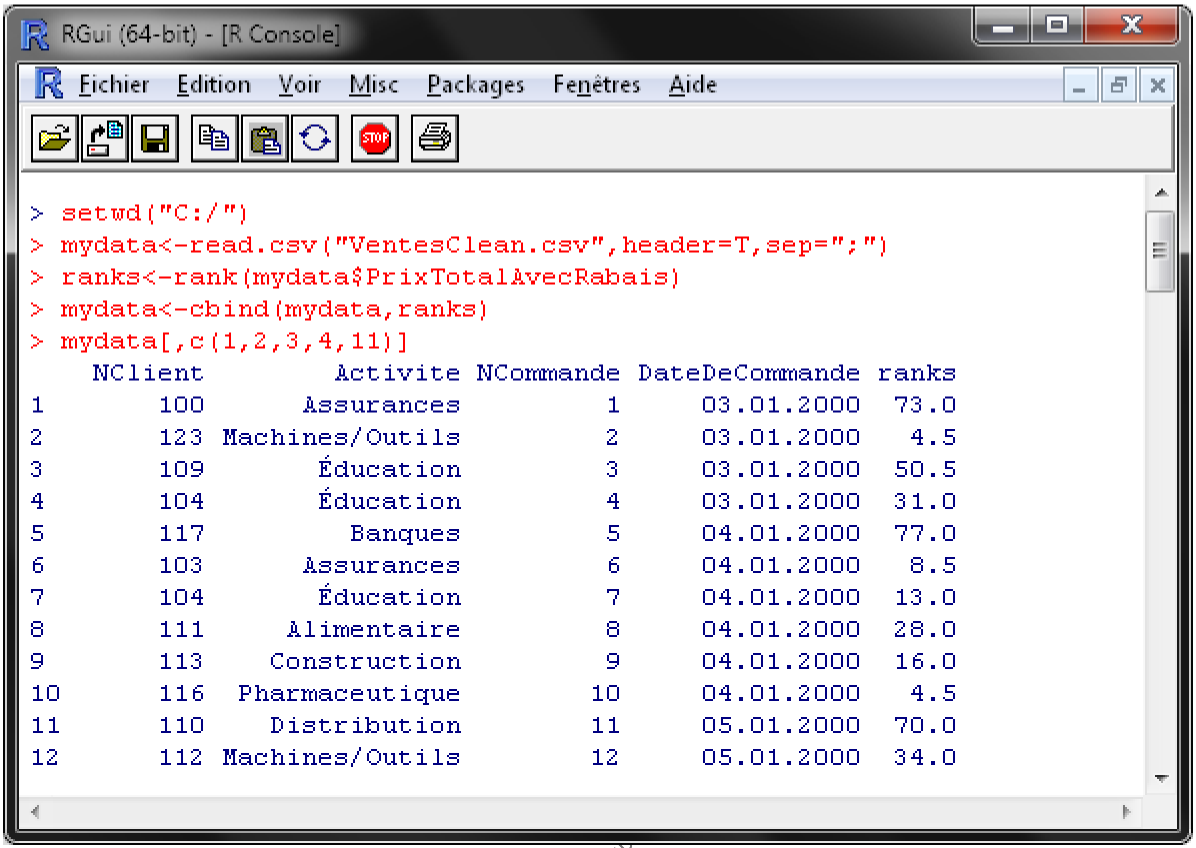
$$decreasing = TRUE$$

à la commande **order( )** pour avoir le tri dans l'ordre décroissant.

Pour faire deux tris (ou plus) dans un ordre différent, voici la syntaxe:

```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata[order(mydata$NClient,-mydata$Quantite),]
    NClient        Activite NCommande DateDeCommande           Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
50      100      Assurances        50     13.01.2000     AST Intel 150       13         2690  1.50%            34445.45          Oui
1       100      Assurances         1     03.01.2000 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
45      100      Assurances        45     12.01.2000     AST Intel 200        4         3190  0.00%            12760.00          Oui
23      100      Assurances        23     06.01.2000           IBM 500        3         2299  0.00%             6897.00          Oui
19      101    Construction        19     05.01.2000 Compaq Presario 100       14         1650  1.50%            22753.50          Non
69      101    Construction        69     19.01.2000 Compaq Presario 100       13         1650  1.50%            21128.25          Non
76      101    Construction        76     21.01.2000     AST Intel 200       10         3190  1.50%            31421.50          Non
35      101    Construction        35     11.01.2000     AST Intel 200        6         3190  1.50%            18852.90          Oui
71      101    Construction        71     19.01.2000           IBM 500        5         2299  0.00%            11495.00          Non
100     101    Construction       100     28.01.2000 Compaq Presario 100        5         1650  0.00%             8250.00          Non
41      101    Construction        41     12.01.2000     AST Intel 200        1         3190  0.00%             3190.00          Oui
90      102 Machines/Outils        90     26.01.2000           IBM 500        7         2299  1.50%            15851.60          Non
52      102 Machines/Outils        52     14.01.2000 Compaq Presario 100        5         1650  0.00%             8250.00          Oui
107     102 Machines/Outils       107     31.01.2000     AST Intel 200        4         3190  0.00%            12760.00          Non
70      103      Assurances        70     19.01.2000     AST Intel 150       12         2690  1.50%            31795.80          Non
74      103      Assurances        74     20.01.2000 Compaq Presario 100        8         1650  1.50%            13002.00          Non
```

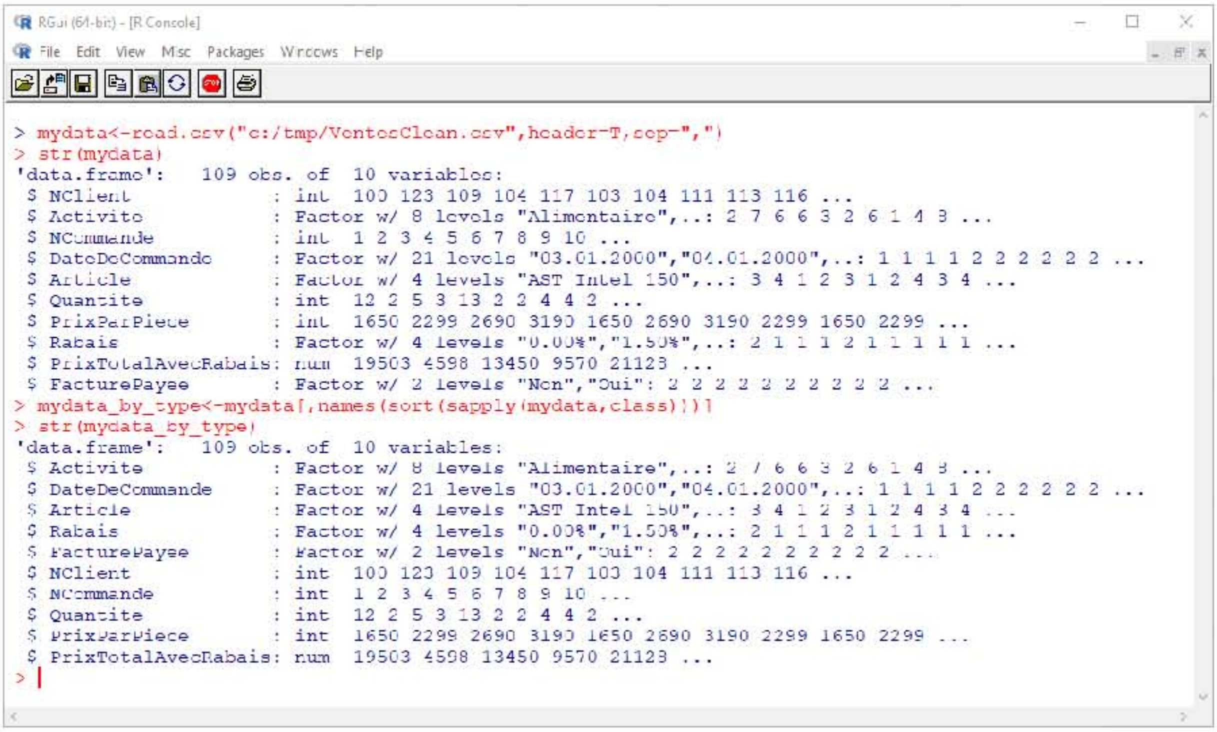
Et on peut s'amuser à faire un ranking avec la commande **rank( )** et ajouter cette colonne au data frame:



Et pour faire un tri sur mesure:

```
> mydata<-read.csv("C:/VentesClean.csv",header=T,sep=";")
> mydata<-mydata[order(factor(mydata$Article,
+ levels=c("IBM 500","AST Intel 200","Compaq Presario 100","AST Intel 150"
+ ))),]
> mydata[,1:5]
    NClient        Activite NCommande DateDeCommande             Article
2       123 Machines/Outils         2     03.01.2000             IBM 500
8       111     Alimentaire         8     04.01.2000             IBM 500
10      116  Pharmaceutique        10     04.01.2000             IBM 500
13      123 Machines/Outils        13     05.01.2000             IBM 500
20      116  Pharmaceutique        20     06.01.2000             IBM 500
23      100      Assurances        23     06.01.2000             IBM 500
27      121  Pharmaceutique        27     07.01.2000             IBM 500
33      118       Éducation        33     10.01.2000             IBM 500
34      127     Alimentaire        34     10.01.2000             IBM 500
38      106    Construction        38     11.01.2000             IBM 500
43      113    Construction        43     12.01.2000             IBM 500
47      113    Construction        47     13.01.2000             IBM 500
51      119    Distribution        51     14.01.2000             IBM 500
54      117         Banques        54     17.01.2000             IBM 500
55      103      Assurances        55     18.01.2000             IBM 500
59      105         Banques        59     18.01.2000             IBM 500
61      118       Éducation        61     18.01.2000             IBM 500
67      120         Banques        67     19.01.2000             IBM 500
71      101    Construction        71     19.01.2000             IBM 500
78      125    Construction        78     21.01.2000             IBM 500
87      105         Banques        87     25.01.2000             IBM 500
90      102 Machines/Outils        90     26.01.2000             IBM 500
95      113    Construction        95     27.01.2000             IBM 500
96      103      Assurances        96     27.01.2000             IBM 500
101     104       Éducation       101     28.01.2000             IBM 500
104     115    Distribution       104     28.01.2000             IBM 500
108     118       Éducation       108     31.01.2000             IBM 500
4       104       Éducation         4     03.01.2000       AST Intel 200
7       104       Éducation         7     04.01.2000       AST Intel 200
11      110    Distribution        11     05.01.2000       AST Intel 200
```

Et encore un tri des colonnes parfois demandé (trier les colonnes par type):



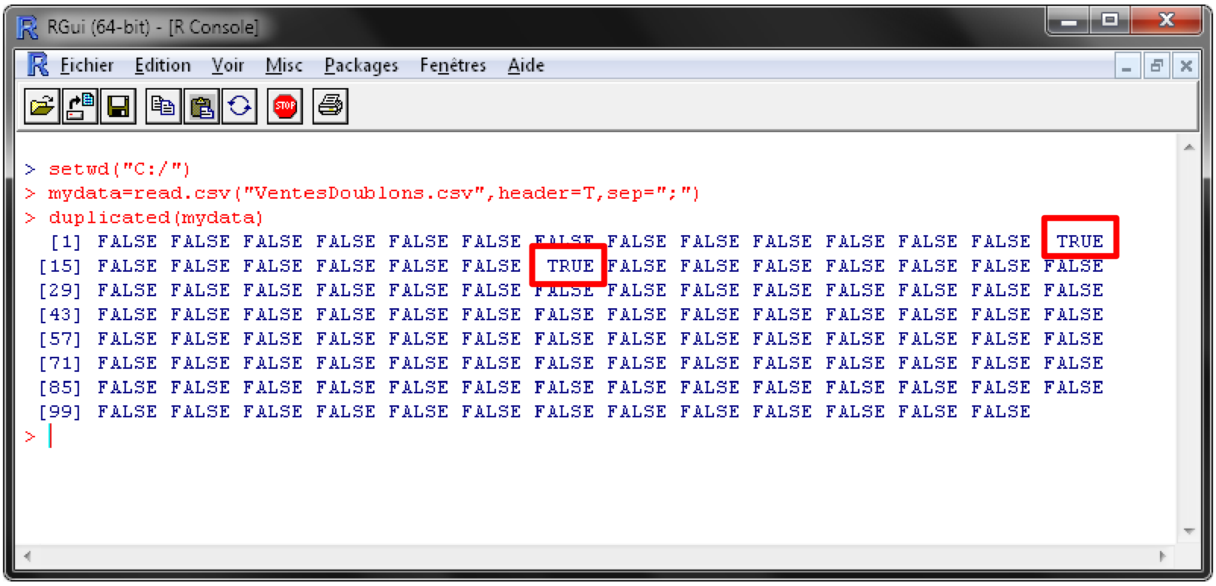
```
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=",")
> str(mydata)
'data.frame':	109 obs. of  10 variables:
 $ NClient            : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Activite           : Factor w/ 8 levels "Alimentaire",..: 2 7 6 6 3 2 6 1 4 8 ...
 $ NCommande          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande     : Factor w/ 21 levels "03.01.2000","04.01.2000",..: 1 1 1 1 2 2 2 2 2 2 ...
 $ Article            : Factor w/ 4 levels "AST Intel 150",..: 3 4 1 2 3 1 2 4 3 4 ...
 $ Quantite           : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece       : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais             : Factor w/ 4 levels "0.00%","1.50%",..: 2 1 1 1 2 1 1 1 1 1 ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21123 ...
 $ FacturePayee       : Factor w/ 2 levels "Non","Oui": 2 2 2 2 2 2 2 2 2 2 ...
> mydata_by_type<-mydata[,names(sort(sapply(mydata,class)))]
> str(mydata_by_type)
'data.frame':	109 obs. of  10 variables:
 $ Activite           : Factor w/ 8 levels "Alimentaire",..: 2 7 6 6 3 2 6 1 4 8 ...
 $ DateDeCommande     : Factor w/ 21 levels "03.01.2000","04.01.2000",..: 1 1 1 1 2 2 2 2 2 2 ...
 $ Article            : Factor w/ 4 levels "AST Intel 150",..: 3 4 1 2 3 1 2 4 3 4 ...
 $ Rabais             : Factor w/ 4 levels "0.00%","1.50%",..: 2 1 1 1 2 1 1 1 1 1 ...
 $ FacturePayee       : Factor w/ 2 levels "Non","Oui": 2 2 2 2 2 2 2 2 2 2 ...
 $ NClient            : int  100 123 109 104 117 103 104 111 113 116 ...
 $ NCommande          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Quantite           : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece       : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21123 ...
>
```

Exemple 89.: Chercher et supprimer des doublons

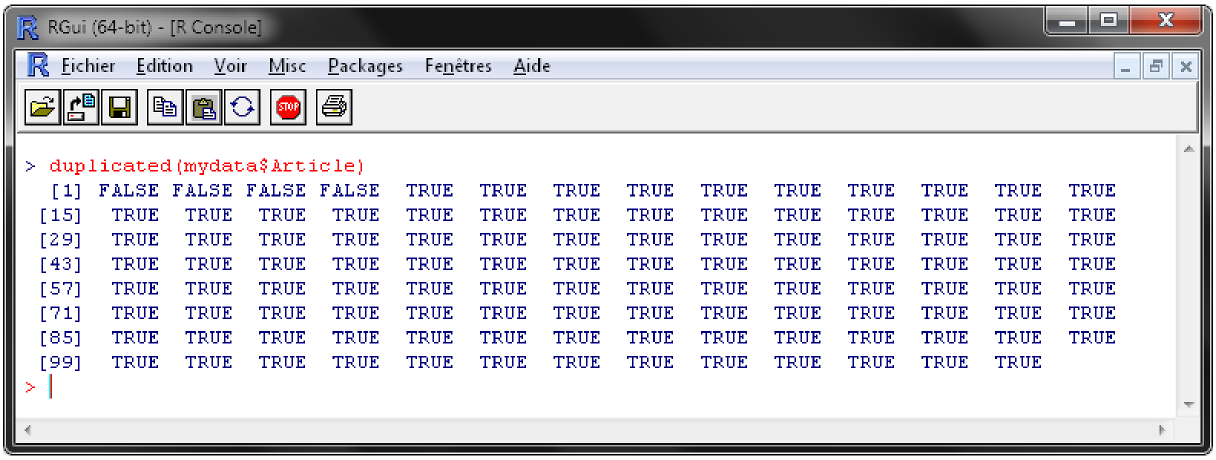
R 3.0.2

En utilisant la commande **duplicated( )** vous pouvez chercher les doublons se trouvant dans un data frame dans ou une ou plusieurs colonnes.

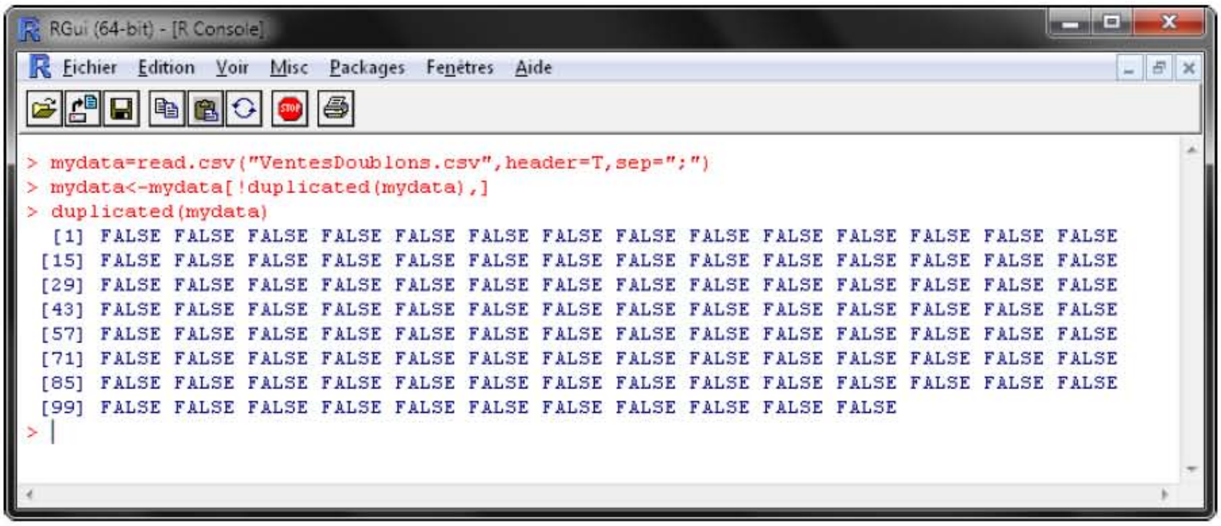
Voyons cela d'abord avec notre data frame habituel en entier:



et avec une seule colonne:



et pour supprimer les doublons:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> mydata=read.csv("VentesDoublons.csv",header=T,sep=";")
> mydata<-mydata[!duplicated(mydata),]
> duplicated(mydata)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[15] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[29] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[43] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[57] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[71] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[99] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>

Exemple 90.: Remplacer des valeurs

R 3.6.0

Remplacer une valeur par une autre est assez trivial mais comme d'habitude, ce n'est pas forcément intuitif si l'on n'a pas déjà vu un exemple au moins une fois.

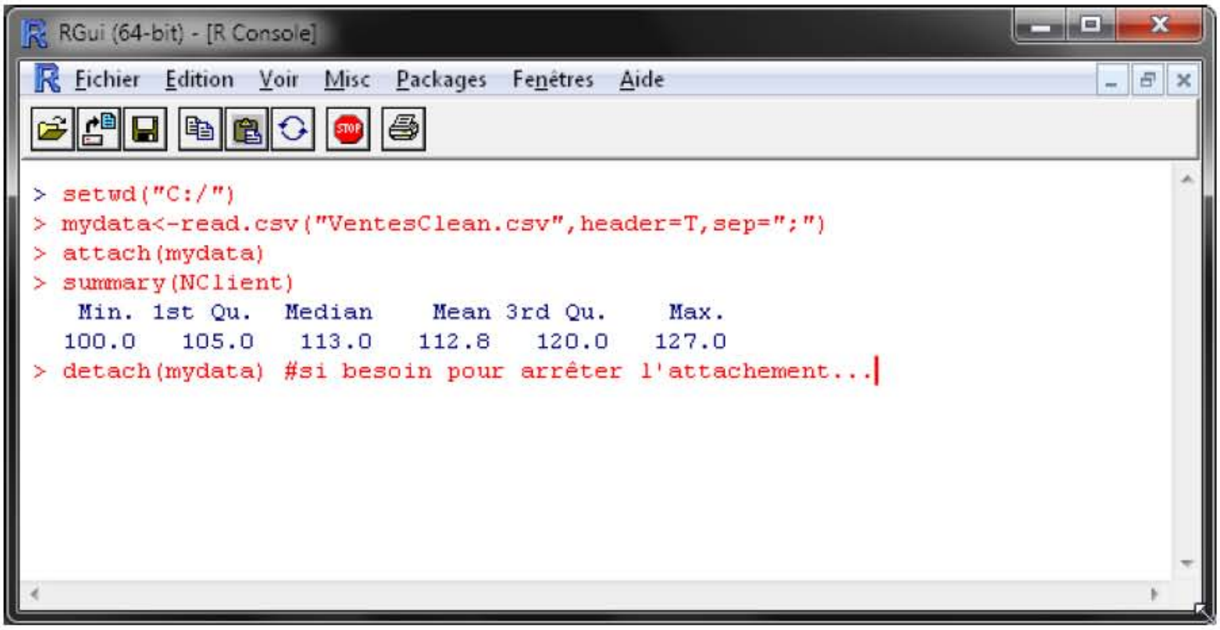
Voyons donc cela en remplaçant un des nomes d'articles:

```
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=",",stringsAsFactors=FALSE)
> head(mydata)
  NClient        Activite NCommande DateDeCommande            Article Quantite
1     100      Assurances         1     03.01.2000 Compaq Presario 100       12
2     123 Machines/Outils         2     03.01.2000            IBM 500        2
3     109       Éducation         3     03.01.2000      AST Intel 150        5
4     104       Éducation         4     03.01.2000      AST Intel 200        3
5     117         Banques         5     04.01.2000 Compaq Presario 100       13
6     103      Assurances         6     04.01.2000      AST Intel 150        2
  PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1         1650  1.50%            19503.00          Oui
2         2299  0.00%             4598.00          Oui
3         2690  0.00%            13450.00          Oui
4         3190  0.00%             9570.00          Oui
5         1650  1.50%            21128.25          Oui
6         2690  0.00%             5380.00          Oui
> mydata[mydata$Article=="Compaq Presario 100",]<-"Compaq 100"
> head(mydata)
     NClient        Activite  NCommande DateDeCommande       Article   Quantite
1 Compaq 100      Compaq 100 Compaq 100     Compaq 100    Compaq 100 Compaq 100
2        123 Machines/Outils          2     03.01.2000       IBM 500          2
3        109       Éducation          3     03.01.2000 AST Intel 150          5
4        104       Éducation          4     03.01.2000 AST Intel 200          3
5 Compaq 100      Compaq 100 Compaq 100     Compaq 100    Compaq 100 Compaq 100
6        103      Assurances          6     04.01.2000 AST Intel 150          2
  PrixParPiece     Rabais PrixTotalAvecRabais FacturePayee
1   Compaq 100 Compaq 100          Compaq 100   Compaq 100
2         2299      0.00%                4598          Oui
3         2690      0.00%               13450          Oui
4         3190      0.00%                9570          Oui
5   Compaq 100 Compaq 100          Compaq 100   Compaq 100
6         2690      0.00%                5380          Oui
>
> #et pour finir proprement
> mydata[, "Article"] <- as.factor(mydata[, "Article"])
> 
```

Exemple 91.: Accélérer l'accès aux colonnes

R 3.0.2

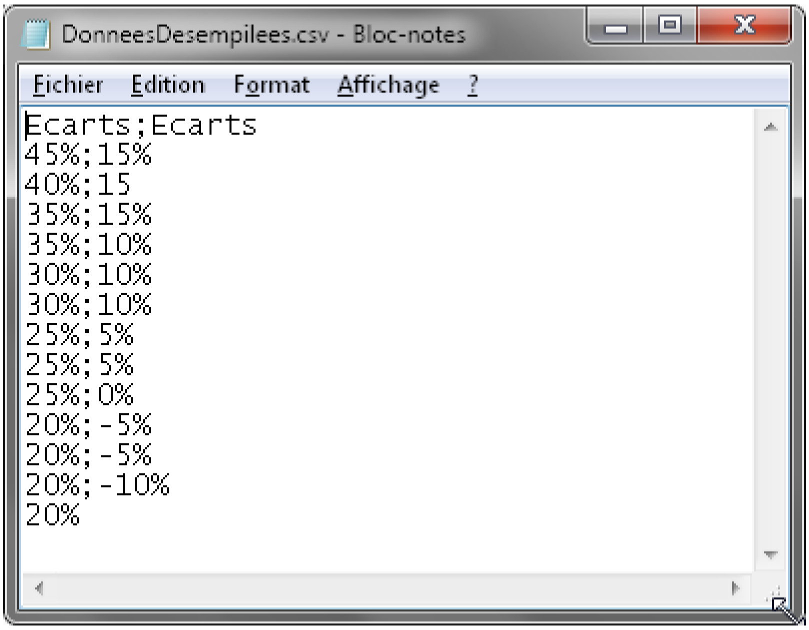
Si la syntaxe *NomDuDataframe.$Colonne* vous ennuie à la longue, vous pouvez utiliser la commande **attach( )** comme indiqué ci-dessous:



Exemple 92.: Empiler/Désempiler des données de data frame

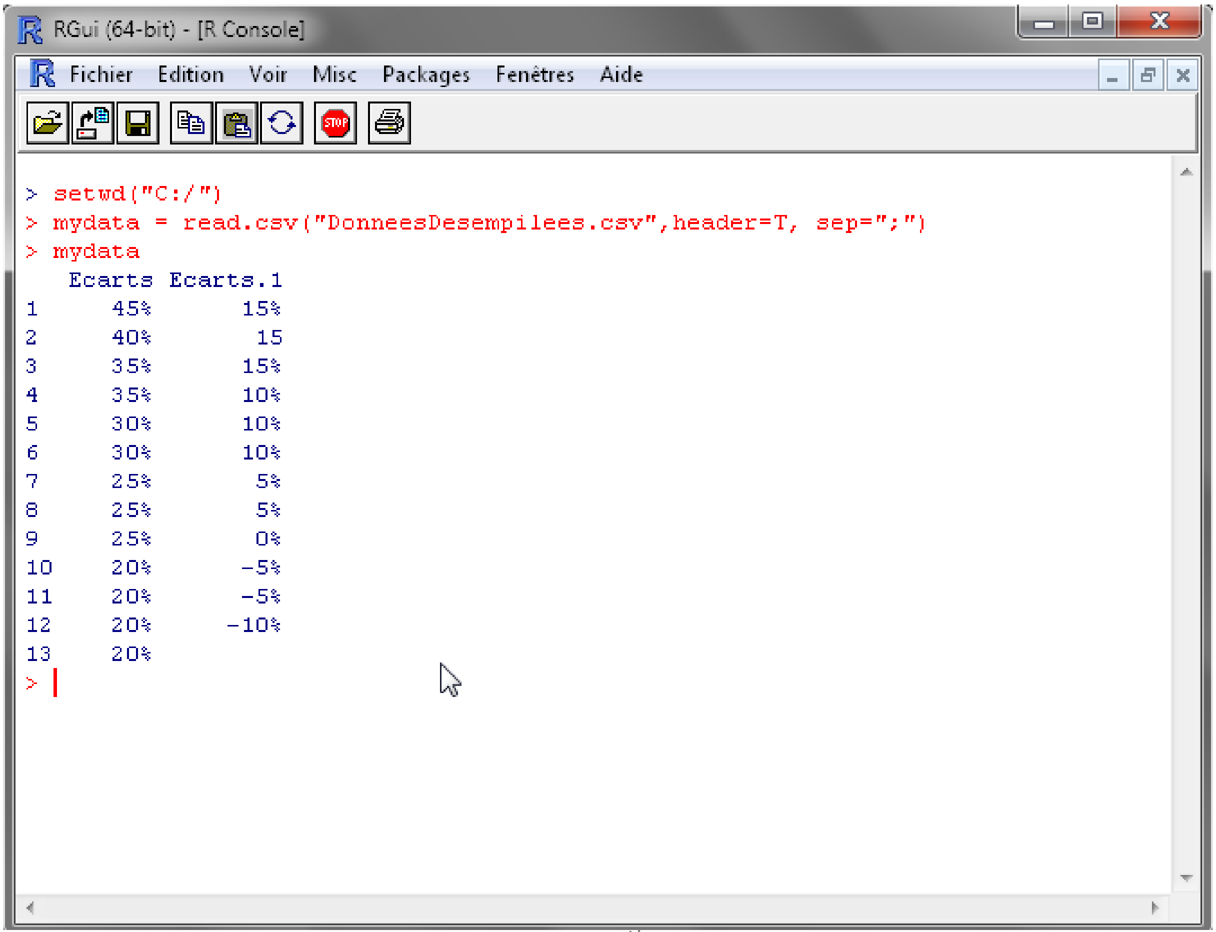
R 3.0.2

Il arrivera souvent dans la pratique que l'utilisateur copie/colle des données provenant de Microsoft Excel sous la forme suivante:

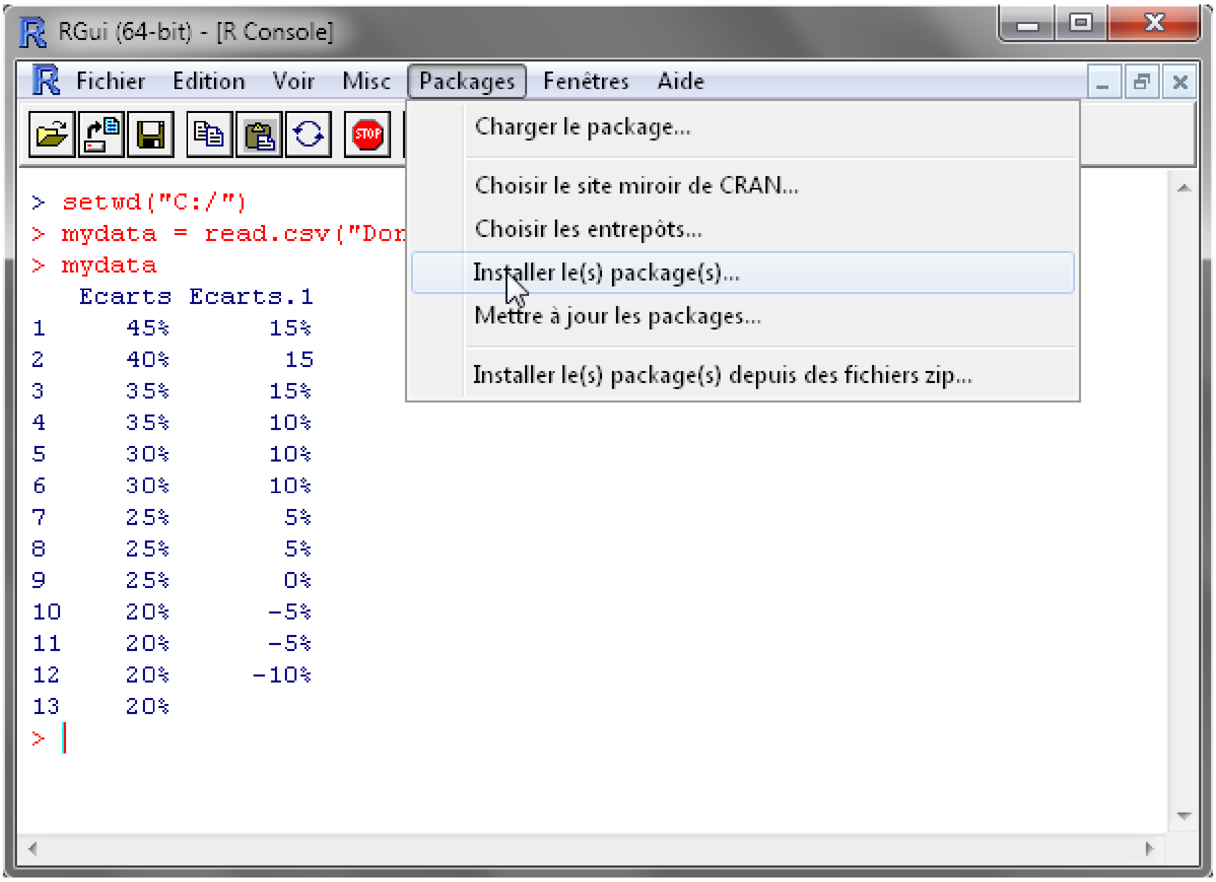


ou avec encore plus de colonnes. Suivant les outils que nous verrons par la suite, il est nécessaire de mettre les données les unes sous les autres. Certes, un simple copier/coller suffit mais R a aussi une commande prévue à cet effet.

D'abord commençons par le classique:

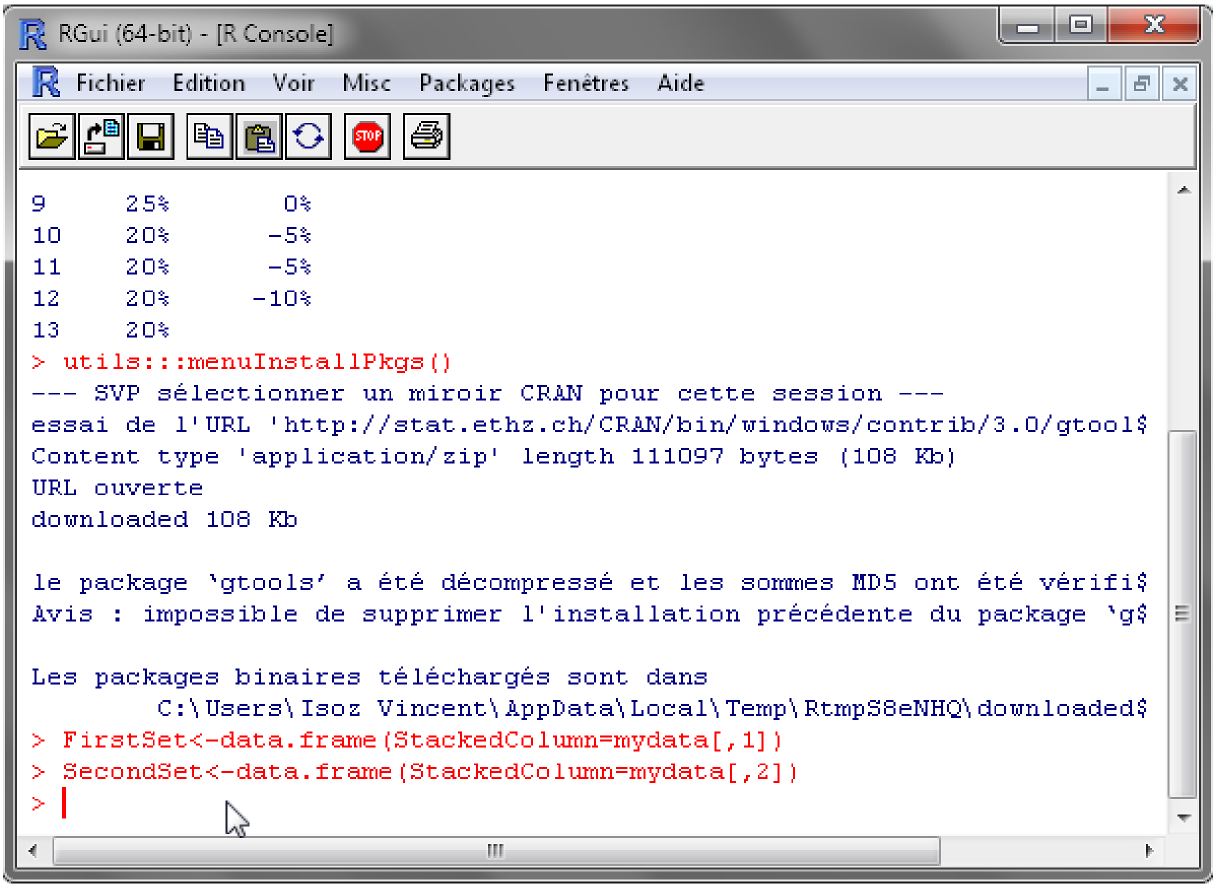


Nous pouvons ensuite continuer notre travail. Il nous faut pour cela installer le package **gtools** comme indiqué ci-dessous:

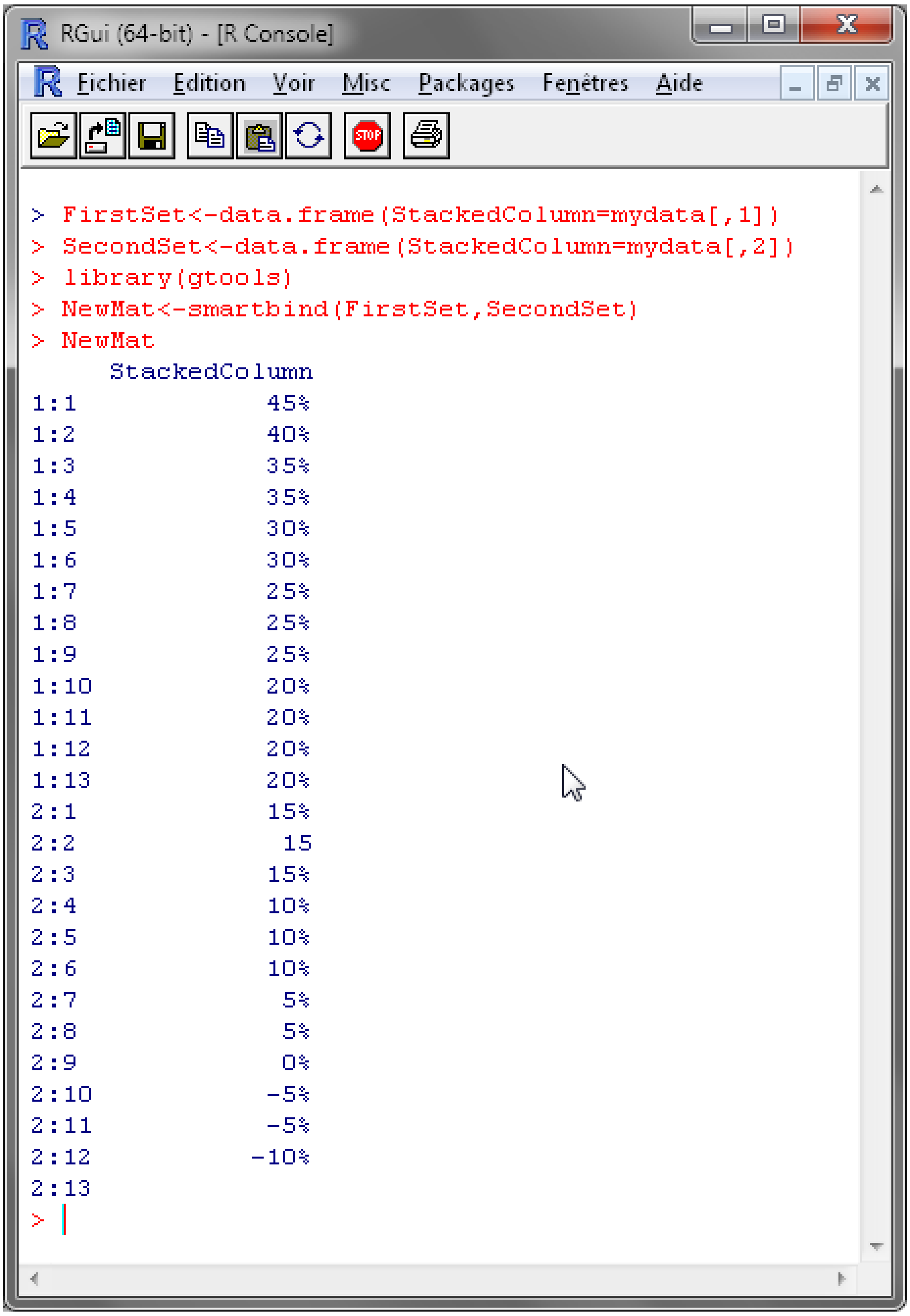




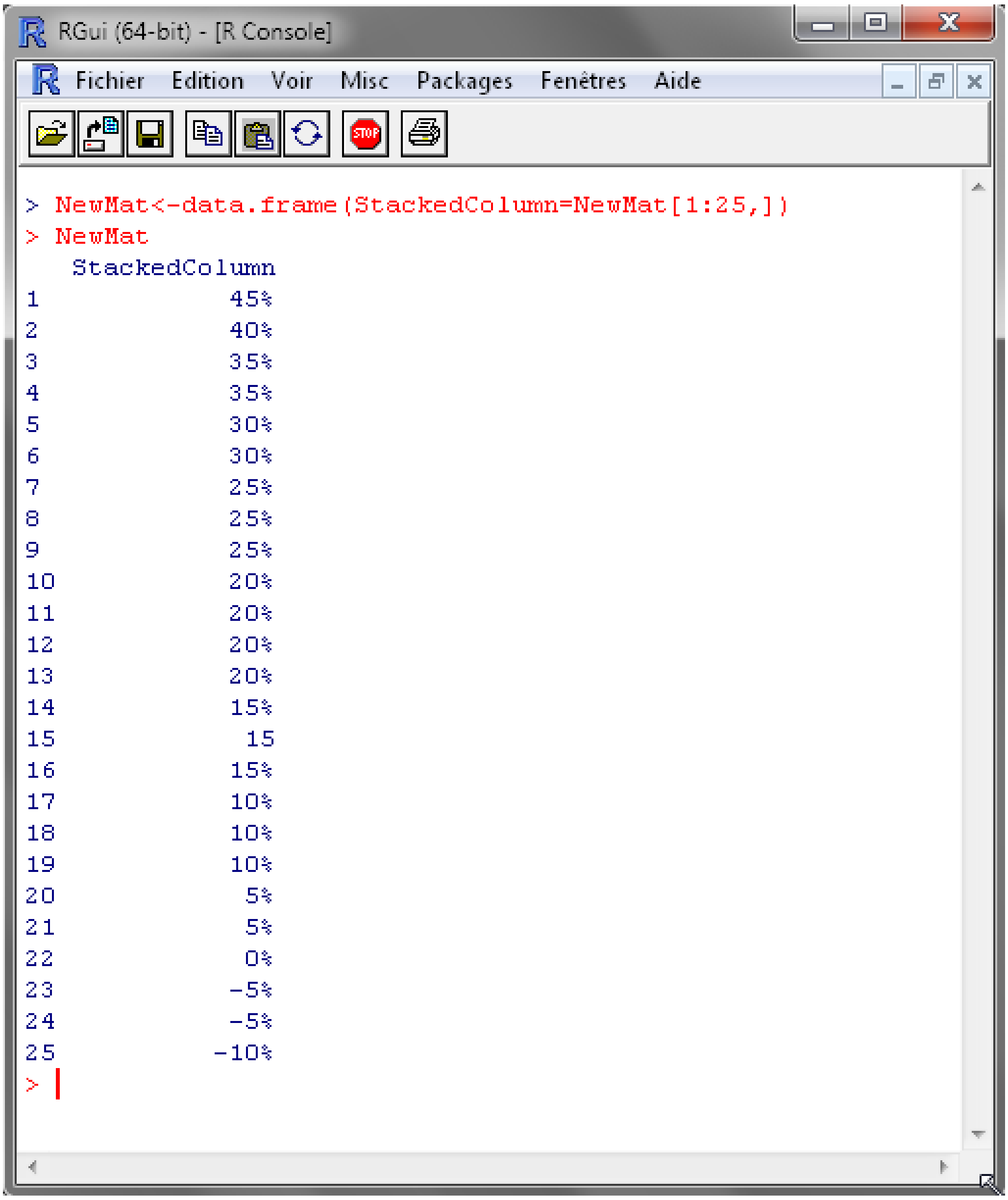
Nous validons par **OK** autant de fois que nécessaire. Ensuite, nous retournons à nos commandes:



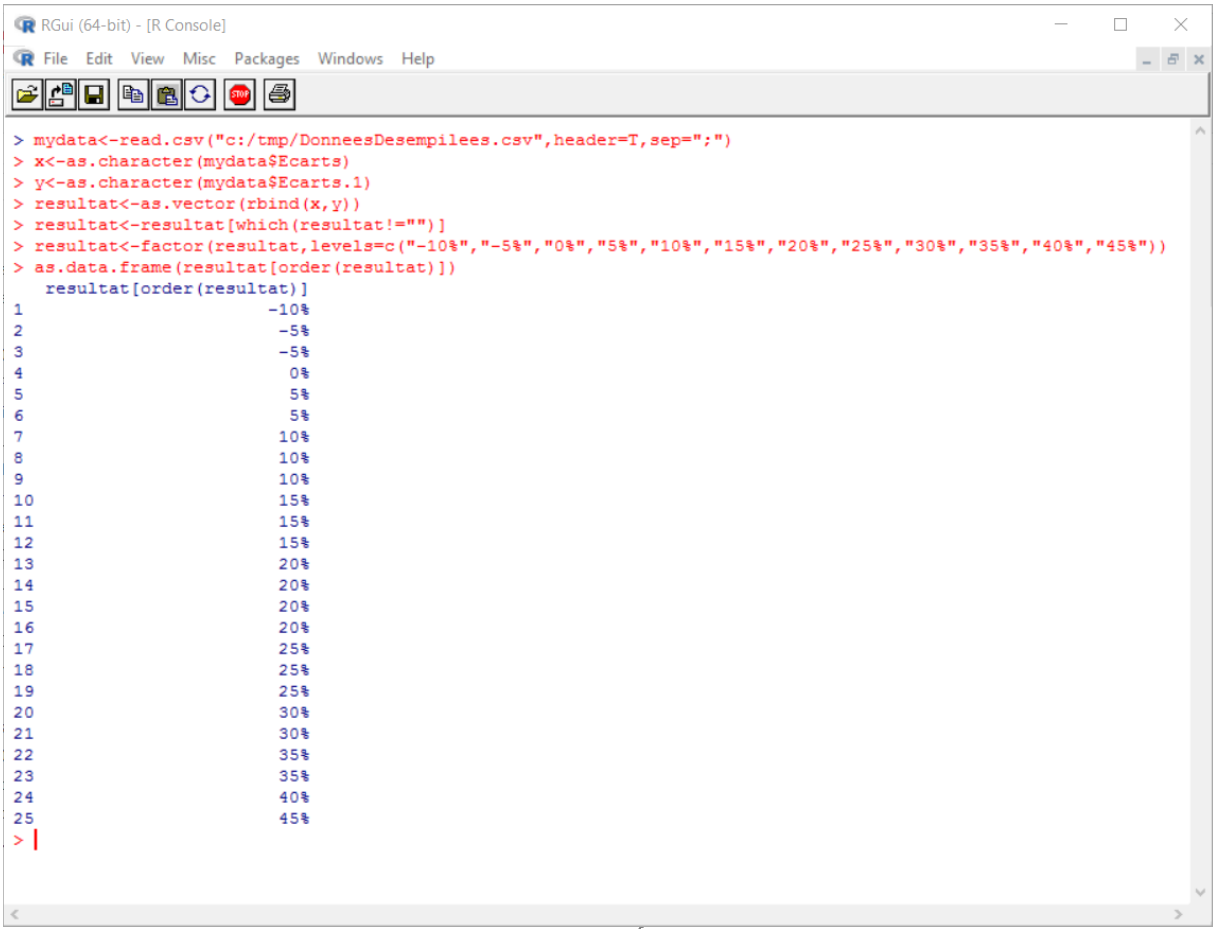
et nous pouvons enfin utiliser la colonne **smartbind( )** du package **gtools**:



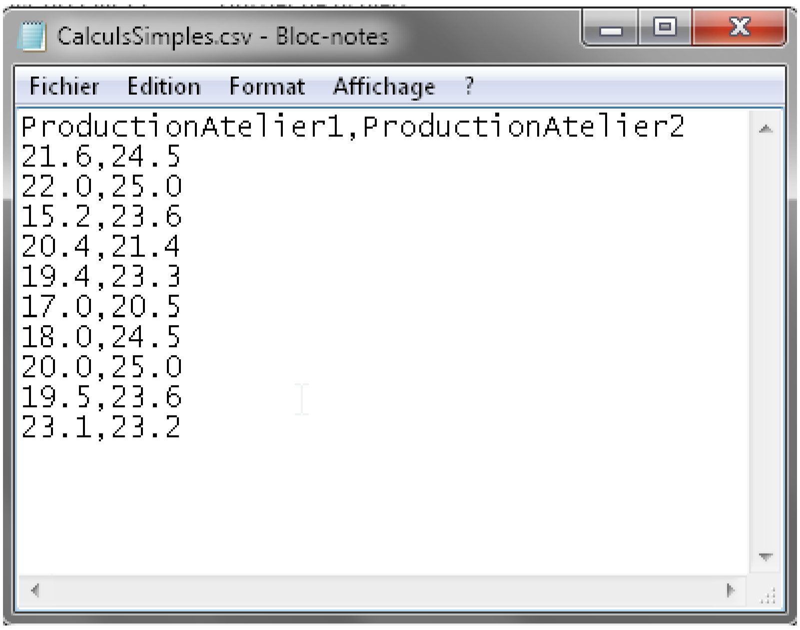
Ensuite, pour faire un peu le ménage:



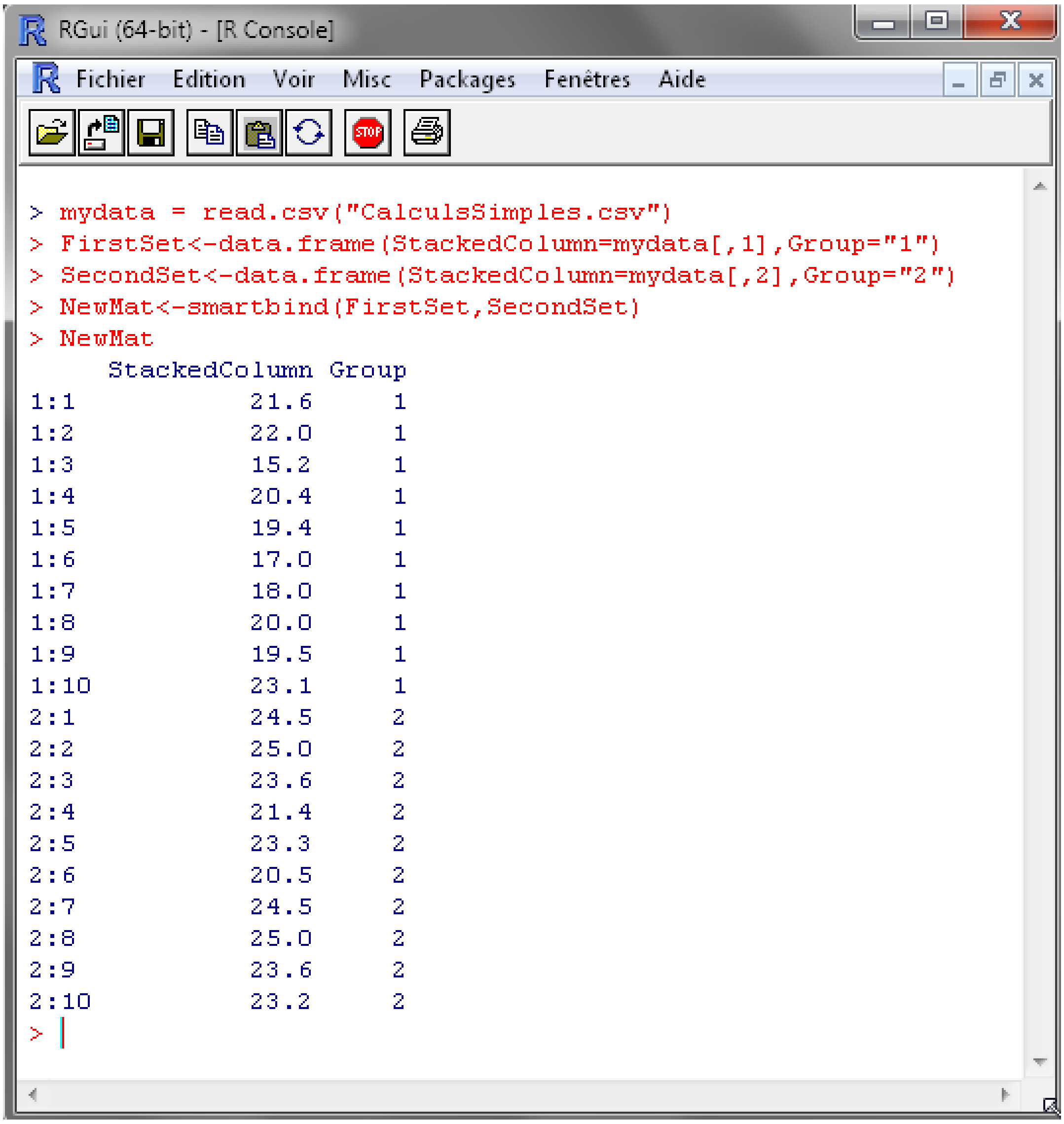
ou sans packages en utilisant judicieusement **rbind( )** , **as.vector( )** , **factor( )** et **as.data.frame( ) :**



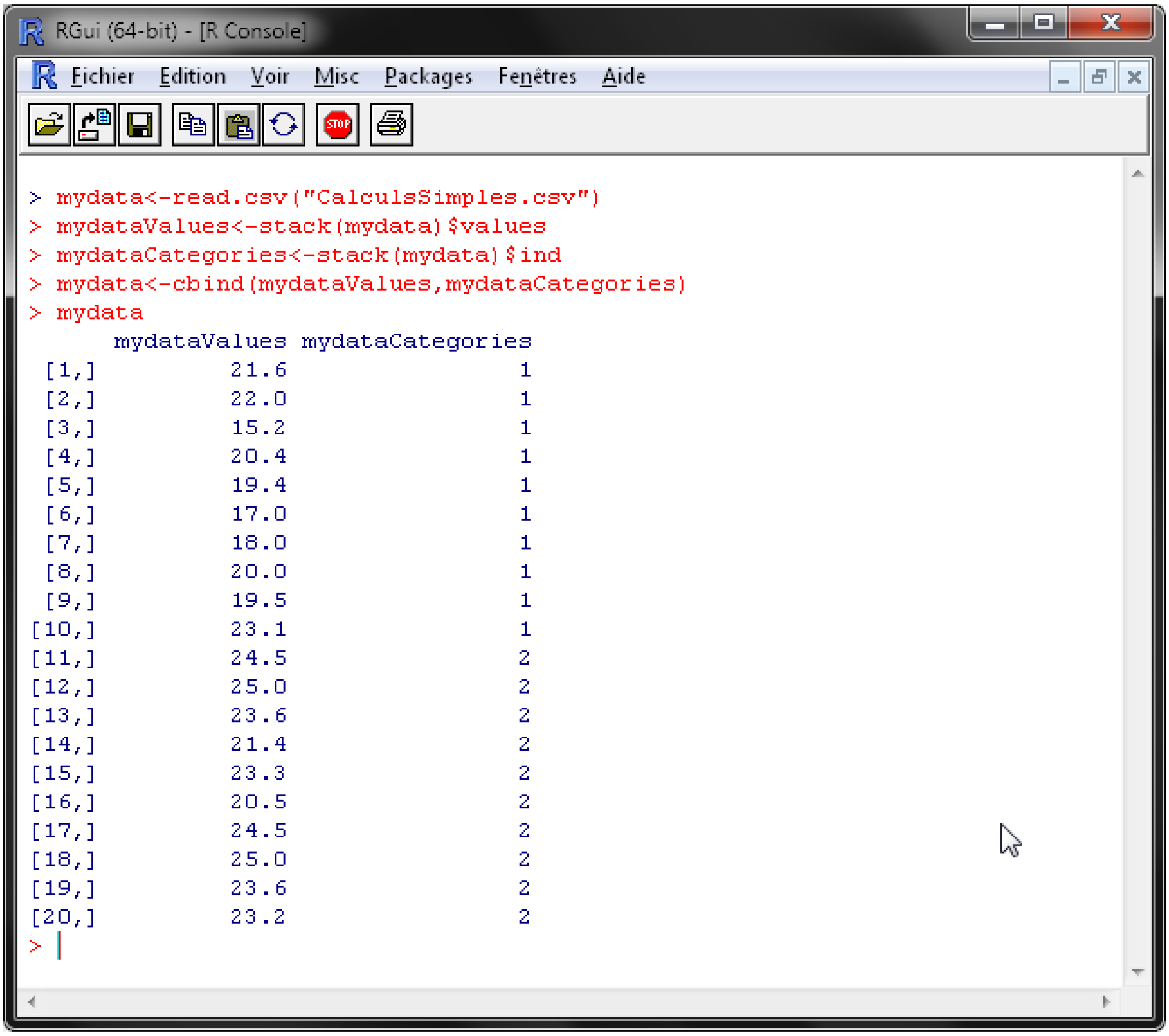
Nous souhaiterions faire la même chose qu'avant mais avec une petite subtilité nécessaire (disons plutôt "préférée" par certains utilisateurs). Il s'agit d'empiler les données mais avec une variable qualitative de l'appartenance de groupe. Pour cela, nous reprenons le fichier suivant:



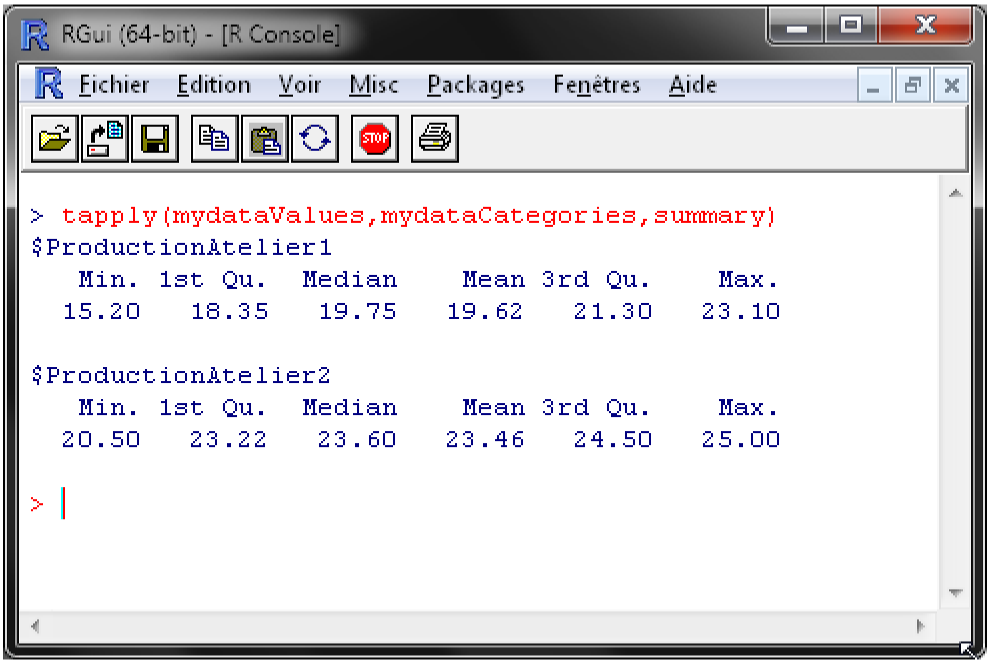
Une méthode possible et similaire à celle d'avant mais avec un petit ajout consiste à utiliser la fonction **smartbind( )** du package **gtools**:



Nous pouvons faire la même chose sans package (je m'en suis rendu compte un peu tard...) avec la commande native **stack( )** :



Ensuite, avec la commande **tapply( )** que nous retrouvons à de nombreuses reprises nous pouvons nous amuser un peu:



et avec un petit graphique rapide:

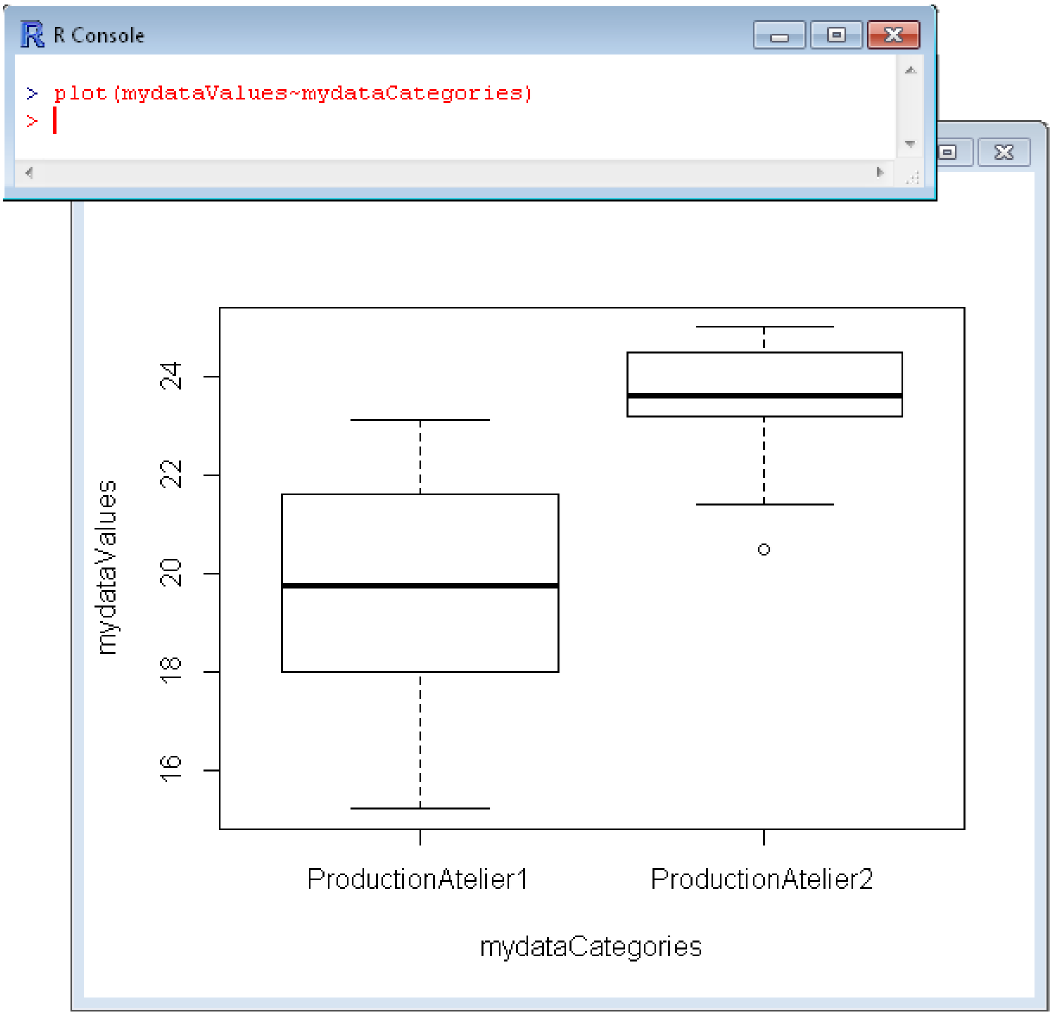
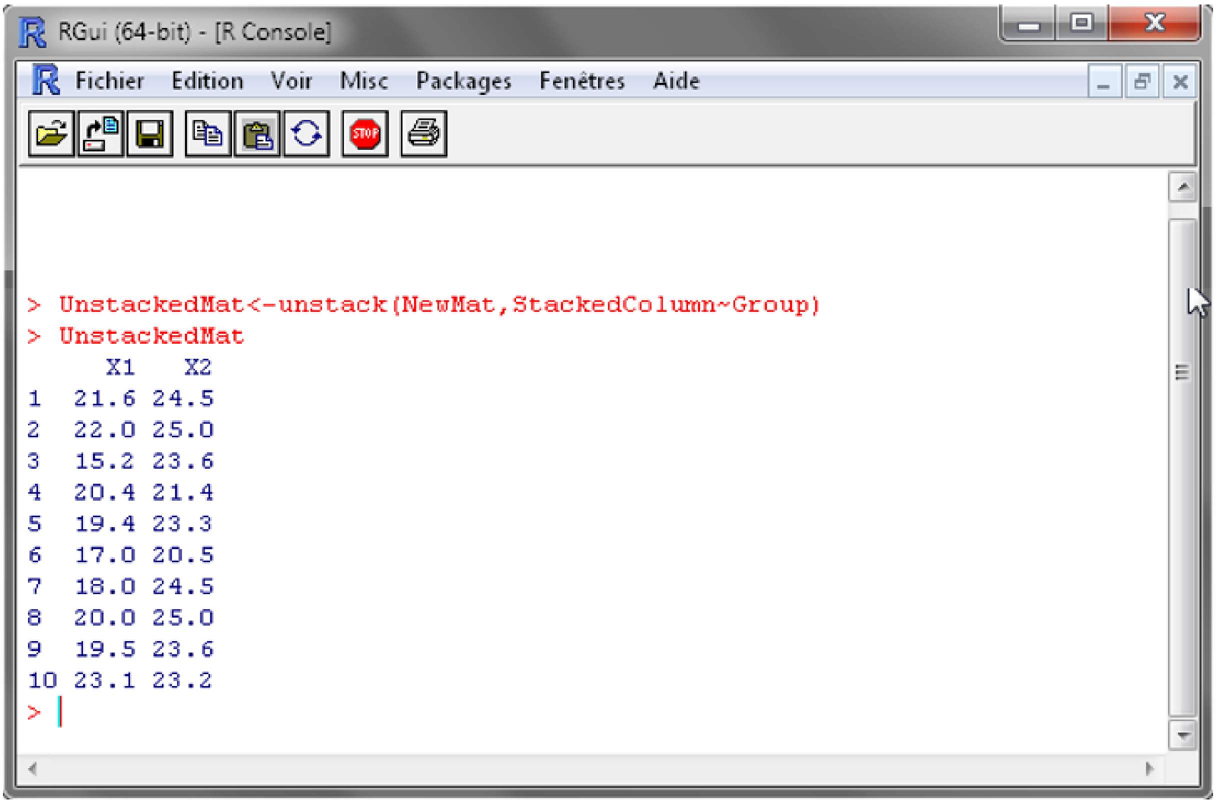


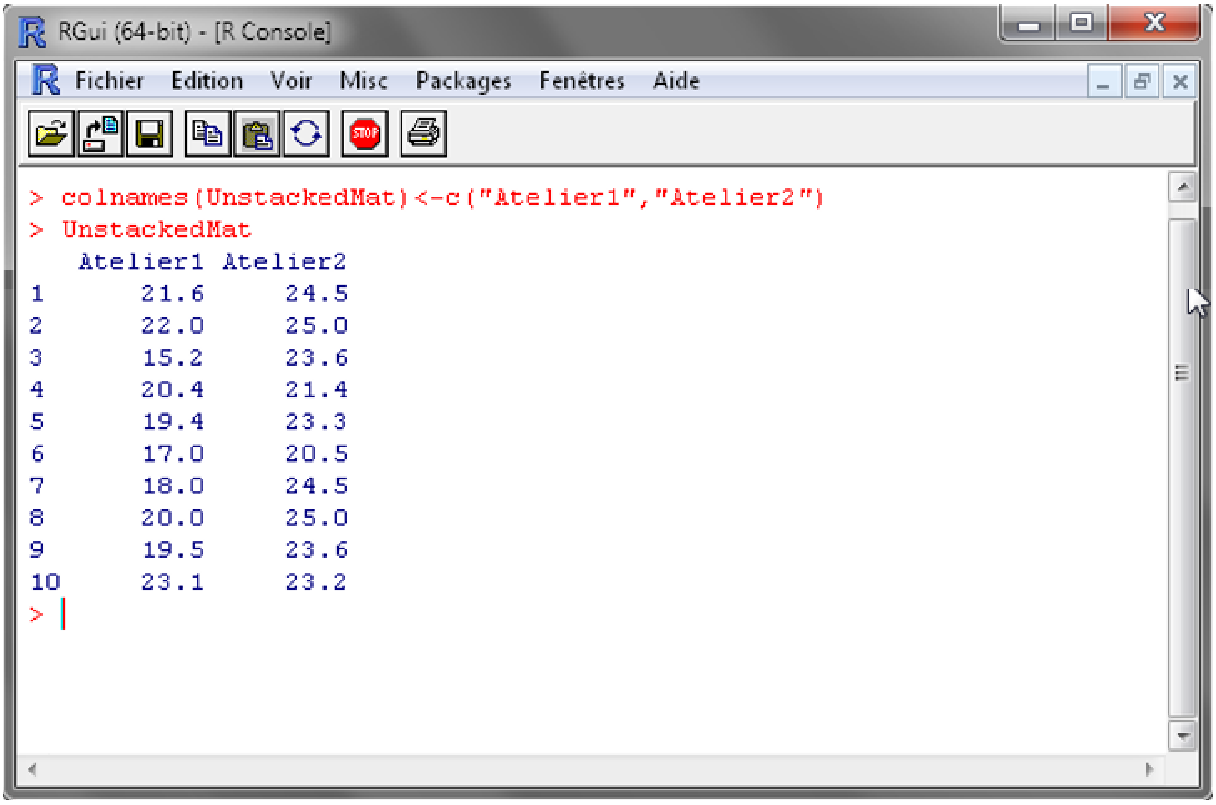
Figure 28 Graph Box-Whisker (boîtes à moustaches)

Maintenant, en imaginant que nous avons importé le tableau précédent, supposons que nous souhaiterions faire le contraire: désempiler.

Pour cela, nous allons utiliser la commande native **unstack( )** (du package **utils**) sur le jeu de données *NewMat* que nous avions créé plus haut:



et on renomme tout cela vite fait:



Exemple 93.: Empiler (fusionner) des données non-homogènes

R 3.6.1

Il peut être pénible pour certaines personnes de fusionner des jeux de données qui n'ont ni le mêmes colonnes, ni dans le même ordre avec la fonction **cbind( )**. Voyons donc comment nous pouvons nous simplifier la tâche avec la fonction **smartbind( )** du package **gtools**:

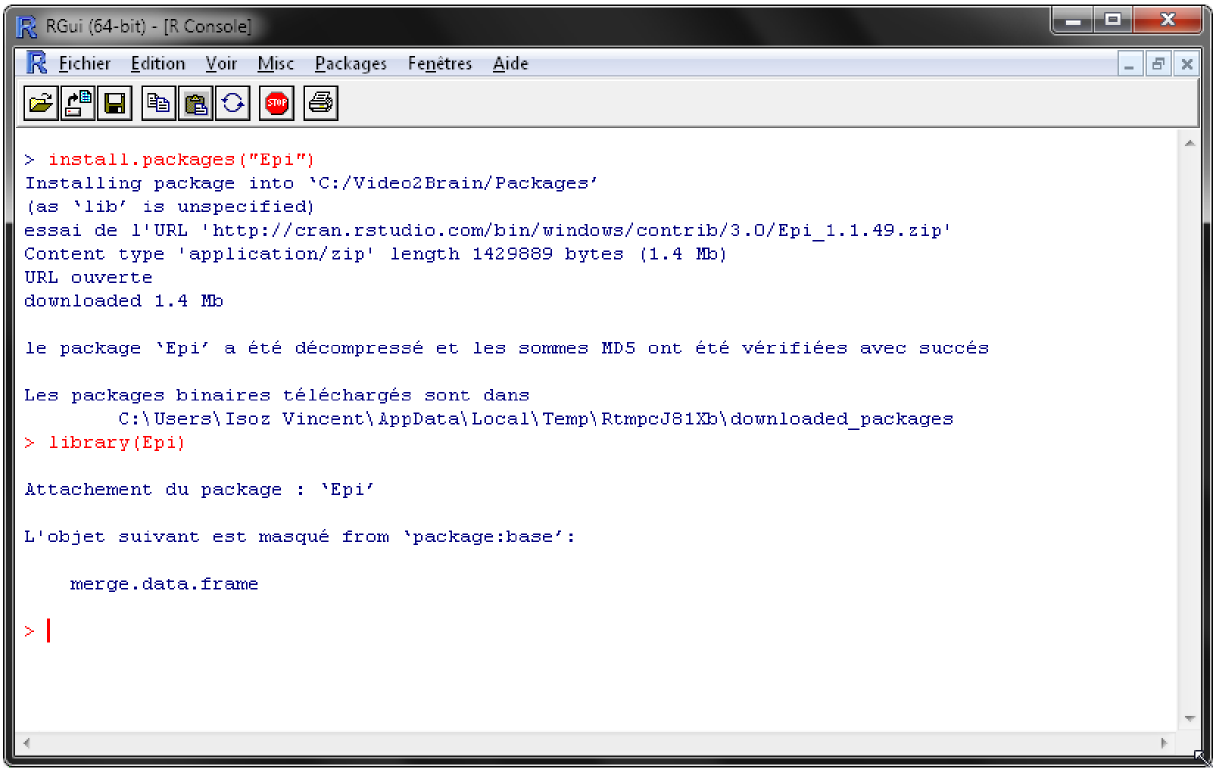
```
> library("gtools")
> (df1<-data.frame(ID=c(1:3),Article=c("IBM 500","AST Intel 150","AST Intel 200")))
  ID       Article
1  1       IBM 500
2  2 AST Intel 150
3  3 AST Intel 200
> (df2<-data.frame(Quantité=c(3,8),ID=c(4:5),Article=c("Compar Presario 100","IBM 500")))
  Quantité ID             Article
1        3  4 Compar Presario 100
2        8  5             IBM 500
> #pour fusionner plein de datafram qui commencent par "df"
> (fusion<-do.call("smartbind", mget(ls(pattern = "df"))))
      ID             Article Quantité
df1:1  1             IBM 500       NA
df1:2  2       AST Intel 150       NA
df1:3  3       AST Intel 200       NA
df2:1  4 Compar Presario 100        3
df2:2  5             IBM 500        8
> rownames(fusion) <- NULL
> fusion
  ID             Article Quantité
1  1             IBM 500       NA
2  2       AST Intel 150       NA
3  3       AST Intel 200       NA
4  4 Compar Presario 100        3
5  5             IBM 500        8
>
```

Exemple 94.: Synthétiser/Aggréger des données (tables de contingences)

R 3.0.2

Nous allons voir ici quelques techniques élémentaires de synthèse de données plus élaborées que la simple fonction **summary( )**. Nous sommes bien évidemment très loin de ce qu'il est possible de faire beaucoup plus rapidement et esthétiquement avec le tableur Microsoft Excel (voir mes formations sur les tableaux croisés dynamiques sur Video2Brain.com et Alphorm.com).

Voyons une première manière sympathique de synthétiser des données dans un tableau à double entrée avec des lignes de séparation. Nous installons et chargeons d'abord le package **Epi**:



et ensuite nous faisons la synthèse avec la commande **stat.table( )** :

```
> library("Epi")
> mydata<-read.csv(file="c:/tmp/VentesClean.csv",header=TRUE,sep=",",encoding="ANSI")
> stat.table(index=list(Article,FacturePayee),data=mydata,contents=list(mean(Quantite)),margins=c(TRUE,TRUE))
 ----------------------------------------------
                      ------FacturePayee-------
 Article                   Non     Oui   Total
 ----------------------------------------------
 AST Intel 150            7.56    9.00    8.38
 AST Intel 200            9.92    5.00    7.00
 Compaq Presario 100     13.36    9.87   11.55
 IBM 500                  5.67    5.40    5.52

 Total                    9.42    7.08    8.11
 ----------------------------------------------
> stat.table(index=list(Article,FacturePayee),data=mydata,contents=list(mean(Quantite),count(NClient))
+ ,margins=c(TRUE,TRUE))
 ----------------------------------------------
                      ------FacturePayee-------
 Article                   Non     Oui   Total
 ----------------------------------------------
 AST Intel 150            7.56    9.00    8.38
                             7      11      14

 AST Intel 200            9.92    5.00    7.00
                            11      13      18

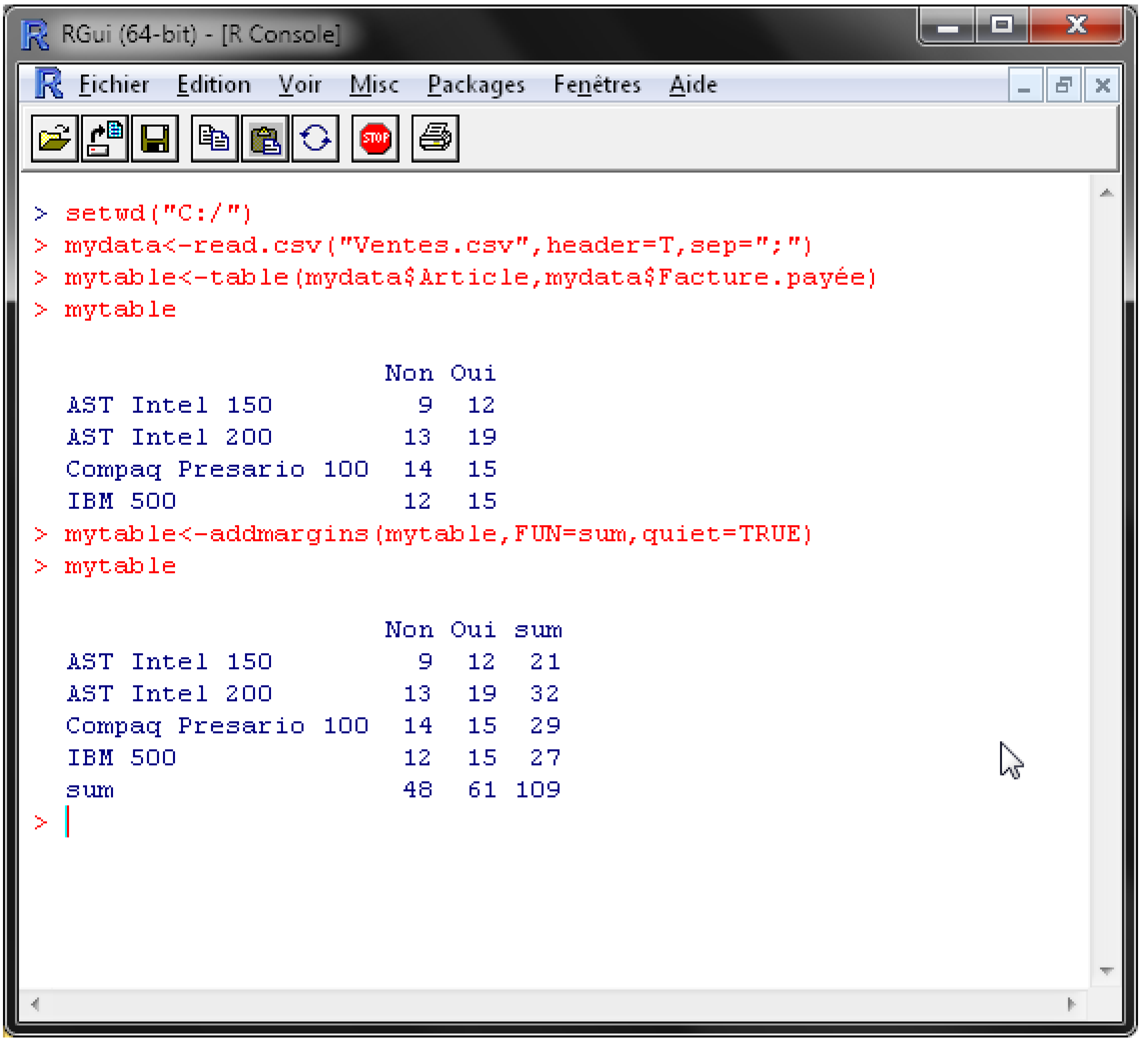
 Compaq Presario 100     13.36    9.87   11.55
                            11      15      23

 IBM 500                  5.67    5.40    5.52
                            12      12      19

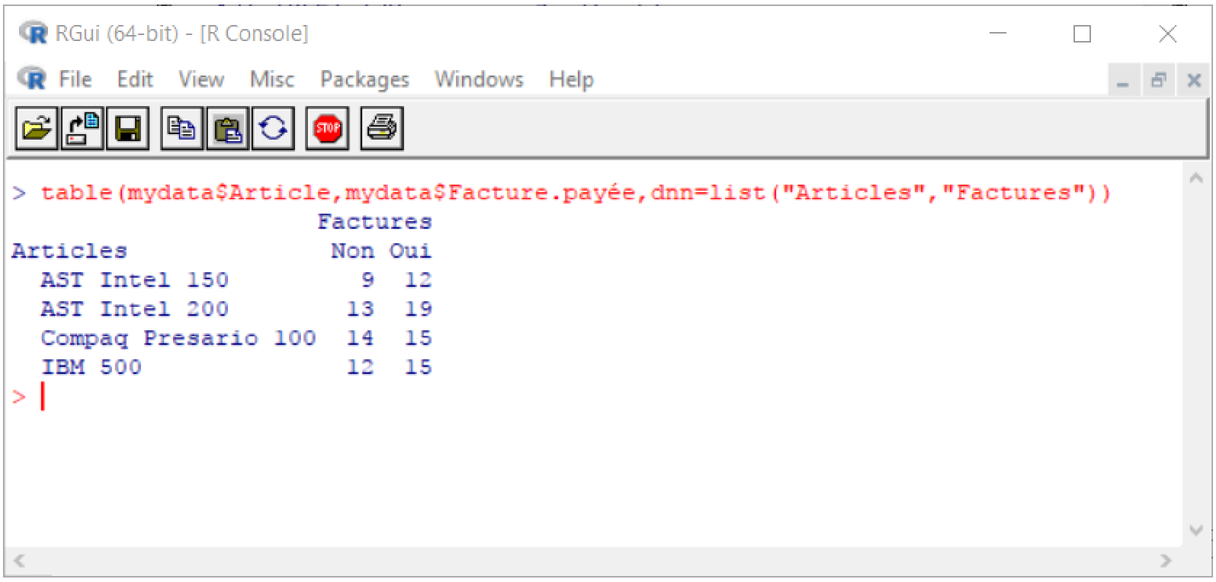

 Total                    9.42    7.08    8.11
                            24      26      28
 ----------------------------------------------
> |
```

Voici une deuxième manière de synthétiser des données en une table de contingences avec les commandes **table( )** (cette dernière peut être utilisée avec une seule variable aussi!) et **addmargins( )** et ce sans packages[10]:

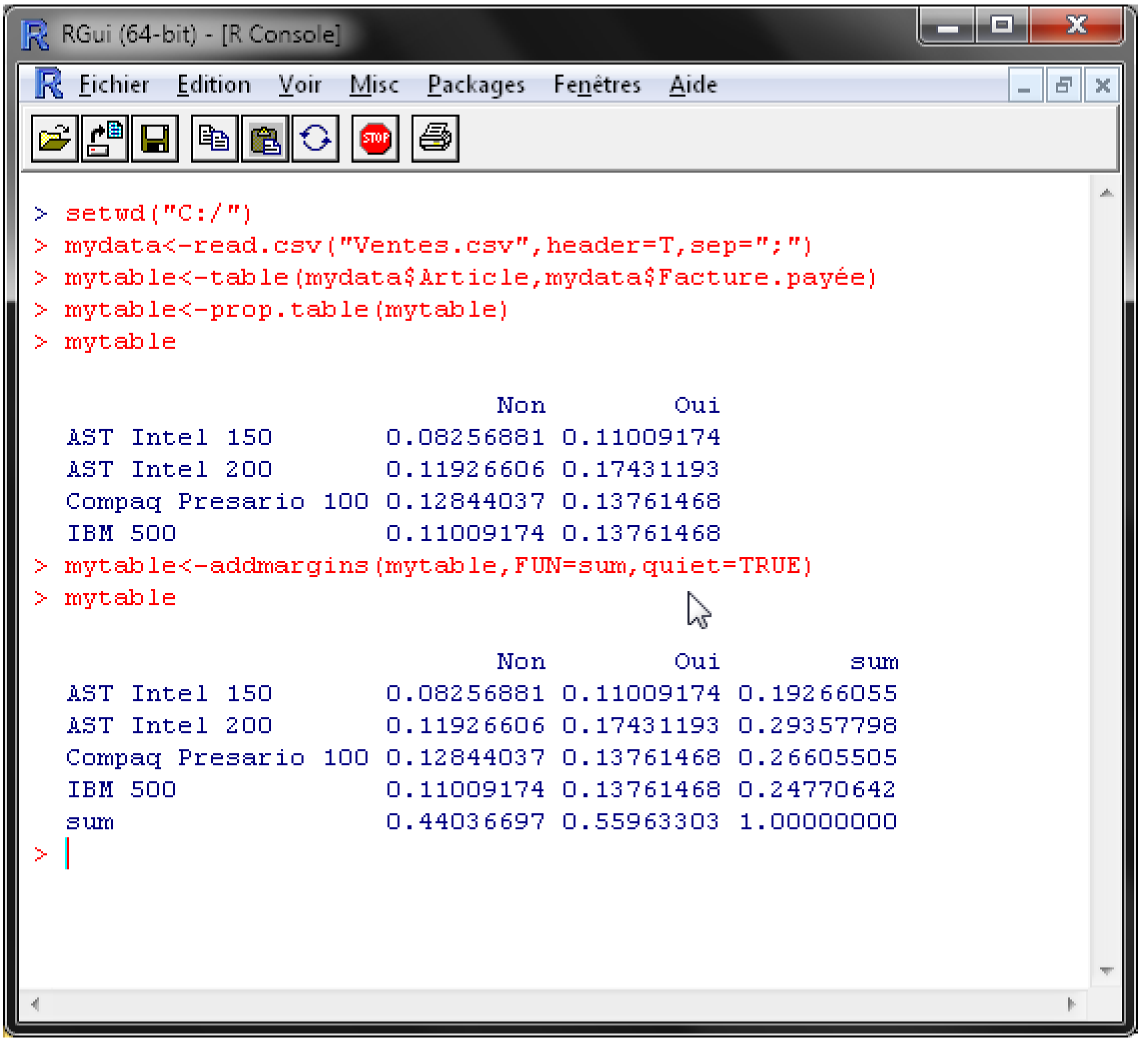
[10] La commande **xtable( )** que nous verrons lors de notre étude des graphiques permet d'avoir le même résultat



On a aussi l'argument **dnn** de la fonction **table( )** qui permet de mettre des titres:



Sinon, avec les proportions, en utilisant la commande **prop.table( )** cela donne:

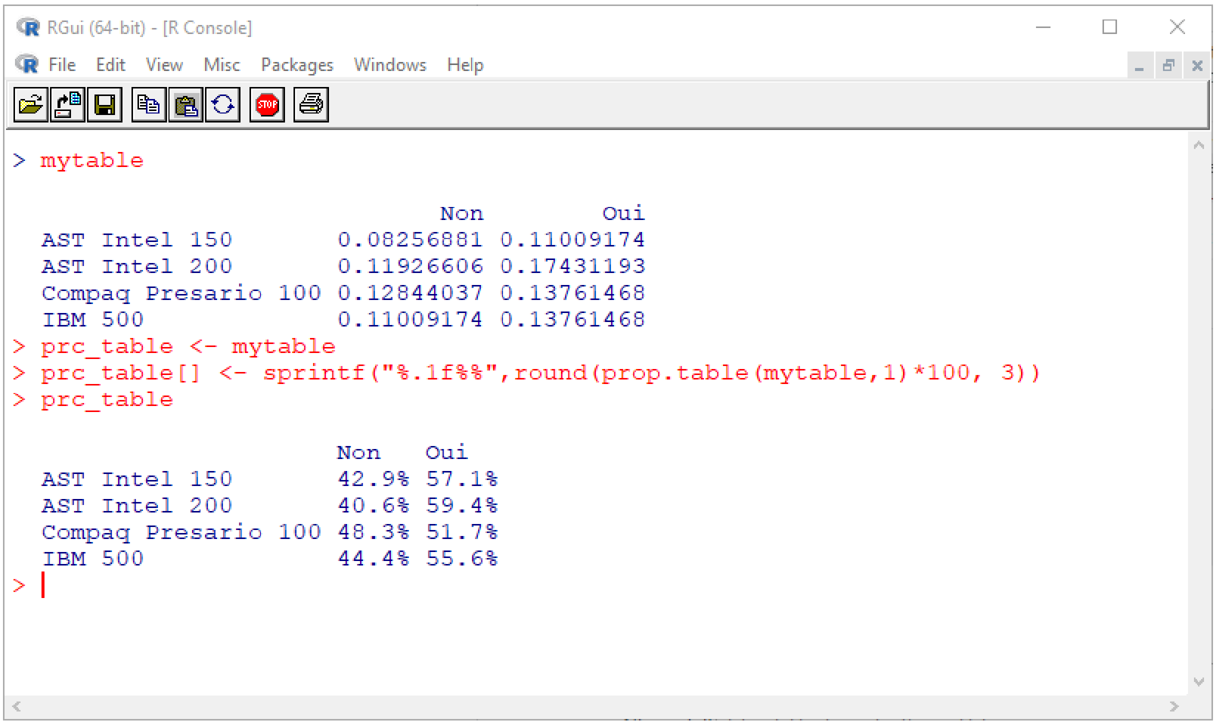


Remarques:

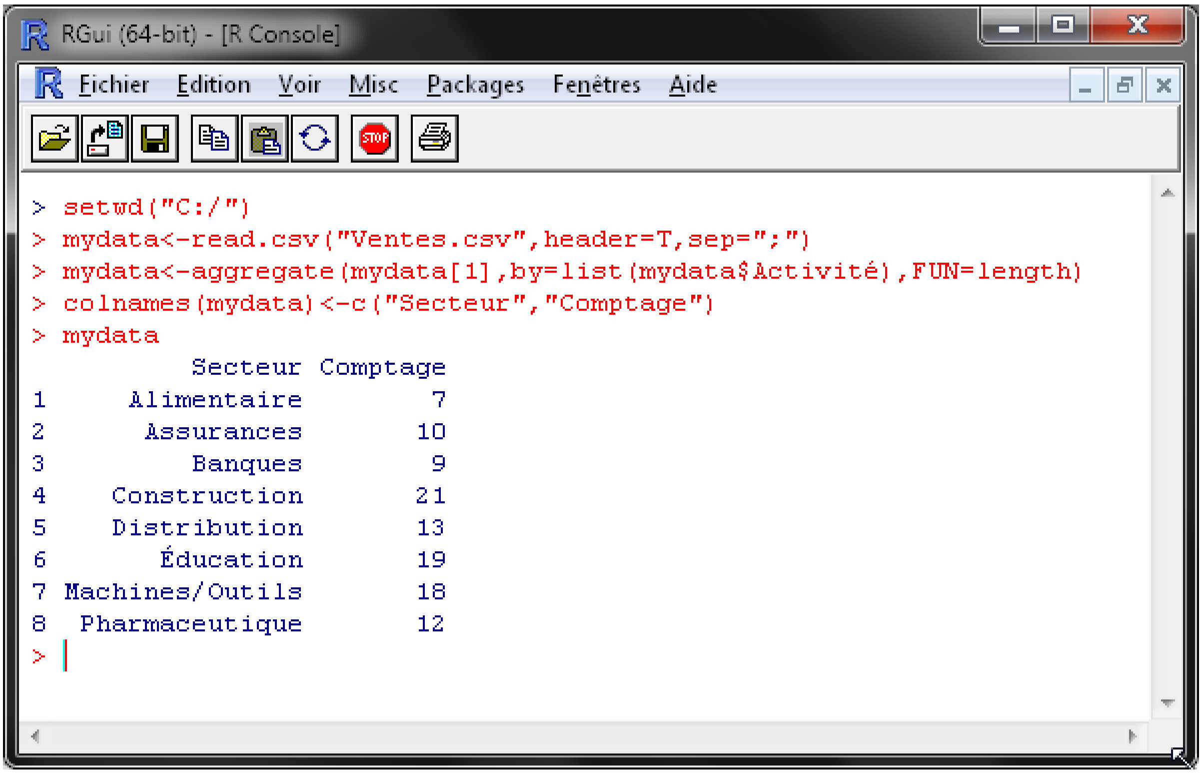
- Pour extraire les valeurs dans des vecteurs il est possible de transformer la table en un data frame en utilisant la commande **as.data.frame(***nom_de_la_table***)**.

- Il suffit d'écrire: **prop.table( )*100** pour avoir toutes les valeurs en %

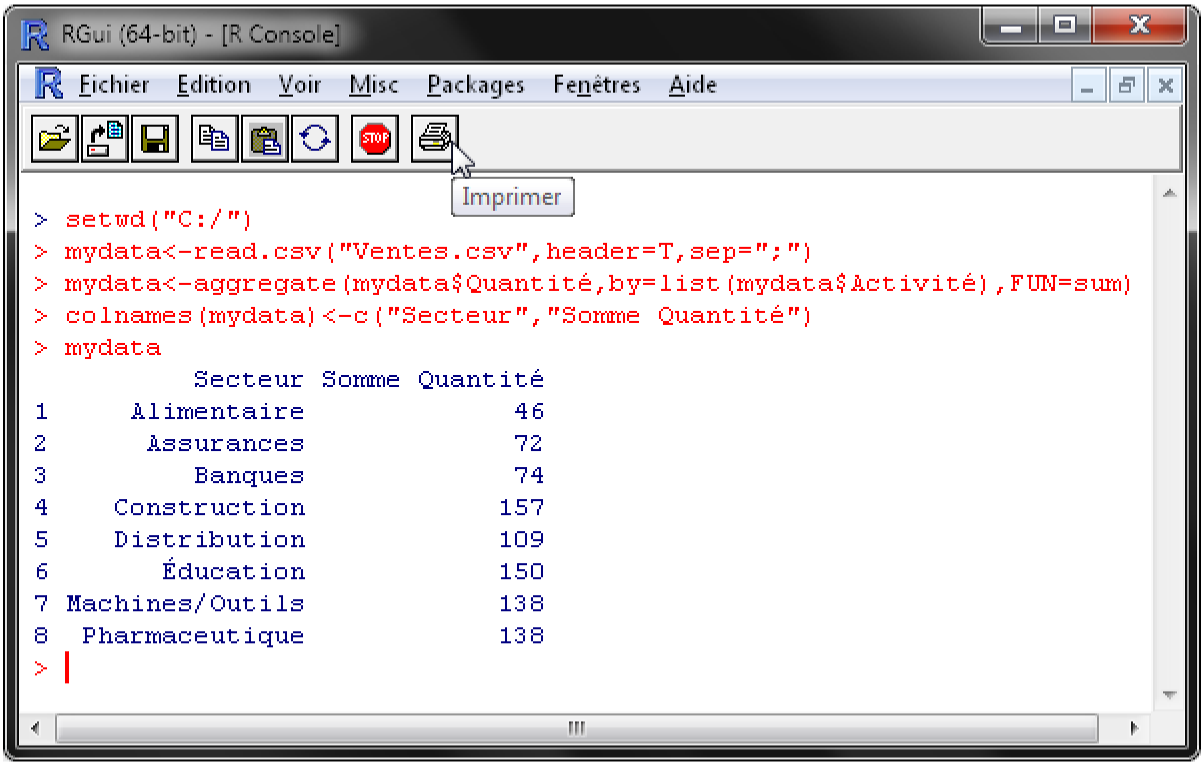
Enfin une question classique: comment mettre le symbole "%"? Eh bien ce n'est pas intuitif mais voilà en utilisant **sprintf( )**:



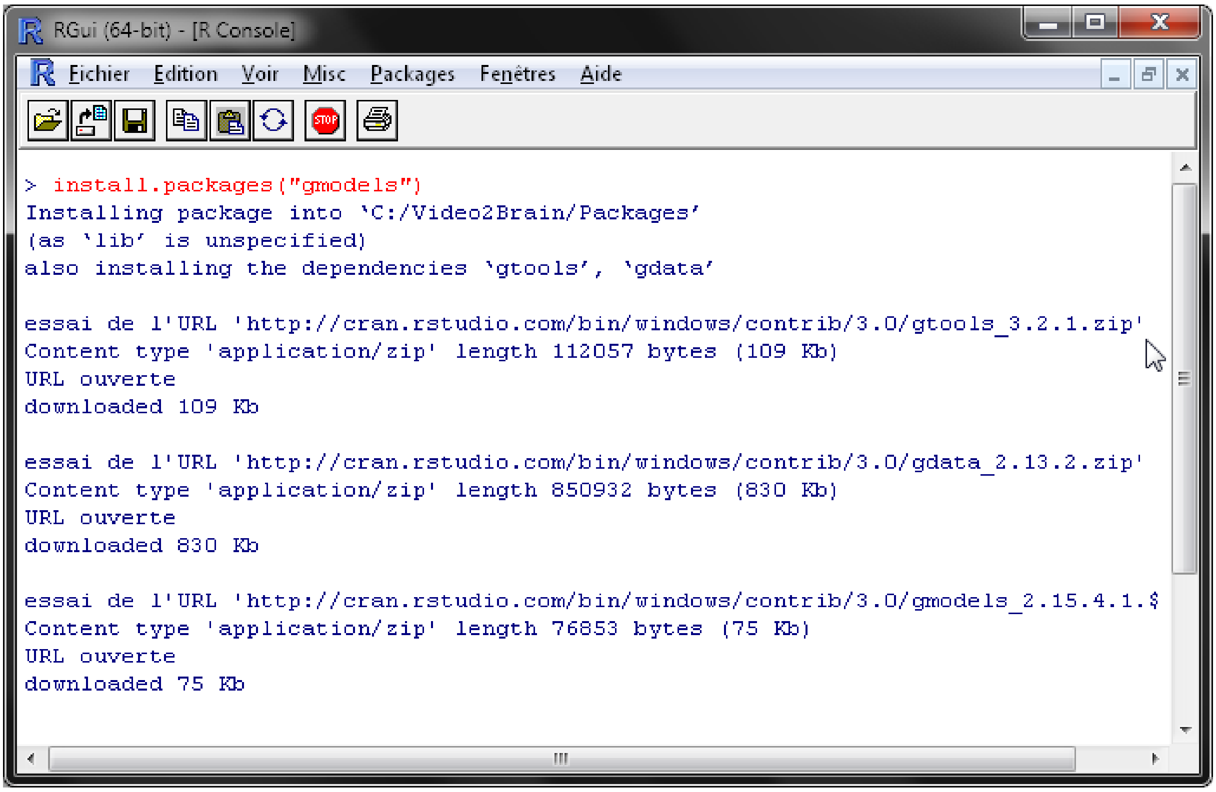
Nous pouvons également synthétiser des données (très utile pour des diagrammes à barre ou des camemberts), avec la commande **aggregate**( ) :



Ou avec autre chose qu'un simple comptage:



Voyons maintenance ce qu'il est possible de faire avec le package nommé **gmodels**:



Nous avons alors avec sa fonction **CrossTable( )**:

```
> library(gmodels)
> CrossTable(mydata$FacturePayee,mydata$Article)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

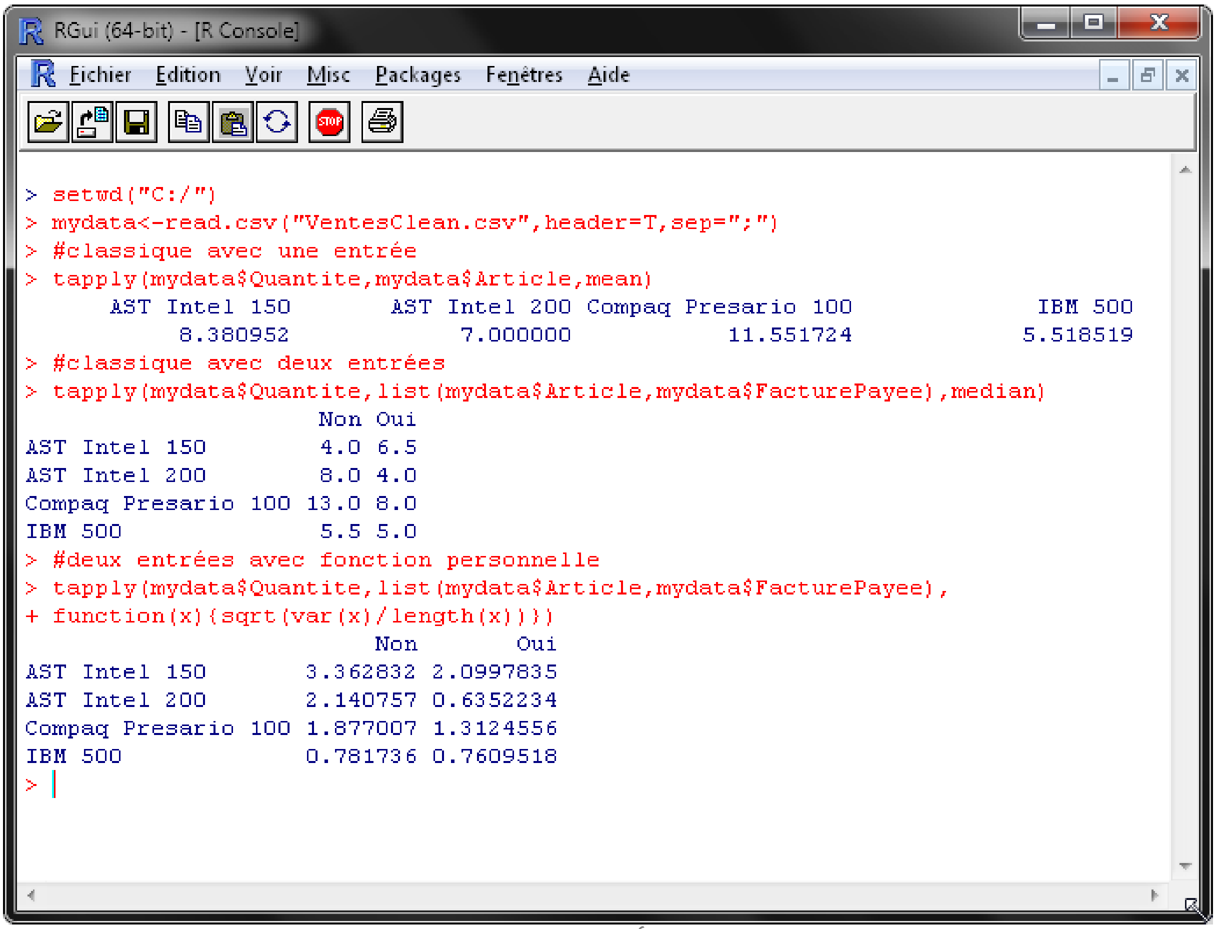

Total Observations in Table:  109


                    | mydata$Article
mydata$FacturePayee |      AST Intel 150 |      AST Intel 200 | Compaq Presario 100 |            IBM 500 |          Row Total |
--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
                Non |                  9 |                 13 |                 14 |                 12 |                 48 |
                    |              0.007 |              0.085 |              0.118 |              0.001 |                    |
                    |              0.188 |              0.271 |              0.292 |              0.250 |              0.440 |
                    |              0.429 |              0.406 |              0.483 |              0.444 |                    |
                    |              0.083 |              0.119 |              0.128 |              0.110 |                    |
--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
                Oui |                 12 |                 19 |                 15 |                 15 |                 61 |
                    |              0.005 |              0.067 |              0.093 |              0.001 |                    |
                    |              0.197 |              0.311 |              0.246 |              0.246 |              0.560 |
                    |              0.571 |              0.594 |              0.517 |              0.556 |                    |
                    |              0.110 |              0.174 |              0.138 |              0.138 |                    |
--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
       Column Total |                 21 |                 32 |                 29 |                 27 |                109 |
                    |              0.193 |              0.294 |              0.266 |              0.248 |                    |
--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
```

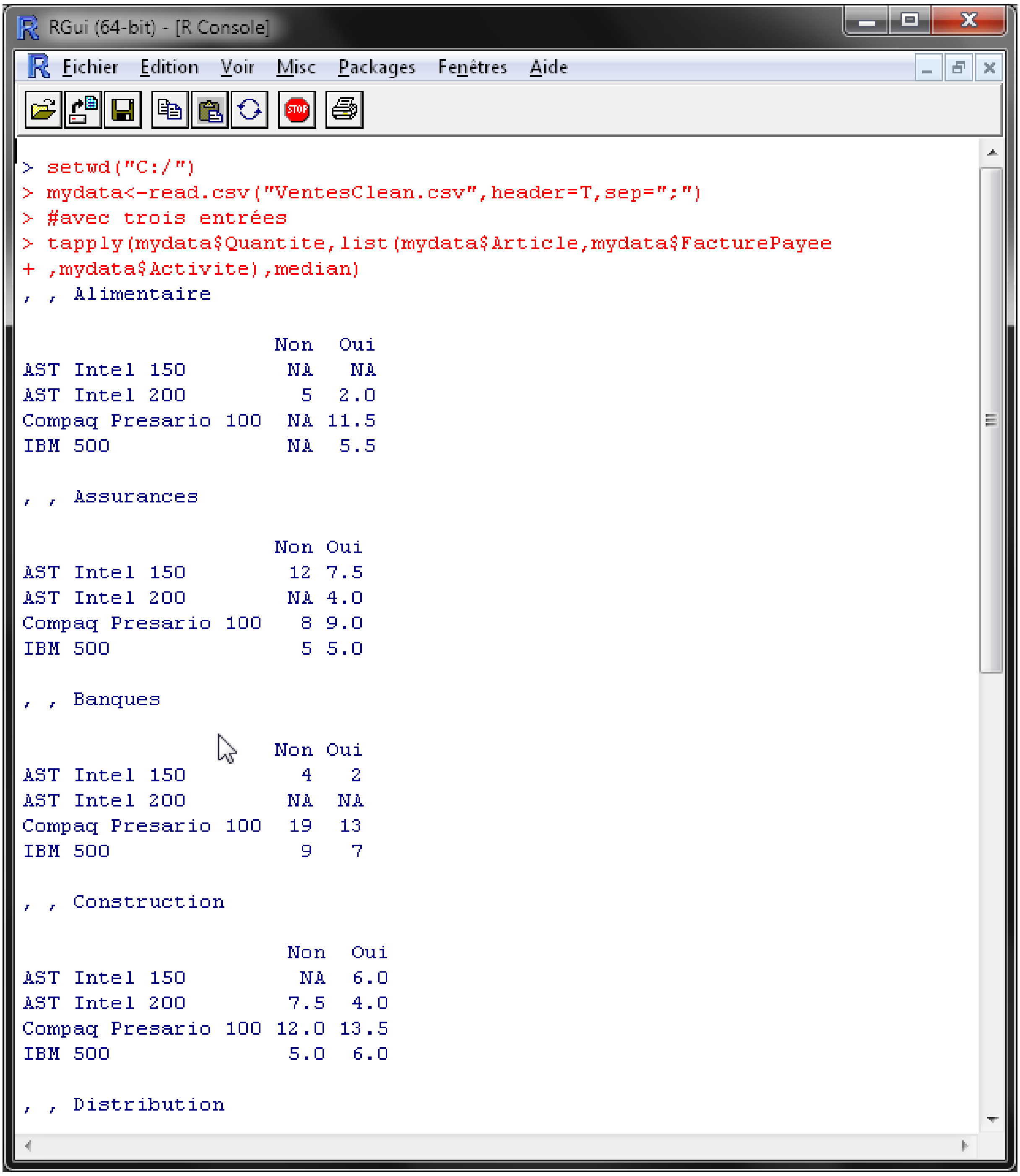
Ou pour revenir à du plus simple sans package utilisant la commande native **sapply( )** relativement souvent utilisée dans des scripts R:

```
> setwd("C:/")
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> sapply(mydata[c("Quantité","Prix.total.avec.rabais")], function(x) (c(mean=mean(x),sd=sd(x))))
     Quantité Prix.total.avec.rabais
mean 8.110092               18784.54
sd   6.314797               15122.27
> 
```

Ou encore sans packages en utilisant la commande native **tapply( )** :



Ou avec trois entrées:



```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> #avec trois entrées
> tapply(mydata$Quantite,list(mydata$Article,mydata$FacturePayee
+ ,mydata$Activite),median)
, , Alimentaire

                     Non  Oui
AST Intel 150         NA   NA
AST Intel 200          5  2.0
Compaq Presario 100   NA 11.5
IBM 500               NA  5.5

, , Assurances

                     Non Oui
AST Intel 150         12 7.5
AST Intel 200         NA 4.0
Compaq Presario 100    8 9.0
IBM 500                5 5.0

, , Banques

                     Non Oui
AST Intel 150          4   2
AST Intel 200         NA  NA
Compaq Presario 100   19  13
IBM 500                9   7

, , Construction

                      Non  Oui
AST Intel 150          NA  6.0
AST Intel 200         7.5  4.0
Compaq Presario 100  12.0 13.5
IBM 500               5.0  6.0

, , Distribution
```

et en utilisant par dessus la commande générique **ftable( )** pour formater cela un peu mieux:

```
> ftable(tapply(mydata$Quantite,list(mydata$Activite,mydata$FacturePayee
+ ,mydata$Article),median))
                                AST Intel 150 AST Intel 200 Compaq Presario 100 IBM 500

Alimentaire      Non                       NA           5.0                  NA      NA
                 Oui                       NA           2.0                11.5     5.5
Assurances       Non                     12.0            NA                 8.0     5.0
                 Oui                      7.5           4.0                 9.0     5.0
Banques          Non                      4.0            NA                19.0     9.0
                 Oui                      2.0            NA                13.0     7.0
Construction     Non                       NA           7.5                12.0     5.0
                 Oui                      6.0           4.0                13.5     6.0
Distribution     Non                       NA          15.5                10.0     3.0
                 Oui                       NA           7.0                 8.0     2.0
Éducation        Non                      1.5            NA                21.0     2.5
                 Oui                     12.0           3.0                11.0     6.5
Machines/Outils  Non                      5.0           4.0                 6.0     7.0
                 Oui                      6.0           3.0                 6.0     4.0
Pharmaceutique   Non                       NA          17.5                15.0     7.5
                 Oui                      7.0           4.0                  NA     2.0
> 
```

Ce dernier cas ressemble aux tableaux croisés dynamique du tableur Microsoft Excel mais en beaucoup moins flexible, rapide et esthétique.

Enfin signalons la fonction **xyTable( )** que nous utiliserons plus loin pour faire des régressions particulières. Considérons par exemple que nous souhaitons compter le nombre de quantités de commandes qui se trouvent près d'un arrondi au millier donnés des montants de facture. Nous avons alors:

```
> x<-mydata$Quantité
> y<-mydata$Prix.total.avec.rabais
> df<-xyTable(x,y,digit=1);df
$x
 [1]  1  2  2  3  3  3  4  4  4  5  5  5  6  6  7  7  8  8  8  9  9 10 10 10 20 20 20 20 20
[30] 20 30 30

$y
 [1]  3000  5000  6000  7000  8000 10000  7000  9000 10000  8000 10000 20000 10000 20000
[15] 10000 20000 10000 20000 30000 20000 30000 20000 30000 40000 20000 30000 40000 50000
[29] 60000 70000 50000 90000

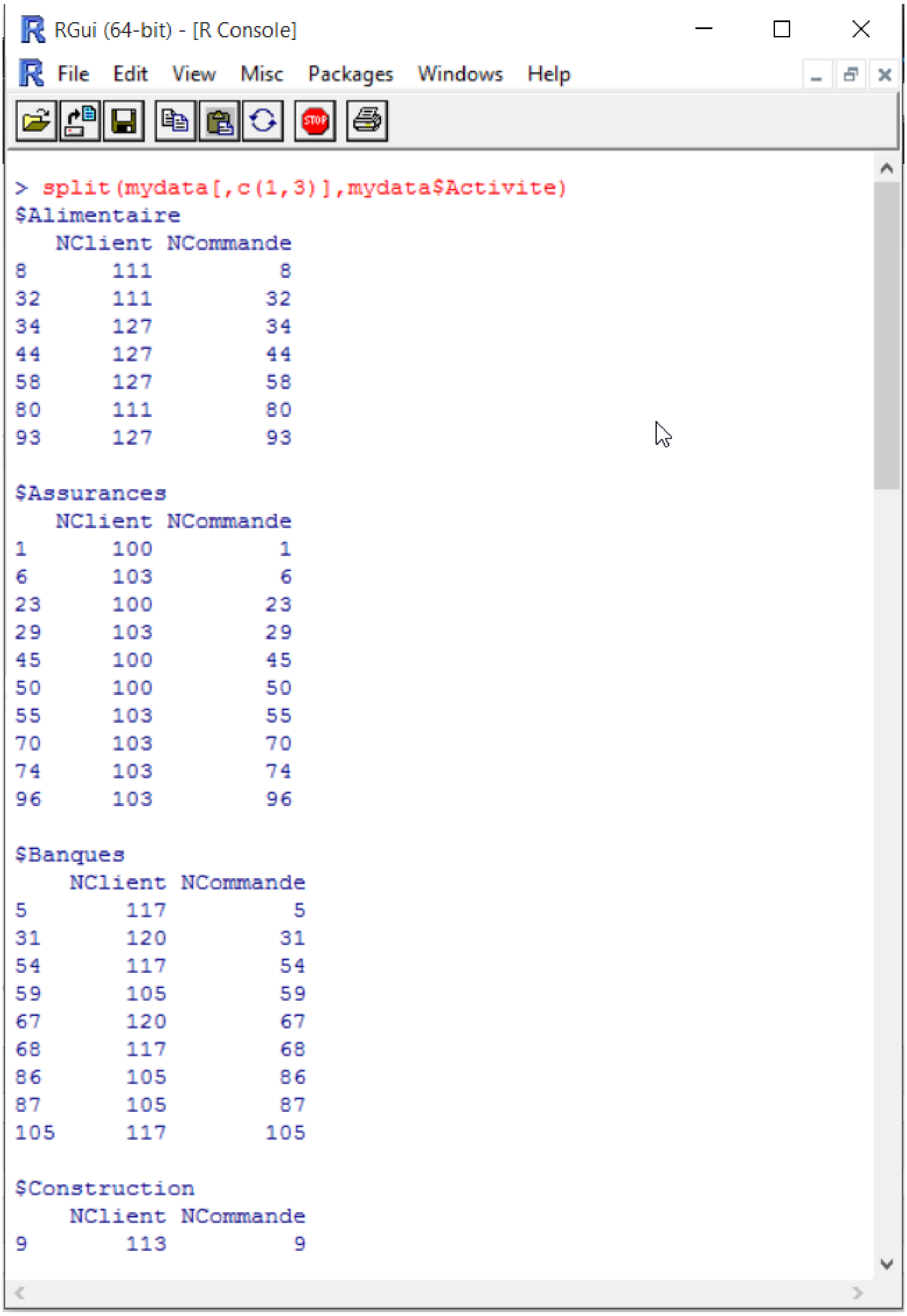
$number
 [1]  2  8  5  4  2  3  1  3  7  4  5  2  7  5  1  7  3  1  4  2  1 12  6  1  2  3  1  1  1
[30]  3  1  1

> 
```

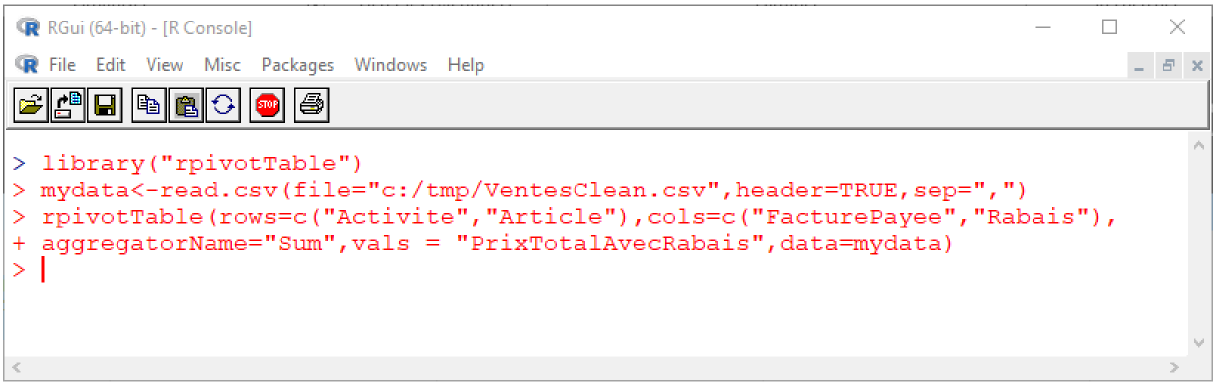
Ainsi il y a:

- 2 commandes dont la quantité est 1 et le montant de facture près de 3'000
- 8 commandes dont la quantité est 2 et le montant de facture près de 5'000
- 5 commandes dont la quantité est 2 et le montant de facture près de 6'000
- Et ainsi de suite…

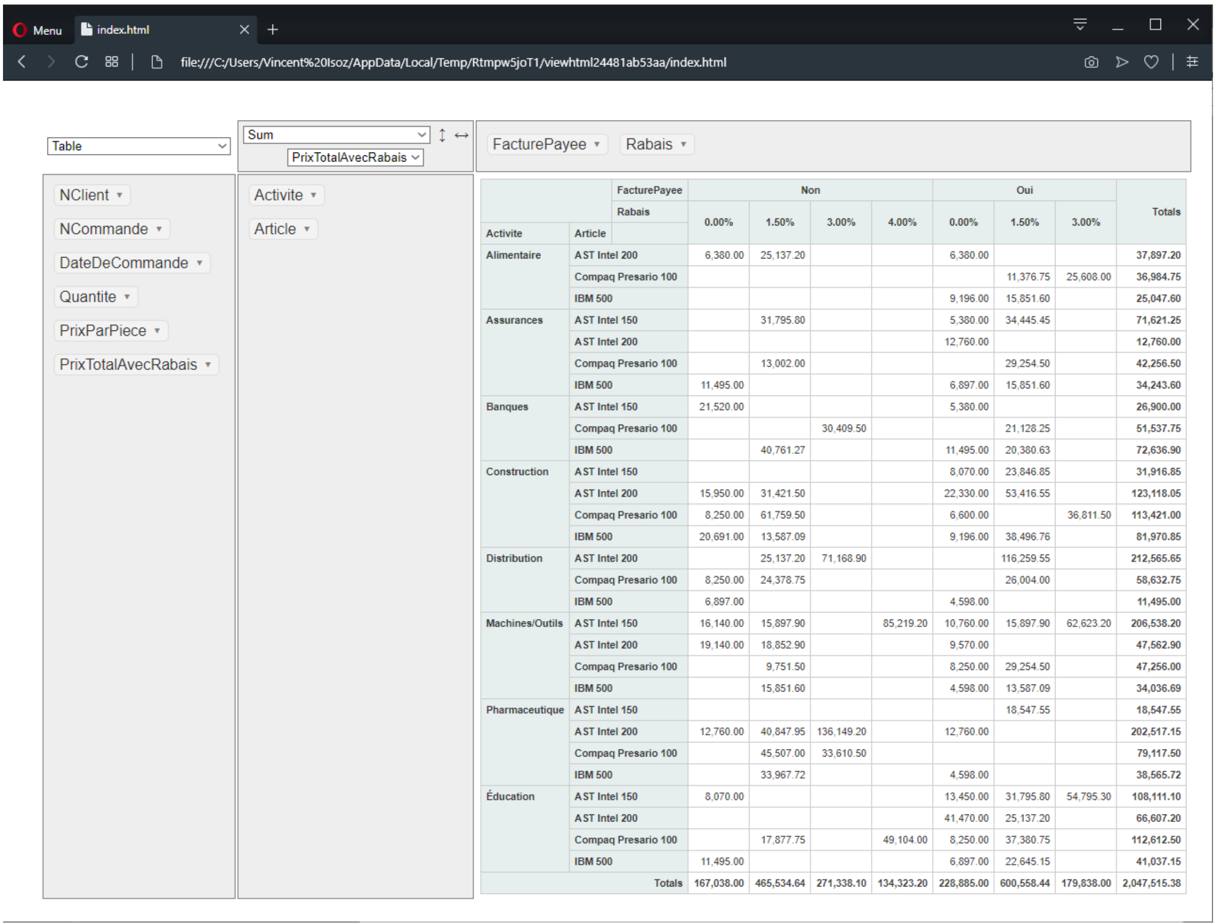
Voyons maintenant la commande native **split( )** agit sur des data frames:



Enfin, même si je ne suis pas un fan des application R qui ouvrent le navigateur Internet… pour ceux qui aiment vraiment les visuels très proches des tableaux croisés dynamiques de Microsoft Excel, nous avons le package **rpivotTable** et sa fonction du même nom **rpivotTable( )**:



Ce qui donnera dans votre navigateur par défaut:



	FacturePayee	Non				Oui			Totals
	Rabais	0.00%	1.50%	3.00%	4.00%	0.00%	1.50%	3.00%	
Activite	Article								
Alimentaire	AST Intel 200	6,380.00	25,137.20			6,380.00			37,897.20
	Compaq Presario 100						11,376.75	25,608.00	36,984.75
	IBM 500					9,196.00	15,851.60		25,047.60
Assurances	AST Intel 150		31,795.80			5,380.00	34,445.45		71,621.25
	AST Intel 200					12,760.00			12,760.00
	Compaq Presario 100		13,002.00				29,254.50		42,256.50
	IBM 500	11,495.00				6,897.00	15,851.60		34,243.60
Banques	AST Intel 150	21,520.00				5,380.00			26,900.00
	Compaq Presario 100			30,409.50			21,128.25		51,537.75
	IBM 500		40,761.27			11,495.00	20,380.63		72,636.90
Construction	AST Intel 150					8,070.00	23,846.85		31,916.85
	AST Intel 200	15,950.00	31,421.50			22,330.00	53,416.55		123,118.05
	Compaq Presario 100	8,250.00	61,759.50			6,600.00		36,811.50	113,421.00
	IBM 500	20,691.00	13,587.09			9,196.00	38,496.76		81,970.85
Distribution	AST Intel 200		25,137.20	71,168.90			116,259.55		212,565.65
	Compaq Presario 100	8,250.00	24,378.75				26,004.00		58,632.75
	IBM 500	6,897.00				4,598.00			11,495.00
Machines/Outils	AST Intel 150	16,140.00	15,897.90		85,219.20	10,760.00	15,897.90	62,623.20	206,538.20
	AST Intel 200	19,140.00	18,852.90			9,570.00			47,562.90
	Compaq Presario 100		9,751.50			8,250.00	29,254.50		47,256.00
	IBM 500		15,851.60			4,598.00	13,587.09		34,036.69
Pharmaceutique	AST Intel 150						18,547.55		18,547.55
	AST Intel 200	12,760.00	40,847.95	136,149.20		12,760.00			202,517.15
	Compaq Presario 100		45,507.00	33,610.50					79,117.50
	IBM 500		33,967.72			4,598.00			38,565.72
Éducation	AST Intel 150	8,070.00				13,450.00	31,795.80	54,795.30	108,111.10
	AST Intel 200					41,470.00	25,137.20		66,607.20
	Compaq Presario 100		17,877.75		49,104.00	8,250.00	37,380.75		112,612.50
	IBM 500	11,495.00				6,897.00	22,645.15		41,037.15
	Totals	167,038.00	465,534.64	271,338.10	134,323.20	228,885.00	600,558.44	179,838.00	2,047,515.38

Exemple 95.: Transformer des données longitudinales en transervales (et l'inverse) (dépivoter vs pivoter)

R 3.0.2

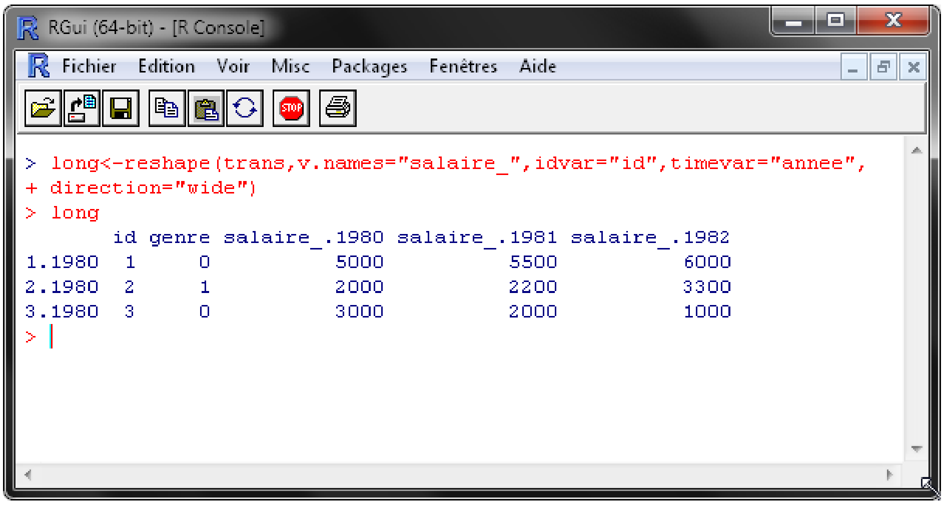
Voilà une situation dans laquelle nous nous retrouvons fréquemment lorsque nous travaillons avec des gens utilisant un tableur. Considérons le fichier suivant:

```
> mydata<-read.csv("C:/DonneesLongitudinales.csv",header=T,sep=";")
> mydata
  id genre salaire_1980 salaire_1981 salaire_1982
1  1     0         5000         5500         6000
2  2     1         2000         2200         3300
3  3     0         3000         2000         1000
>
```

avec la commande **reshape( )** du package **reshape2** nous allons transformer cela en données transversales (la syntaxe de la transformation est peu intuitive):


```
> library(reshape2)
> trans<-reshape(mydata,idvar="id",varying=list(names(mydata)[3:5]),
+ v.names="salaire_",timevar="annee",times=1980:1982,direction="long")
> trans
       id genre annee salaire_
1.1980  1     0  1980     5000
2.1980  2     1  1980     2000
3.1980  3     0  1980     3000
1.1981  1     0  1981     5500
2.1981  2     1  1981     2200
3.1981  3     0  1981     2000
1.1982  1     0  1982     6000
2.1982  2     1  1982     3300
3.1982  3     0  1982     1000
>
```

et l'opération inverse (dont la syntaxe est beaucoup plus aisée):

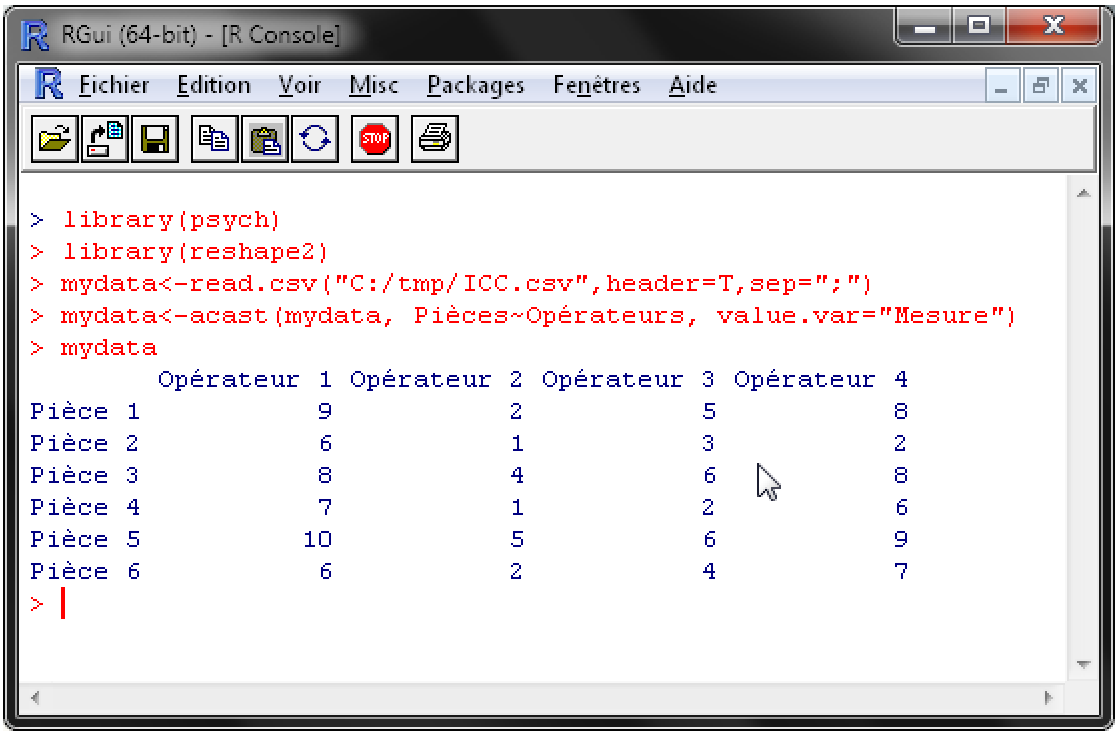


Voyons un autre exemple:

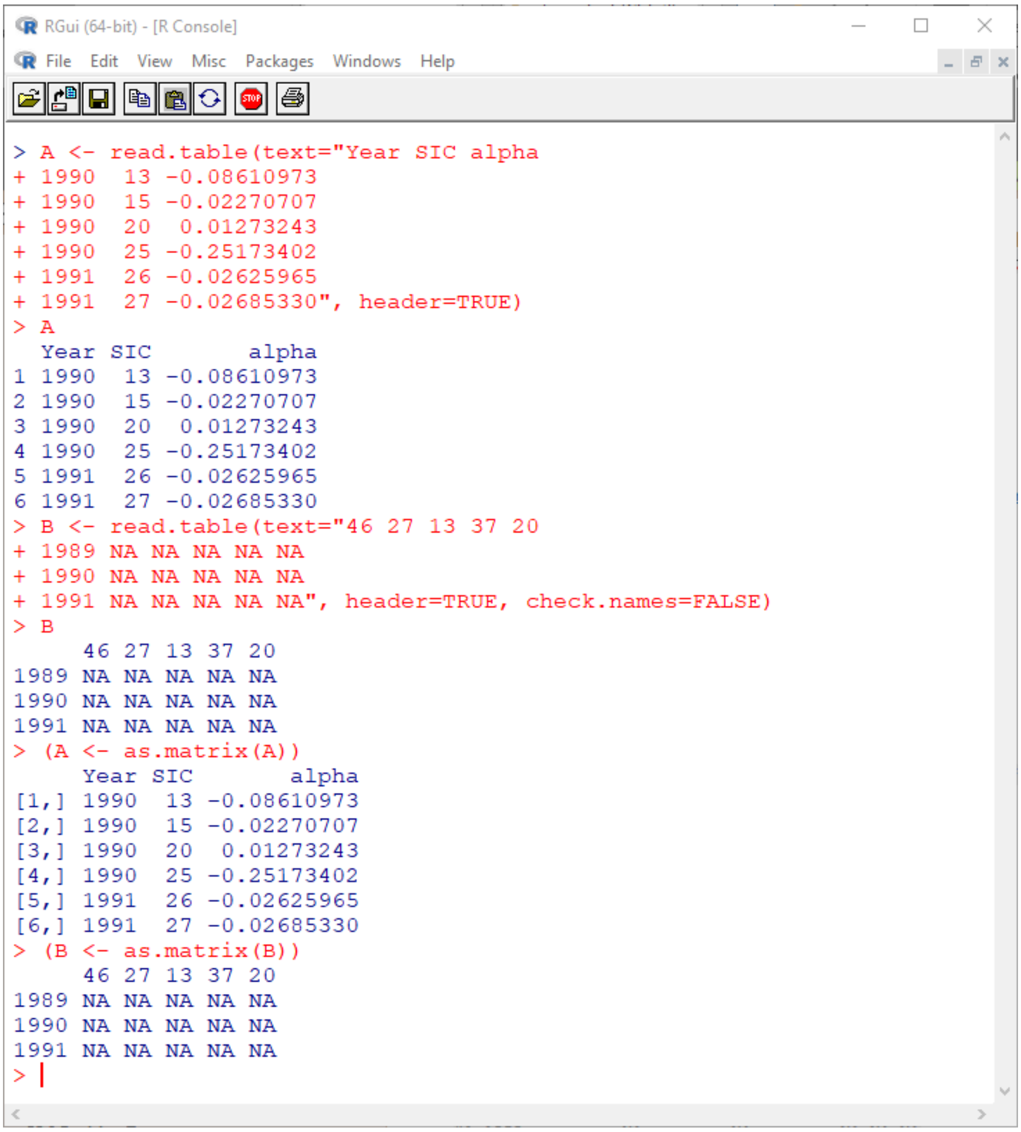


Nous utilisons le package **psych** pour faire l'analyse avec la fonction **ICC( )** et le package **reshape2** avec la fonction **acast( )**[11] pour restructurer (pivoter) les données:

[11] Le "a" de acast( ) signifie "array" car la fonction dcast( ) du même package renvoie un "data frame".



Nous pouvons traiter des cas compliqués et assez courants dans le monde Microsoft Excel. Pour cela, construisons d'abord les deux jeux de données nécessaires pour l'exemple:



Nous souhaiterions retrouver les valeurs de la table *A* dans la matrice *B*. Pour cela subtilement, il faudra écrire:

```
> #on transforme d'abord la matrice B
> Bm <- stack(as.data.frame(B))
> Bm$ind <- as.character(Bm$ind)
> Bm$year <- rownames(B)
> Bm
   values ind year
1      NA  46 1989
2      NA  46 1990
3      NA  46 1991
4      NA  27 1989
5      NA  27 1990
6      NA  27 1991
7      NA  13 1989
8      NA  13 1990
9      NA  13 1991
10     NA  37 1989
11     NA  37 1990
12     NA  37 1991
13     NA  20 1989
14     NA  20 1990
15     NA  20 1991
> #et on fusionne
> C <- merge(Bm[,c("ind", "year")],as.data.frame(A),by.x=c("ind", "year"),
+            by.y=c("SIC", "Year"),all.x=TRUE)
> C
   ind year       alpha
1   13 1989          NA
2   13 1990 -0.08610973
3   13 1991          NA
4   20 1989          NA
5   20 1990  0.01273243
6   20 1991          NA
7   27 1989          NA
8   27 1990          NA
9   27 1991 -0.02685330
10  37 1989          NA
11  37 1990          NA
12  37 1991          NA
13  46 1989          NA
14  46 1990          NA
15  46 1991          NA
> #et on reformatte en longitudinal
> library(reshape2)
> dcast(C, year~ind)
Using alpha as value column: use value.var to override.
  year          13         20         27 37 46
1 1989          NA         NA         NA NA NA
2 1990 -0.08610973 0.01273243         NA NA NA
3 1991          NA         NA -0.0268533 NA NA
> 
```

Enfin, voyons encore un autre exemple relativement courant et qui pose problème dans le monde Microsoft Excel et Microsoft Power BI, mais qui s'avère très facile avec R en utilisant les fonctions **dcast( )**, **setDT( )** et **rowid( )** du package data.table:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("data.table")
> df <- iris
> #nous voulons pivoter les noms des espèces relativement à la longuer de sépales
> head(iris,10)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1:          5.1         3.5          1.4         0.2  setosa
 2:          4.9         3.0          1.4         0.2  setosa
 3:          4.7         3.2          1.3         0.2  setosa
 4:          4.6         3.1          1.5         0.2  setosa
 5:          5.0         3.6          1.4         0.2  setosa
 6:          5.4         3.9          1.7         0.4  setosa
 7:          4.6         3.4          1.4         0.3  setosa
 8:          5.0         3.4          1.5         0.2  setosa
 9:          4.4         2.9          1.4         0.2  setosa
10:          4.9         3.1          1.5         0.1  setosa
> dcast(setDT(df), rowid(Species)~Species, value.var = "Sepal.Length")
    Species setosa versicolor virginica
 1:       1    5.1        7.0       6.3
 2:       2    4.9        6.4       5.8
 3:       3    4.7        6.9       7.1
 4:       4    4.6        5.5       6.3
 5:       5    5.0        6.5       6.5
 6:       6    5.4        5.7       7.6
 7:       7    4.6        6.3       4.9
 8:       8    5.0        4.9       7.3
 9:       9    4.4        6.6       6.7
10:      10    4.9        5.2       7.2
11:      11    5.4        5.0       6.5
12:      12    4.8        5.9       6.4
13:      13    4.8        6.0       6.8
14:      14    4.3        6.1       5.7
15:      15    5.8        5.6       5.8
16:      16    5.7        6.7       6.4
17:      17    5.4        5.6       6.5
18:      18    5.1        5.8       7.7
19:      19    5.7        6.2       7.7
20:      20    5.1        5.6       6.0
21:      21    5.4        5.9       6.9
22:      22    5.1        6.1       5.6
23:      23    4.6        6.3       7.7
24:      24    5.1        6.1       6.3
25:      25    4.8        6.4       6.7
26:      26    5.0        6.6       7.2
27:      27    5.0        6.8       6.2
28:      28    5.2        6.7       6.1
29:      29    5.2        6.0       6.4
30:      30    4.7        5.7       7.2
31:      31    4.8        5.5       7.4
32:      32    5.4        5.5       7.9
33:      33    5.2        5.8       6.4
34:      34    5.5        6.0       6.3
35:      35    4.9        5.4       6.1
36:      36    5.0        6.0       7.7
```

Bingo! Opération réussie!

Exemple 96.: Tableau disjonctif complet et tableau de Burt

R 3.6.3

Comme nous le savons de par notre étude théorique de l'analyse factorielle des correspondance et l'analyse factorielle multiple, ces dernières utilisent toutes deux un tableau disjonctif complet et le tableau de Burt. Voyons comment obtenir cela à partir d'un simple exemple vérifiable à la main à l'aide du package **Laterality** et sa fonction **laterAFCM( )**:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("Laterality")
>
> (mydata <- data.frame(Genre=c("Femme", "Homme", "Homme", "Femme")
+ ,Couleur=c("Rouge", "Bleu", "Vert", "Rouge")
+ ,Age=c(20,25,30,35)))
  Genre Couleur Age
1 Femme   Rouge  20
2 Homme    Bleu  25
3 Homme    Vert  30
4 Femme   Rouge  35
>
> laterAFCM(mydata)$Datadisj
  Genre.Femme Genre.Homme Couleur.Bleu Couleur.Rouge Couleur.Vert Age.20 Age.25 Age.30 Age.35
1           1           0            0             1            0      1      0      0      0
2           0           1            1             0            0      0      1      0      0
3           0           1            0             0            1      0      0      1      0
4           1           0            0             1            0      0      0      0      1
> laterAFCM(mydata)$SumcolDatadisj
  Genre.Femme   Genre.Homme  Couleur.Bleu Couleur.Rouge  Couleur.Vert        Age.20        Age.25        Age.30        Age.35
            2             2             1             2             1             1             1             1             1
> laterAFCM(mydata)$Databurt
              Genre.Femme Genre.Homme Couleur.Bleu Couleur.Rouge Couleur.Vert Age.20 Age.25 Age.30 Age.35
Genre.Femme             2           0            0             2            0      1      0      0      1
Genre.Homme             0           2            1             0            1      0      1      1      0
Couleur.Bleu            0           1            1             0            0      0      1      0      0
Couleur.Rouge           2           0            0             2            0      1      0      0      1
Couleur.Vert            0           1            0             0            1      0      0      1      0
Age.20                  1           0            0             1            0      1      0      0      0
Age.25                  0           1            1             0            0      0      1      0      0
Age.30                  0           1            0             0            1      0      0      1      0
Age.35                  1           0            0             1            0      0      0      0      1
> |
```

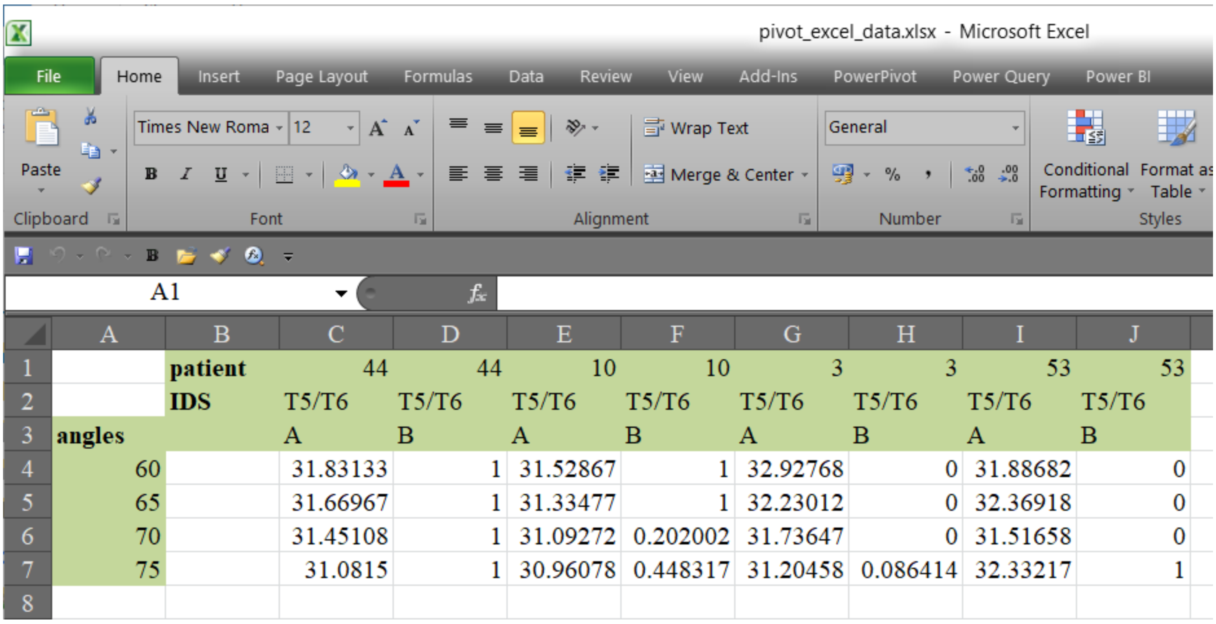
Exemple 97.: Dépivoter des données de Microsoft Excel

R 3.5.1

L'exemple qui suit est totalement copié de l'excellent billet de blog suivant de Abhijit Dasgupta:

https://webbedfeet.netlify.com/post/tidying-messy-excel-data-tidyxl/

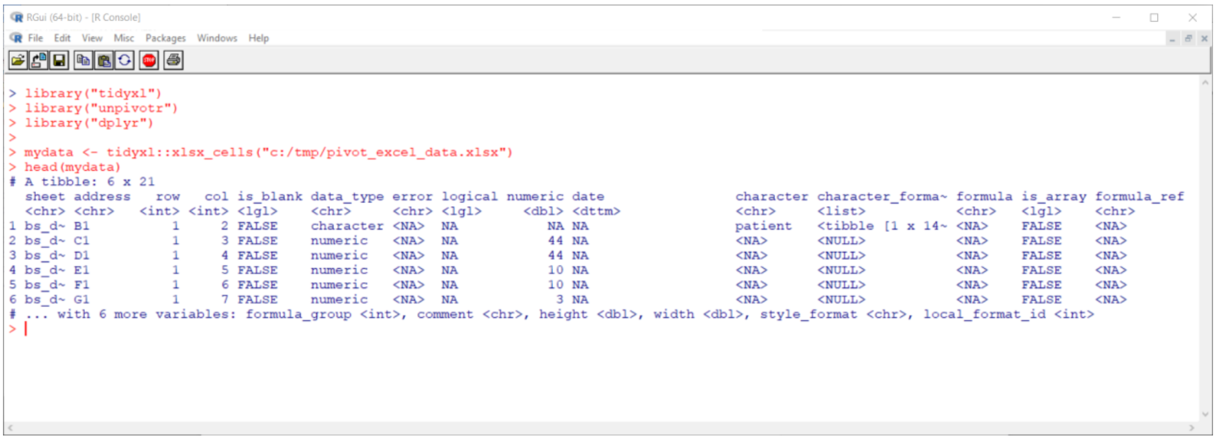
Considérons le fichier Microsoft Excel suivant dont les données sont assez typiquement structurées pour ceux qui connaissent les tableaux croisés dynamiques:



	A	B	C	D	E	F	G	H	I	J
1		patient	44	44	10	10	3	3	53	53
2		IDS	T5/T6	T5/T6	T5/T6	T5/T6	T5/T6	T5/T6	T5/T6	T5/T6
3	angles		A	B	A	B	A	B	A	B
4	60		31.83133	1	31.52867	1	32.92768	0	31.88682	0
5	65		31.66967	1	31.33477	1	32.23012	0	32.36918	0
6	70		31.45108	1	31.09272	0.202002	31.73647	0	31.51658	0
7	75		31.0815	1	30.96078	0.448317	31.20458	0.086414	32.33217	1
8										

Notre objectif va être ici de mettre à plat ces données. Pour cela nous allons faire usage des packages **unpivotr**, **tidyxl** et **dplyr** et des fonctions majeures suivantes de ces packages: **xlsx_cells( )** , **behead( )** , **filter( )** et **spatter( )** .

L'ensemble du script (non-trivial comme le dit l'auteur lui-même) permettant d'arriver au résultat attendu est le suivant d'abord pour l'import:



```
> library("tidyxl")
> library("unpivotr")
> library("dplyr")
>
> mydata <- tidyxl::xlsx_cells("c:/tmp/pivot_excel_data.xlsx")
> head(mydata)
# A tibble: 6 x 21
  sheet address   row   col is_blank data_type error logical numeric date                character character_forma~ formula is_array formula_ref
  <chr> <chr>   <int> <int> <lgl>    <chr>     <chr> <lgl>     <dbl> <dttm>              <chr>     <list>            <chr>   <lgl>    <chr>
1 bs_d~ B1          1     2 FALSE    character <NA>  NA           NA NA                  patient   <tibble [1 x 14~  <NA>    FALSE    <NA>
2 bs_d~ C1          1     3 FALSE    numeric   <NA>  NA           44 NA                  <NA>      <NULL>            <NA>    FALSE    <NA>
3 bs_d~ D1          1     4 FALSE    numeric   <NA>  NA           44 NA                  <NA>      <NULL>            <NA>    FALSE    <NA>
4 bs_d~ E1          1     5 FALSE    numeric   <NA>  NA           10 NA                  <NA>      <NULL>            <NA>    FALSE    <NA>
5 bs_d~ F1          1     6 FALSE    numeric   <NA>  NA           10 NA                  <NA>      <NULL>            <NA>    FALSE    <NA>
6 bs_d~ G1          1     7 FALSE    numeric   <NA>  NA            3 NA                  <NA>      <NULL>            <NA>    FALSE    <NA>
# ... with 6 more variables: formula_group <int>, comment <chr>, height <dbl>, width <dbl>, style_format <chr>, local_format_id <int>
>
```

et ensuite pour le traitement:

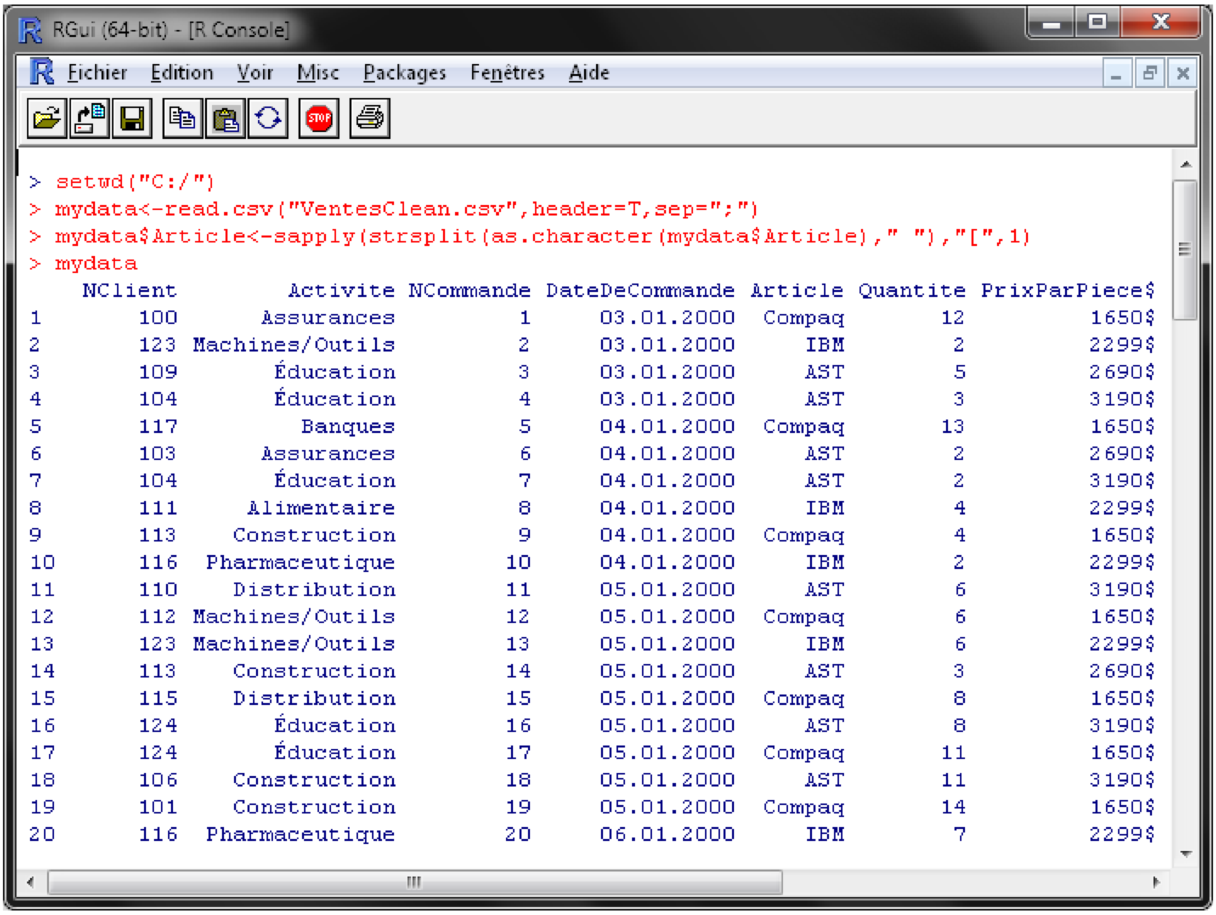
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> cleanData <- function(d) {
+   final <- d %>%
+     select(row, col, data_type, numeric, character) %>%
+     behead('N', ID) %>%  # Extract column headers
+     behead('N', spine) %>%
+     behead('N', variable) %>%
+     behead('W', angles) %>% # angles as row header
+     select(numeric, ID:angles, data_type, -spine) %>% # all vars are numeric
+     filter(variable %in% c('A', 'B')) %>% # Kills off some extra columns
+     spatter(variable) # Spreads, using data_type, numeric
+   return(final)
+ }
>
> (cleanData(mydata))
# A tibble: 16 x 4
   ID    angles     A      B
   <chr>  <dbl> <dbl>  <dbl>
 1 10        60  31.5 1
 2 10        65  31.3 1
 3 10        70  31.1 0.202
 4 10        75  31.0 0.448
 5 3         60  32.9 0
 6 3         65  32.2 0
 7 3         70  31.7 0
 8 3         75  31.2 0.0864
 9 44        60  31.8 1
10 44        65  31.7 1
11 44        70  31.5 1
12 44        75  31.1 1
13 53        60  31.9 0
14 53        65  32.4 0
15 53        70  31.5 0
16 53        75  32.3 1
>
```

Exemple 98.: Traiter un vecteur de texte

R 3.0.2

Voyons un cas courant dans les data frame consistant à décomposer le contenu d'une colonne contenant des chaînes de caractères en en prenant qu'une seule partie en appliquant la commande **sapply( )** dont la syntaxe est loi d'être intuitive dans le cas présent (il faut au préalable convertir le typage de la colonne des textes avec la commande **as.character( )**):

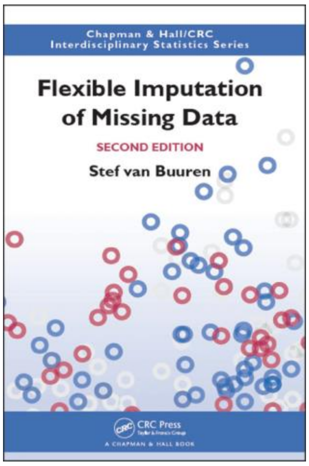



```
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata$Article<-sapply(strsplit(as.character(mydata$Article)," "),"[",1)
> mydata
   NClient        Activite NCommande DateDeCommande Article Quantite PrixParPiece$
1      100     Assurances         1     03.01.2000  Compaq       12         1650$
2      123 Machines/Outils         2     03.01.2000     IBM        2         2299$
3      109       Éducation         3     03.01.2000     AST        5         2690$
4      104       Éducation         4     03.01.2000     AST        3         3190$
5      117         Banques         5     04.01.2000  Compaq       13         1650$
6      103      Assurances         6     04.01.2000     AST        2         2690$
7      104       Éducation         7     04.01.2000     AST        2         3190$
8      111     Alimentaire         8     04.01.2000     IBM        4         2299$
9      113    Construction         9     04.01.2000  Compaq        4         1650$
10     116  Pharmaceutique        10     04.01.2000     IBM        2         2299$
11     110    Distribution        11     05.01.2000     AST        6         3190$
12     112 Machines/Outils        12     05.01.2000  Compaq        6         1650$
13     123 Machines/Outils        13     05.01.2000     IBM        6         2299$
14     113    Construction        14     05.01.2000     AST        3         2690$
15     115    Distribution        15     05.01.2000  Compaq        8         1650$
16     124       Éducation        16     05.01.2000     AST        8         3190$
17     124       Éducation        17     05.01.2000  Compaq       11         1650$
18     106    Construction        18     05.01.2000     AST       11         3190$
19     101    Construction        19     05.01.2000  Compaq       14         1650$
20     116  Pharmaceutique        20     06.01.2000     IBM        7         2299$
```

Exemple 99.: Travailler avec des valeurs absentes (NA)

R 3.0.2

Bon je ne suis pas un fan de ce sujet car je considère que le travail doit être correctement fait en amont pour éviter ce type de situations (c'est évidemment une opinion très personnelle). Notons et rappelons que ceux qui sont intéressés à approfondir le sujet, pourront se reporter au superbe ouvrage suivant sur le sujet:



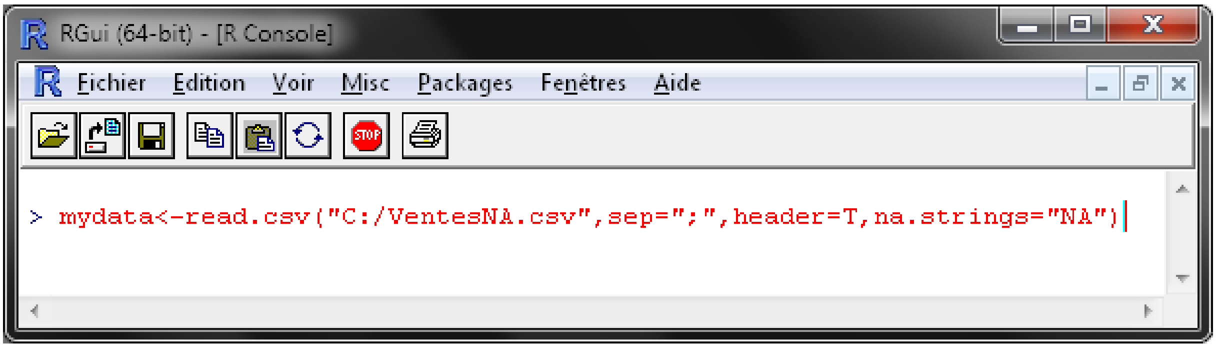
Il existe quelques paramètres et commandes dans R permettant de se débarrasser des valeurs absentes de type NA (Not Available) se trouvant dans un data frame.

Considérons le fichier suivant (remarquez les NA dans la colonne *Activité* et *Quantité*):

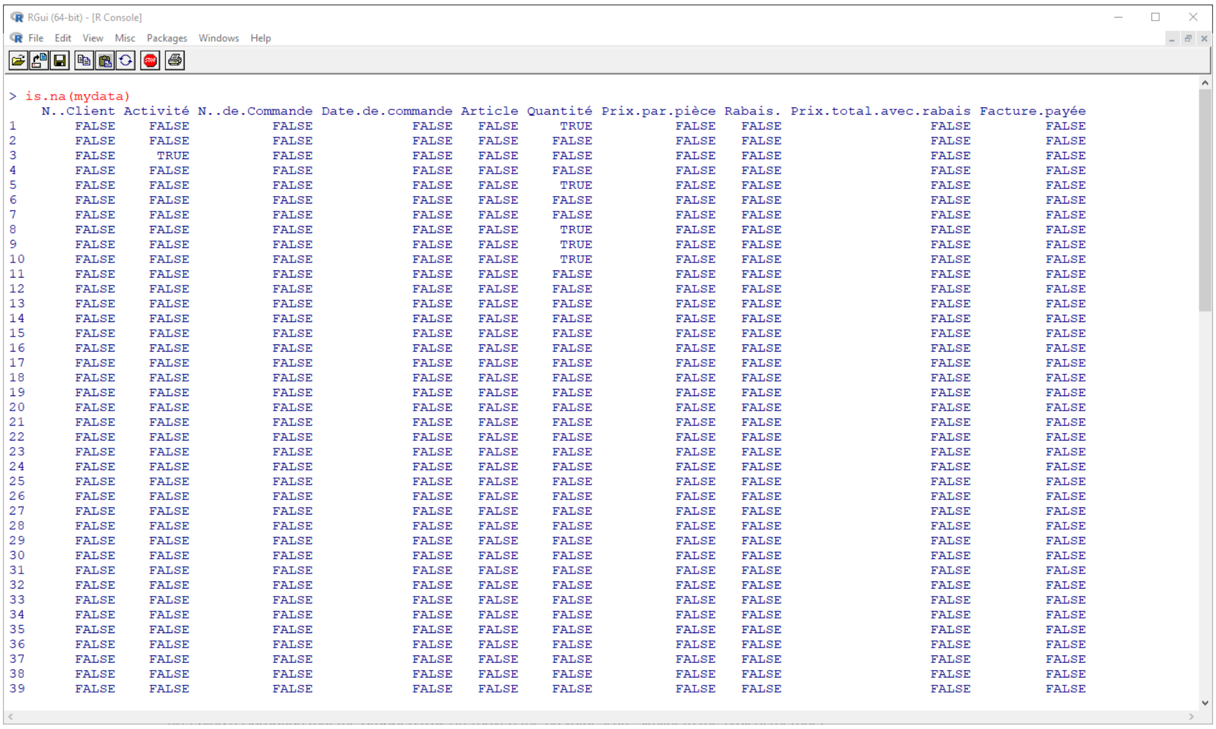
```
VentesNA.csv - Notepad
File Edit Format View Help
"N..Client";"Activité";"N..de.Commande";"Date.de.commande";"Article";"Quantité";"Prix.par.pièce";"Rabais.";"Prix.total.avec.rabais";"Facture.payée"
"1";100;"Assurances";1;"03.01.2000";"Compaq Presario 100";NA;1650;"1.50%";19503;"Oui"
"2";123;"Machines/Outils";2;"03.01.2000";"IBM 500";2;2299;"0.00%";4598;"Oui"
"3";109;NA;3;"03.01.2000";"AST Intel 150";5;2690;"0.00%";13450;"Oui"
"4";104;"Éducation";4;"03.01.2000";"AST Intel 200";3;3190;"0.00%";9570;"Oui"
"5";117;"Banques";5;"04.01.2000";"Compaq Presario 100";NA;1650;"1.50%";21128.25;"Oui"
"6";103;"Assurances";6;"04.01.2000";"AST Intel 150";2;2690;"0.00%";5380;"Oui"
"7";104;"Éducation";7;"04.01.2000";"AST Intel 200";2;3190;"0.00%";6380;"Oui"
"8";111;"Alimentaire";8;"04.01.2000";"IBM 500";NA;2299;"0.00%";9196;"Oui"
"9";113;"Construction";9;"04.01.2000";"Compaq Presario 100";NA;1650;"0.00%";6600;"Oui"
"10";116;"Pharmaceutique";10;"04.01.2000";"IBM 500";NA;2299;"0.00%";4598;"Oui"
"11";110;"Distribution";11;"05.01.2000";"AST Intel 200";6;3190;"1.50%";18852.9;"Oui"
"12";112;"Machines/Outils";12;"05.01.2000";"Compaq Presario 100";6;1650;"1.50%";9751.5;"Oui"
"13";123;"Machines/Outils";13;"05.01.2000";"IBM 500";6;2299;"1.50%";13587.09;"Oui"
"14";113;"Construction";14;"05.01.2000";"AST Intel 150";3;2690;"0.00%";8070;"Oui"
"15";115;"Distribution";15;"05.01.2000";"Compaq Presario 100";8;1650;"1.50%";13002;"Oui"
"16";124;"Éducation";16;"05.01.2000";"AST Intel 200";8;3190;"1.50%";25137.2;"Oui"
"17";124;"Éducation";17;"05.01.2000";"Compaq Presario 100";11;1650;"1.50%";17877.75;"Oui"
"18";106;"Construction";18;"05.01.2000";"AST Intel 200";11;3190;"1.50%";34563.65;"Oui"
"19";101;"Construction";19;"05.01.2000";"Compaq Presario 100";14;1650;"1.50%";22753.5;"Non"
"20";116;"Pharmaceutique";20;"06.01.2000";"IBM 500";7;2299;"1.50%";15851.605;"Non"
"21";112;"Machines/Outils";21;"06.01.2000";"AST Intel 150";6;2690;"1.50%";15897.9;"Oui"
"22";125;"Construction";22;"06.01.2000";"Compaq Presario 100";23;1650;"3.00%";36811.5;"Oui"
"23";100;"Assurances";23;"06.01.2000";"IBM 500";3;2299;"0.00%";6897;"Oui"
"24";125;"Construction";24;"06.01.2000";"AST Intel 200";2;3190;"0.00%";6380;"Oui"
"25";104;"Éducation";25;"07.01.2000";"AST Intel 150";12;2690;"1.50%";31795.8;"Oui"
"26";126;"Machines/Outils";26;"07.01.2000";"AST Intel 150";24;2690;"3.00%";62623.2;"Oui"
"27";121;"Pharmaceutique";27;"07.01.2000";"IBM 500";8;2299;"1.50%";18116.12;"Non"
"28";114;"Distribution";28;"07.01.2000";"AST Intel 200";9;3190;"1.50%";28279.35;"Oui"
"29";103;"Assurances";29;"07.01.2000";"Compaq Presario 100";6;1650;"1.50%";9751.5;"Oui"
```

Identifier les valeurs manquantes NA

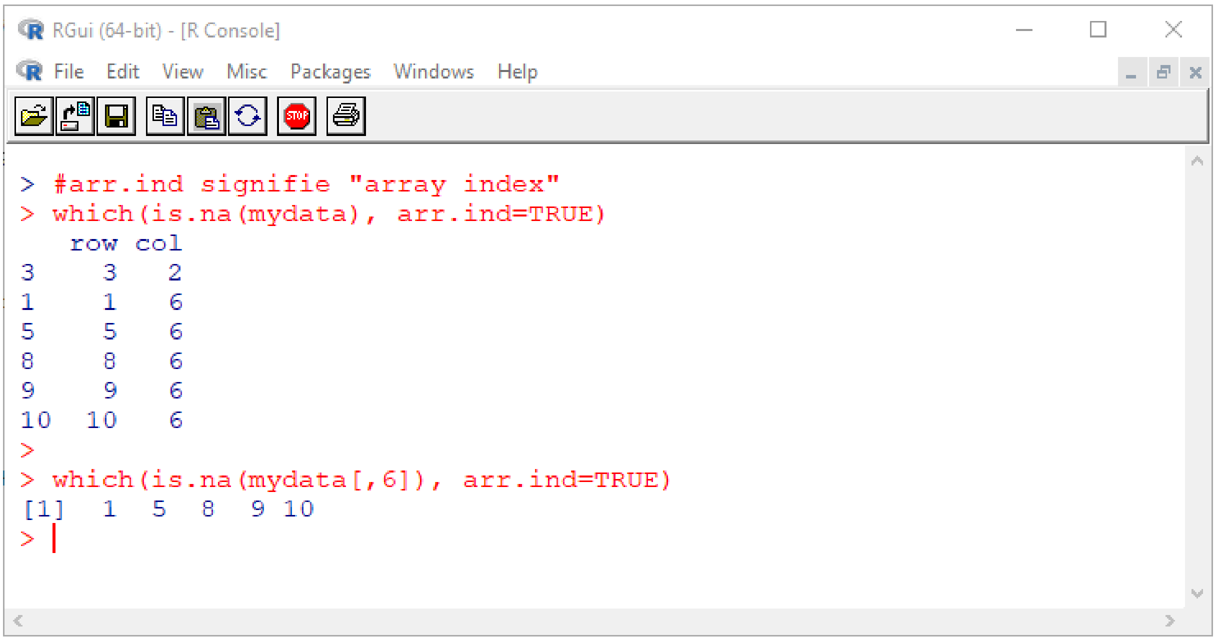
Alors quand nous importons dans R il faut spécifier ce qu'est une valeur "vide":



Et pour diagnostiquer simplement et bêtement les valeurs manquantes, nous pouvons utiliser la fonction **is.na( )** de R:

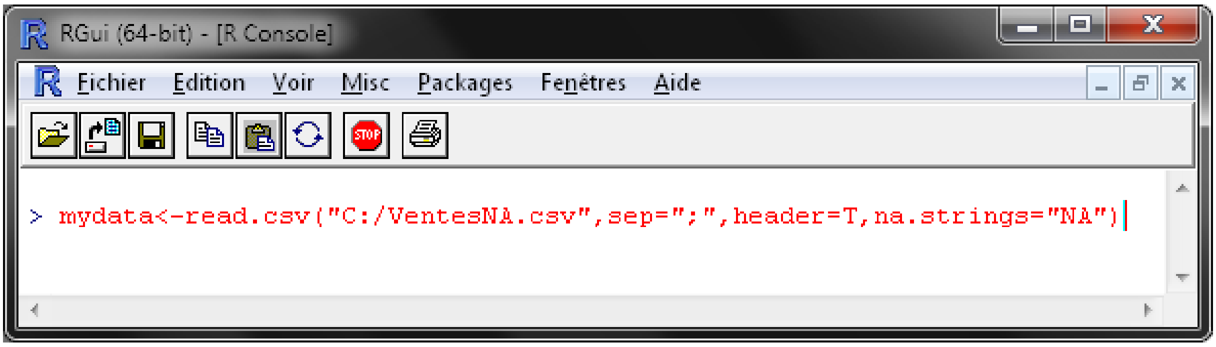


On peut faire mieux esthétiquement (en sachant que les valeurs retournées peuvent être très utiles dans de nombreux cas pratiques!):

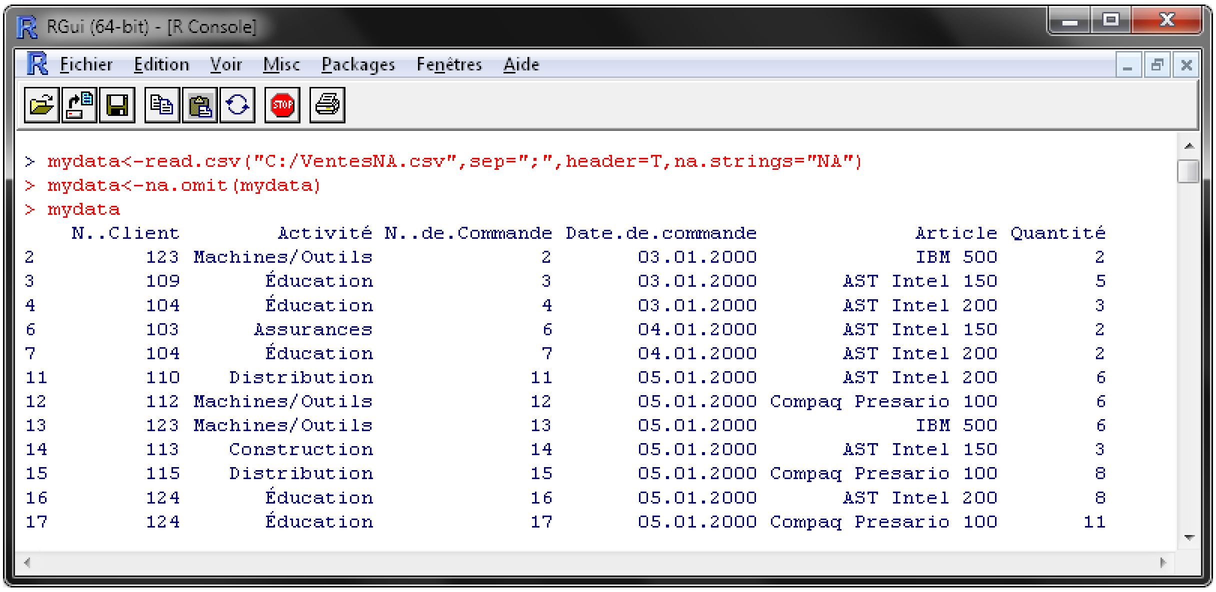


Supprimer les valeurs manquantes NA

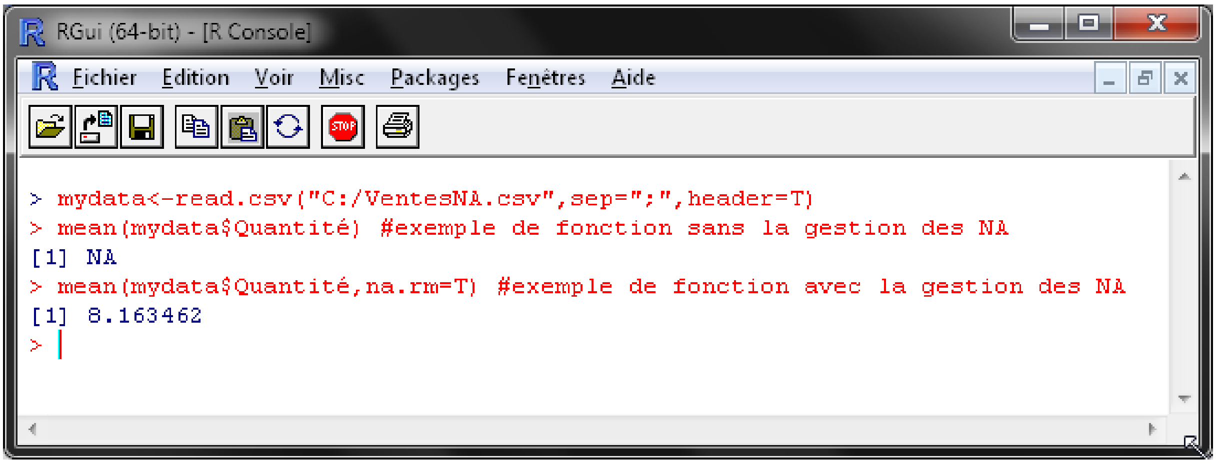
À nouveau, nous partons de:



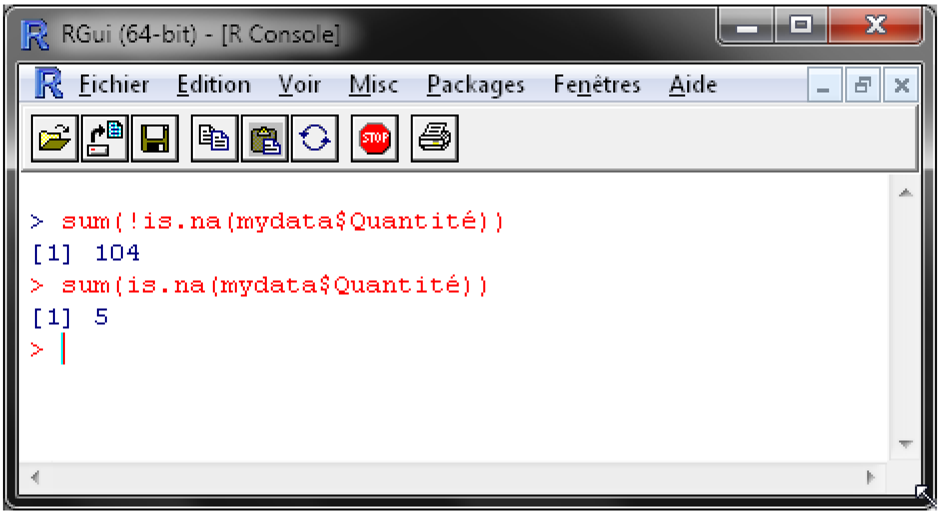
Maintenant pour se débarrasser des lignes avec des NA nous utilisons la commande **na.omit( )** :



Sinon certaines fonctions statistiques ont directement une option pour gérer ce genre de situation. Effectivement:



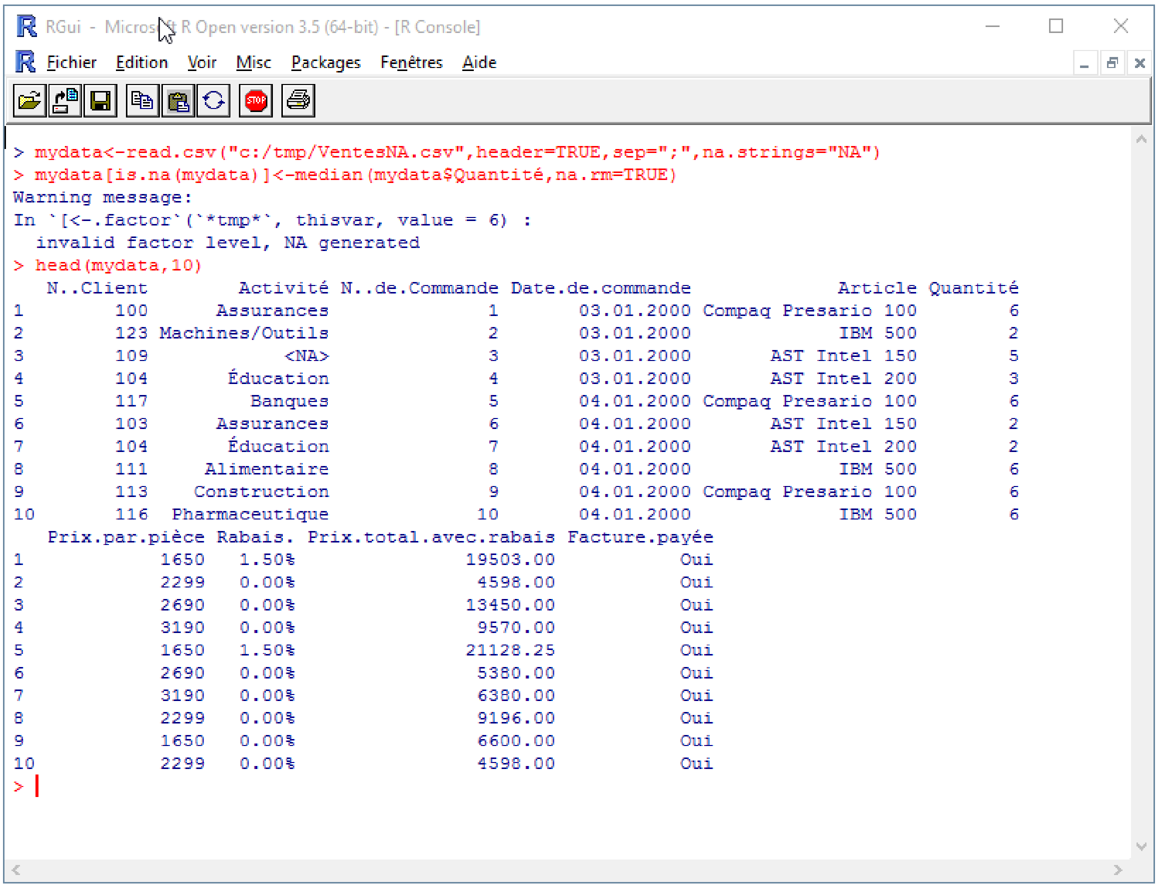
avec **is.na( )** nous pouvons aussi savoir combien il y a des valeurs absentes ou non afin de savoir à quoi nous attendre...:



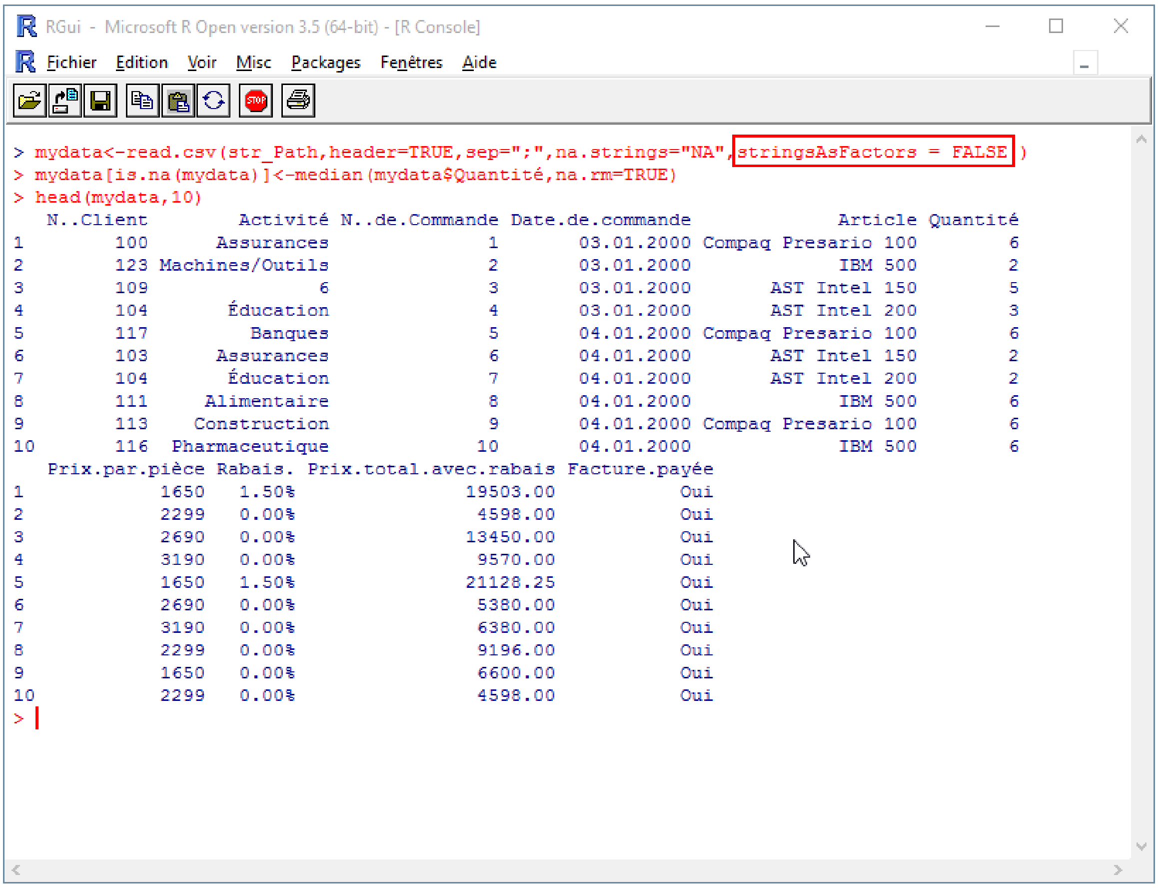
nous verrons plus loin que pour nous débarrasser des NA d'un data frame il suffit d'écrire **na.omit(***mon.data.frame***).**

Remplacer les valeurs manquantes NA dans un dataframe

Parfois il est souvent utile en finance et en statistiques de remplacer les NA par une valeur précise. Pour cela nous utilisons toujours la même fonction:

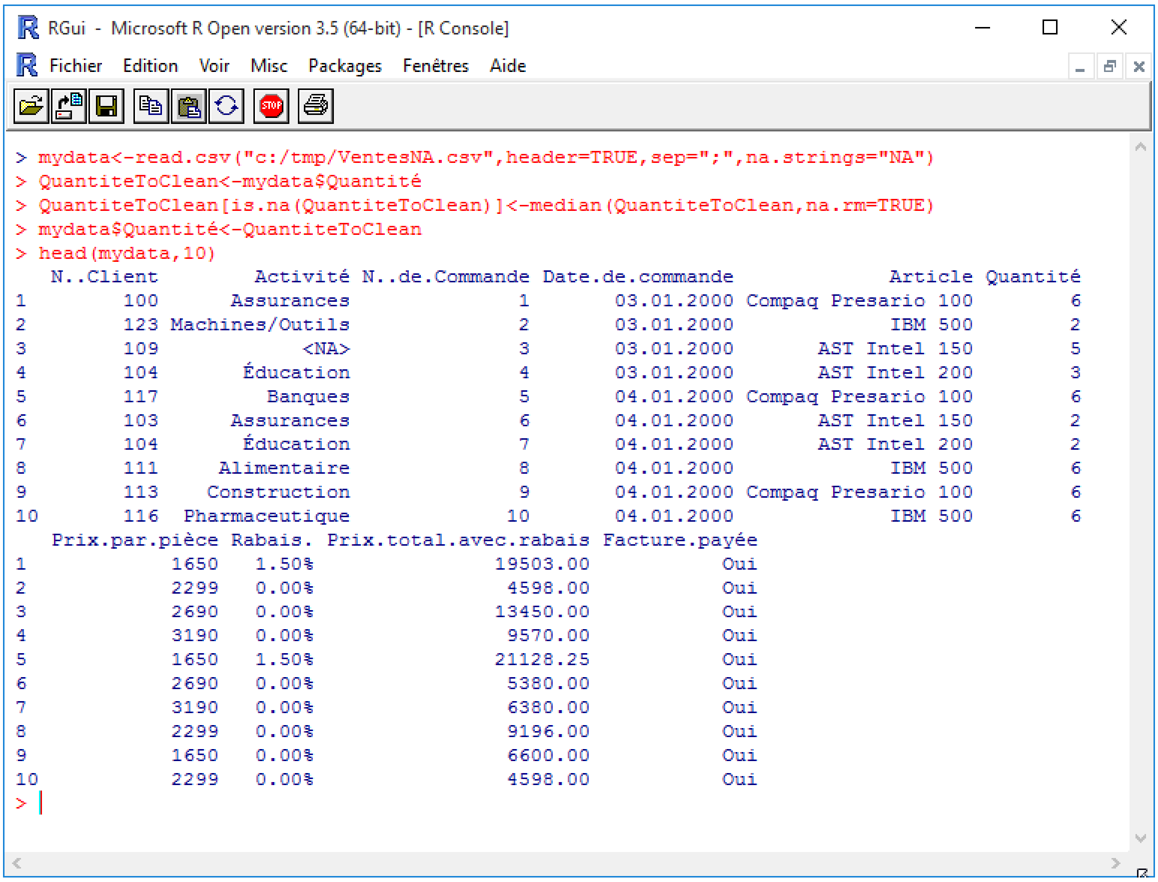


À comparer avec:



Remplacer les valeurs manquantes dans une colonne numérique

Si on veut faire une imputation que sur une unique colonne, on écrira explicitement:



```
> mydata<-read.csv("c:/tmp/VentesNA.csv",header=TRUE,sep=";",na.strings="NA")
> QuantiteToClean<-mydata$Quantité
> QuantiteToClean[is.na(QuantiteToClean)]<-median(QuantiteToClean,na.rm=TRUE)
> mydata$Quantité<-QuantiteToClean
> head(mydata,10)
   N..Client        Activité N..de.Commande Date.de.commande          Article Quantité
1        100      Assurances              1       03.01.2000 Compaq Presario 100        6
2        123 Machines/Outils              2       03.01.2000          IBM 500        2
3        109            <NA>              3       03.01.2000    AST Intel 150        5
4        104       Éducation              4       03.01.2000    AST Intel 200        3
5        117         Banques              5       04.01.2000 Compaq Presario 100        6
6        103      Assurances              6       04.01.2000    AST Intel 150        2
7        104       Éducation              7       04.01.2000    AST Intel 200        2
8        111     Alimentaire              8       04.01.2000          IBM 500        6
9        113    Construction              9       04.01.2000 Compaq Presario 100        6
10       116  Pharmaceutique             10       04.01.2000          IBM 500        6
   Prix.par.pièce Rabais. Prix.total.avec.rabais Facture.payée
1            1650   1.50%               19503.00           Oui
2            2299   0.00%                4598.00           Oui
3            2690   0.00%               13450.00           Oui
4            3190   0.00%                9570.00           Oui
5            1650   1.50%               21128.25           Oui
6            2690   0.00%                5380.00           Oui
7            3190   0.00%                6380.00           Oui
8            2299   0.00%                9196.00           Oui
9            1650   0.00%                6600.00           Oui
10           2299   0.00%                4598.00           Oui
> 
```

Remplacer les valeurs manquantes dans une colonne de facteurs

ToDo

Remplacer les valeurs manquantes par la précédente valeur non-absente

Pour cela on a le package **mefa** et sa fonction **fill.na( )** mais remarquons que bien évidemment il y a un problème si la première valeur est absente (ce qui peut être facilement résolu comme le montre la capture d'écran suivante):

```
RGui - Microsoft R Open version 3.5 (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> mydata<-read.csv("c:/tmp/VentesNA.csv",header=TRUE,sep=";",na.strings="NA")
> mydata$Quantité
  [1] NA  2  5  3 NA  2  2 NA NA NA  6  6  6  3  8  8 11 11 14  7  6 23  3  2 12 24  8  9  6  4  2
 [32] 16  3  7  6  5 23  4  7 12  1  9  6  2  4  1 11  8  7 13  2  5  3  5  7 21  7  7  9  8 10  4
 [63] 12  3  4  5 12  5 13 12  5 22 13  8 33 10 11  4 13  2 15  2  4 22 21 19  6  2  8  7  4  5  8
 [94]  5  6  5  6  6 31  5  2  6 11  3  3 15  4  3  2
> library("mefa")
> fill.na(mydata$Quantité)
Error in fill.na(mydata$Quantité) : cannot replace 'NA's in first place
> mydata$Quantité[1]<-0
> mydata$Quantité
  [1]  0  2  5  3 NA  2  2 NA NA NA  6  6  6  3  8  8 11 11 14  7  6 23  3  2 12 24  8  9  6  4  2
 [32] 16  3  7  6  5 23  4  7 12  1  9  6  2  4  1 11  8  7 13  2  5  3  5  7 21  7  7  9  8 10  4
 [63] 12  3  4  5 12  5 13 12  5 22 13  8 33 10 11  4 13  2 15  2  4 22 21 19  6  2  8  7  4  5  8
 [94]  5  6  5  6  6 31  5  2  6 11  3  3 15  4  3  2
> fill.na(mydata$Quantité)
  [1]  0  2  5  3  3  2  2  2  2  2  6  6  6  3  8  8 11 11 14  7  6 23  3  2 12 24  8  9  6  4  2
 [32] 16  3  7  6  5 23  4  7 12  1  9  6  2  4  1 11  8  7 13  2  5  3  5  7 21  7  7  9  8 10  4
 [63] 12  3  4  5 12  5 13 12  5 22 13  8 33 10 11  4 13  2 15  2  4 22 21 19  6  2  8  7  4  5  8
 [94]  5  6  5  6  6 31  5  2  6 11  3  3 15  4  3  2
> |
```

Analyser les données manquantes

Nous allons voir ici en quoi la package **VIM** peut nous aider à analyser les valeurs manquantes d'un jeu de données.

Nous chargeons pour l'exemple un jeu de données nommé *sleep* associé au package:

```
> library(VIM)
> data(sleep);sleep
   BodyWgt BrainWgt NonD Dream Sleep  Span  Gest Pred Exp Danger
1  6654.000  5712.00   NA    NA   3.3  38.6 645.0    3   5      3
2     1.000     6.60  6.3   2.0   8.3   4.5  42.0    3   1      3
3     3.385    44.50   NA    NA  12.5  14.0  60.0    1   1      1
4     0.920     5.70   NA    NA  16.5    NA  25.0    5   2      3
5  2547.000  4603.00  2.1   1.8   3.9  69.0 624.0    3   5      4
6    10.550   179.50  9.1   0.7   9.8  27.0 180.0    4   4      4
7     0.023     0.30 15.8   3.9  19.7  19.0  35.0    1   1      1
8   160.000   169.00  5.2   1.0   6.2  30.4 392.0    4   5      4
9     3.300    25.60 10.9   3.6  14.5  28.0  63.0    1   2      1
10   52.160   440.00  8.3   1.4   9.7  50.0 230.0    1   1      1
11    0.425     6.40 11.0   1.5  12.5   7.0 112.0    5   4      4
12  465.000   423.00  3.2   0.7   3.9  30.0 281.0    5   5      5
13    0.550     2.40  7.6   2.7  10.3    NA    NA    2   1      2
14  187.100   419.00   NA    NA   3.1  40.0 365.0    5   5      5
15    0.075     1.20  6.3   2.1   8.4   3.5  42.0    1   1      1
16    3.000    25.00  8.6   0.0   8.6  50.0  28.0    2   2      2
17    0.785     3.50  6.6   4.1  10.7   6.0  42.0    2   2      2
18    0.200     5.00  9.5   1.2  10.7  10.4 120.0    2   2      2
19    1.410    17.50  4.8   1.3   6.1  34.0    NA    1   2      1
20   60.000    81.00 12.0   6.1  18.1   7.0    NA    1   1      1
21  529.000   680.00   NA   0.3    NA  28.0 400.0    5   5      5
22   27.660   115.00  3.3   0.5   3.8  20.0 148.0    5   5      5
23    0.120     1.00 11.0   3.4  14.4   3.9  16.0    3   1      2
24  207.000   406.00   NA    NA  12.0  39.3 252.0    1   4      1
25   85.000   325.00  4.7   1.5   6.2  41.0 310.0    1   3      1
26   36.330   119.50   NA    NA  13.0  16.2  63.0    1   1      1
27    0.101     4.00 10.4   3.4  13.8   9.0  28.0    5   1      3
28    1.040     5.50  7.4   0.8   8.2   7.6  68.0    5   3      4
29  521.000   655.00  2.1   0.8   2.9  46.0 336.0    5   5      5
30  100.000   157.00   NA    NA  10.8  22.4 100.0    1   1      1
31   35.000    56.00   NA    NA    NA  16.3  33.0    3   5      4
32    0.005     0.14  7.7   1.4   9.1   2.6  21.5    5   2      4
33    0.010     0.25 17.9   2.0  19.9  24.0  50.0    1   1      1
34   62.000  1320.00  6.1   1.9   8.0 100.0 267.0    1   1      1
35    0.122     3.00  8.2   2.4  10.6    NA  30.0    2   1      1
36    1.350     8.10  8.4   2.8  11.2    NA  45.0    3   1      3
37    0.023     0.40 11.9   1.3  13.2   3.2  19.0    4   1      3
38    0.048     0.33 10.8   2.0  12.8   2.0  30.0    4   1      3
39    1.700     6.30 13.8   5.6  19.4   5.0  12.0    2   1      1
40    3.500    10.80 14.3   3.1  17.4   6.5 120.0    2   1      1
41  250.000   490.00   NA   1.0    NA  23.6 440.0    5   5      5
```

Dont voici la description (suite à la demande d'une personne):

```
Data from which conclusions were drawn in the article "Sleep in
Mammals: Ecological and Constitutional Correlates" by Allison, T. and
Cicchetti, D. (1976), _Science_, November 12, vol. 194, pp. 732-734.
Includes brain and body weight, life span, gestation time, time
sleeping, and predation and danger indices for 62 mammals.


Variables below (from left to right) for Mammals Data Set:

species of animal

body weight in kg

brain weight in g

slow wave ("nondreaming") sleep (hrs/day)

paradoxical ("dreaming") sleep (hrs/day)

total sleep (hrs/day)  (sum of slow wave and paradoxical sleep)

maximum life span (years)

gestation time (days)

predation index (1-5)
                1 = minimum (least likely to be preyed upon)
                5 = maximum (most likely to be preyed upon)

sleep exposure index (1-5)
                1 = least exposed (e.g. animal sleeps in a
                    well-protected den)
                5 = most exposed

overall danger index (1-5)
                (based on the above two indices and other information)
                1 = least danger (from other animals)
                5 = most danger (from other animals)

Note: Missing values denoted by -999.0


For more details, see

Allison, Truett and Cicchetti, Domenic V. (1976), "Sleep in Mammals:
Ecological and Constitutional Correlates", _Science_, November 12,
vol. 194, pp. 732-734.


African elephant         6654.000 5712.000  -999.0  -999.0     3.3    38.6   645.0         3         5         3
African giant pouched rat   1.000    6.600     6.3     2.0     8.3     4.5    42.0         3         1         3
Arctic Fox                  3.385   44.500  -999.0  -999.0    12.5    14.0    60.0         1         1         1
Arctic ground squirrel       .920    5.700  -999.0  -999.0    16.5  -999.0    25.0         5         2         3
Asian elephant           2547.000 4603.000     2.1     1.8     3.9    69.0   624.0         3         5         4
Baboon                     10.550  179.500     9.1      .7     9.8    27.0   180.0         4         4         4
Big brown bat                .023     .300    15.8     3.9    19.7    19.0    35.0         1         1         1
Brazilian tapir           160.000  169.000     5.2     1.0     6.2    30.4   392.0         4         5         4
Cat                         3.300   25.600    10.9     3.6    14.5    28.0    63.0         1         2         1
```

Avec la fonction **aggr( )** du package **VIM** nous pouvons avoir un état des lieux du jeu de données:

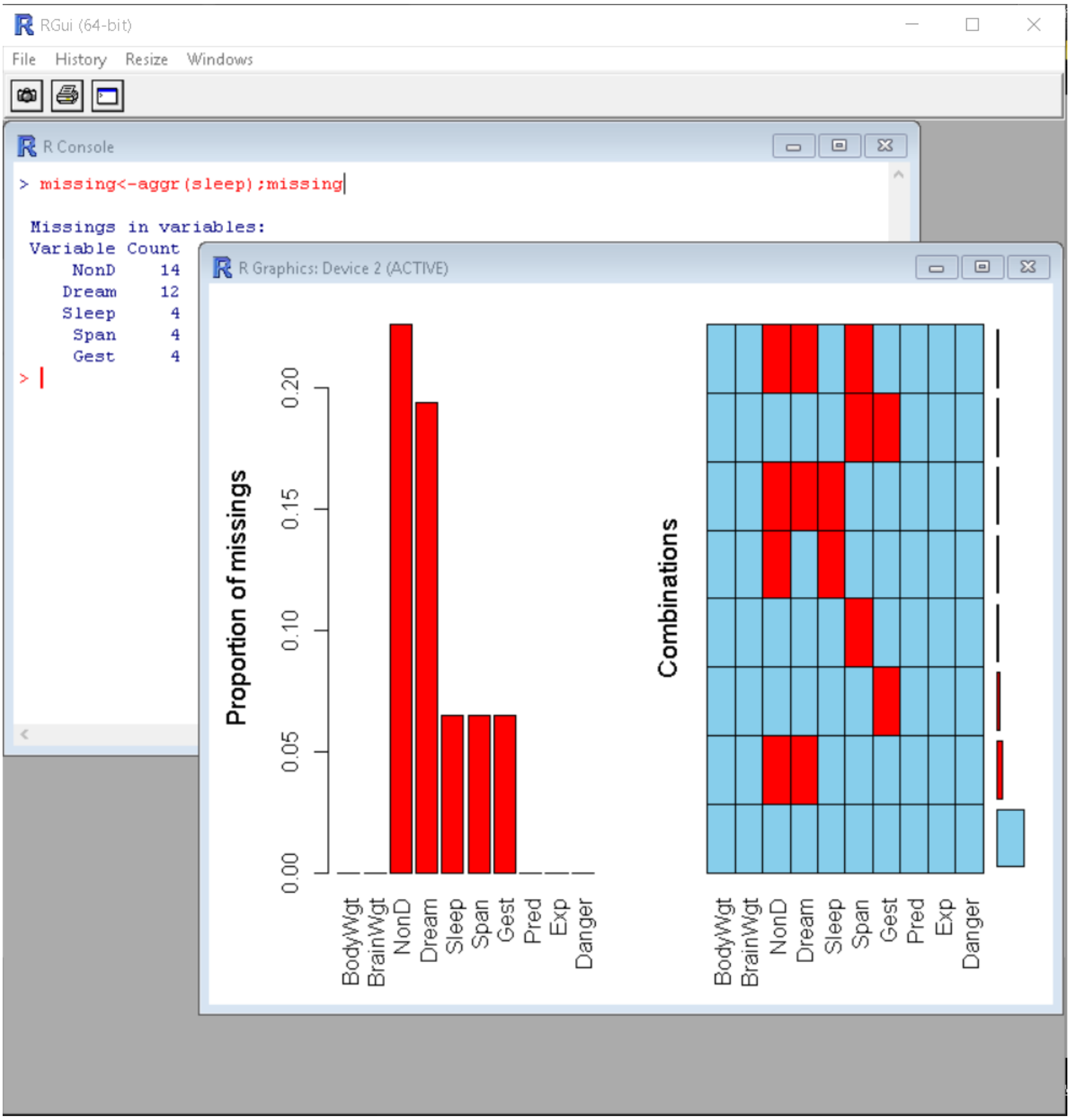


Figure 29 Graph combiné d'analyse de valeurs manquantes

ou encore:

```
> summary(aggr(sleep))

 Missings per variable:
 Variable Count
  BodyWgt     0
 BrainWgt     0
     NonD    14
    Dream    12
    Sleep     4
     Span     4
     Gest     4
     Pred     0
      Exp     0
   Danger     0

 Missings in combinations of variables:
        Combinations Count   Percent
 0:0:0:0:0:0:0:0:0:0    42 67.741935
 0:0:0:0:0:0:1:0:0:0     3  4.838710
 0:0:0:0:0:1:0:0:0:0     2  3.225806
 0:0:0:0:0:1:1:0:0:0     1  1.612903
 0:0:1:0:1:0:0:0:0:0     2  3.225806
 0:0:1:1:0:0:0:0:0:0     9 14.516129
 0:0:1:1:0:1:0:0:0:0     1  1.612903
 0:0:1:1:1:0:0:0:0:0     2  3.225806
>
```

Pour une variable donnée et ses intervalles nous pouvons afficher le nombre d'éléments non disponibles d'une autre variable choisie à l'aide de la fonction **barMiss)**:

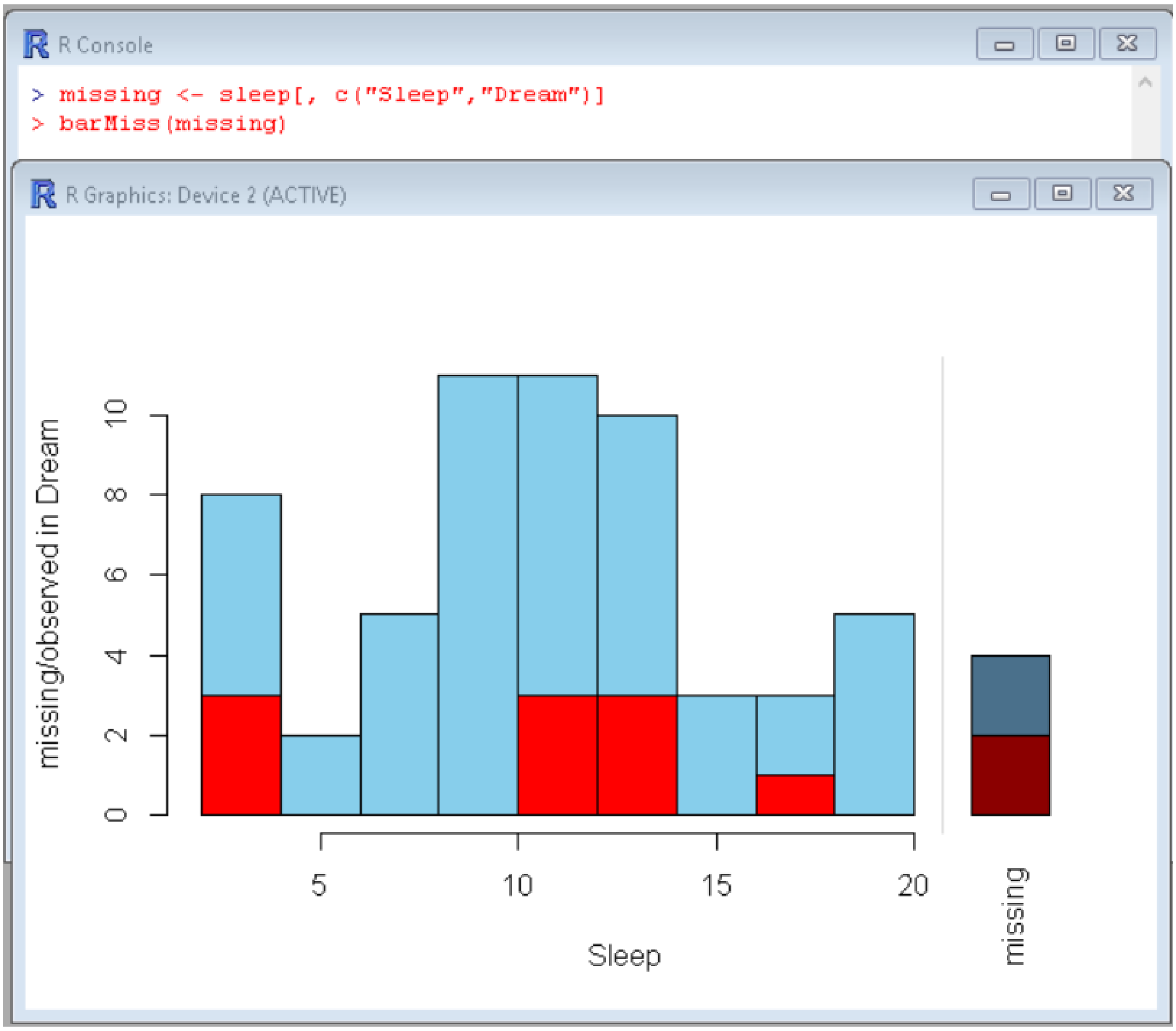


Figure 30 Graphe d'histogramme de valeurs manquantes

Nous pouvons également analyser les statistiques d'une variable (*sleep* dans le cas présent) sous forme d'un box-plot et séparant aussi la statistique en deux sous-box-plot avec seulement et sans les valeurs manquantes d'une autre variable (*dream* dans le cas présent) à l'aide de la commande **pbox( )**:

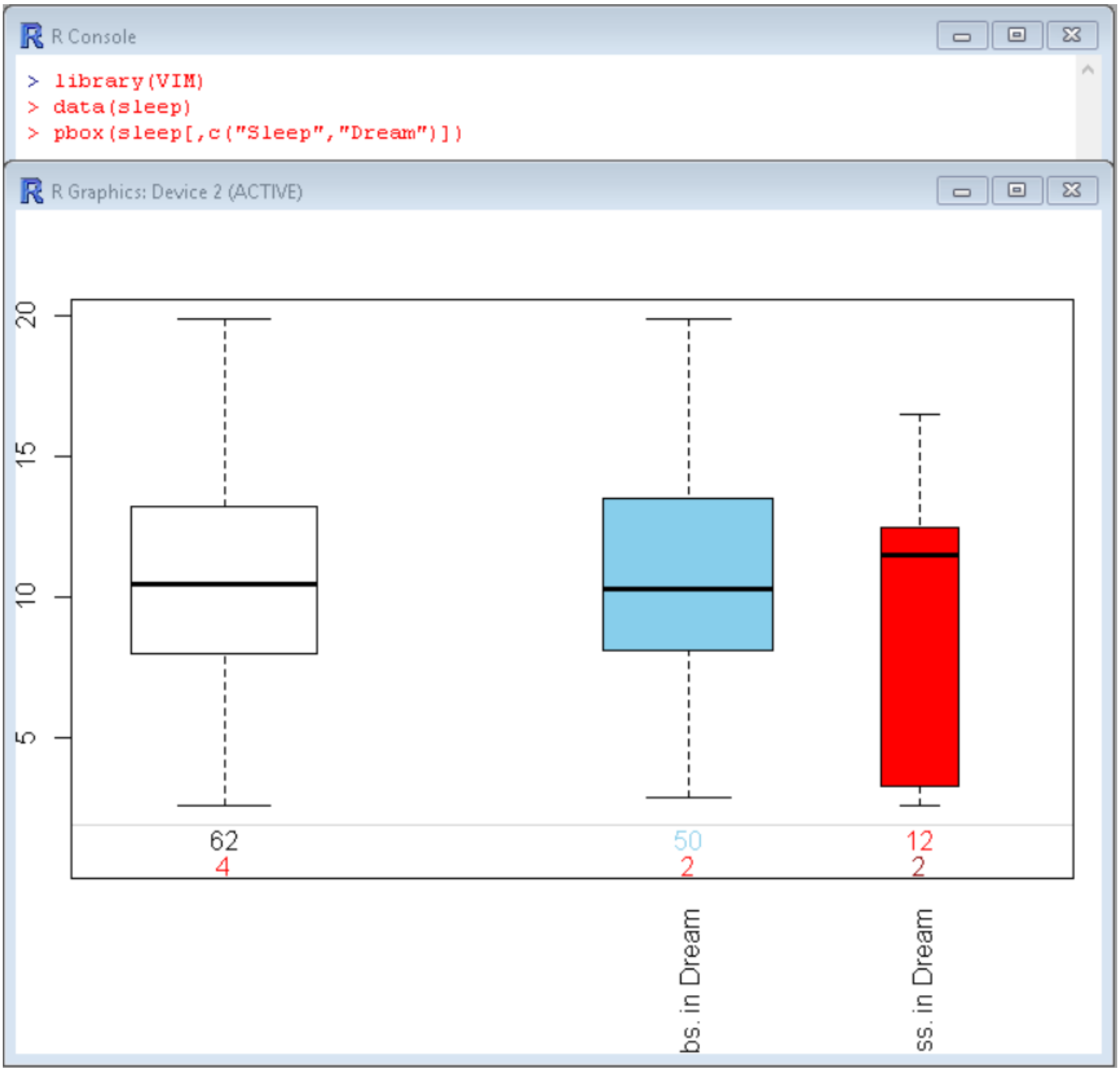


Figure 31 Graphe Box-Whiser (boîtes à moustaches) de valeurs manquantes

Nous avons aussi la fonction **md.pattern( )** du package **mice**:

```
> library("VIM")
> library("mice")
> md.pattern(sleep)
   BodyWgt BrainWgt Pred Exp Danger Sleep Span Gest Dream NonD
42       1        1    1   1      1     1    1    1     1    1  0
 2       1        1    1   1      1     1    0    1     1    1  1
 3       1        1    1   1      1     1    1    0     1    1  1
 9       1        1    1   1      1     1    1    1     0    0  2
 2       1        1    1   1      1     0    1    1     1    0  2
 1       1        1    1   1      1     1    0    0     1    1  2
 2       1        1    1   1      1     0    1    1     0    0  3
 1       1        1    1   1      1     1    0    1     0    0  3
         0        0    0   0      0     4    4    4    12   14 38
>
```

On peut lire du tableau ci-dessus, notamment à la première ligne, qu'il y a 42 observations ou entreprises qui sont complètes pour toutes les variables. On peut lire également à la huitième ligne que 1 observation est complète sauf pour 3 variables.

Pour d'autres outils, le lecteur pourra se reporter à la page 516.

Imputer les valeurs manquantes statistiquement

À ce jour le package le plus complet à notre connaissance permettant d'imputer des valeurs manquantes est le package **mice** avec la fonction **mice( )** qui inclut presque une vingtaine de méthodes mathématiques pour accomplir le travail en question.

Voyons donc un exemple (notez les *NA* dans le jeu de données d'origine!) mais d'abord observons la liste des méthodes impressionnante de méthodes d'imputation disponibles dans mice:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

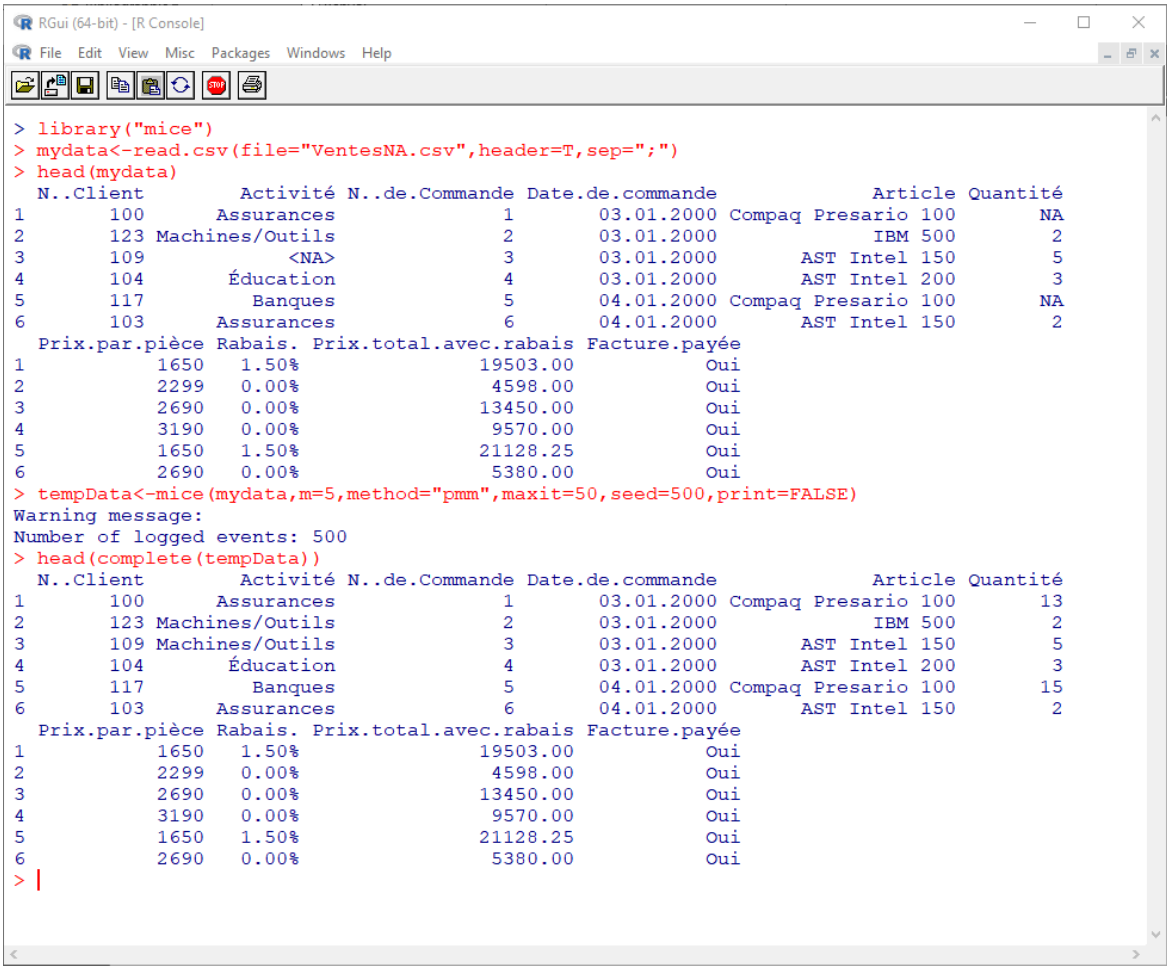
> library("mice")
> methods(mice)
 [1] mice.impute.2l.bin         mice.impute.2l.lmer
 [3] mice.impute.2l.norm        mice.impute.2l.pan
 [5] mice.impute.2lonly.mean    mice.impute.2lonly.norm
 [7] mice.impute.2lonly.pmm     mice.impute.cart
 [9] mice.impute.jomoImpute     mice.impute.lda
[11] mice.impute.logreg         mice.impute.logreg.boot
[13] mice.impute.mean           mice.impute.midastouch
[15] mice.impute.norm           mice.impute.norm.boot
[17] mice.impute.norm.nob       mice.impute.norm.predict
[19] mice.impute.panImpute      mice.impute.passive
[21] mice.impute.pmm            mice.impute.polr
[23] mice.impute.polyreg        mice.impute.quadratic
[25] mice.impute.rf             mice.impute.ri
[27] mice.impute.sample         mice.mids
[29] mice.theme
see '?methods' for accessing help and source code
Warning message:
In .S3methods(generic.function, class, envir) :
  function 'mice' appears not to be S3 generic; found functions that$
> 
```

Ou de manière plus détaillée en allant lire la documentation du package lui-même:

pmm	any	Predictive mean matching
midastouch	any	Weighted predictive mean matching
sample	any	Random sample from observed values
cart	any	Classification and regression trees

rf	any	Random forest imputations
mean	numeric	Unconditional mean imputation
norm	numeric	Bayesian linear regression
norm.nob	numeric	Linear regression ignoring model error
norm.boot	numeric	Linear regression using bootstrap
norm.predict	numeric	Linear regression, predicted values
quadratic	numeric	Imputation of quadratic terms
ri	numeric	Random indicator for nonignorable data
logreg	binary	Logistic regression
logreg.boot	binary	Logistic regression with bootstrap
polr	ordered	Proportional odds model
polyreg	unordered	Polytomous logistic regression
lda	unordered	Linear discriminant analysis
2l.norm	numeric	Level-1 normal heteroscedastic
2l.lmer	numeric	Level-1 normal homoscedastic, lmer
2l.pan	numeric	Level-1 normal homoscedastic, pan
2l.bin	binary	Level-1 logistic, glmer
2lonly.mean	numeric	Level-2 class mean
2lonly.norm	numeric	Level-2 class normal
2lonly.pmm	any	Level-2 class predictive mean matching

Et allons-y pour un petit cas pratique non-exhaustif!:


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("mice")
> mydata<-read.csv(file="VentesNA.csv",header=T,sep=";")
> head(mydata)
  N..Client        Activité N..de.Commande Date.de.commande            Article Quantité
1       100      Assurances              1       03.01.2000 Compaq Presario 100       NA
2       123 Machines/Outils              2       03.01.2000             IBM 500        2
3       109            <NA>              3       03.01.2000       AST Intel 150        5
4       104       Éducation              4       03.01.2000       AST Intel 200        3
5       117         Banques              5       04.01.2000 Compaq Presario 100       NA
6       103      Assurances              6       04.01.2000       AST Intel 150        2
  Prix.par.pièce Rabais. Prix.total.avec.rabais Facture.payée
1           1650   1.50%               19503.00           Oui
2           2299   0.00%                4598.00           Oui
3           2690   0.00%               13450.00           Oui
4           3190   0.00%                9570.00           Oui
5           1650   1.50%               21128.25           Oui
6           2690   0.00%                5380.00           Oui
> tempData<-mice(mydata,m=5,method="pmm",maxit=50,seed=500,print=FALSE)
Warning message:
Number of logged events: 500
> head(complete(tempData))
  N..Client        Activité N..de.Commande Date.de.commande            Article Quantité
1       100      Assurances              1       03.01.2000 Compaq Presario 100       13
2       123 Machines/Outils              2       03.01.2000             IBM 500        2
3       109 Machines/Outils              3       03.01.2000       AST Intel 150        5
4       104       Éducation              4       03.01.2000       AST Intel 200        3
5       117         Banques              5       04.01.2000 Compaq Presario 100       15
6       103      Assurances              6       04.01.2000       AST Intel 150        2
  Prix.par.pièce Rabais. Prix.total.avec.rabais Facture.payée
1           1650   1.50%               19503.00           Oui
2           2299   0.00%                4598.00           Oui
3           2690   0.00%               13450.00           Oui
4           3190   0.00%                9570.00           Oui
5           1650   1.50%               21128.25           Oui
6           2690   0.00%                5380.00           Oui
> |
```

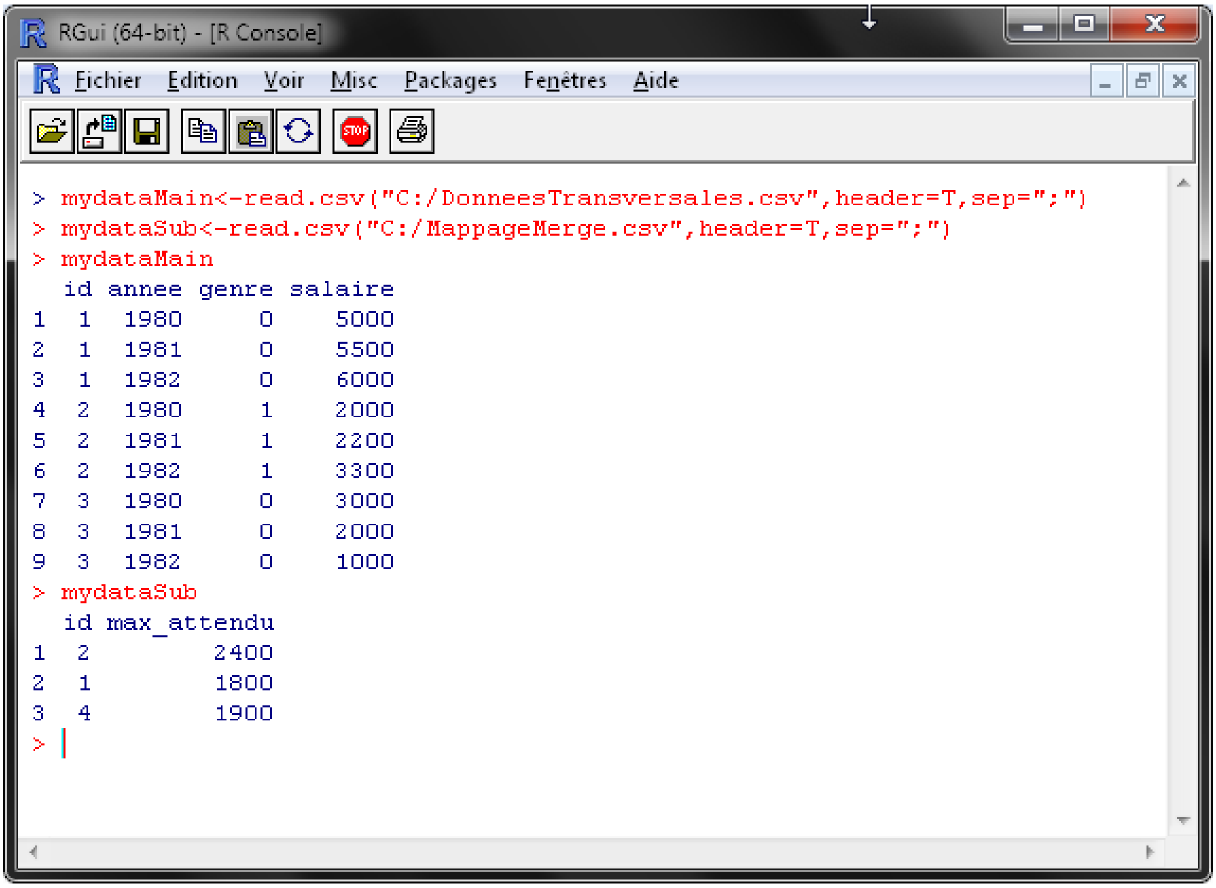
Attention!!! Pour que le package mice impute les variables catégorielles, il faut importer les données catégorielles en tant que facteurs. Donc évitez le stringsAsFactors=FALSE dans read.csv( ) ou autre…!

Exemple 100.: Fusionner des données par clé de mappage (RECHERCHEV)

R 3.0.2

Dans certaines situations simples plutôt que de faire appel à l'artillerie lourde du SQL pour associer deux data frame, une solution simple est parfois possible. Elle consiste à faire un mappage entre deux tables en utilisant la commande native **merge( )** de R.

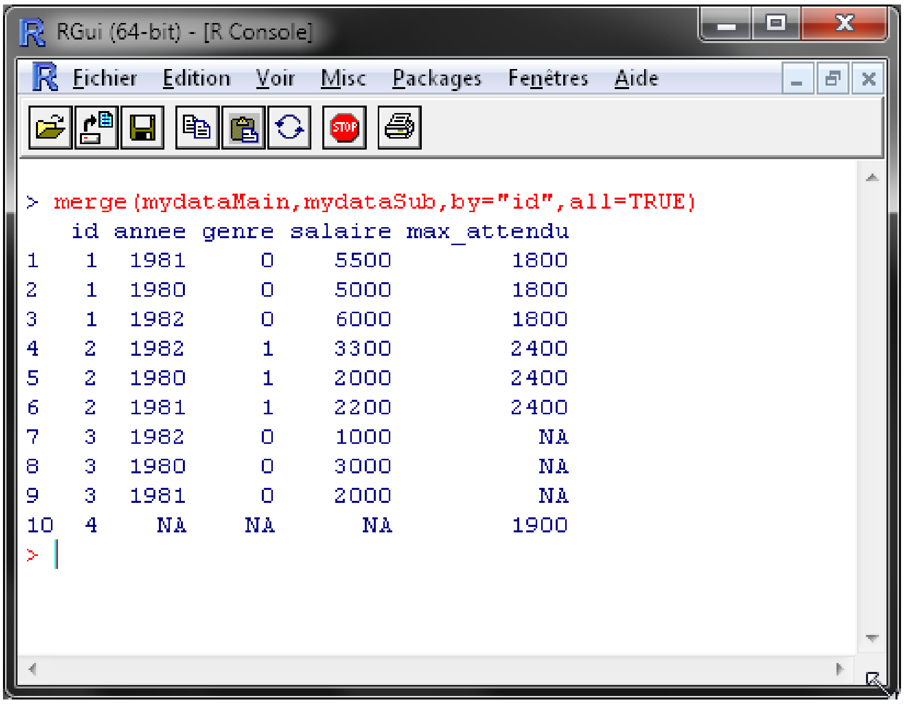
Considérons pour l'exemple les deux jeux de données suivants:




```
> mydataMain<-read.csv("C:/DonneesTransversales.csv",header=T,sep=";")
> mydataSub<-read.csv("C:/MappageMerge.csv",header=T,sep=";")
> mydataMain
  id annee genre salaire
1  1  1980     0    5000
2  1  1981     0    5500
3  1  1982     0    6000
4  2  1980     1    2000
5  2  1981     1    2200
6  2  1982     1    3300
7  3  1980     0    3000
8  3  1981     0    2000
9  3  1982     0    1000
> mydataSub
  id max_attendu
1  2        2400
2  1        1800
3  4        1900
> 
```

Full (outer) join

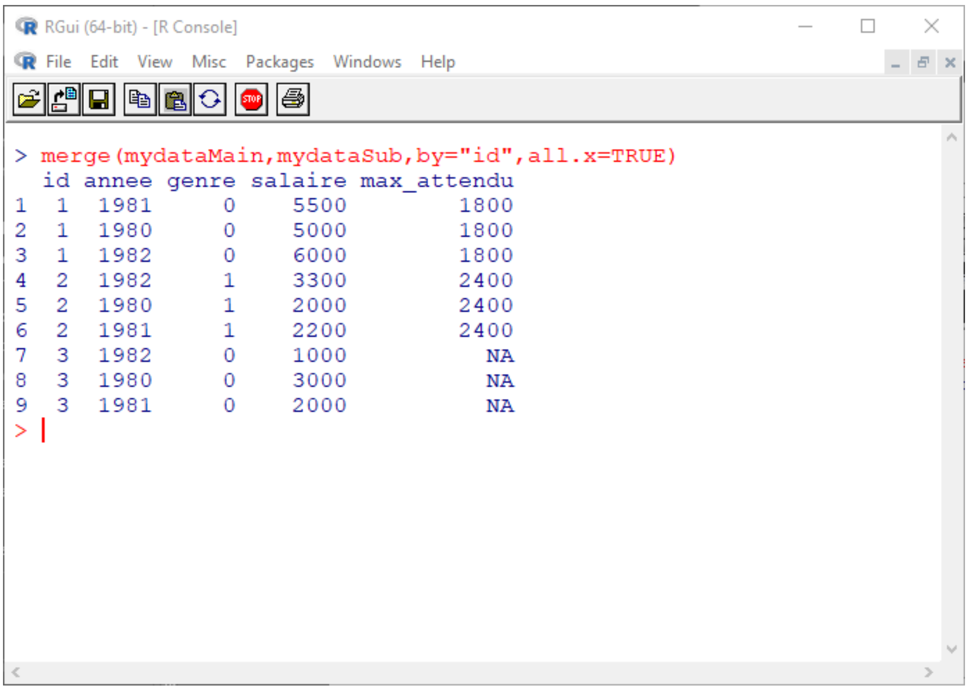
Nous allons faire un mappage et observer le résultat (la conclusion devrait ne pas nécessiter de commentaires de notre part):



Il s'agit donc de l'équivalent du fameux RECHERCHEV de Microsoft Excel (VLOOKUP pour ceux qui ont la version anglaise).

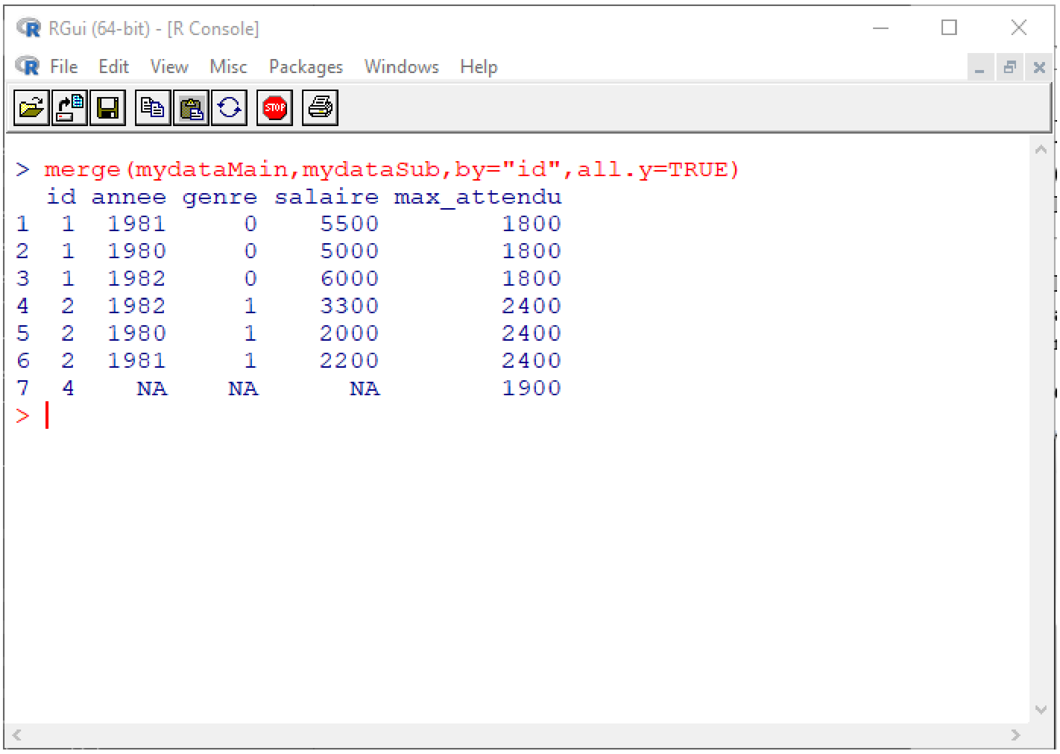
Left (outer) join

Pour la jointure gauche nous utiliserons une syntaxe similaire:



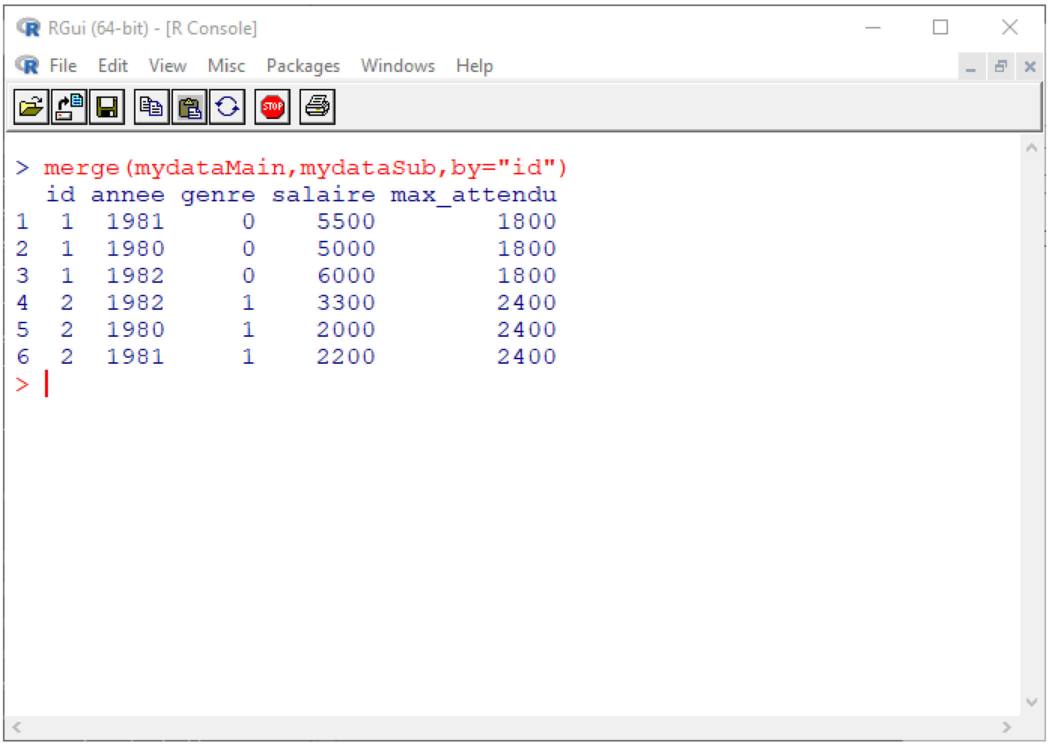
Right (outer) join

Pour la jointure droite nous utiliserons une syntaxe similaire:



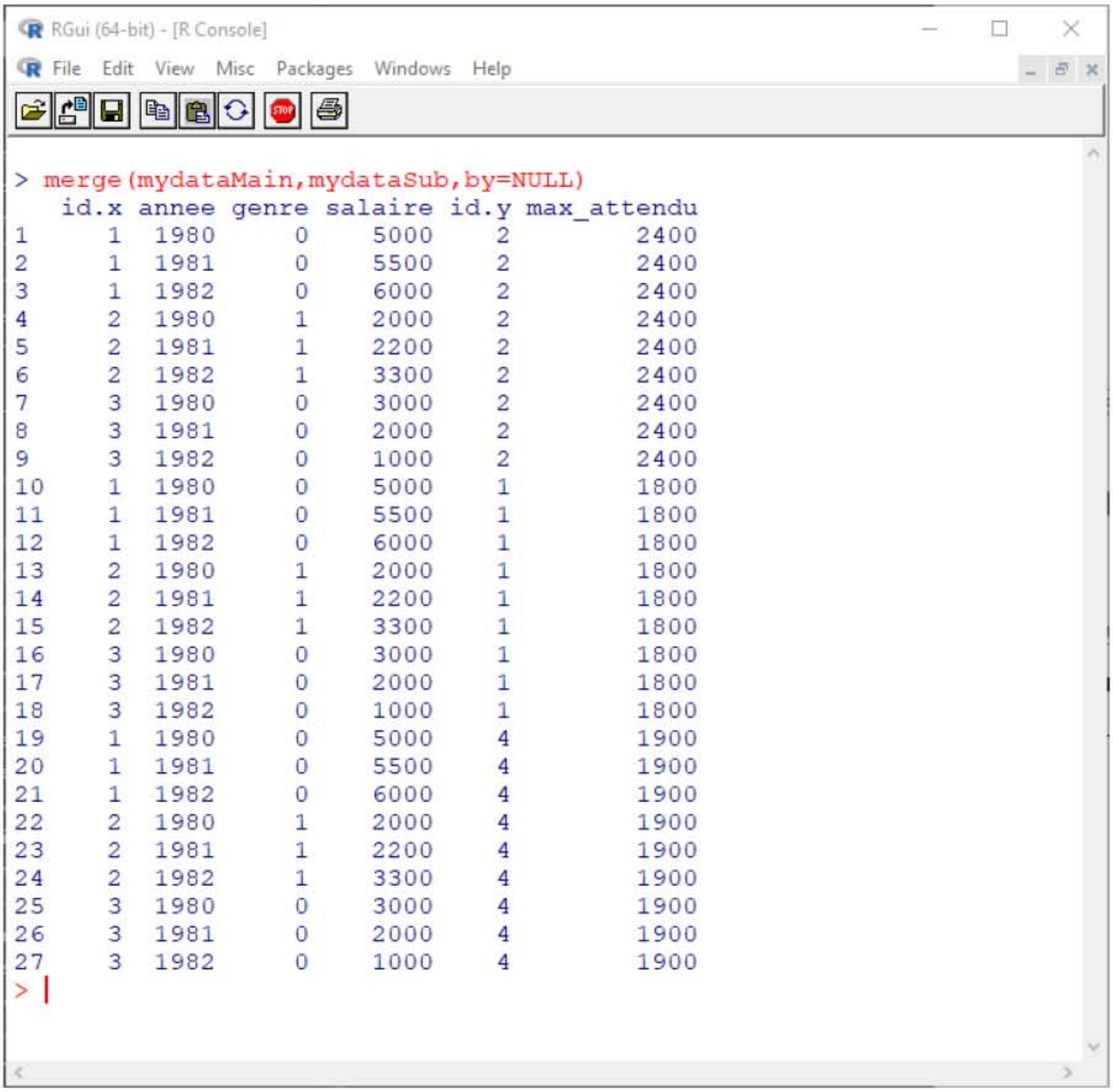
Inner join

Pour la jointure interne nous utiliserons une syntaxe similaire:



Cross join

Pour la jointure cartésienne nous utiliserons une syntaxe similaire:



```
> merge(mydataMain,mydataSub,by=NULL)
   id.x annee genre salaire id.y max_attendu
1     1  1980     0    5000    2        2400
2     1  1981     0    5500    2        2400
3     1  1982     0    6000    2        2400
4     2  1980     1    2000    2        2400
5     2  1981     1    2200    2        2400
6     2  1982     1    3300    2        2400
7     3  1980     0    3000    2        2400
8     3  1981     0    2000    2        2400
9     3  1982     0    1000    2        2400
10    1  1980     0    5000    1        1800
11    1  1981     0    5500    1        1800
12    1  1982     0    6000    1        1800
13    2  1980     1    2000    1        1800
14    2  1981     1    2200    1        1800
15    2  1982     1    3300    1        1800
16    3  1980     0    3000    1        1800
17    3  1981     0    2000    1        1800
18    3  1982     0    1000    1        1800
19    1  1980     0    5000    4        1900
20    1  1981     0    5500    4        1900
21    1  1982     0    6000    4        1900
22    2  1980     1    2000    4        1900
23    2  1981     1    2200    4        1900
24    2  1982     1    3300    4        1900
25    3  1980     0    3000    4        1900
26    3  1981     0    2000    4        1900
27    3  1982     0    1000    4        1900
> |
```

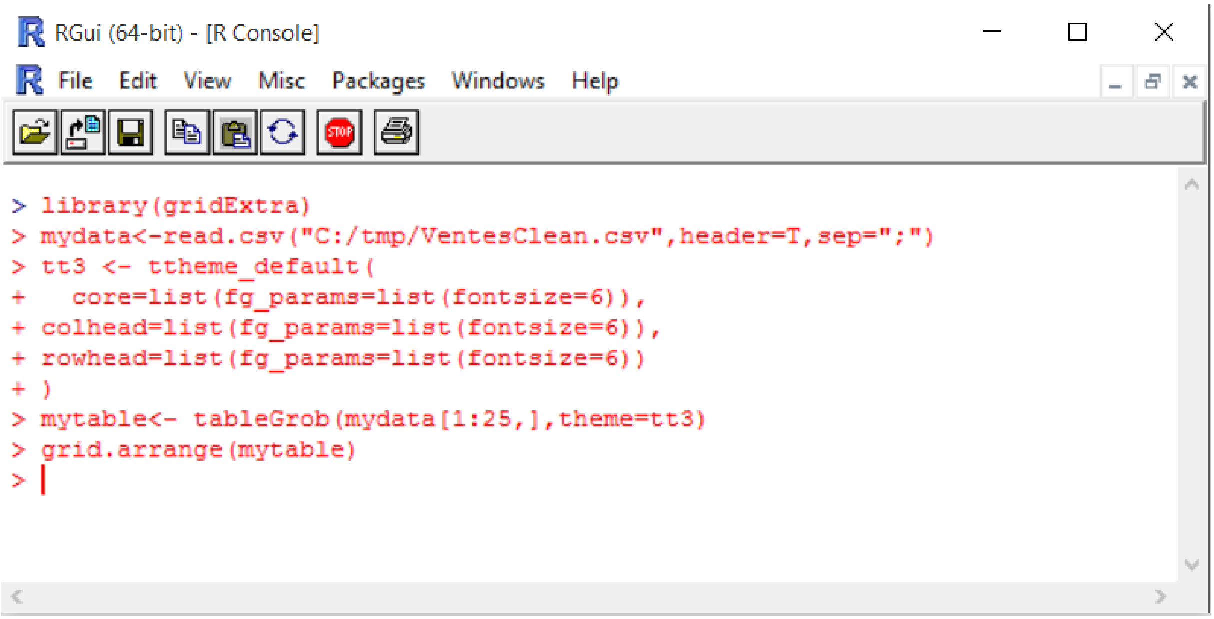
Exemple 101.: Afficher un data frame en tant que table

R 3.2.4

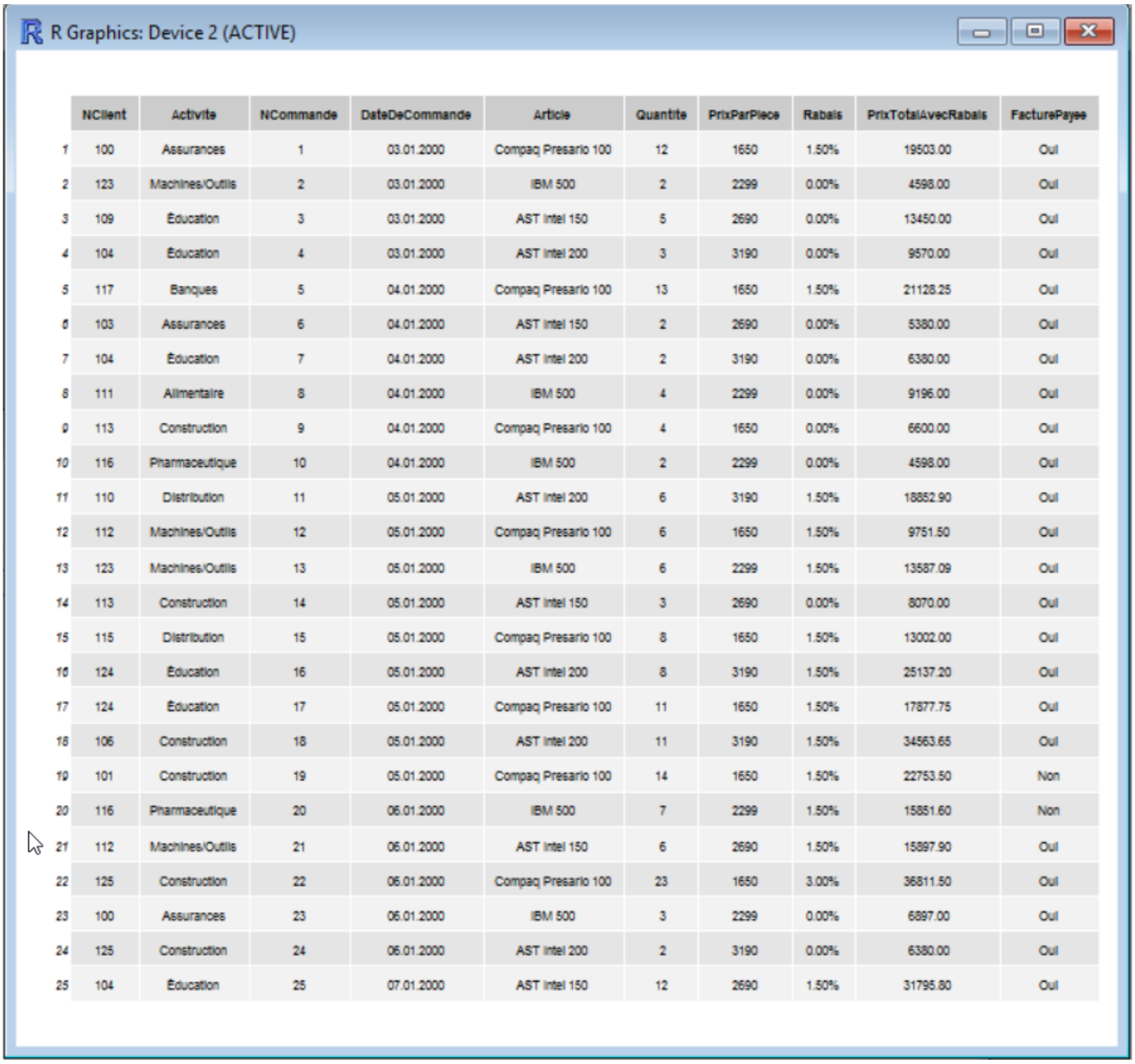
Nous allons voici ici quelques possibilités du package **gridExtra** dont il existe une excellente documentation ici:

https://github.com/baptiste/gridextra/wiki/tableGrob

Donc voyons un example avec notre jeu de données habituel et en utilisant la fonction **tableGrob**:



Ce qui donne:



	NClient	Activite	NCommande	DateDeCommande	Article	Quantite	PrixParPiece	Rabais	PrixTotalAvecRabais	FacturePayee
1	100	Assurances	1	03.01.2000	Compaq Presario 100	12	1650	1.50%	19503.00	Oui
2	123	Machines/Outils	2	03.01.2000	IBM 500	2	2299	0.00%	4598.00	Oui
3	109	Éducation	3	03.01.2000	AST Intel 150	5	2690	0.00%	13450.00	Oui
4	104	Éducation	4	03.01.2000	AST Intel 200	3	3190	0.00%	9570.00	Oui
5	117	Banques	5	04.01.2000	Compaq Presario 100	13	1650	1.50%	21128.25	Oui
6	103	Assurances	6	04.01.2000	AST Intel 150	2	2690	0.00%	5380.00	Oui
7	104	Éducation	7	04.01.2000	AST Intel 200	2	3190	0.00%	6380.00	Oui
8	111	Alimentaire	8	04.01.2000	IBM 500	4	2299	0.00%	9196.00	Oui
9	113	Construction	9	04.01.2000	Compaq Presario 100	4	1650	0.00%	6600.00	Oui
10	116	Pharmaceutique	10	04.01.2000	IBM 500	2	2299	0.00%	4598.00	Oui
11	110	Distribution	11	05.01.2000	AST Intel 200	6	3190	1.50%	18852.90	Oui
12	112	Machines/Outils	12	05.01.2000	Compaq Presario 100	6	1650	1.50%	9751.50	Oui
13	123	Machines/Outils	13	05.01.2000	IBM 500	6	2299	1.50%	13587.09	Oui
14	113	Construction	14	05.01.2000	AST Intel 150	3	2690	0.00%	8070.00	Oui
15	115	Distribution	15	05.01.2000	Compaq Presario 100	8	1650	1.50%	13002.00	Oui
16	124	Éducation	16	05.01.2000	AST Intel 200	8	3190	1.50%	25137.20	Oui
17	124	Éducation	17	05.01.2000	Compaq Presario 100	11	1650	1.50%	17877.75	Oui
18	106	Construction	18	05.01.2000	AST Intel 200	11	3190	1.50%	34563.65	Oui
19	101	Construction	19	05.01.2000	Compaq Presario 100	14	1650	1.50%	22753.50	Non
20	116	Pharmaceutique	20	06.01.2000	IBM 500	7	2299	1.50%	15851.60	Non
21	112	Machines/Outils	21	06.01.2000	AST Intel 150	6	2690	1.50%	15897.90	Oui
22	125	Construction	22	06.01.2000	Compaq Presario 100	23	1650	3.00%	36811.50	Oui
23	100	Assurances	23	06.01.2000	IBM 500	3	2299	0.00%	6897.00	Oui
24	125	Construction	24	06.01.2000	AST Intel 200	2	3190	0.00%	6380.00	Oui
25	104	Éducation	25	07.01.2000	AST Intel 150	12	2690	1.50%	31795.80	Oui

Figure 32 Graph d'un data.frame

D'autres complications viendront par la suite car la syntaxe est un peu pénible à ce jour!

ToDo: Faire d'autres exemples

Exemple 102.: Différences entre apply, lapply, sapply et autres…

R 3.2.4

Comme la question est fréquemment demandée dans les formations, voici – avant de passer à l'étude du package **dplyr** - un résumé concernant les différences entre ses trois fonctions que l'on retrouve fréquemment (si nous omettons **vapply( )**, **tapply( )**, **eapply( )**, **rollapply( )**[12]).

Alors allons-y (on peut obtenir les mêmes informations en tapant **??apply** dans la console et en allant regarder la dernière page du résultat):

zoo::rollapply	Apply Rolling Functions
base::apply	Apply Functions Over Array Margins
base::.subset	Internal Objects in Package 'base'
base::by	Apply a Function to a Data Frame Split by Factors
base::eapply	Apply a Function Over Values in an Environment
base::lapply	Apply a Function over a List or Vector
base::mapply	Apply a Function to Multiple List or Vector Arguments
base::rapply	Recursively Apply a Function to a List
base::tapply	Apply a Function Over a Ragged Array
grid::gEdit	Create and Apply Edit Objects

- `base::by` Apply a function to a data frame split by Factors

  Permet d'appliquer une fonction R ou personnelle sur un ensemble de colonnes d'un data frame par catégories d'une variable.

- `base::apply(array/matrix, margin, fun, …)` Apply Functions Over Array Margins

  Permet d'appliquer une fonction R ou personnelle sur un ensemble de colonnes d'une matrice. Comme nous l'avons montré avec les exemples aux pages 277, 2512 ou encore 2931.

- `base::lapply(list/vector, fun, …)` Apply Functions and return a list

  Permet d'appliquer une fonction R ou personnelle sur une liste et renvoie une liste. C'est un cas relativement rare mais nous le lecteur pourra en voir un exemple à la page 2779.

  Remarque: Passer le paramètre `simplify=FALSE` à **sapply( )** donne le même résultat que **lapply**: `sapply(x, fun, simplify=F)`

- `base::sapply(list/vector, fun, …)` Apply Functions and return a vector (or optionaly a list)

[12] En tant qu'auteurs du présent document, nous n'en avons jamais eu usage et donc ne voyons que faiblement l'intérêt de s'attarder dessus.

Permet d'appliquer une fonction R ou personnelle sur une liste vecteur et renvoie une liste ou vecteur. C'est le cas le plus utilisé dans R et lecteur pourra en voir des exemples à la page 571, 233, 2484, 2916.

Mais pour ceux qui ne souhaiteraient pas avoir à reparcourir toutes les pages mentionnées en référence, voici un résumé francophone repris de l'excellent article de blog suivant de Neil Saunders:

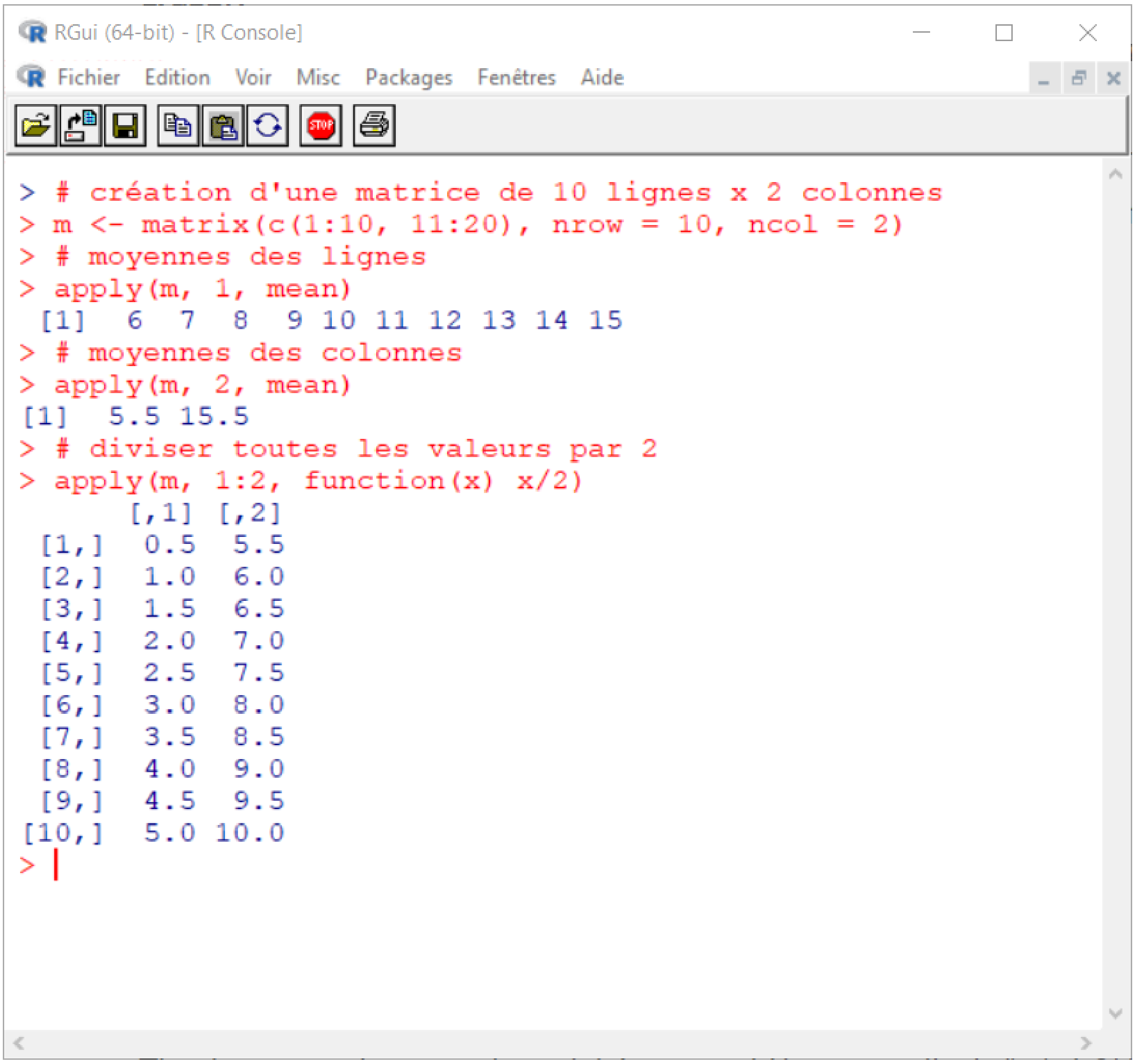
https://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/

Remarque: On parle souvent de "vectorisation" lorsqu'on utilise les fonctions ci-dessous. Mais en réalité, en allant voir le code source de R, on peut observer qu'il s'agit simplement de fonctions qui cachent en arrière plan des boucles écrites en R pour apply( ) ou des boucles en C qui appellent des fonctions en R pour les autres (donc ce sont juste des fonctions qui le plus souvent simplifient l'écriture d'un script R!).

apply

Rappel: Renvoie un vecteur ou un tableau ou une liste de valeurs obtenus en appliquant une fonction aux marges d'un tableau ou d'une matrice.

Voyons un exemple:



by

Rappel: La fonction 'by' est un wrapper orienté objet pour 'tapply' appliqué aux trames de données.

Exemple:

```
> attach(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> # obtenir la moyenne des 4 premières variables, par espèce
> by(iris[, 1:4], Species, colMeans)
Species: setosa
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.006        3.428        1.462        0.246
---------------------------------------------------
Species: versicolor
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
---------------------------------------------------
Species: virginica
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       6.588        2.974        5.552        2.026
> 
```

eapply

Rappel: eapply applique FUN aux valeurs nommées d'un environnement et renvoie les résultats sous forme de liste.

Exemple:

```
> # un nouvel environnement
> e <- new.env()
> # deux variables d'environnement a et b
> e$a <- 1:10
> e$b <- 11:20
> # moyenne des variables
> eapply(e, mean)
$`a`
[1] 5.5

$b
[1] 15.5

>
```

lapply

Rappel: lapply renvoie une liste de la même longueur que X, dont chaque élément est le résultat de l'application de FUN à l'élément correspondant de X.

Exemple:


```
> # créer une liste avec 2 éléments
> l <- list(a = 1:10, b = 11:20)
> # la moyenne des valeurs dans chaque élément
> lapply(l, mean)
$`a`
[1] 5.5

$b
[1] 15.5

> # la somme des valeurs dans chaque élément
> lapply(l, sum)
$`a`
[1] 55

$b
[1] 155

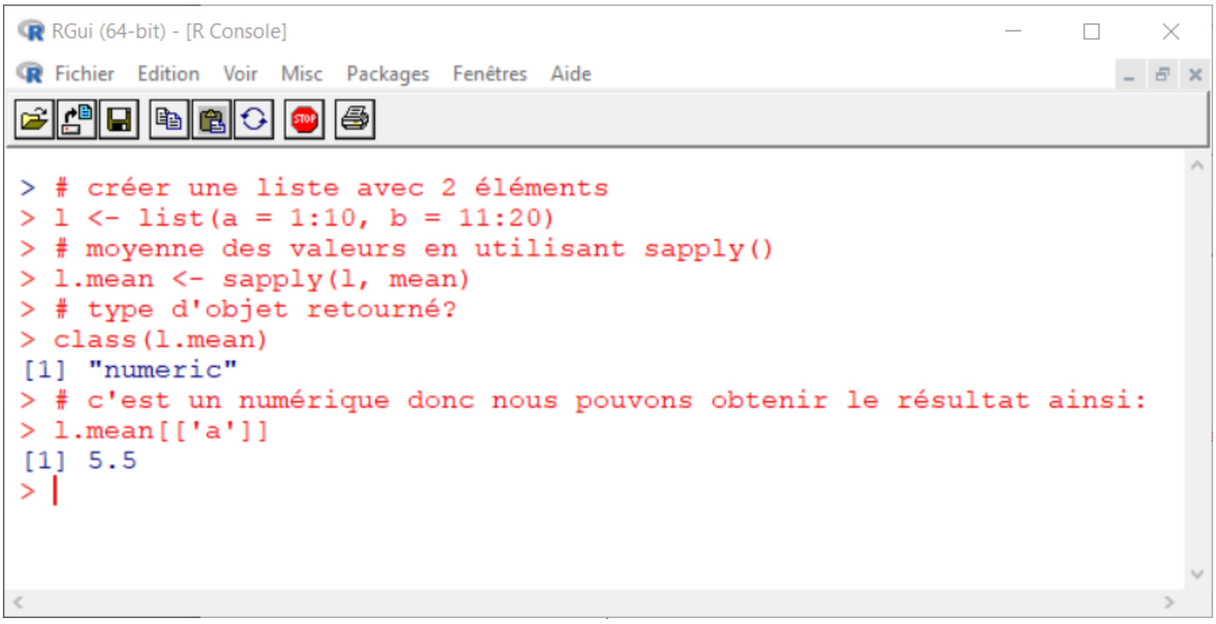
>
```

La documentation sur **lapply( )** nous dit de consulter la documentation pour **sapply( )** , **vapply( )** . Faisons cela.

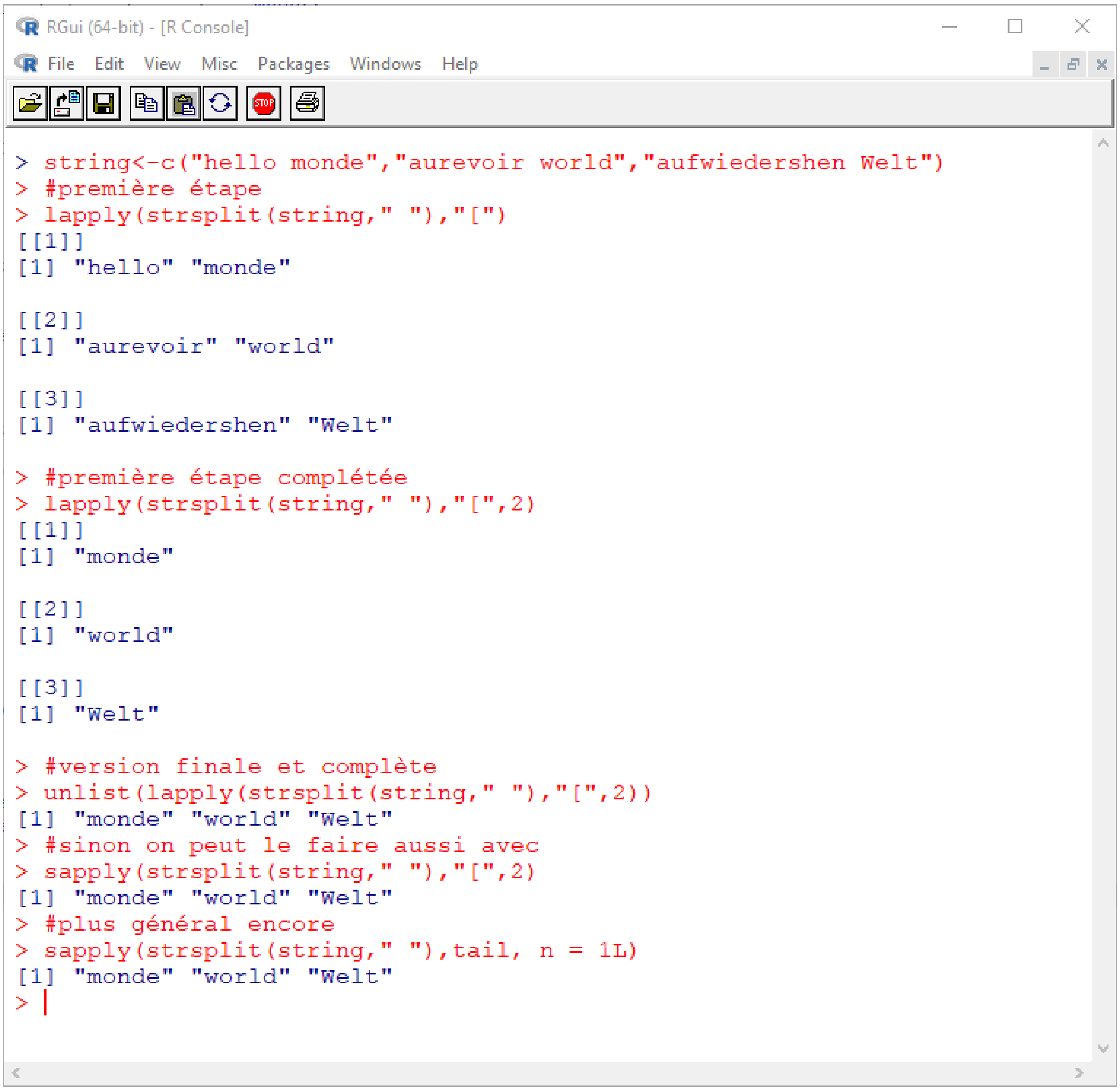
sapply

Rappel: sapply est une version conviviale de lapply par défaut renvoyant un vecteur ou une matrice si nécessaire.

Exemple:



On peut voir la différence en revenant sur un exemple que nous avions fait lors de notre études sur la manipulation de textes:



vapply

Rappel: **vapply( )** est similaire à **sapply( )** , mais a un type de valeur de retour pré-spécifié, donc il peut être plus sûr (et parfois plus rapide) à utiliser.

Exemple:

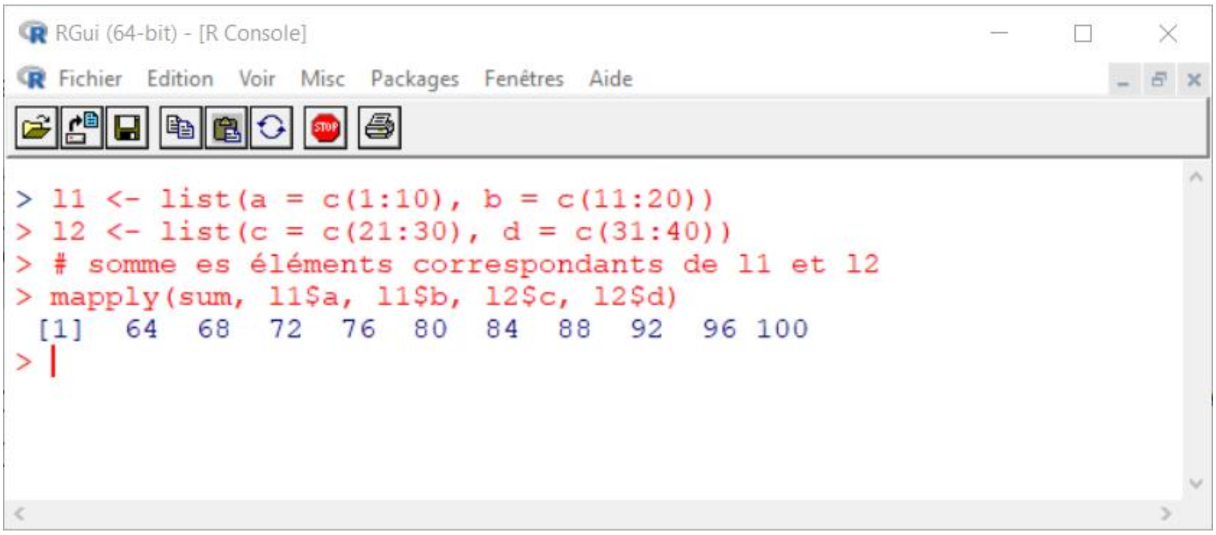
Un troisième argument est fourni à **vapply( )** , que vous pouvez considérer comme une sorte de modèle pour la sortie. La documentation utilise la fonction **fivenum** comme exemple, alors allons-y avec ça:



mapply

Rappel: **mapply( )** est une version multivariée de **sapply( )** . **mapply( )** applique FUN aux premiers éléments de chaque argument (...), aux deuxièmes éléments, aux troisièmes éléments, etc.

La documentation de **mapply( )** est pleine d'exemples assez complexes, mais voici un exemple simple et stupide:



Ici, nous sommons l1$a[1]+l1$b[1]+l2$c[1]+l2$d[1]=(1 + 11 + 21 + 31) pour obtenir 64, le premier élément du retour liste. C'est le même principe jusqu'à l1$a[10]+l1$b[10]+l2$c[10]+l2$d[10]=(10+20+30+40)=100, le dernier élément.

rapply

Rappel: Rapply est une version récursive[13] de **lapply( )** .

Exemple:

```
> # commencons avec notre exemple classique de liste
> l <- list(a = 1:10, b = 11:20)
> # log2 de chaque élément de la liste
> rapply(l, log2)
      a1       a2       a3       a4       a5       a6       a7       a8 
0.000000 1.000000 1.584963 2.000000 2.321928 2.584963 2.807355 3.000000 
      a9      a10       b1       b2       b3       b4       b5       b6 
3.169925 3.321928 3.459432 3.584963 3.700440 3.807355 3.906891 4.000000 
      b7       b8       b9      b10 
4.087463 4.169925 4.247928 4.321928 
> # log2 de chaque valeur de la liste
> rapply(l, log2, how = "list")
$`a`
 [1] 0.000000 1.000000 1.584963 2.000000 2.321928 2.584963 2.807355
 [8] 3.000000 3.169925 3.321928

$b
 [1] 3.459432 3.584963 3.700440 3.807355 3.906891 4.000000 4.087463
 [8] 4.169925 4.247928 4.321928

> # et qu'advient-t-il si la fonctione est la moyenne?
> rapply(l, mean)
   a    b 
 5.5 15.5 
> rapply(l, mean, how = "list")
$`a`
[1] 5.5

$b
[1] 15.5

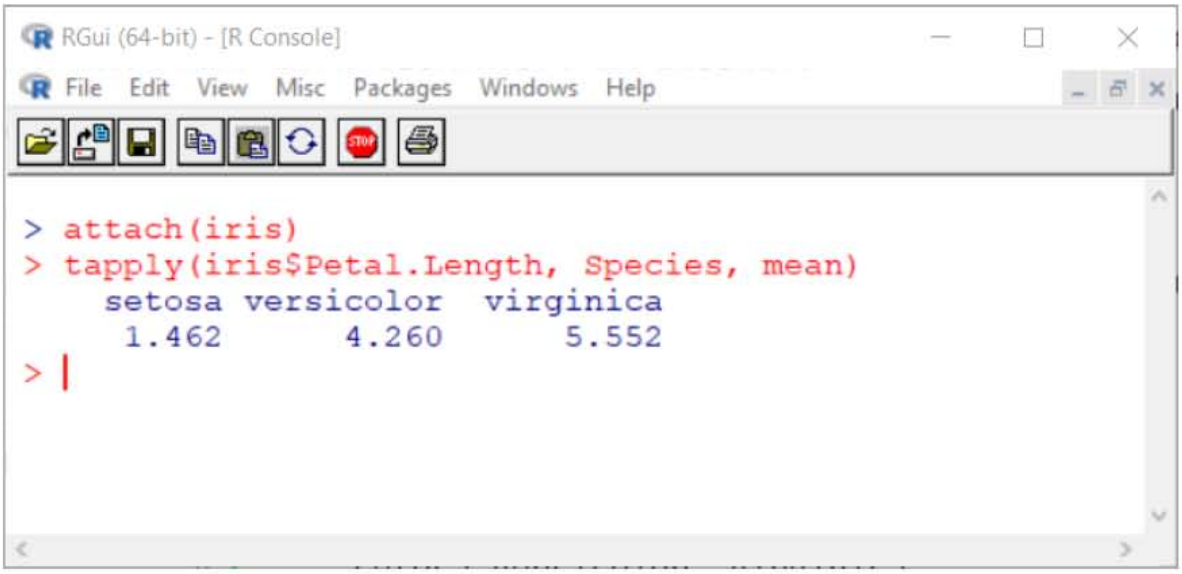
> 
```

[13] Je pense que "récursif" est un peu trompeur. Ce que fait le **rapply( )** est d'appliquer des fonctions aux listes de différentes manières, en fonction des arguments fournis.

tapply

Rappel: Applique une fonction à chaque cellule d'un tableau décalé, c'est-à-dire à chaque groupe de valeurs (non vide) donné par une combinaison unique des niveaux de certains facteurs.

Ok comme cela ce n'est pas clair du tout. Voyons un simple exemple ce sera plus parlant…



Data Wrangling

Avant de commencer, un petit visual qui benchmark des technologies pour le traitement de gros jeux de données, est plus qu'intéressant:



Packages sqldf et tidyquery

Pour la traitement de données rien qu'avec du SQL (langage relativement important en Data Science), le lecteur peut se référer à la page 446.

Sinon voici déjà une petite entrée…:

```
> library("tidyquery")
> library("nycflights13")
> query(
+     "SELECT origin, dest,
+        COUNT(flight) AS num_flts,
+        round(SUM(seats)) AS num_seats,
+        round(AVG(arr_delay)) AS avg_delay
+     FROM flights f LEFT OUTER JOIN planes p
+        ON f.tailnum = p.tailnum
+     WHERE distance BETWEEN 200 AND 300
+        AND air_time IS NOT NULL
+     GROUP BY origin, dest
+     HAVING num_flts > 3000
+     ORDER BY num_seats DESC, avg_delay ASC
+     LIMIT 2;"
+ )
# A tibble: 2 x 5
  origin dest  num_flts num_seats avg_delay
  <chr>  <chr>    <int>     <dbl>     <dbl>
1 LGA    DCA       4468    712643         6
2 EWR    BOS       5247    611192         5
>
```

Package dplyr

Je recommande toujours d'éviter au maximum d'utiliser des packages pour faire ce que les packages natifs de R font déjà. Mais certaines personnes n'écoutant pas… le package **dplyr**[14] (qui est une itération du package **plyr**) reste toutefois beaucoup utilisé de par sa rapidité et c'est pourquoi nous allons exceptionnellement voir de quoi il s'agit relativement en détails (même si je déteste faire cela) mais sans toutefois étudier toutes les combinaisons possibles (car la combinatoire tend vers l'infini…).

Remarque: Il semblerait que le package **data.table** et **dplyr** sont comparables en vitesse. Mais comme les packages évoluent (raison pour laquelle nous ne fournirons pas de comparaison chiffrée!), il faut régulièrement les tester et les comparer. Cependant il est clair et net que **dplyr** à une syntaxe plus lisible que celle de **data.table** et permet également un accès plus facile à des bases de données externes.

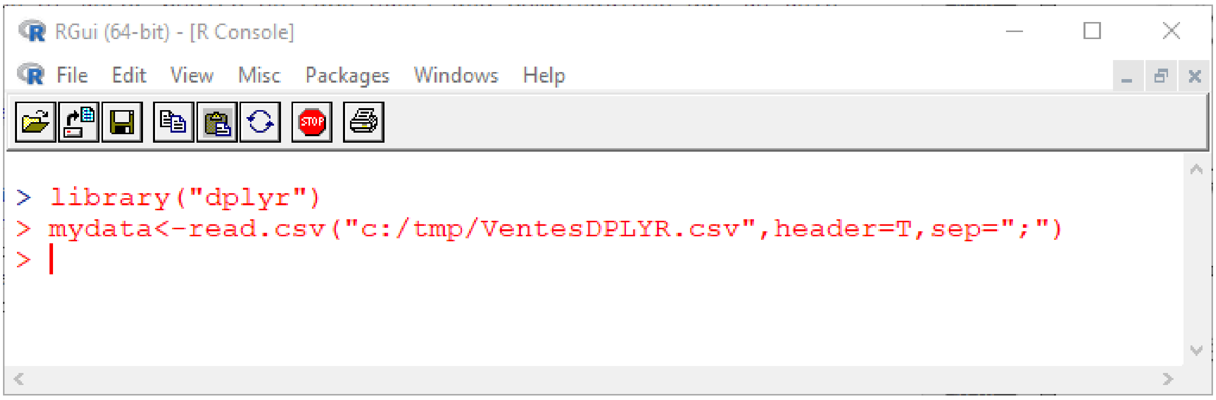
Attention!!! Comme pour tout package non-natif, les noms des fonctions changent parfois d'une itération à une autre…

Pour en savoir plus le lecteur pour se référer à:

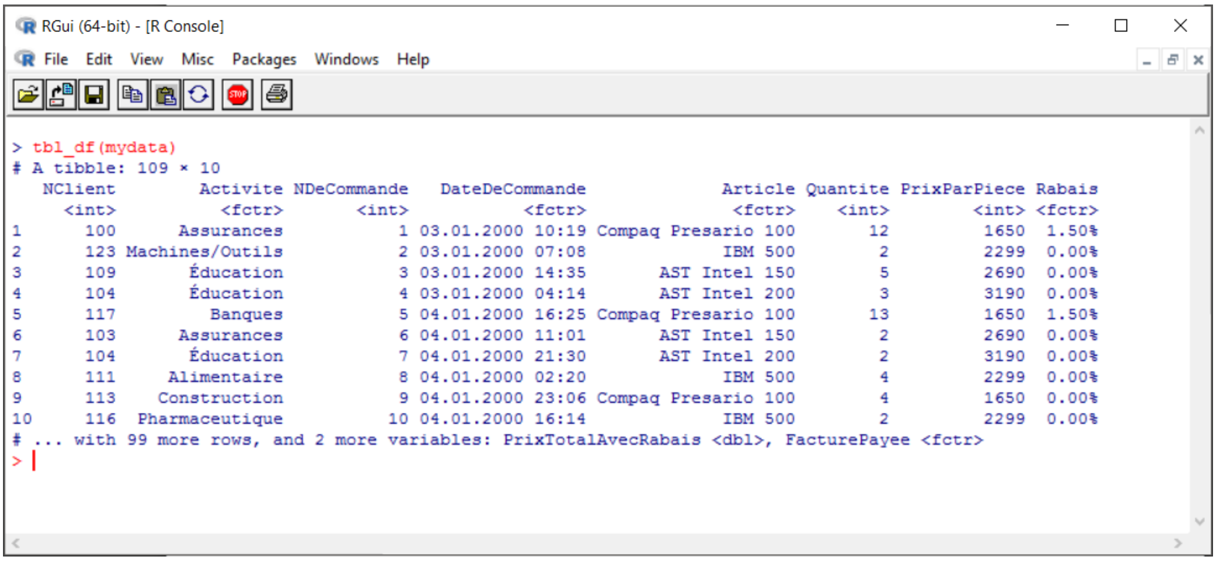
[14] Le "d" est pour dataframes, le "plyr" et pour évoquer des"pliers" ("pinces" en français)

https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

Bon chargeons le package et le dataframe que nous allons utiliser:

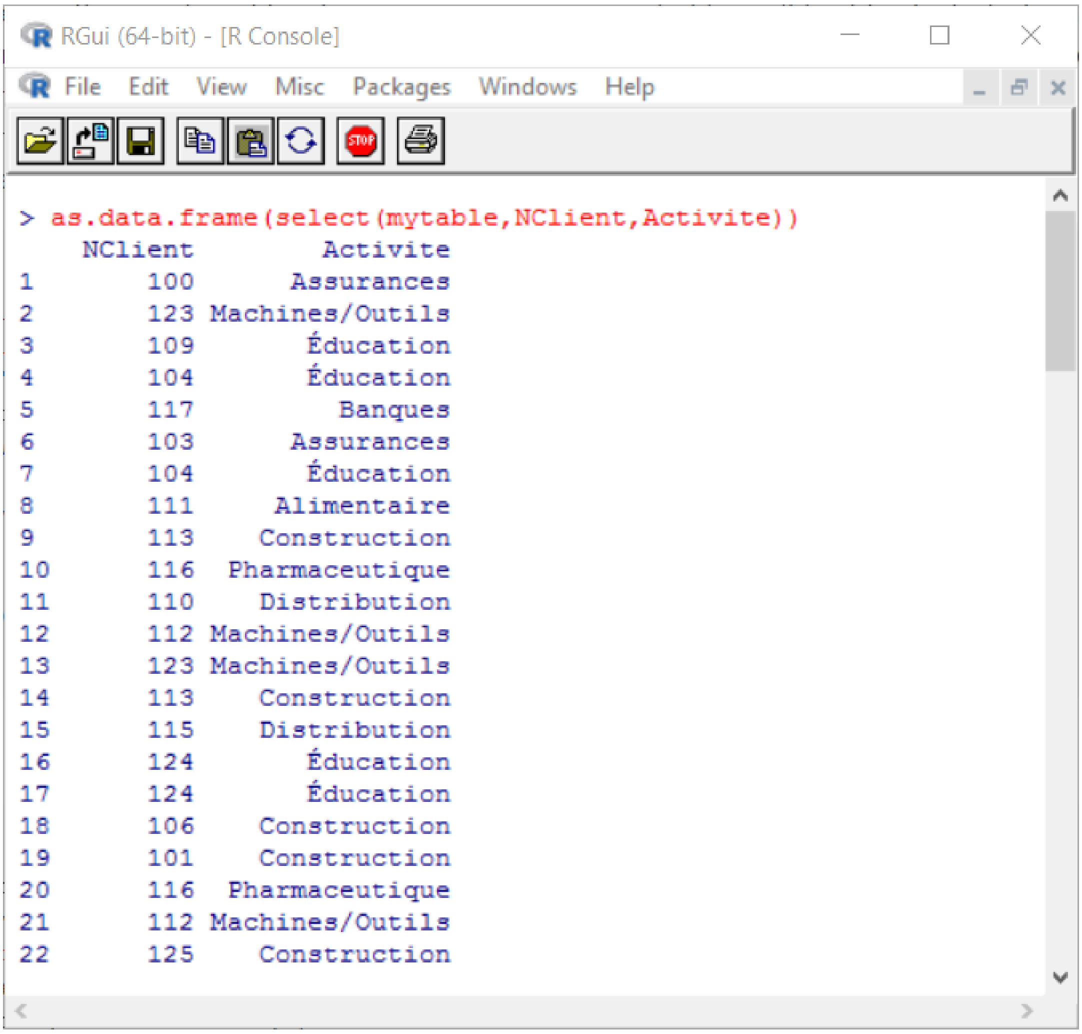


Voyons comment convertir le jeu de données en classe *tbl* avec la fonction **tbl_df( )**. Les *tbl* sont plus faciles à explorer que les data frames:

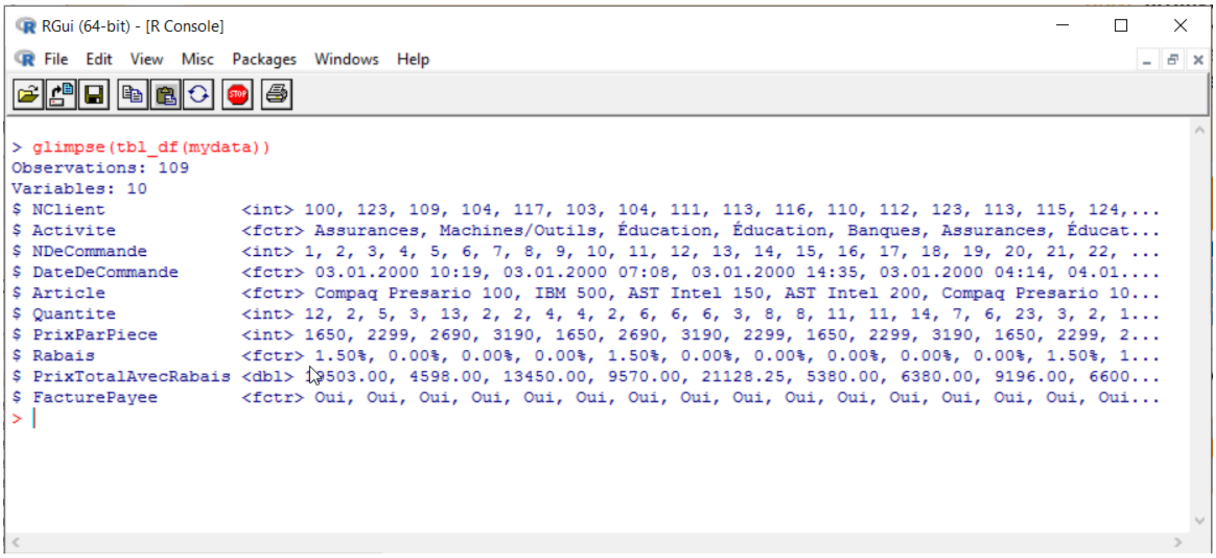


R n'affiche que les données adaptées à la taille de l'écran!

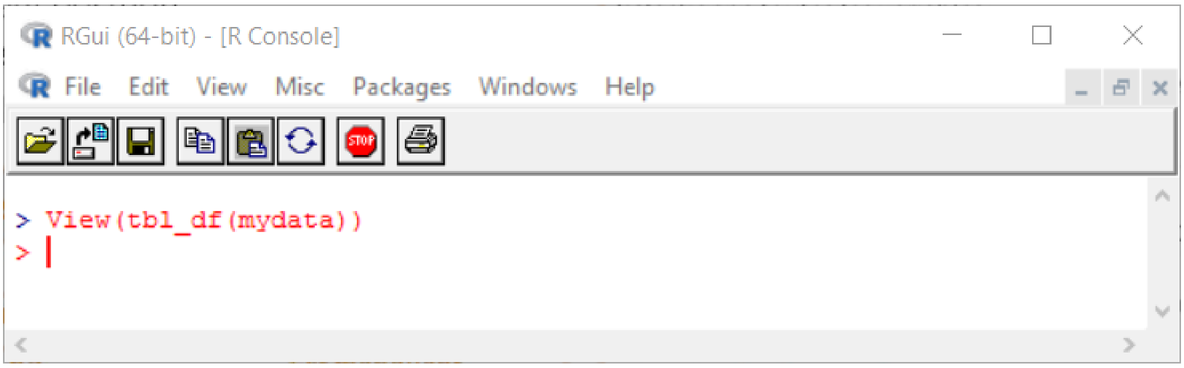
Pour l'opération inverse, nous utilisons la fonction **as.data.frame( )**:



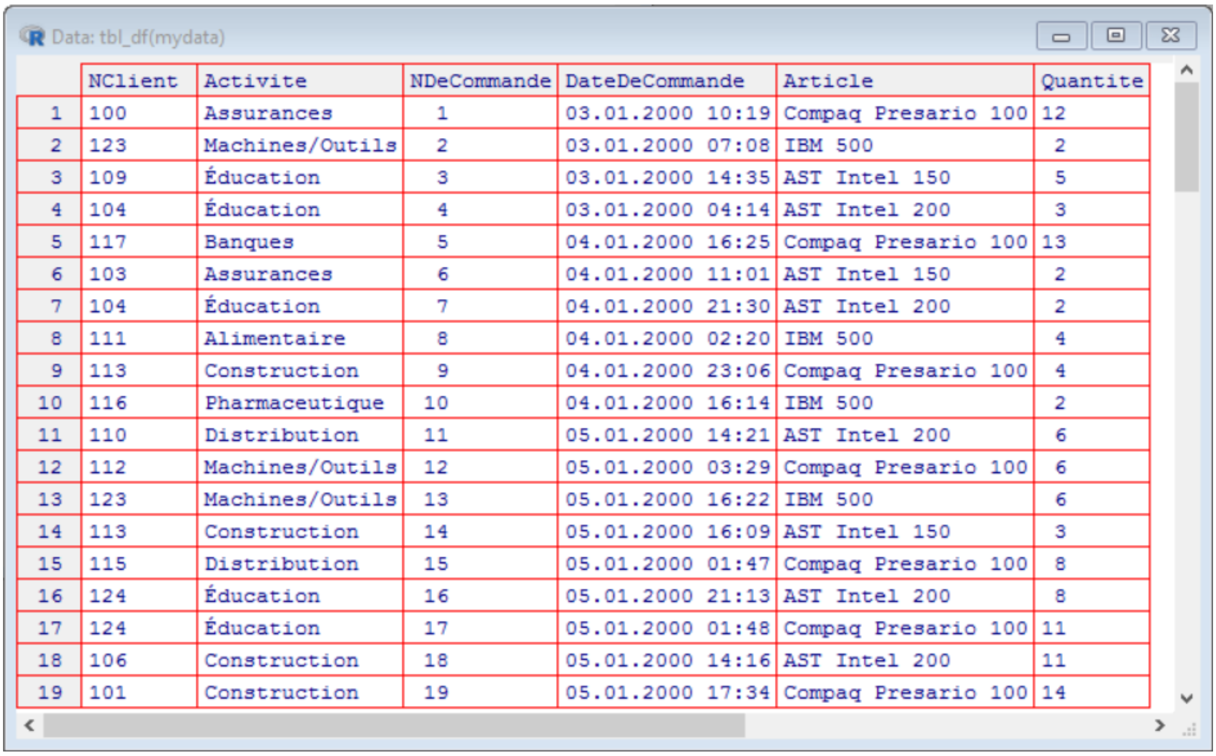
Pour avoir un résumé, nous utilisons la fonction **glimpse( )**:



Comme d'habitude nous pouvons utiliser la fonction **View( )** pour afficher les données dans un tableur:



Ce qui donne:


Data: tbl_df(mydata)

	NClient	Activite	NDeCommande	DateDeCommande	Article	Quantite
1	100	Assurances	1	03.01.2000 10:19	Compaq Presario 100	12
2	123	Machines/Outils	2	03.01.2000 07:08	IBM 500	2
3	109	Éducation	3	03.01.2000 14:35	AST Intel 150	5
4	104	Éducation	4	03.01.2000 04:14	AST Intel 200	3
5	117	Banques	5	04.01.2000 16:25	Compaq Presario 100	13
6	103	Assurances	6	04.01.2000 11:01	AST Intel 150	2
7	104	Éducation	7	04.01.2000 21:30	AST Intel 200	2
8	111	Alimentaire	8	04.01.2000 02:20	IBM 500	4
9	113	Construction	9	04.01.2000 23:06	Compaq Presario 100	4
10	116	Pharmaceutique	10	04.01.2000 16:14	IBM 500	2
11	110	Distribution	11	05.01.2000 14:21	AST Intel 200	6
12	112	Machines/Outils	12	05.01.2000 03:29	Compaq Presario 100	6
13	123	Machines/Outils	13	05.01.2000 16:22	IBM 500	6
14	113	Construction	14	05.01.2000 16:09	AST Intel 150	3
15	115	Distribution	15	05.01.2000 01:47	Compaq Presario 100	8
16	124	Éducation	16	05.01.2000 21:13	AST Intel 200	8
17	124	Éducation	17	05.01.2000 01:48	Compaq Presario 100	11
18	106	Construction	18	05.01.2000 14:16	AST Intel 200	11
19	101	Construction	19	05.01.2000 17:34	Compaq Presario 100	14

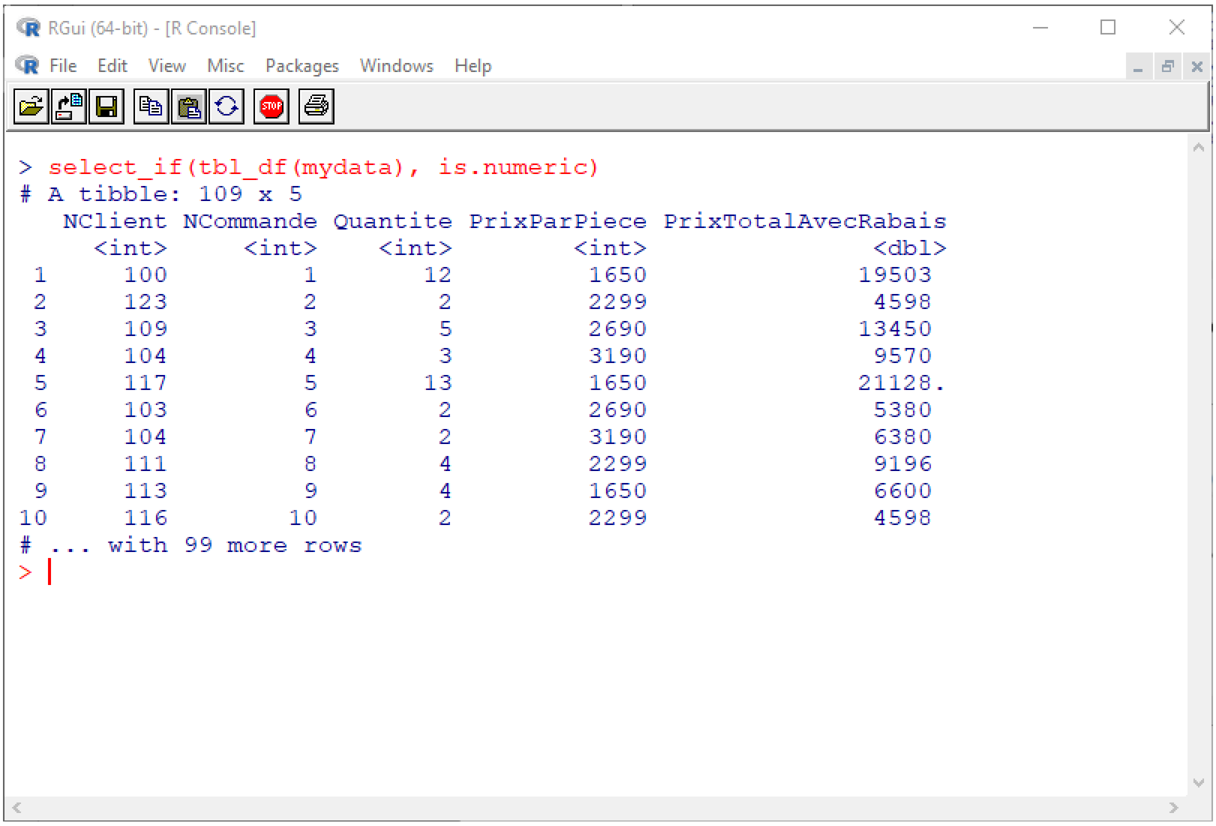
On peut sélectionner seulement certaines colonnes à choix avec la fonction **select( )**:

```
> select(mytable,NClient,Activite)
# A tibble: 109 × 2
   NClient         Activite
     <int>           <fctr>
1      100       Assurances
2      123 Machines/Outils
3      109        Éducation
4      104        Éducation
5      117          Banques
6      103       Assurances
7      104        Éducation
8      111      Alimentaire
9      113     Construction
10     116   Pharmaceutique
# ... with 99 more rows
>
```

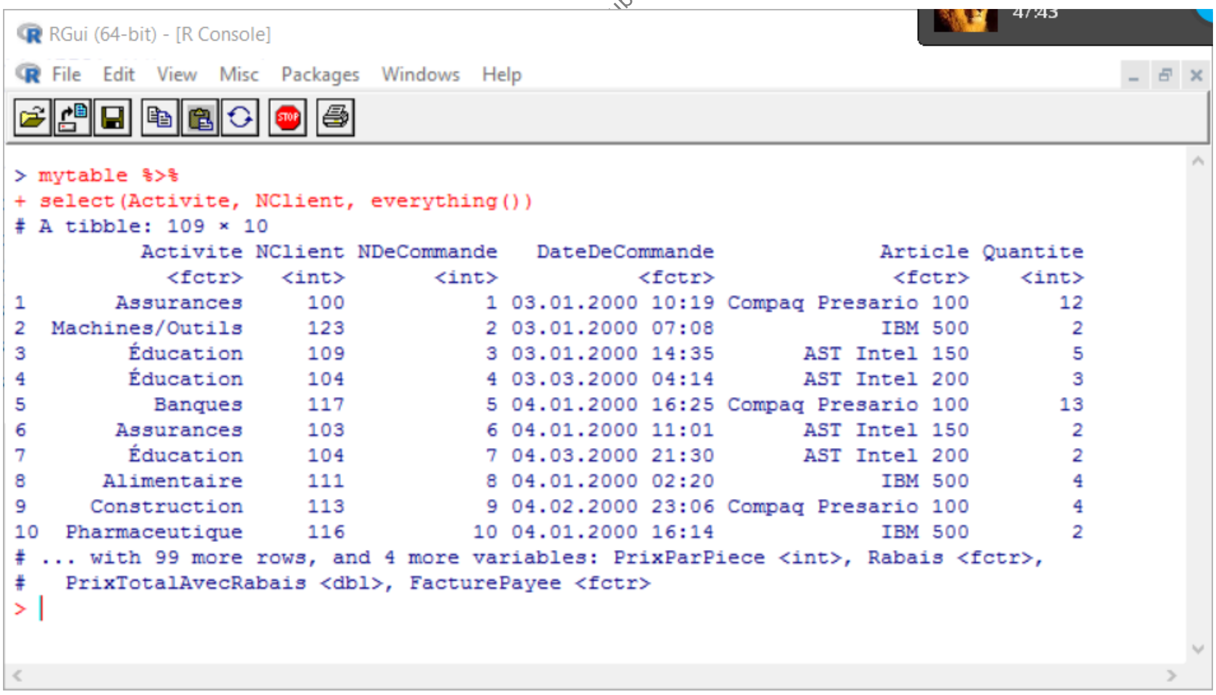
ou pour retirer une ou plusieurs colonnes:

```
> mytable %>%
+ select(-NClient,-Activite)
# A tibble: 109 × 8
   NDeCommande   DateDeCommande              Article Quantite PrixParPiece Rabais
         <int>           <fctr>               <fctr>    <int>        <int> <fctr>
1            1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%
2            2 03.01.2000 07:08              IBM 500        2         2299  0.00%
3            3 03.01.2000 14:35        AST Intel 150        5         2690  0.00%
4            4 03.03.2000 04:14        AST Intel 200        3         3190  0.00%
5            5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
6            6 04.01.2000 11:01        AST Intel 150        2         2690  0.00%
7            7 04.03.2000 21:30        AST Intel 200        2         3190  0.00%
8            8 04.01.2000 02:20              IBM 500        4         2299  0.00%
9            9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%
10          10 04.01.2000 16:14              IBM 500        2         2299  0.00%
# ... with 99 more rows, and 2 more variables: PrixTotalAvecRabais <dbl>,
#   FacturePayee <fctr>
>
```

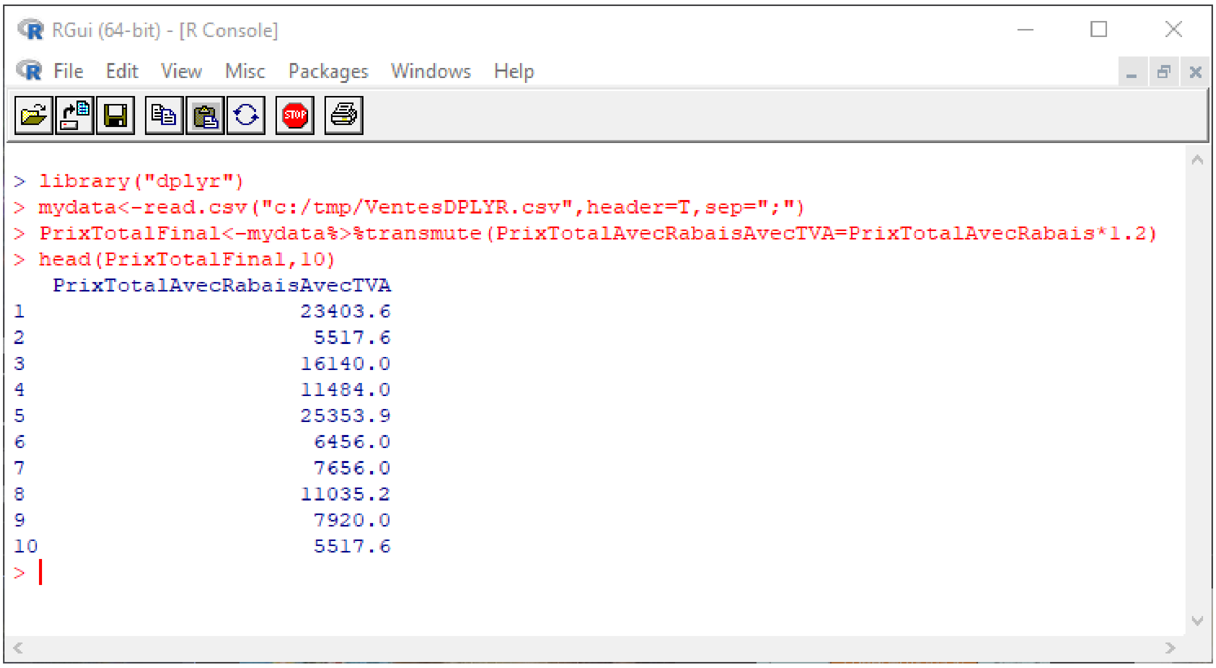
Ou pour retirer toutes les colonnes non-numériques:



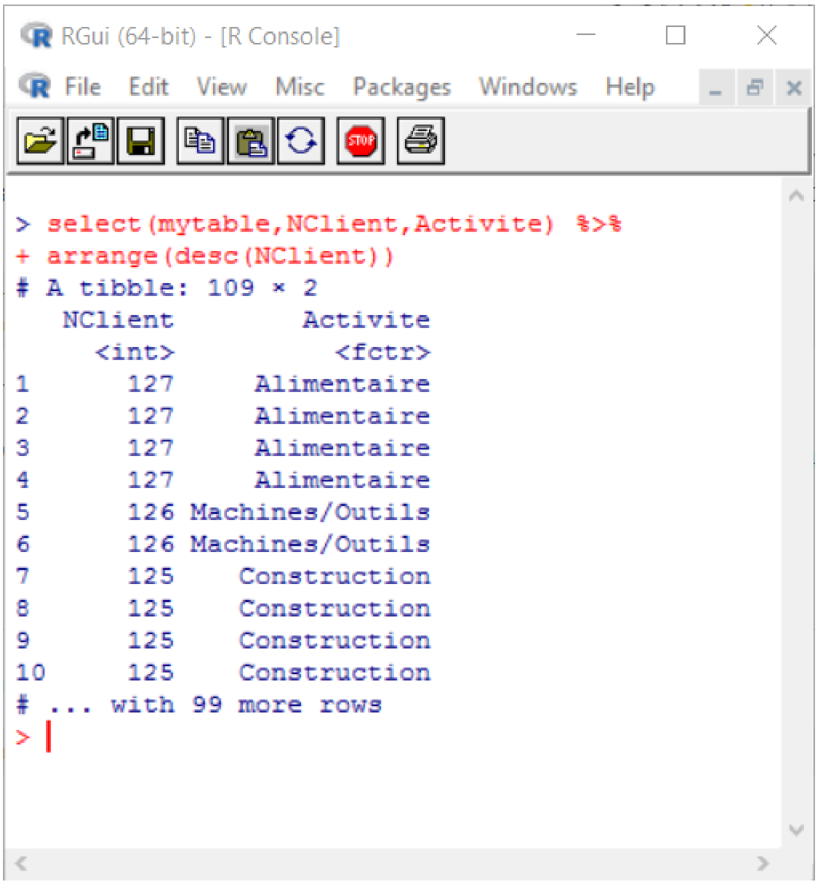
Ou encore pour changer l'ordre de certaines colonnes seulement en préservant le reste des colonnes:



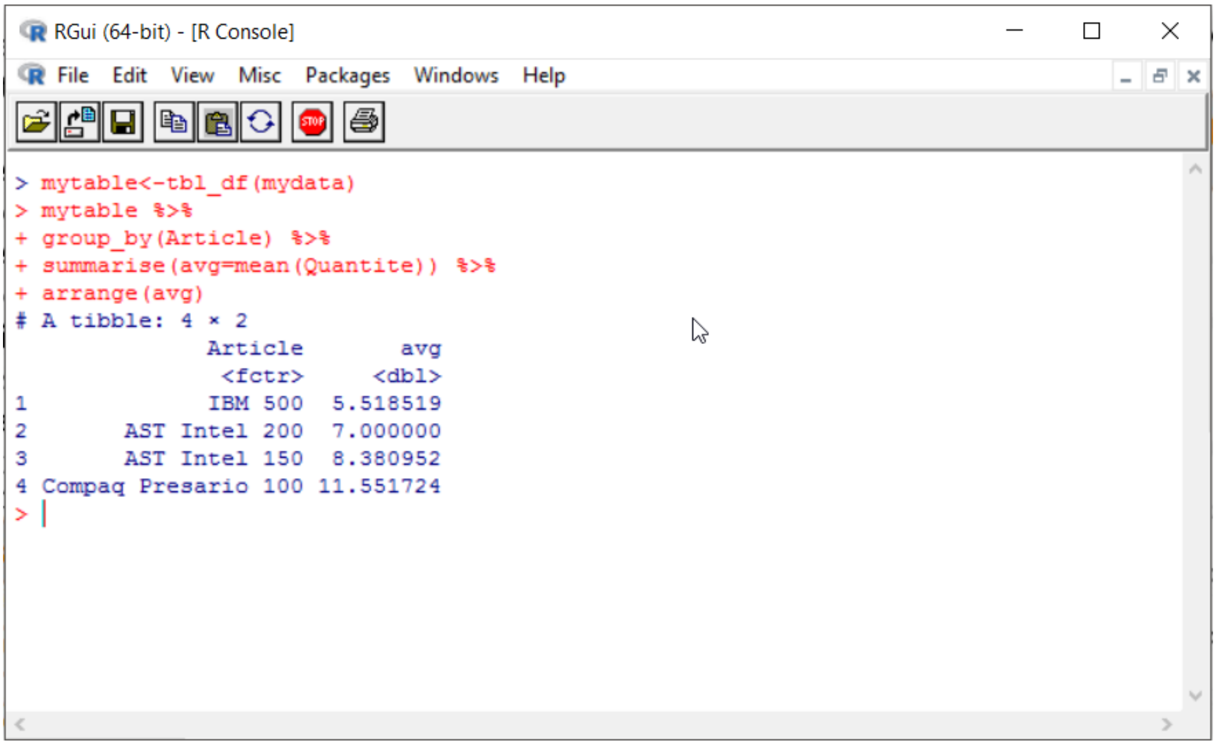
Nous pouvons aussi faire en très peu de lignes un calcul et en préserver uniquement le résultat sans utiliser la commande select (il suffit d'utiliser transmute et d'affecter le résultat à une variable):



Il est possible de faire un tri avec la fonction **arrange( )** où nous découvrons le symbole **%>%** (appelé "pipe" en anglais ou "puis" en français comme "…et puis…"):

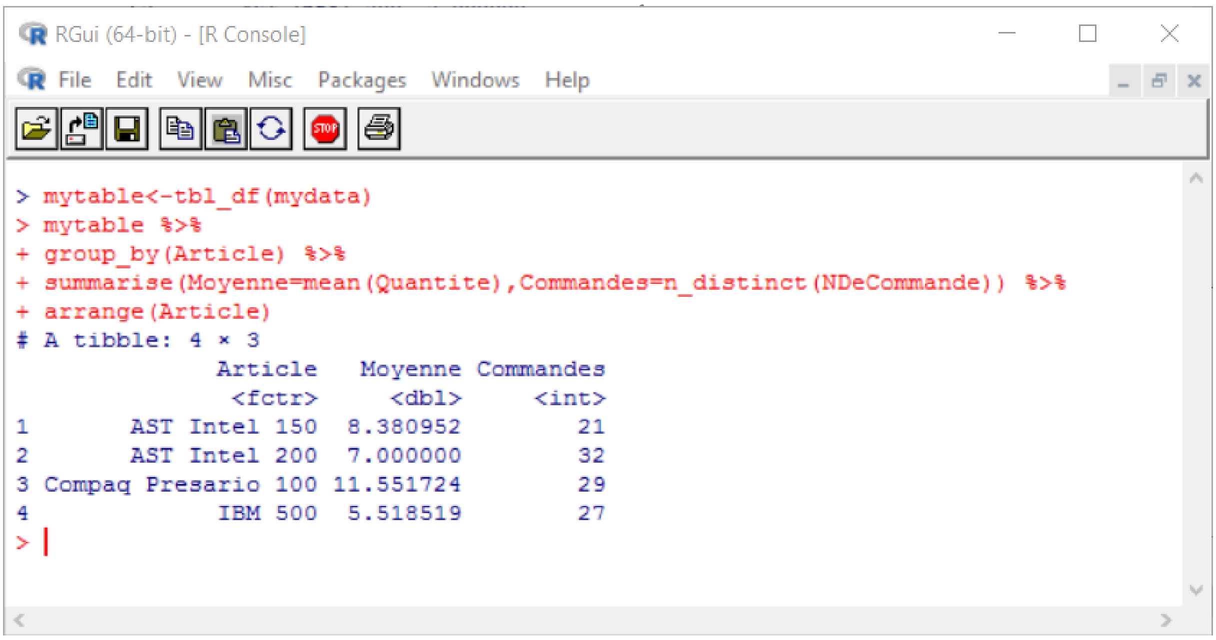


Symbole que nous pouvons généraliser:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mytable<-tbl_df(mydata)
> mytable %>%
+ group_by(Article) %>%
+ summarise(avg=mean(Quantite)) %>%
+ arrange(avg)
# A tibble: 4 × 2
              Article       avg
               <fctr>     <dbl>
1             IBM 500  5.518519
2       AST Intel 200  7.000000
3       AST Intel 150  8.380952
4 Compaq Presario 100 11.551724
>
```

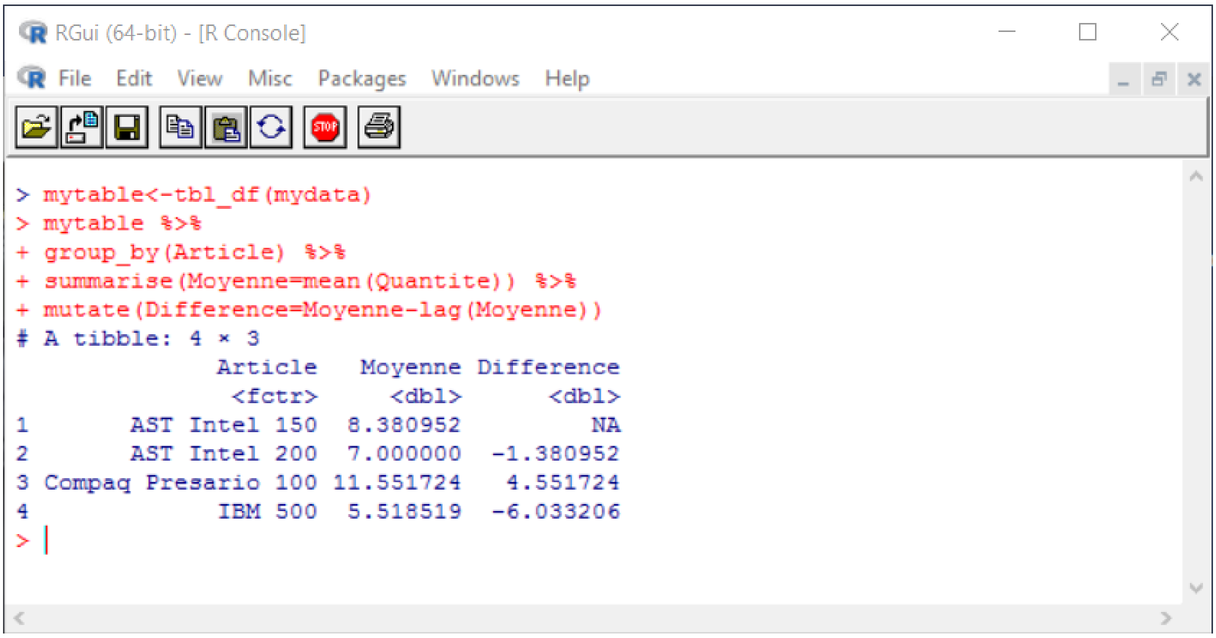
ou encore avec plus de finesse pour prendre un autre exemple typique (cela fait bien évidemment penser aux tableaux croisés dynamiques de nombreux tableurs):



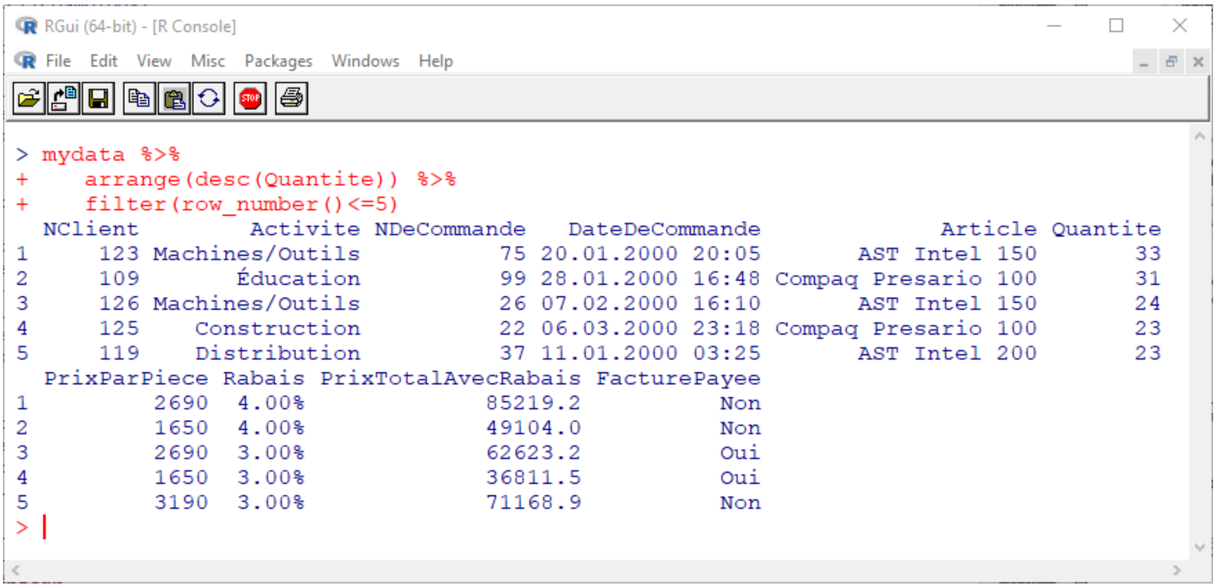
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mytable<-tbl_df(mydata)
> mytable %>%
+ group_by(Article) %>%
+ summarise(Moyenne=mean(Quantite),Commandes=n_distinct(NDeCommande)) %>%
+ arrange(Article)
# A tibble: 4 × 3
              Article   Moyenne Commandes
               <fctr>     <dbl>     <int>
1       AST Intel 150  8.380952        21
2       AST Intel 200  7.000000        32
3 Compaq Presario 100 11.551724        29
4             IBM 500  5.518519        27
>
```

ou encore si on veut des différences successives (très utilisé en finance avec des complications significatives[15]!):

[15] Transformation en pourcent, ou calcul du logarithme natural des ratios des rendements, etc.



On peut aussi (cas hyper classique!) vouloir extraire que les *n* plus grands éléments d'un data frame:



Sinon on peut aussi utiliser:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

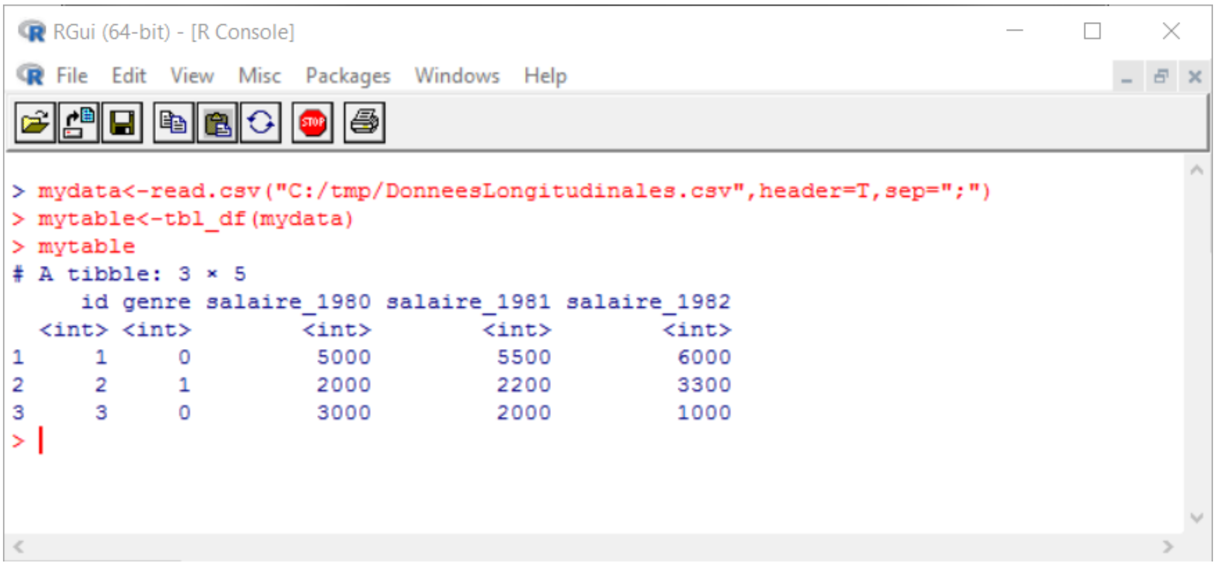
> top_n(mydata,5,Quantite)
  NClient           Activite NDeCommande    DateDeCommande             Article Quantite
1     125       Construction          22 06.03.2000 23:18 Compaq Presario 100       23
2     126 Machines/Outils             26 07.02.2000 16:10       AST Intel 150       24
3     119       Distribution          37 11.01.2000 03:25       AST Intel 200       23
4     123 Machines/Outils             75 20.01.2000 20:05       AST Intel 150       33
5     109          Éducation          99 28.01.2000 16:48 Compaq Presario 100       31
  PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1         1650  3.00%             36811.5          Oui
2         2690  3.00%             62623.2          Oui
3         3190  3.00%             71168.9          Non
4         2690  4.00%             85219.2          Non
5         1650  4.00%             49104.0          Non
> |
```

Nous pouvons aussi calculer les rangs:

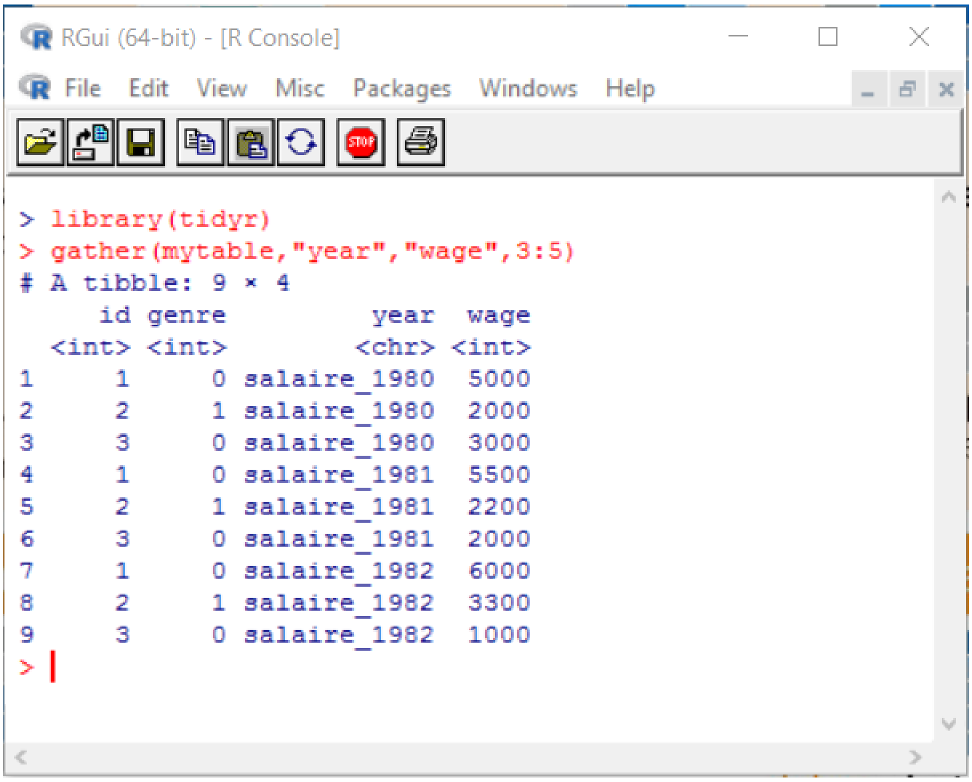
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("dplyr")
> mydata<-read.csv("c:/tmp/VentesDPLYR.csv",header=T,sep=";")
>
> mydata %>%
+ group_by(NClient) %>%
+ summarise(SumQuantite=sum(Quantite)) %>%
+ arrange(desc(SumQuantite)) %>%
+ mutate(rank = rank(-SumQuantite))
# A tibble: 28 x 3
   NClient SumQuantite  rank
     <int>       <int> <dbl>
 1     109          69   1
 2     123          61   2
 3     121          58   3
 4     101          54   4
 5     108          50   5
 6     103          40   6.5
 7     106          40   6.5
 8     126          36   8
 9     104          34   9.5
10     105          34   9.5
# ... with 18 more rows
> |
```

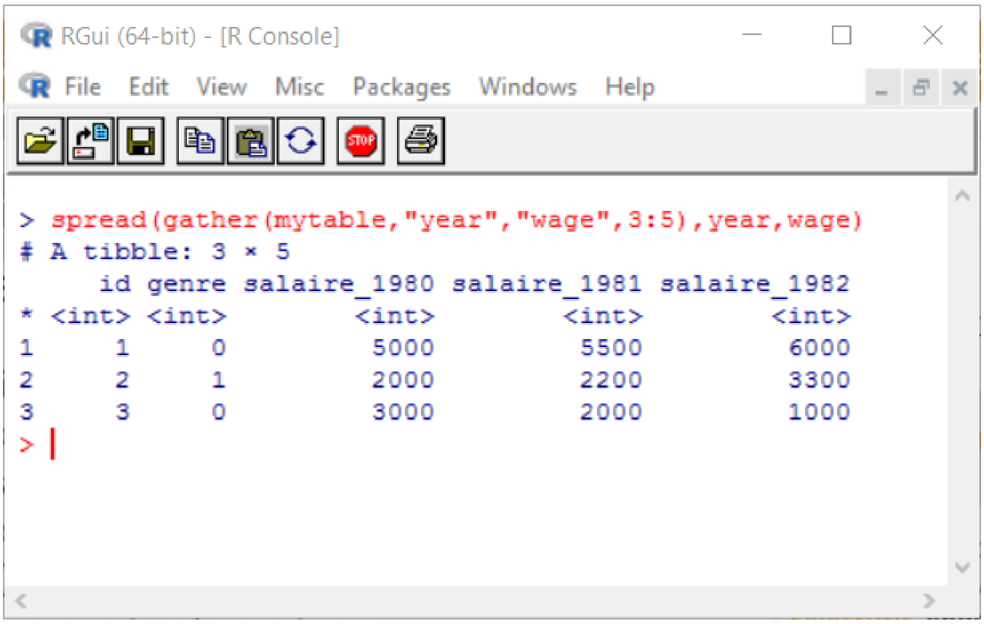
Maintenant considérons le petit fichier de données longitudinales (wide-format) suivant:



et avec le package **tidyr** et la fonction **gather( )** nous obtenons (long-format):

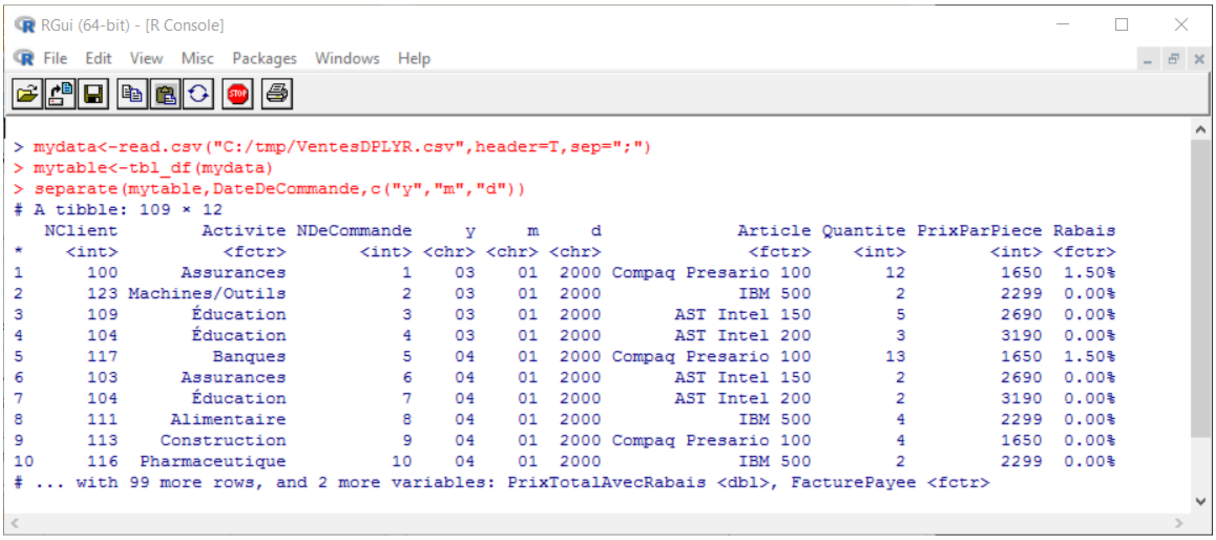


On peut faire l'opération inverse (long to wide) avec la fonction **spread( )** toujours du package **tidyr**:



```
> spread(gather(mytable,"year","wage",3:5),year,wage)
# A tibble: 3 × 5
     id genre salaire_1980 salaire_1981 salaire_1982
* <int> <int>        <int>        <int>        <int>
1     1     0         5000         5500         6000
2     2     1         2000         2200         3300
3     3     0         3000         2000         1000
>
```

Avec la fonction **separate( )** on peut scinder une colonne de dates (remarquez l'association erronée des années et des jours!!!!):



```
> mydata<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mytable<-tbl_df(mydata)
> separate(mytable,DateDeCommande,c("y","m","d"))
# A tibble: 109 × 12
   NClient        Activite NDeCommande     y     m     d            Article Quantite PrixParPiece Rabais
*    <int>          <fctr>       <int> <chr> <chr> <chr>             <fctr>    <int>        <int> <fctr>
1      100      Assurances           1    03    01  2000 Compaq Presario 100       12         1650  1.50%
2      123 Machines/Outils           2    03    01  2000            IBM 500        2         2299  0.00%
3      109       Éducation           3    03    01  2000      AST Intel 150        5         2690  0.00%
4      104       Éducation           4    03    01  2000      AST Intel 200        3         3190  0.00%
5      117         Banques           5    04    01  2000 Compaq Presario 100       13         1650  1.50%
6      103      Assurances           6    04    01  2000      AST Intel 150        2         2690  0.00%
7      104       Éducation           7    04    01  2000      AST Intel 200        2         3190  0.00%
8      111     Alimentaire           8    04    01  2000            IBM 500        4         2299  0.00%
9      113    Construction           9    04    01  2000 Compaq Presario 100        4         1650  0.00%
10     116   Pharmaceutique         10    04    01  2000            IBM 500        2         2299  0.00%
# ... with 99 more rows, and 2 more variables: PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
```

Et pour ceux qui connaissent bien les tableaux croisés dynamiques, nous pouvons faire une analyse similaire avec répétition des lignes de groupement (comportement exclusif de dplyr):

```
> mytable<-tbl_df(mydata)
> mytable<-separate(mytable,DateDeCommande,c("y","m","d"))
Warning message:
Too many values at 109 locations: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...
> mytable %>%
+ group_by(d,y) %>%
+ summarise(mn_amt=mean(Quantite))
Source: local data frame [21 x 3]
Groups: d [?]

       d     y    mn_amt
   <chr> <chr>     <dbl>
1   2000    03  5.500000
2   2000    04  4.500000
3   2000    05  8.111111
4   2000    06  8.200000
5   2000    07 10.500000
6   2000    10  7.000000
7   2000    11  9.000000
8   2000    12  5.666667
9   2000    13  8.000000
10  2000    14  3.500000
# ... with 11 more rows
>
```

On peut refaire l'inverse avec la fonction **unite_( )**:

```
> unite_(separate(mytable,DateDeCommande,c("y","m","d")),"DateCommande",c("y","m","d"),".")
# A tibble: 109 × 10
   NClient        Activite NDeCommande DateCommande            Article Quantite PrixParPiece Rabais
*    <int>          <fctr>       <int>        <chr>             <fctr>    <int>        <int> <fctr>
1      100     Assurances           1   03.01.2000 Compaq Presario 100       12         1650  1.50%
2      123 Machines/Outils          2   03.01.2000             IBM 500        2         2299  0.00%
3      109       Éducation          3   03.01.2000      AST Intel 150        5         2690  0.00%
4      104       Éducation          4   03.01.2000      AST Intel 200        3         3190  0.00%
5      117         Banques          5   04.01.2000 Compaq Presario 100       13         1650  1.50%
6      103      Assurances          6   04.01.2000      AST Intel 150        2         2690  0.00%
7      104       Éducation          7   04.01.2000      AST Intel 200        2         3190  0.00%
8      111     Alimentaire          8   04.01.2000             IBM 500        4         2299  0.00%
9      113    Construction          9   04.01.2000 Compaq Presario 100        4         1650  0.00%
10     116  Pharmaceutique         10   04.01.2000             IBM 500        2         2299  0.00%
# ... with 99 more rows, and 2 more variables: PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
Warning message:
Too many values at 109 locations: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2$
>
```

Voyons un autre cas d'application de **separate( )** (rappelons que cette dernière permet de séparer des colonnes) et **unite_( )** (qui permet de fusionner des colonnes**)**:

```
> library("dplyr")
> library("tidyverse")
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> head(unite_(mtcars, "vs_am", c("vs","am")))
                   mpg cyl disp  hp drat    wt  qsec vs_am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46   0_1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02   0_1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61   1_1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44   1_0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02   0_0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22   1_0    3    1
> mtcars %>%
+   unite("vs_am", vs, am) %>%
+   separate(vs_am, c("vs", "am"))
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

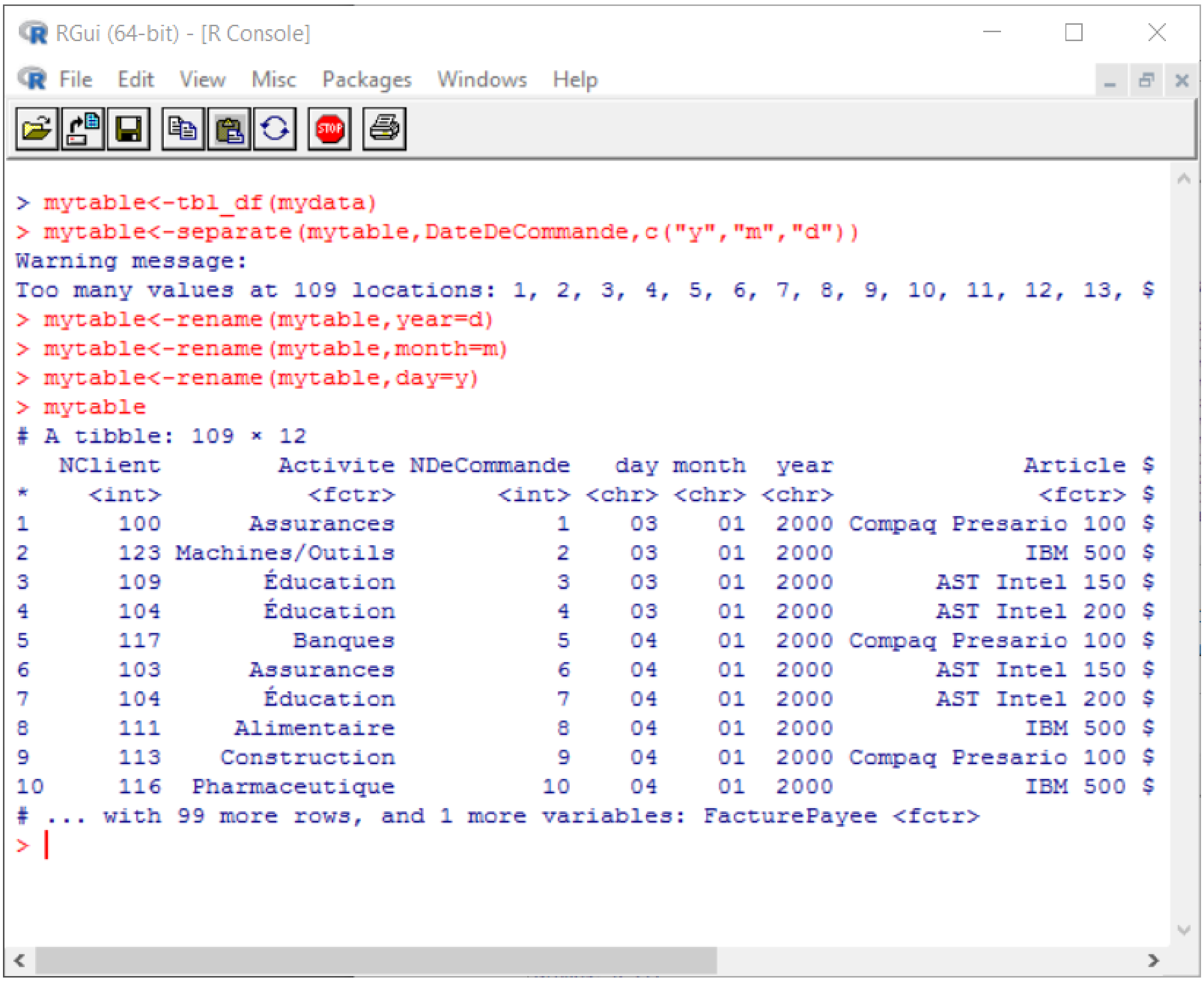
Enfin un autre exemple classique (et malheureux) d'application de **unite_( )** pour corriger des stockages de données mal conçus (merci à une de mes post-doctorante pour m'avoir soumis le problème car je sais qu'elle n'est pas la seule dans ce genre de situation m*******).
Considérons les données suivantes:

```
> library("dplyr")
> library("tidyverse")
> (my_bad_data<- read.csv("C:/tmp/unite.csv", header=T, sep=","))
            StartDate            EndDate Duration LocationLatitude LocationLongitude
1  2019-01-02 09:35 2019-01-02 09:36       51         46.99181          6.931000
2  2019-01-02 10:19 2019-01-02 10:20       72         46.99181          6.931000
3  2019-01-03 05:35 2019-01-03 05:43      467         46.99181          6.931000
4  2019-01-03 05:49 2019-01-03 05:51      136         46.99181          6.931000
5  2019-01-03 05:51 2019-01-03 05:53       95         46.99181          6.931000
6  2019-01-03 05:53 2019-01-03 05:54       80         46.99181          6.931000
7  2019-01-03 06:15 2019-01-03 06:17      121         46.99181          6.931000
8  2019-01-03 06:18 2019-01-03 06:19       55         46.99181          6.931000
9  2019-01-03 09:29 2019-01-03 09:39      625         46.51601          6.632797
10 2019-01-07 03:13 2019-01-07 03:24      664         46.99181          6.931000
   DistributionChannel UserLanguage Filler1_NEG Filler1_NEG.1 Filler1_NEG.2 Filler1_NEG.3
1              preview           EN          NA            NA            NA            NA
2              preview           EN          NA            NA            NA            NA
3              preview           EN          NA            NA            NA            NA
4              preview           EN          NA            NA            NA            NA
5              preview           EN          NA            NA            NA            NA
6              preview           EN          NA            NA            NA            NA
7              preview           EN          NA            NA            NA             3
8              preview           EN          NA             4            NA            NA
9            anonymous           EN          NA            NA            NA            NA
10           anonymous           FR          NA            NA            NA            NA
   Filler1_NEG.4 Filler1_NEG.5 Filler1_NEG.6 Filler1_NEG.7 Filler1_NEG.8 Filler1_NEG.9
1             NA             4            NA            NA            NA            NA
2             NA            NA             5            NA            NA            NA
3             NA            NA            NA            NA            NA            NA
4             NA             4            NA            NA            NA            NA
5             NA            NA            NA            NA             4            NA
6             NA            NA            NA            NA            NA             2
7             NA            NA            NA            NA            NA            NA
8             NA            NA            NA            NA            NA            NA
9             NA            NA            NA             1            NA            NA
10            NA            NA            NA            NA            NA             1
   Filler1_NEG.10 Sexe Age Education
1              NA    1  25         4
2              NA    2  25         5
3               4    3  26         4
4              NA    1  36         8
5              NA    3  81         2
6              NA    3  58         2
7              NA    2  28         7
8              NA    3  52         8
9              NA    1  31         8
10             NA    1  47         8
>
```

Pour les nettoyer afin de fusionner toutes les colonnes (qui à l'origine ont toutes le même nom dans le fichier *.csv…) il faut un script non-trivial que voici:

```
> my_bad_data <-unite_(my_bad_data, col="Filler1_NEG_All", from=c("Filler1_NEG",
+   "Filler1_NEG.1","Filler1_NEG.2","Filler1_NEG.3","Filler1_NEG.4","Filler1_NEG.5",
+    "Filler1_NEG.6","Filler1_NEG.7","Filler1_NEG.8", "Filler1_NEG.9", "Filler1_NEG.10"))%>%
+   mutate(Filler1_NEG_All = stringr::str_replace_all(Filler1_NEG_All, 'NA_,?', ''))%>%
+   mutate(Filler1_NEG_All = stringr::str_replace_all(Filler1_NEG_All, '_NA,?', ''))
> my_bad_data
          StartDate          EndDate Duration LocationLatitude LocationLongitude
1  2019-01-02 09:35 2019-01-02 09:36       51         46.99181          6.931000
2  2019-01-02 10:19 2019-01-02 10:20       72         46.99181          6.931000
3  2019-01-03 05:35 2019-01-03 05:43      467         46.99181          6.931000
4  2019-01-03 05:49 2019-01-03 05:51      136         46.99181          6.931000
5  2019-01-03 05:51 2019-01-03 05:53       95         46.99181          6.931000
6  2019-01-03 05:53 2019-01-03 05:54       80         46.99181          6.931000
7  2019-01-03 06:15 2019-01-03 06:17      121         46.99181          6.931000
8  2019-01-03 06:18 2019-01-03 06:19       55         46.99181          6.931000
9  2019-01-03 09:29 2019-01-03 09:39      625         46.51601          6.632797
10 2019-01-07 03:13 2019-01-07 03:24      664         46.99181          6.931000
   DistributionChannel UserLanguage Filler1_NEG_All Sexe Age Education
1              preview           EN               4    1  25         4
2              preview           EN               5    2  25         5
3              preview           EN               4    3  26         4
4              preview           EN               4    1  36         8
5              preview           EN               4    3  81         2
6              preview           EN               2    3  58         2
7              preview           EN               3    2  28         7
8              preview           EN               4    3  52         8
9            anonymous           EN               1    1  31         8
10           anonymous           FR               1    1  47         8
> 
```

Nous pouvons aussi renommer les colonnes avec la fonction **rename( )**:



```
> mytable<-tbl_df(mydata)
> mytable<-separate(mytable,DateDeCommande,c("y","m","d"))
Warning message:
Too many values at 109 locations: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, $
> mytable<-rename(mytable,year=d)
> mytable<-rename(mytable,month=m)
> mytable<-rename(mytable,day=y)
> mytable
# A tibble: 109 × 12
  NClient          Activite NDeCommande   day month  year          Article $
*   <int>            <fctr>       <int> <chr> <chr> <chr>           <fctr> $
1     100       Assurances           1    03    01  2000 Compaq Presario 100 $
2     123 Machines/Outils            2    03    01  2000         IBM 500 $
3     109         Éducation           3    03    01  2000   AST Intel 150 $
4     104         Éducation           4    03    01  2000   AST Intel 200 $
5     117           Banques           5    04    01  2000 Compaq Presario 100 $
6     103       Assurances           6    04    01  2000   AST Intel 150 $
7     104         Éducation           7    04    01  2000   AST Intel 200 $
8     111      Alimentaire           8    04    01  2000         IBM 500 $
9     113     Construction           9    04    01  2000 Compaq Presario 100 $
10    116   Pharmaceutique          10    04    01  2000         IBM 500 $
# ... with 99 more rows, and 1 more variables: FacturePayee <fctr>
>
```

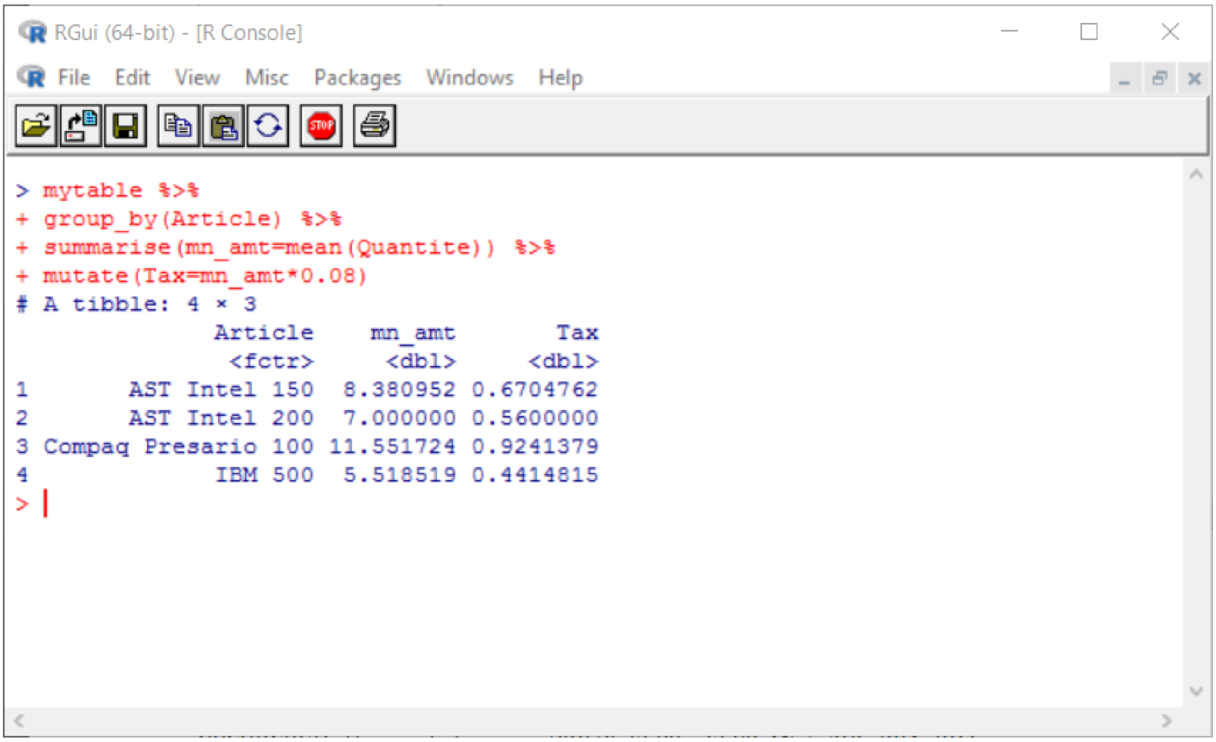
avec la fonction **mutate( )** nous pouvons entre autres, changer le typage (as.numeric( ), as.string( ), factor( )) d'une colonne:

```
> mytable %>% mutate(year=as.numeric(year))
# A tibble: 109 × 12
   NClient        Activite NDeCommande   day month  year              Article
     <int>          <fctr>       <int> <chr> <chr> <dbl>               <fctr>
1      100      Assurances           1    03    01  2000 Compaq Presario 100
2      123 Machines/Outils           2    03    01  2000             IBM 500
3      109       Éducation           3    03    01  2000        AST Intel 150
4      104       Éducation           4    03    01  2000        AST Intel 200
5      117         Banques           5    04    01  2000 Compaq Presario 100
6      103      Assurances           6    04    01  2000        AST Intel 150
7      104       Éducation           7    04    01  2000        AST Intel 200
8      111     Alimentaire           8    04    01  2000             IBM 500
9      113    Construction           9    04    01  2000 Compaq Presario 100
10     116  Pharmaceutique          10    04    01  2000             IBM 500
# ... with 99 more rows, and 5 more variables: Quantite <int>,
#   PrixParPiece <int>, Rabais <fctr>, PrixTotalAvecRabais <dbl>,
#   FacturePayee <fctr>
> 
```

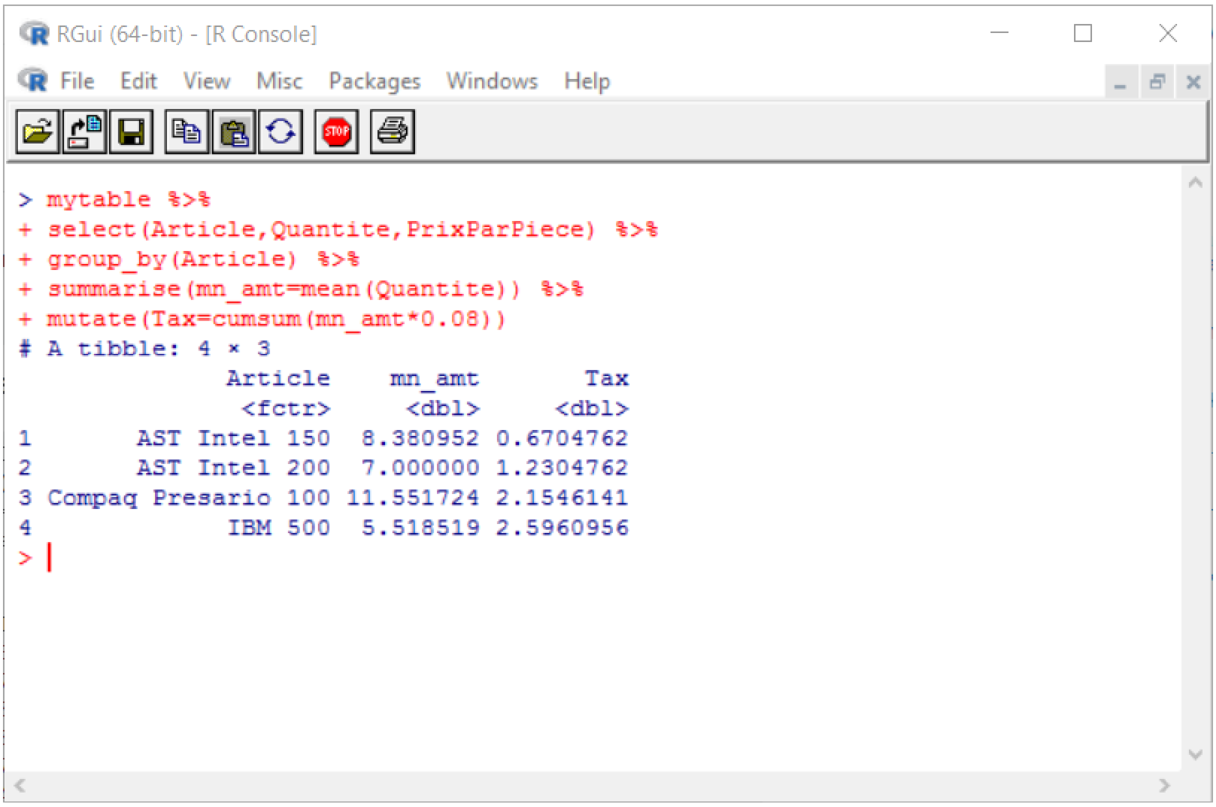
ou encore un exemple de mutate( ) avec les facteurs:

```
> mytable %>% mutate(Article=factor(Article,
+ levels=c("AST Intel 150", "AST Intel 200","IBM 500","Compaq Presario 100"),
+ labels=c("ASTI150","ASTI200","IBM500","CP100")))
# A tibble: 109 x 10
   NClient Activite NDeCommande DateDeCommande Article Quantite PrixParPiece Rabais
     <int> <fct>          <int> <fct>          <fct>      <int>        <int> <fct>
 1     100 Assuran~           1 03.01.2000 10~ CP100         12         1650 1.50%
 2     123 Machine~           2 03.01.2000 07~ IBM500         2         2299 0.00%
 3     109 Éducati~           3 03.01.2000 14~ ASTI150        5         2690 0.00%
 4     104 Éducati~           4 03.01.2000 04~ ASTI200        3         3190 0.00%
 5     117 Banques            5 04.01.2000 16~ CP100         13         1650 1.50%
 6     103 Assuran~           6 04.01.2000 11~ ASTI150        2         2690 0.00%
 7     104 Éducati~           7 04.01.2000 21~ ASTI200        2         3190 0.00%
 8     111 Aliment~           8 04.01.2000 02~ IBM500         4         2299 0.00%
 9     113 Constru~           9 04.01.2000 23~ CP100          4         1650 0.00%
10     116 Pharmac~          10 04.01.2000 16~ IBM500         2         2299 0.00%
# ... with 99 more rows, and 2 more variables: PrixTotalAvecRabais <dbl>,
#   FacturePayee <fct>
> 
```

ou créer de nouvelles variables en mélangeant plusieurs notions vues jusqu'à maintenant:



En associant des fonctions on peut aussi faire par exemple:



ou encore plus subtil avec un **if_else( )** :

```
> mytable %>%
+ select(NDeCommande,FacturePayee)%>%
+ mutate(FacturePayee_New=if_else(FacturePayee=="Oui", 1, 0))
# A tibble: 109 × 3
   NDeCommande FacturePayee FacturePayee_New
         <int>       <fctr>            <dbl>
1            1          Oui                1
2            2          Oui                1
3            3          Oui                1
4            4          Oui                1
5            5          Oui                1
6            6          Oui                1
7            7          Oui                1
8            8          Oui                1
9            9          Oui                1
10          10          Oui                1
# ... with 99 more rows
> 
```


ou si nous avons beaucoup de critères nous pouvons faire usage de **case_when( )** [16]:

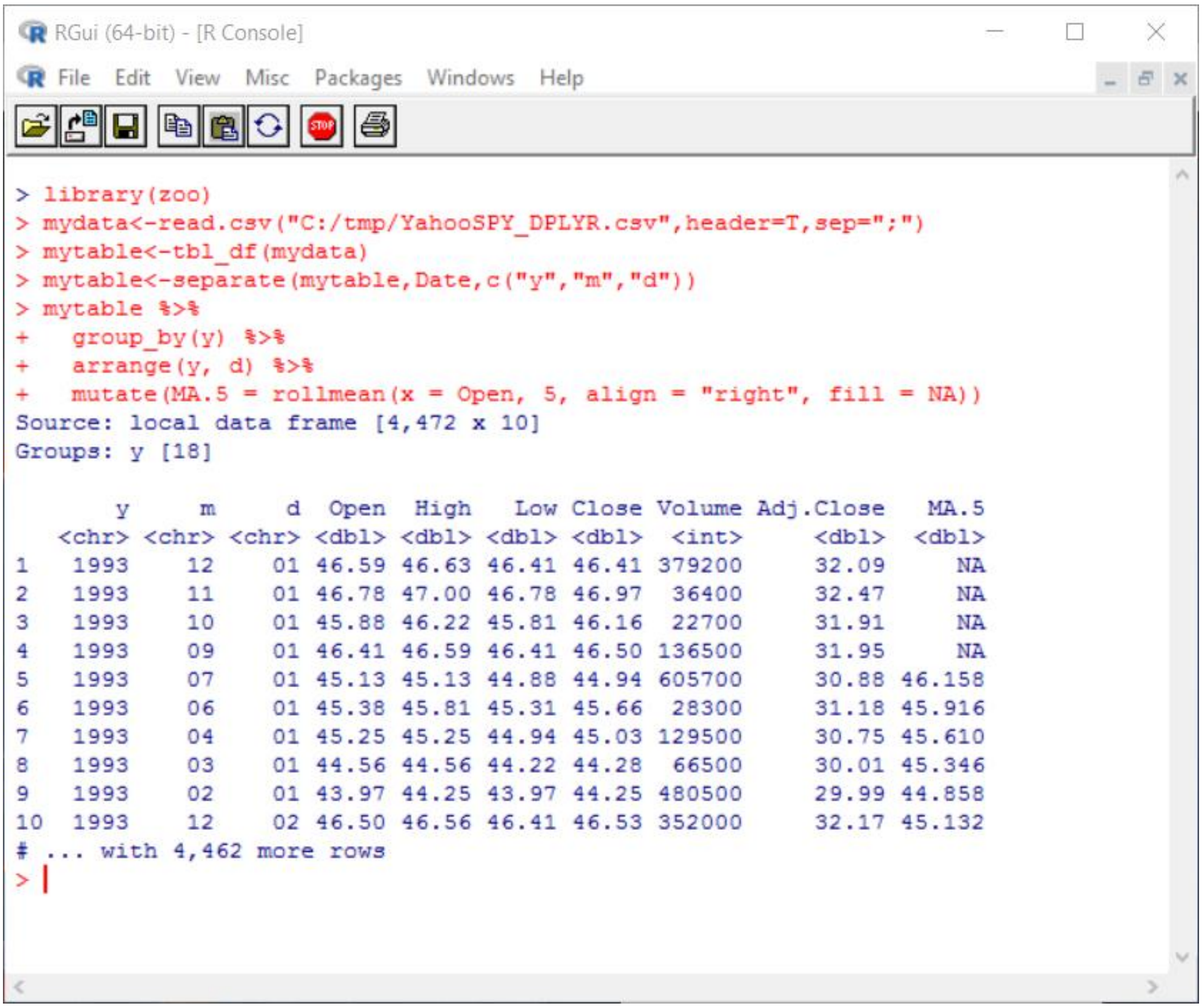
[16] La nécessité de mettre le $ est un bug qui devrait être corrigé un jour futur….

```
> mytable %>%
+ select(NDeCommande,FacturePayee,Rabais)%>%
+ mutate(FacturePayee_New=case_when(.$FacturePayee == "Oui" ~ 1,
+ .$FacturePayee == "Non" ~ 2))
# A tibble: 109 × 4
   NDeCommande FacturePayee Rabais FacturePayee_New
         <int>       <fctr> <fctr>            <dbl>
1            1          Oui  1.50%                1
2            2          Oui  0.00%                1
3            3          Oui  0.00%                1
4            4          Oui  0.00%                1
5            5          Oui  1.50%                1
6            6          Oui  0.00%                1
7            7          Oui  0.00%                1
8            8          Oui  0.00%                1
9            9          Oui  0.00%                1
10          10          Oui  0.00%                1
# ... with 99 more rows
>
```

ou encore plus subtil avec le fichier suivant:


```
"Date";"Open";"High";"Low";"Close";"Volume";"Adj.Close"
"44";2010-10-29;118.28;118.72;118.07;118.49;144305500;110.71
"45";2010-10-28;119.06;119.11;117.83;118.4;168576000;110.62
"46";2010-10-27;117.89;118.51;117.26;118.38;190024000;110.61
"47";2010-10-26;118.1;118.84;117.87;118.72;158982900;110.92
"48";2010-10-25;119.14;119.76;118.61;118.7;151145700;110.9
"49";2010-10-22;118.31;118.53;118;118.35;108212400;110.58
"50";2010-10-21;118.4;119.09;117.21;118.13;221585500;110.37
"51";2010-10-20;116.94;118.44;116.87;117.87;200051800;110.13
"52";2010-10-19;117.19;117.85;116.02;116.73;280604700;109.06
"53";2010-10-18;117.74;118.67;117.31;118.28;141204800;110.51
"54";2010-10-15;118.28;118.35;116.76;117.7;243705000;109.97
"55";2010-10-14;117.81;118.01;116.72;117.46;217764300;109.75
"56";2010-10-13;117.66;118.55;117.38;117.92;194347200;110.18
"57";2010-10-12;116.27;117.35;115.65;117.01;182210000;109.33
"58";2010-10-11;116.72;116.97;116.25;116.65;103098300;108.99
"59";2010-10-08;116.05;116.86;115.61;116.54;177760100;108.89
"60";2010-10-07;116.5;116.53;115.19;115.89;164860000;108.28
"61";2010-10-06;116.02;116.33;115.56;116.03;148626600;108.41
"62";2010-10-05;114.8;116.32;114.67;116.04;229634100;108.42
"63";2010-10-04;114.37;114.85;113.18;113.75;166153200;106.28
```

Faire une moyenne mobile d'ordre 5 avec le package **zoo** et sa fonction **rollmean( )** :

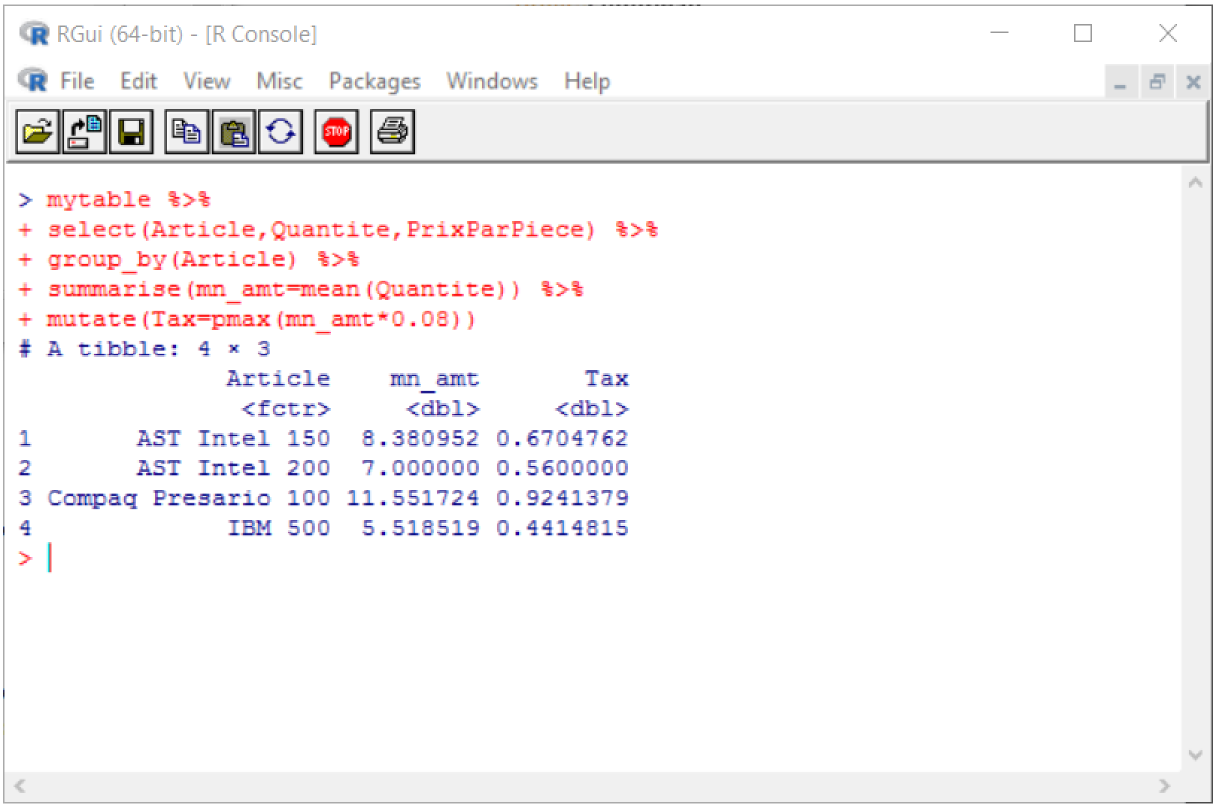


et il existe d'autres fonctions:

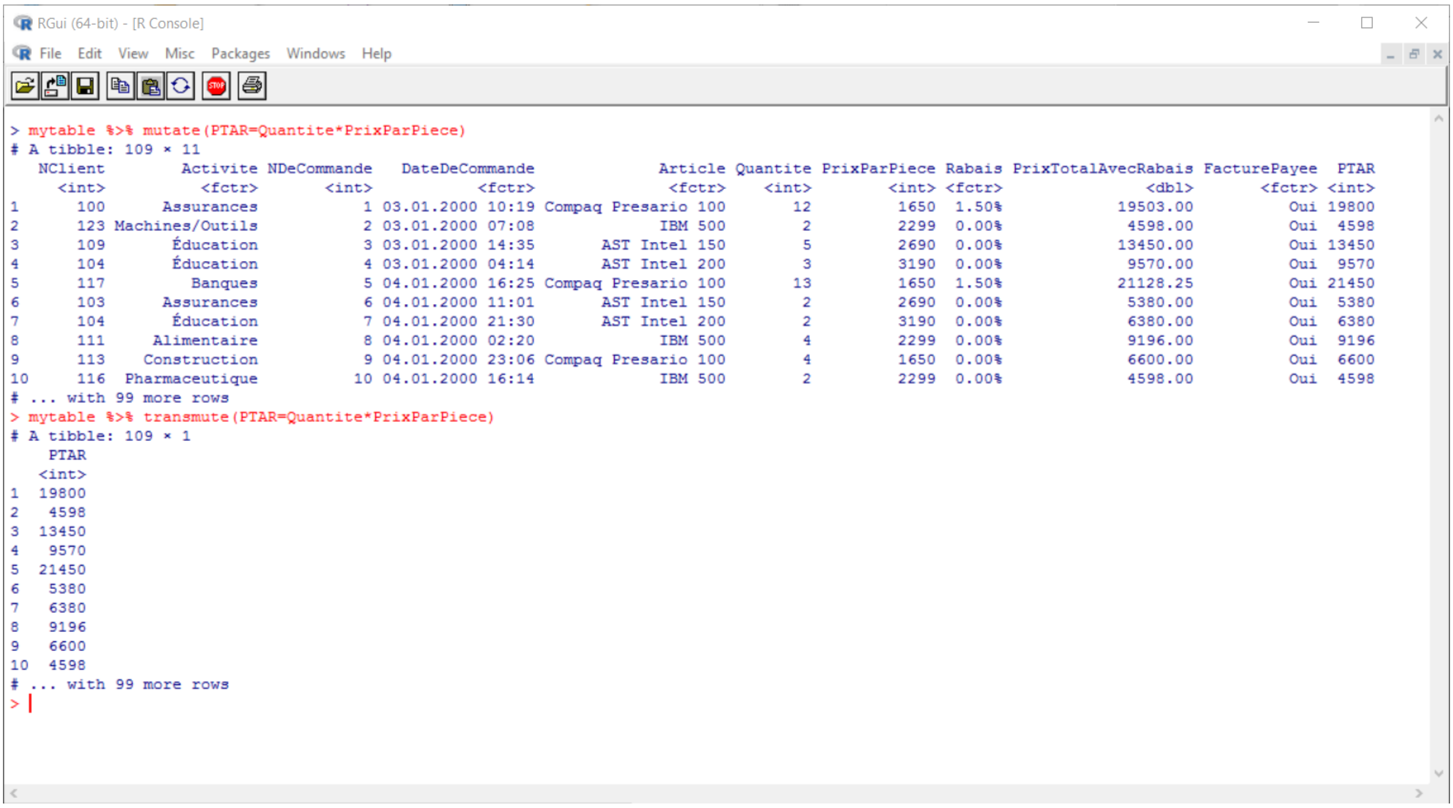
dplyr::lead
Copier avec des valeurs décalées à gauche
dplyr::lag
Copier avec des valeurs décalées à droite
dplyr::dense_rank
Ordonne sans sauts de rangs
dplyr::min_rank
Ordonne avec sauts de rangs
dplyr::percent_rank
Rangs de (**min_rank**) entre [0, 1].
dplyr::row_number
Ordonne en affectant aux liens la première position.
dplyr::ntile
Divise en n groupes.
dplyr::between
Les valeurs sont-elles entre a et b?

dplyr::cume_dist
Distribution cumulée
dplyr::cumall
Cumul tant que vrai
dplyr::cumany
Cumul dès que vrai
dplyr::cummean
Moyenne glissante
cumsum
Somme cumulée
cummax
Maximum cumulé
cummin
Minimum cumulé
cumprod
Produit cumulé
pmax
Maximum par élément
pmin
Minimum par élément

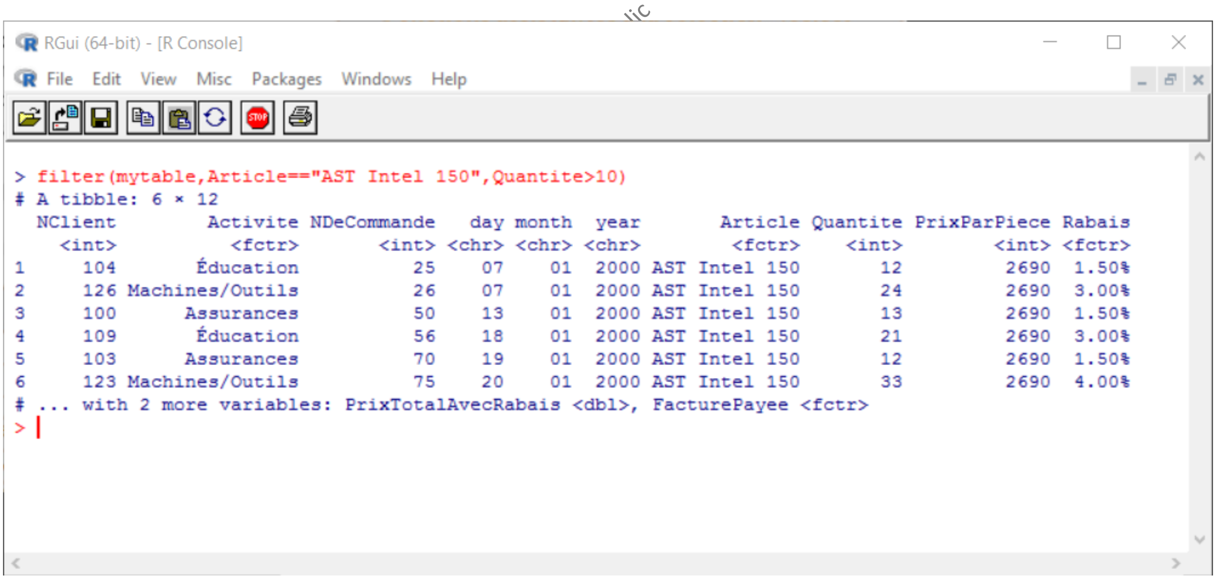
Ce qui montre que nous sommes très proches de ce que font les PowerPivot dans Microsoft Excel:



Avec la fonction **transmute( )** on peut garder que la nouvelle variable comme le montre l'exemple suivant qui la compare avec **mutate( ):**



Pour choisir uniquement certaines lignes, nous utiliserons la fonction **select( )** avec les opérateurs (<,>,==,<=,>=, !=, %in%, is.na, !is.na, &, |, xor, any, all):



Il est aussi possible de faire des sélections automatiques de colonnes du type:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> mytable<-tbl_df(mydata)
> select(mytable,contains("prix"))
# A tibble: 109 × 2
   PrixParPiece PrixTotalAvecRabais
          <int>               <dbl>
1          1650            19503.00
2          2299             4598.00
3          2690            13450.00
4          3190             9570.00
5          1650            21128.25
6          2690             5380.00
7          3190             6380.00
8          2299             9196.00
9          1650             6600.00
10         2299             4598.00
# ... with 99 more rows
> 
```

Bref voici que l'on a venant de la CheatSheet de dplyr:

Fonctions assistantes à la sélection - ?select

`select(iris, contains("."))`
Sélectionne les variables contenant la chaîne de caractères **"."**

`select(iris, ends_with("Length"))`
Sélectionne les variables se terminant par la chaîne de caractères **"Length"**

`select(iris, everything())`
Sélectionne toutes les variables

`select(iris, matches(".t."))`
Sélectionne toutes les variables qui correspondent à l' expression régulière **.t.**

`select(iris, num_range("x", 1:5))`
Sélectionne les variables nommées x1, x2, x3, x4, x5.

`select(iris, one_of(c("Species", "Genus")))`
Sélectionne les variables dans la liste de noms spécifiée

`select(iris, starts_with("Sepal"))`
Sélectionne les variables débutant par la chaîne de caractères **"Sepal"**

`select(iris, Sepal.Length:Petal.Width)`
Sélectionne toutes les variables de Sepal.Length à Petal.Width (incluses).

`select(iris, - Species)`
Sélectionne toutes les variables sauf *Species*.

Maintenant si le lecteur affiche toutes les données avec la fonction **print.data.frame( )** il pourra voir qu'il y a par endroits des valeurs NA:

```
> mydata<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mytable<-tbl_df(mydata)
> print.data.frame(mytable)
   NClient        Activite NDeCommande   DateDeCommande            Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1      100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
2      123 Machines/Outils           2 03.01.2000 07:08            IBM 500        2         2299  0.00%             4598.00          Oui
3      109       Éducation           3 03.01.2000 14:35      AST Intel 150        5         2690  0.00%            13450.00          Oui
4      104       Éducation           4 03.01.2000 04:14      AST Intel 200        3         3190  0.00%             9570.00          Oui
5      117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%            21128.25          Oui
6      103      Assurances           6 04.01.2000 11:01      AST Intel 150        2         2690  0.00%             5380.00          Oui
7      104       Éducation           7 04.01.2000 21:30      AST Intel 200        2         3190  0.00%             6380.00          Oui
8      111     Alimentaire           8 04.01.2000 02:20            IBM 500        4         2299  0.00%             9196.00          Oui
9      113    Construction           9 04.01.2000 23:06 Compaq Presario 100        4         1650  0.00%             6600.00          Oui
10     116  Pharmaceutique          10 04.01.2000 16:14            IBM 500        2         2299  0.00%             4598.00          Oui
11     110    Distribution          11 05.01.2000 14:21      AST Intel 200        6         3190  1.50%            18852.90          Oui
12     112 Machines/Outils          12 05.01.2000 03:29 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
13     123 Machines/Outils          13 05.01.2000 16:22            IBM 500        6         2299  1.50%            13587.09          Oui
14     113    Construction          14 05.01.2000 16:09      AST Intel 150        3         2690  0.00%             8070.00          Oui
15     115    Distribution          15 05.01.2000 01:47 Compaq Presario 100        8         1650  1.50%            13002.00          Oui
16     124       Éducation          16 05.01.2000 21:13      AST Intel 200        8         3190  1.50%            25137.20          Oui
17     124       Éducation          17 05.01.2000 01:48 Compaq Presario 100       11         1650  1.50%            17877.75          Oui
18     106    Construction          18 05.01.2000 14:16      AST Intel 200       11         3190  1.50%            34563.65          Oui
19     101    Construction          19 05.01.2000 17:34 Compaq Presario 100       14         1650  1.50%            22753.50          Non
20     116  Pharmaceutique          20 06.01.2000 23:05            IBM 500        7         2299  1.50%            15851.60          Non
21     112 Machines/Outils          21 06.01.2000 12:40      AST Intel 150        6         2690  1.50%            15897.90          Oui
22     125    Construction          22 06.01.2000 23:18 Compaq Presario 100       23         1650  3.00%            36811.50          Oui
23     100      Assurances          23 06.01.2000 06:48            IBM 500        3         2299  0.00%             6897.00          Oui
24     125    Construction          24 06.01.2000 10:16      AST Intel 200        2         3190  0.00%             6380.00          Oui
25     104       Éducation          25 07.01.2000 15:12      AST Intel 150       12         2690  1.50%            31795.80          Oui
26     126 Machines/Outils          26 07.01.2000 16:10      AST Intel 150       24         2690  3.00%            62623.20          Oui
27     121  Pharmaceutique          27 07.01.2000 21:49            IBM 500        8         2299  1.50%            18116.12          Non
28     114    Distribution          28 07.01.2000 14:28      AST Intel 200        9         3190  1.50%            28279.35          Oui
29     103      Assurances          29 07.01.2000 07:31 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
30     125    Construction          30 07.01.2000 13:42      AST Intel 200        4         3190  0.00%            12760.00          Oui
31     120         Banques          31 10.01.2000 15:59      AST Intel 150        2         2690  0.00%             5380.00          Oui
32     111     Alimentaire          32 10.01.2000 08:57 Compaq Presario 100       16         1650  3.00%            25608.00          Oui
33     118       Éducation          33 10.01.2000 23:32            IBM 500        3         2299  0.00%             6897.00          Oui
34     127     Alimentaire          34 10.01.2000 21:31            IBM 500        7         2299  1.50%            15851.60          Oui
35     101    Construction          35 11.01.2000 06:47      AST Intel 200        6         3190  1.50%            18852.90          Oui
36     118       Éducation          36 11.01.2000 21:30 Compaq Presario 100        5         1650  0.00%             8250.00          Oui
```

Si nous souhaitons les éliminer quelles que soient les colonnes, nous pouvons toujours utiliser **na.omit( )** avec **dplyr**:

```
> mytable<-mytable %>% na.omit()
> print.data.frame(mytable)
   NClient        Activite NDeCommande   DateDeCommande            Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1      100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
2      123 Machines/Outils           2 03.01.2000 07:08            IBM 500        2         2299  0.00%             4598.00          Oui
3      109       Éducation           3 03.01.2000 14:35      AST Intel 150        5         2690  0.00%            13450.00          Oui
4      104       Éducation           4 03.01.2000 04:14      AST Intel 200        3         3190  0.00%             9570.00          Oui
5      117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%            21128.25          Oui
6      103      Assurances           6 04.01.2000 11:01      AST Intel 150        2         2690  0.00%             5380.00          Oui
7      104       Éducation           7 04.01.2000 21:30      AST Intel 200        2         3190  0.00%             6380.00          Oui
8      111     Alimentaire           8 04.01.2000 02:20            IBM 500        4         2299  0.00%             9196.00          Oui
9      113    Construction           9 04.01.2000 23:06 Compaq Presario 100        4         1650  0.00%             6600.00          Oui
10     116  Pharmaceutique          10 04.01.2000 16:14            IBM 500        2         2299  0.00%             4598.00          Oui
11     110    Distribution          11 05.01.2000 14:21      AST Intel 200        6         3190  1.50%            18852.90          Oui
12     112 Machines/Outils          12 05.01.2000 03:29 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
13     123 Machines/Outils          13 05.01.2000 16:22            IBM 500        6         2299  1.50%            13587.09          Oui
14     113    Construction          14 05.01.2000 16:09      AST Intel 150        3         2690  0.00%             8070.00          Oui
15     115    Distribution          15 05.01.2000 01:47 Compaq Presario 100        8         1650  1.50%            13002.00          Oui
16     124       Éducation          16 05.01.2000 21:13      AST Intel 200        8         3190  1.50%            25137.20          Oui
17     124       Éducation          17 05.01.2000 01:48 Compaq Presario 100       11         1650  1.50%            17877.75          Oui
18     106    Construction          18 05.01.2000 14:16      AST Intel 200       11         3190  1.50%            34563.65          Oui
19     101    Construction          19 05.01.2000 17:34 Compaq Presario 100       14         1650  1.50%            22753.50          Non
20     116  Pharmaceutique          20 06.01.2000 23:05            IBM 500        7         2299  1.50%            15851.60          Non
21     112 Machines/Outils          21 06.01.2000 12:40      AST Intel 150        6         2690  1.50%            15897.90          Oui
22     125    Construction          22 06.01.2000 23:18 Compaq Presario 100       23         1650  3.00%            36811.50          Oui
23     100      Assurances          23 06.01.2000 06:48            IBM 500        3         2299  0.00%             6897.00          Oui
24     125    Construction          24 06.01.2000 10:16      AST Intel 200        2         3190  0.00%             6380.00          Oui
25     104       Éducation          25 07.01.2000 15:12      AST Intel 150       12         2690  1.50%            31795.80          Oui
26     126 Machines/Outils          26 07.01.2000 16:10      AST Intel 150       24         2690  3.00%            62623.20          Oui
27     121  Pharmaceutique          27 07.01.2000 21:49            IBM 500        8         2299  1.50%            18116.12          Non
28     114    Distribution          28 07.01.2000 14:28      AST Intel 200        9         3190  1.50%            28279.35          Oui
29     103      Assurances          29 07.01.2000 07:31 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
30     125    Construction          30 07.01.2000 13:42      AST Intel 200        4         3190  0.00%            12760.00          Oui
31     120         Banques          31 10.01.2000 15:59      AST Intel 150        2         2690  0.00%             5380.00          Oui
32     111     Alimentaire          32 10.01.2000 08:57 Compaq Presario 100       16         1650  3.00%            25608.00          Oui
33     118       Éducation          33 10.01.2000 23:32            IBM 500        3         2299  0.00%             6897.00          Oui
34     127     Alimentaire          34 10.01.2000 21:31            IBM 500        7         2299  1.50%            15851.60          Oui
35     101    Construction          35 11.01.2000 06:47      AST Intel 200        6         3190  1.50%            18852.90          Oui
36     118       Éducation          36 11.01.2000 21:30 Compaq Presario 100        5         1650  0.00%             8250.00          Oui
37     119    Distribution          37 11.01.2000 03:25      AST Intel 200       23         3190  3.00%            71168.90          Non
```

ou pour une seule colonne:

```
> mydata<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mytable<-tbl_df(mydata)
> filter(mytable,!is.na(NDeCommande))
# A tibble: 107 × 10
   NClient          Activite NDeCommande    DateDeCommande              Article Quantite
     <int>            <fctr>       <int>            <fctr>               <fctr>    <int>
1      100        Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
2      123 Machines/Outils             2 03.01.2000 07:08              IBM 500        2
3      109         Éducation           3 03.01.2000 14:35        AST Intel 150        5
4      104         Éducation           4 03.01.2000 04:14        AST Intel 200        3
5      117           Banques           5 04.01.2000 16:25 Compaq Presario 100       13
6      103        Assurances           6 04.01.2000 11:01        AST Intel 150        2
7      104         Éducation           7 04.01.2000 21:30        AST Intel 200        2
8      111       Alimentaire           8 04.01.2000 02:20              IBM 500        4
9      113      Construction           9 04.01.2000 23:06 Compaq Presario 100        4
10     116   Pharmaceutique           10 04.01.2000 16:14              IBM 500        2
# ... with 97 more rows, and 4 more variables: PrixParPiece <int>, Rabais <fctr>,
#   PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
> |
```

Et si nous voulons remplacer par une certaine valeur dans une certaine colonne seule, nous avons:

```
> mydata<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mytable<-tbl_df(mydata)
> mytable<-mytable %>%
+ mutate(NDeCommande=ifelse(is.na(NDeCommande),0,NDeCommande))
> print.data.frame(mytable)
   NClient          Activite NDeCommande    DateDeCommande              Article Quantite PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
1      100        Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%            19503.00          Oui
2      123 Machines/Outils             2 03.01.2000 07:08              IBM 500        2         2299  0.00%             4598.00          Oui
3      109         Éducation           3 03.01.2000 14:35        AST Intel 150        5         2690  0.00%            13450.00          Oui
4      104         Éducation           4 03.01.2000 04:14        AST Intel 200        3         3190  0.00%             9570.00          Oui
5      117           Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%            21128.25          Oui
6      103        Assurances           6 04.01.2000 11:01        AST Intel 150        2         2690  0.00%             5380.00          Oui
7      104         Éducation           7 04.01.2000 21:30        AST Intel 200        2         3190  0.00%             6380.00          Oui
8      111       Alimentaire           8 04.01.2000 02:20              IBM 500        4         2299  0.00%             9196.00          Oui
9      113      Construction           9 04.01.2000 23:06 Compaq Presario 100        4         1650  0.00%             6600.00          Oui
10     116   Pharmaceutique           10 04.01.2000 16:14              IBM 500        2         2299  0.00%             4598.00          Oui
11     110      Distribution          11 05.01.2000 14:21        AST Intel 200        6         3190  1.50%            18852.90          Oui
12     112 Machines/Outils            12 05.01.2000 03:29 Compaq Presario 100        6         1650  1.50%             9751.50          Oui
13     123 Machines/Outils            13 05.01.2000 16:22              IBM 500        6         2299  1.50%            13587.09          Oui
14     113      Construction          14 05.01.2000 16:09        AST Intel 150        3         2690  0.00%             8070.00          Oui
15     115      Distribution          15 05.01.2000 01:47 Compaq Presario 100        8         1650  1.50%            13002.00          Oui
16     124         Éducation          16 05.01.2000 21:13        AST Intel 200        8         3190  1.50%            25137.20          Oui
17     124         Éducation          17 05.01.2000 01:48 Compaq Presario 100       11         1650  1.50%            17877.75          Oui
18     106      Construction          18 05.01.2000 14:16        AST Intel 200       11         3190  1.50%            34563.65          Oui
19     101      Construction          19 05.01.2000 17:34 Compaq Presario 100       14         1650  1.50%            22753.50          Non
20     116   Pharmaceutique           20 06.01.2000 23:05              IBM 500        7         2299  1.50%            15851.60          Non
21     112 Machines/Outils            21 06.01.2000 12:40        AST Intel 150        6         2690  1.50%            15897.90          Oui
22     125      Construction          22 06.01.2000 23:18 Compaq Presario 100       23         1650  3.00%            36811.50          Oui
23     100        Assurances          23 06.01.2000 06:48              IBM 500        3         2299  0.00%             6897.00          Oui
24     125      Construction          24 06.01.2000 10:16        AST Intel 200        2         3190  0.00%             6380.00          Oui
25     104         Éducation          25 07.01.2000 15:12        AST Intel 150       12         2690  1.50%            31795.80          Oui
```

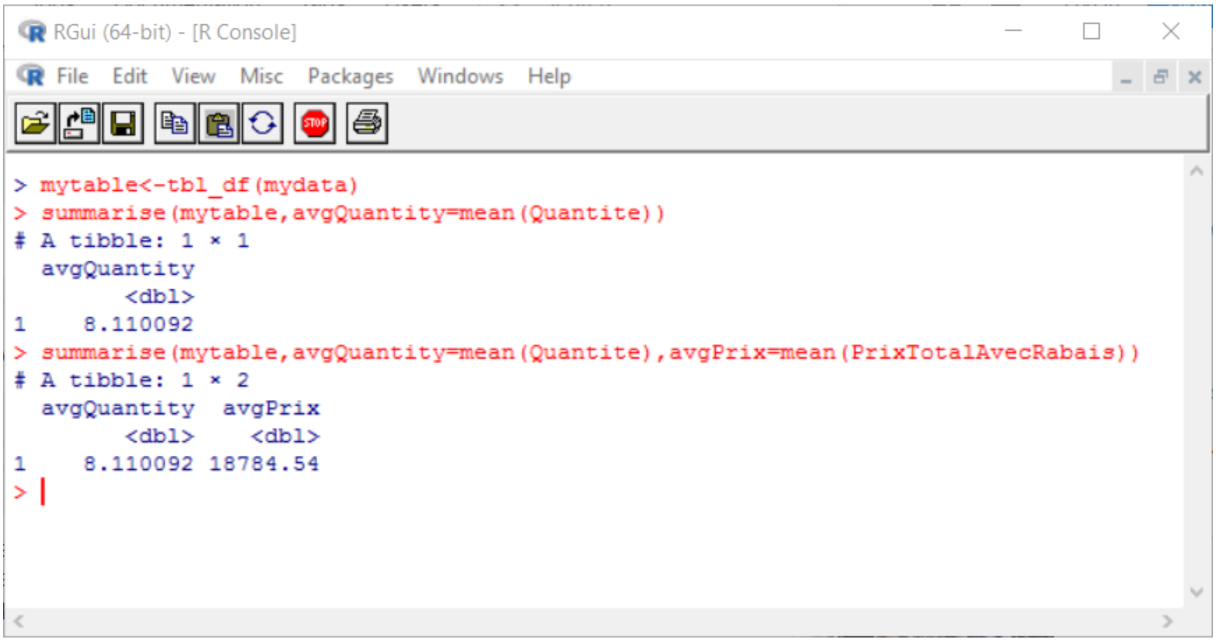
Pour supprimer les doublons avec la fonction **distinct( )**:

```
> distinct(mytable)
# A tibble: 109 × 10
   NClient          Activite NDeCommande   DateDeCommande            Article Quantite
     <int>            <fctr>       <int>           <fctr>             <fctr>    <int>
1      100         Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
2      123 Machines/Outils              2 03.01.2000 07:08            IBM 500        2
3      109         Éducation            3 03.01.2000 14:35      AST Intel 150        5
4      104         Éducation            4 03.01.2000 04:14      AST Intel 200        3
5      117           Banques            5 04.01.2000 16:25 Compaq Presario 100       13
6      103        Assurances            6 04.01.2000 11:01      AST Intel 150        2
7      104         Éducation            7 04.01.2000 21:30      AST Intel 200        2
8      111        Alimentaire           8 04.01.2000 02:20            IBM 500        4
9      113      Construction            9 04.01.2000 23:06 Compaq Presario 100        4
10     116    Pharmaceutique           10 04.01.2000 16:14            IBM 500        2
# ... with 99 more rows, and 4 more variables: PrixParPiece <int>, Rabais <fctr>,
#   PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
>
```

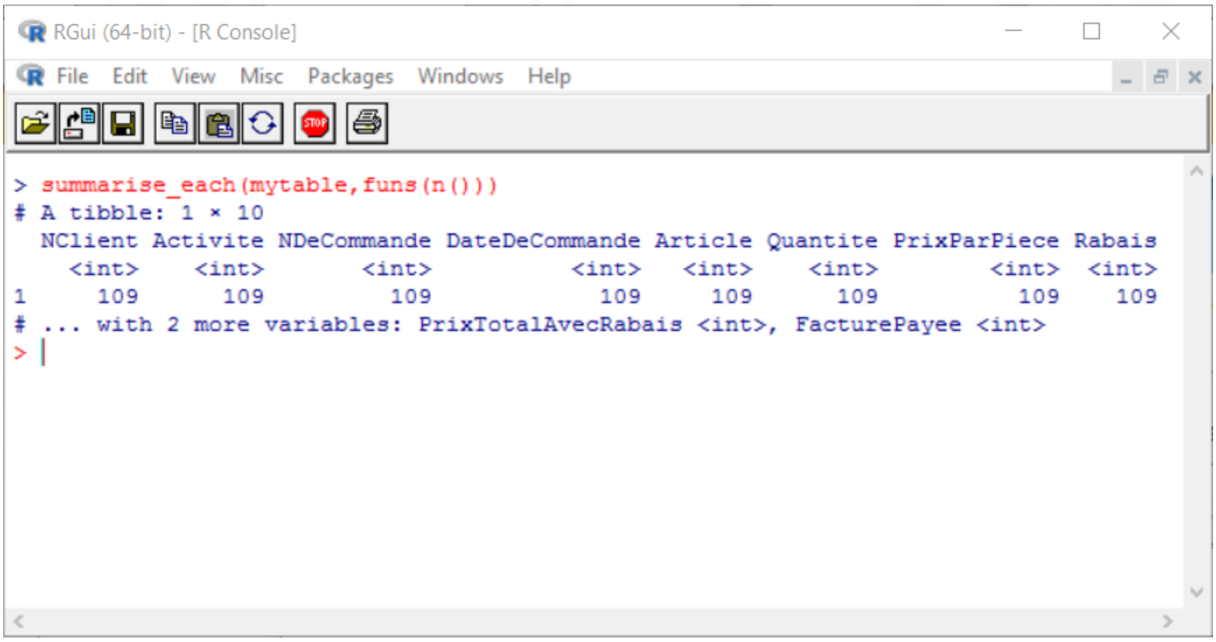
Ou pour des colonnes à choix:

```
> distinct(mytable, NClient, Article)
# A tibble: 74 × 2
   NClient             Article
     <int>              <fctr>
1      100 Compaq Presario 100
2      123             IBM 500
3      109       AST Intel 150
4      104       AST Intel 200
5      117 Compaq Presario 100
6      103       AST Intel 150
7      111             IBM 500
8      113 Compaq Presario 100
9      116             IBM 500
10     110       AST Intel 200
# ... with 64 more rows
>
```

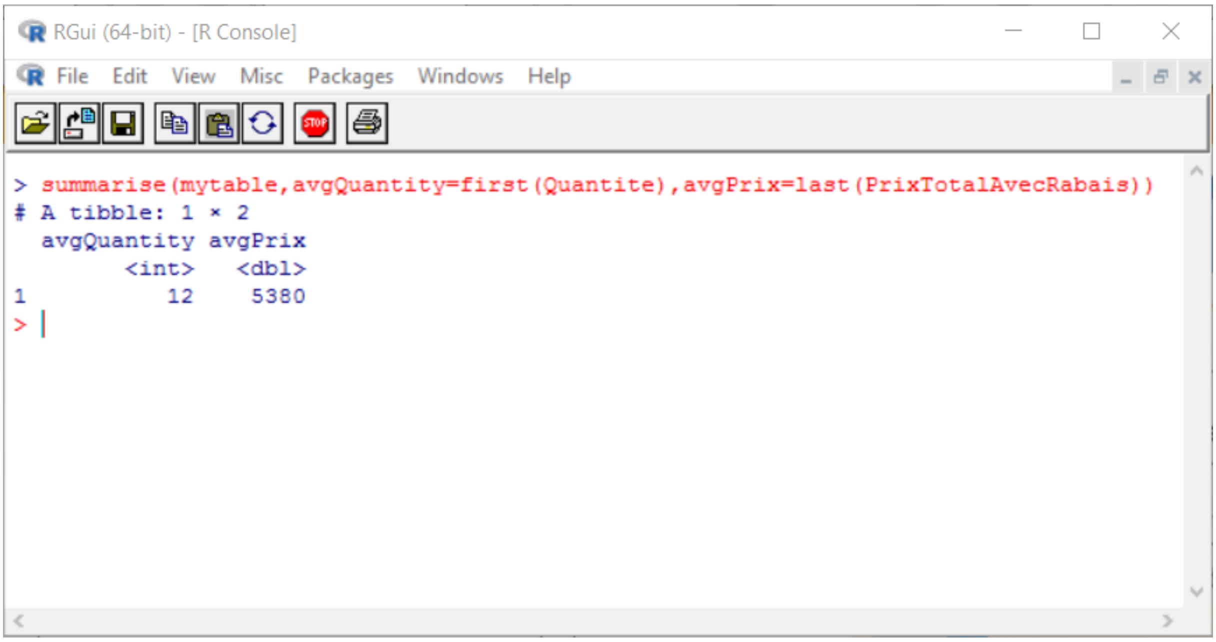
Enfin, nous pouvons avec la fonction **summarise( )** faire des statistiques de base:



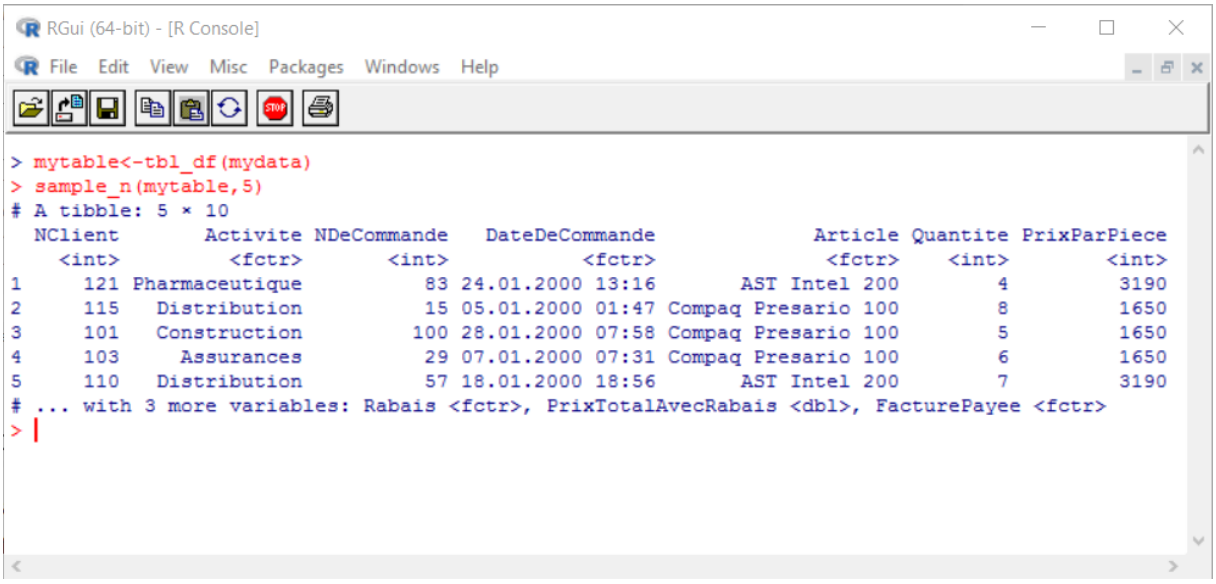
et pour appliquer une fonction identique sur tous les vecteurs on utiliser **summarise_each( )**:



A **summarise( )** nous pouvons passer des fonctions particulières:



Pour ceux qui font des sondages ou du bootstrapping signalons encore la fonction **sample_n( )** pour faire un échantillonnage sans remise:



et pour le tirage avec remise (bootstrap):

```
> sample_n(mytable,5,replace=T)
# A tibble: 5 × 10
  NClient        Activite NDeCommande   DateDeCommande              Article Quantite
    <int>          <fctr>       <int>           <fctr>               <fctr>    <int>
1     127     Alimentaire          93 27.01.2000 23:29        AST Intel 200        8
2     114    Distribution          92 27.01.2000 11:36 Compaq Presario 100        5
3     112 Machines/Outils          88 26.01.2000 07:38        AST Intel 200        2
4     121  Pharmaceutique          83 24.01.2000 13:16        AST Intel 200        4
5     100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
# ... with 4 more variables: PrixParPiece <int>, Rabais <fctr>, PrixTotalAvecRabais <dbl>,
#   FacturePayee <fctr>
>
```

Voyons maintenant un exemple que ceux qui font des tableaux croisés dynamiques dans Microsoft Excel connaissent bien:

```
> library(tidyr)
> mydata<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mytable<-tbl_df(mydata)
> mytable<-separate(mytable,DateDeCommande,c("y","m","d"))
Warning message:
Too many values at 109 locations: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...
> remplacement <- function(y){
+             ls <- unique(y)
+             for(i in ls){
+               specyear <- which(i == y)
+               firstyear <- min(specyear)
+               y[specyear[firstyear!=specyear]]=""
+             }
+             return(y)
+            }
> mytable <- mytable %>%
+            select(m,d,Quantite)%>%
+            group_by(d,m)%>%
+            summarise(Average_value = mean(Quantite))%>%
+            mutate(newyear = remplacement(d))
> as.data.frame(mytable)[,c("newyear","m","Average_value")]
  newyear  m Average_value
1    2000 01      8.120879
2         02      9.800000
3         03      9.000000
4    2001 01      6.428571
5         03      6.000000
6         04      9.000000
>
```

Maintenant voyons un sujet considéré parfois comme le plus important: créer des liaisons entre données (dans la même veine que ce que fait Microsoft Access, PowerPivot, FileMarker, 4D, SQL Server, Oracle, etc.).

Donc nous considérons le même jeu de données qu'avant, mais maintenant nous rajoutons:

```
> mydataMain<-read.csv("C:/tmp/VentesDPLYR.csv",header=T,sep=";")
> mydataSecondary<-read.csv("C:/tmp/BonusDPLYR.csv",header=T,sep=",")
> mydataSecondary
    Status Client  Bonus
1      Bon    100 10.00%
2  Mauvais    101  5.00%
3  Mauvais    102  9.00%
4      Bon    103  4.00%
5      Bon    105  1.00%
6  Mauvais    106  8.00%
7      Bon    107  7.00%
8      Bon    108  1.00%
9  Mauvais    109  8.00%
10 Mauvais    110  9.00%
11 Mauvais    111  1.00%
12     Bon    112 10.00%
13     Bon    114  2.00%
14 Mauvais    115  8.00%
15 Mauvais    116  3.00%
16 Mauvais    117  7.00%
17     Bon    119  9.00%
18     Bon    120  6.00%
19 Mauvais    121  8.00%
20     Bon    122  4.00%
21 Mauvais    123  6.00%
22 Mauvais    124 10.00%
23     Bon    125  7.00%
24     Bon    126  8.00%
25 Mauvais    127  6.00%
26     Bon    128  6.00%
```

et nous utilisons la fonction **left_join( )**:

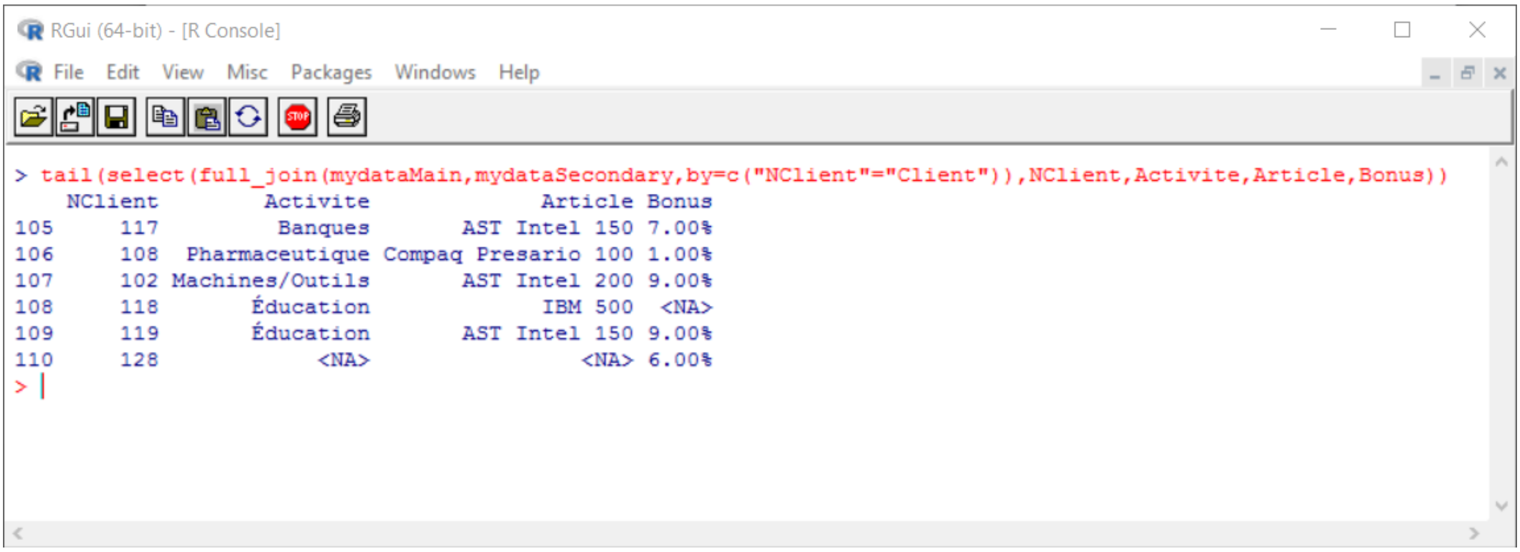
```
> head(select(left_join(mydataMain,mydataSecondary,by=c("NClient"="Client")),NClient,Activite,Article,Bonus))
  NClient         Activite              Article  Bonus
1     100       Assurances Compaq Presario 100 10.00%
2     123 Machines/Outils              IBM 500  6.00%
3     109        Éducation        AST Intel 150  8.00%
4     104        Éducation        AST Intel 200   <NA>
5     117          Banques Compaq Presario 100  7.00%
6     103       Assurances        AST Intel 150  4.00%
> 
```

On peut bien "évidemment" faire un **inner_join( )** [17]:

```
> head(select(inner_join(mydataMain,mydataSecondary,by=c("NClient"="Client")),NClient,Activite,Article,Bonus))
  NClient         Activite              Article  Bonus
1     100       Assurances Compaq Presario 100 10.00%
2     123 Machines/Outils              IBM 500  6.00%
3     109        Éducation        AST Intel 150  8.00%
4     117          Banques Compaq Presario 100  7.00%
5     103       Assurances        AST Intel 150  4.00%
6     111      Alimentaire              IBM 500  1.00%
> 
```

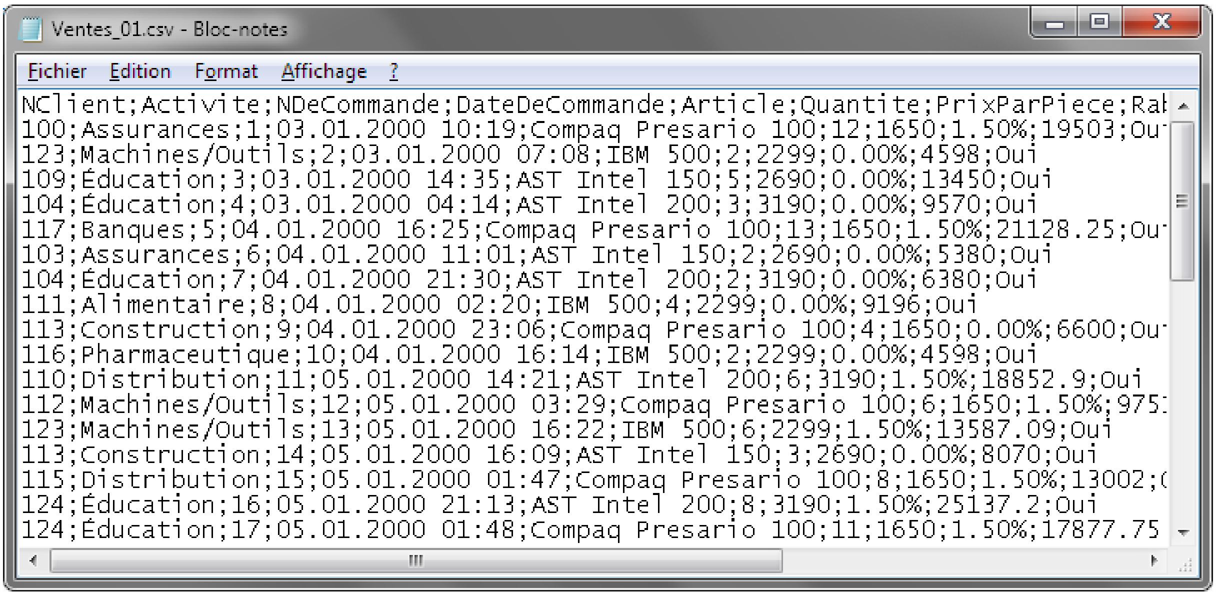
et bien évidemment un **full_join( )** [18]:

[17] Ne rapporte que les lignes pour lesquels il y a des éléments des deux côtés!
[18] Rapport toutes les lignes de deux tables même celles qui n'ont pas forcément de correspondance dans l'une ou dans l'autre!

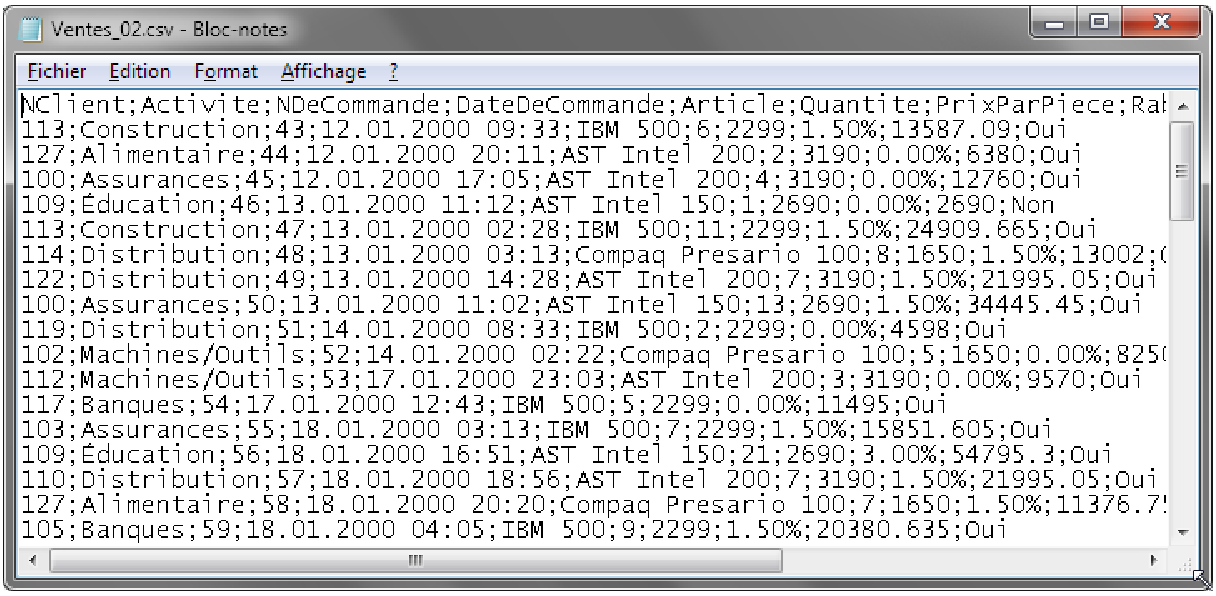


```
> tail(select(full_join(mydataMain,mydataSecondary,by=c("NClient"="Client")),NClient,Activite,Article,Bonus))
    NClient        Activite           Article Bonus
105     117         Banques     AST Intel 150 7.00%
106     108  Pharmaceutique Compaq Presario 100 1.00%
107     102 Machines/Outils     AST Intel 200 9.00%
108     118       Éducation           IBM 500  <NA>
109     119       Éducation     AST Intel 150 9.00%
110     128            <NA>              <NA> 6.00%
>
```

Sinon pour ajouter des données d'un data frame à un autre data frame comme les deux ci-dessous à la suite l'un de l'autre:

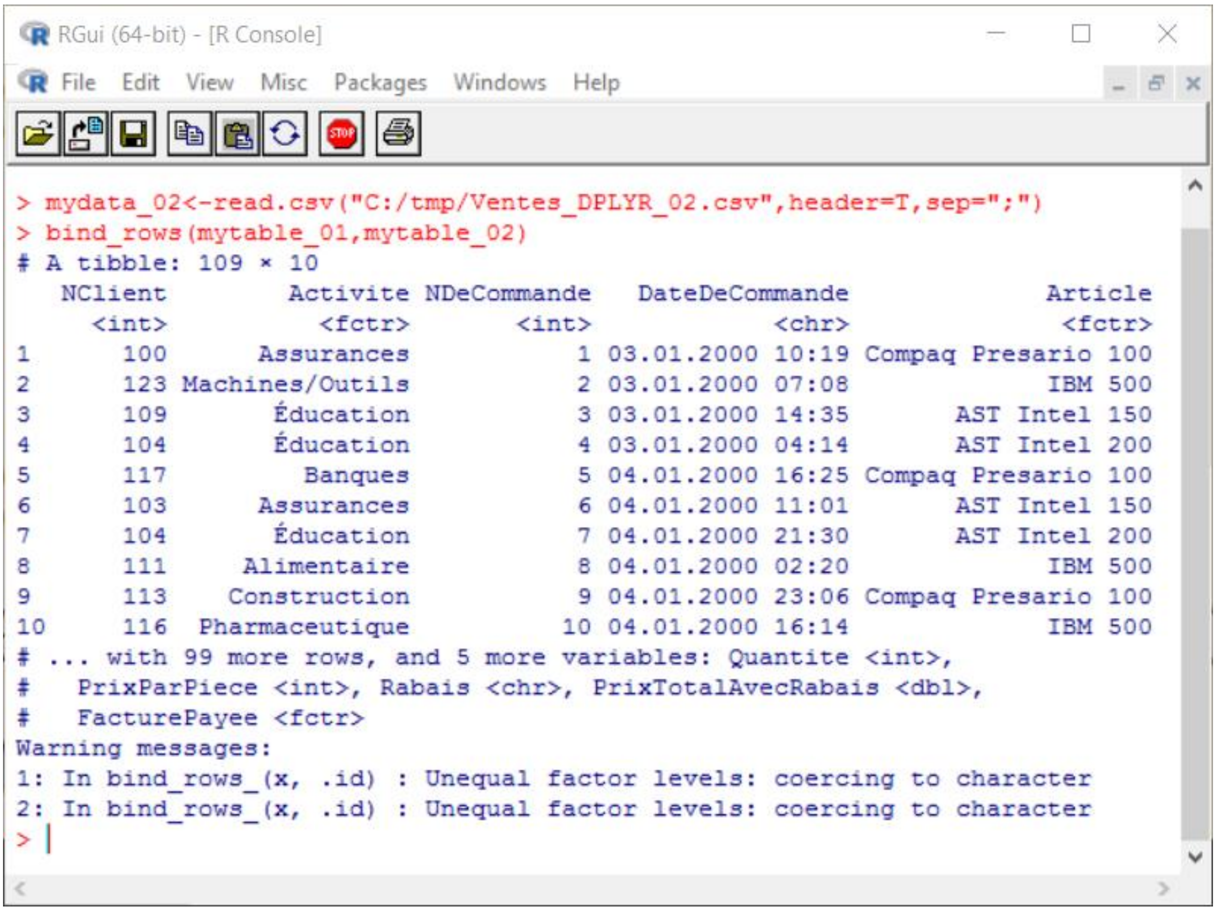


```
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Rab
100;Assurances;1;03.01.2000 10:19;Compaq Presario 100;12;1650;1.50%;19503;Ou
123;Machines/Outils;2;03.01.2000 07:08;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000 14:35;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000 04:14;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000 16:25;Compaq Presario 100;13;1650;1.50%;21128.25;Ou
103;Assurances;6;04.01.2000 11:01;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000 21:30;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000 02:20;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000 23:06;Compaq Presario 100;4;1650;0.00%;6600;Ou
116;Pharmaceutique;10;04.01.2000 16:14;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000 14:21;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000 03:29;Compaq Presario 100;6;1650;1.50%;975
123;Machines/Outils;13;05.01.2000 16:22;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000 16:09;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000 01:47;Compaq Presario 100;8;1650;1.50%;13002;(
124;Éducation;16;05.01.2000 21:13;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000 01:48;Compaq Presario 100;11;1650;1.50%;17877.75
```



```
Ventes_02.csv - Bloc-notes
Fichier Edition Format Affichage ?
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Rab
113;Construction;43;12.01.2000 09:33;IBM 500;6;2299;1.50%;13587.09;Oui
127;Alimentaire;44;12.01.2000 20:11;AST Intel 200;2;3190;0.00%;6380;Oui
100;Assurances;45;12.01.2000 17:05;AST Intel 200;4;3190;0.00%;12760;Oui
109;Éducation;46;13.01.2000 11:12;AST Intel 150;1;2690;0.00%;2690;Non
113;Construction;47;13.01.2000 02:28;IBM 500;11;2299;1.50%;24909.665;Oui
114;Distribution;48;13.01.2000 03:13;Compaq Presario 100;8;1650;1.50%;13002;(
122;Distribution;49;13.01.2000 14:28;AST Intel 200;7;3190;1.50%;21995.05;Oui
100;Assurances;50;13.01.2000 11:02;AST Intel 150;13;2690;1.50%;34445.45;Oui
119;Distribution;51;14.01.2000 08:33;IBM 500;2;2299;0.00%;4598;Oui
102;Machines/Outils;52;14.01.2000 02:22;Compaq Presario 100;5;1650;0.00%;825(
112;Machines/Outils;53;17.01.2000 23:03;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;54;17.01.2000 12:43;IBM 500;5;2299;0.00%;11495;Oui
103;Assurances;55;18.01.2000 03:13;IBM 500;7;2299;1.50%;15851.605;Oui
109;Éducation;56;18.01.2000 16:51;AST Intel 150;21;2690;3.00%;54795.3;Oui
110;Distribution;57;18.01.2000 18:56;AST Intel 200;7;3190;1.50%;21995.05;Oui
127;Alimentaire;58;18.01.2000 20:20;Compaq Presario 100;7;1650;1.50%;11376.7!
105;Banques;59;18.01.2000 04:05;IBM 500;9;2299;1.50%;20380.635;Oui
```

nous utiliserons la fonction **bind_rows( )**:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mydata_02<-read.csv("C:/tmp/Ventes_DPLYR_02.csv",header=T,sep=";")
> bind_rows(mytable_01,mytable_02)
# A tibble: 109 × 10
   NClient        Activite NDeCommande   DateDeCommande             Article
     <int>          <fctr>       <int>            <chr>              <fctr>
1      100      Assurances           1 03.01.2000 10:19 Compaq Presario 100
2      123 Machines/Outils           2 03.01.2000 07:08             IBM 500
3      109       Éducation           3 03.01.2000 14:35       AST Intel 150
4      104       Éducation           4 03.01.2000 04:14       AST Intel 200
5      117         Banques           5 04.01.2000 16:25 Compaq Presario 100
6      103      Assurances           6 04.01.2000 11:01       AST Intel 150
7      104       Éducation           7 04.01.2000 21:30       AST Intel 200
8      111     Alimentaire           8 04.01.2000 02:20             IBM 500
9      113    Construction           9 04.01.2000 23:06 Compaq Presario 100
10     116  Pharmaceutique          10 04.01.2000 16:14             IBM 500
# ... with 99 more rows, and 5 more variables: Quantite <int>,
#   PrixParPiece <int>, Rabais <chr>, PrixTotalAvecRabais <dbl>,
#   FacturePayee <fctr>
Warning messages:
1: In bind_rows_(x, .id) : Unequal factor levels: coercing to character
2: In bind_rows_(x, .id) : Unequal factor levels: coercing to character
> |
```

Et maintenant considérons les deux fichiers suivants:

```
Salaires_Evolution_01.csv - Notepad
id;genre;salaire_1980;salaire_1981;salaire_1982
1;0;5000;5500;6000
2;1;2000;2200;3300
3;0;3000;2000;1000
```

```
Salaires_Evolution_02.csv - Notepad
salaire_1983;salaire_1984;salaire_1985
8000;9500;10000
6000;7200;8300
7000;9000;9500
```

et nous utilisons la fonction **bind_cols( )** pour mettre les données les unes à la suite des autres:

```
> mydata_01<-read.csv("C:/tmp/Salaires_Evolution_01.csv",header=T,sep=";")
> mydata_02<-read.csv("C:/tmp/Salaires_Evolution_02.csv",header=T,sep=";")
> bind_cols(mydata_01,mydata_02)
  id genre salaire_1980 salaire_1981 salaire_1982 salaire_1983 salaire_1984 salaire_1985
1  1     0         5000         5500         6000         8000         9500        10000
2  2     1         2000         2200         3300         6000         7200         8300
3  3     0         3000         2000         1000         7000         9000         9500
> 
```

Pour ce qui est des fonctions suivantes de dplyr:

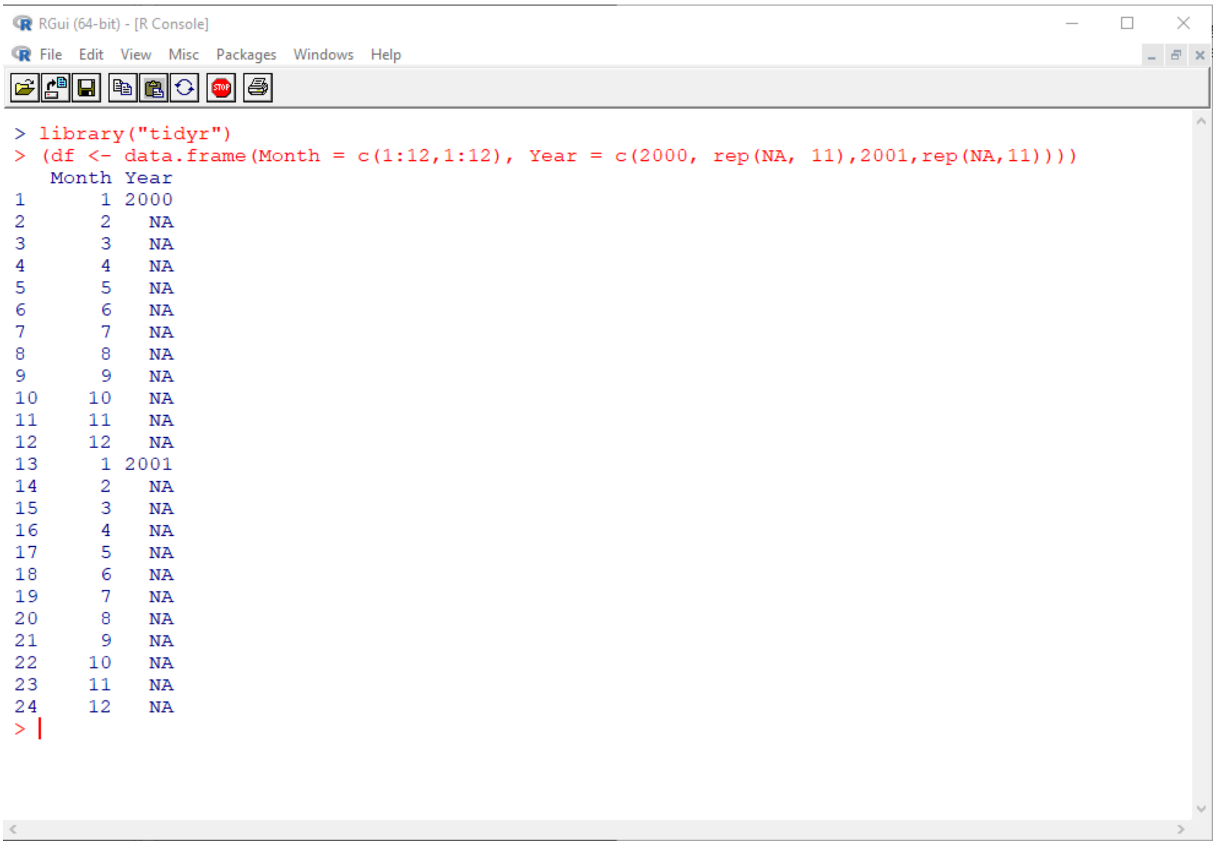
- **union( )** (fusion de deux jeux de données)
- **intersect( )** (éléments communs de deux jeux de données)

- **setdiff( )** (éléments différents de deux jeux de données)

nous ne ferons pas d'exemple ni ne les utiliseront ici car nous les considérerons comme encore non entièrement fonctionnels et satisfaisant à utiliser par rapport à d'autres méthodes!

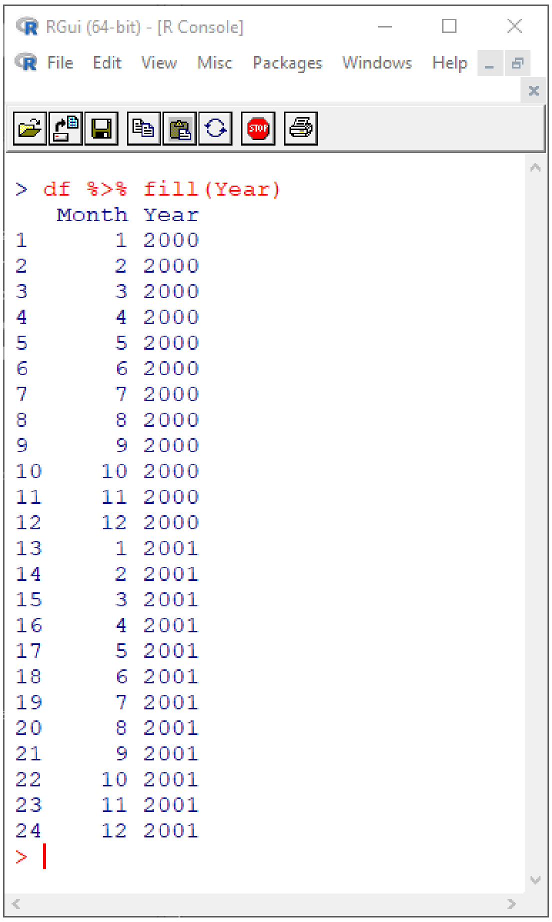
Maintenant voyons la fonction **fill( )** du package tidyr suite à un challenge relativement à une comparaison avec Power Query qui est disponible dans Microsoft Excel ou Microsoft Power BI.

Considérons le jeu de données suivant (on devine évidemment les futurs problèmes):




```
> library("tidyr")
> (df <- data.frame(Month = c(1:12,1:12), Year = c(2000, rep(NA, 11),2001,rep(NA,11))))
   Month Year
1      1 2000
2      2   NA
3      3   NA
4      4   NA
5      5   NA
6      6   NA
7      7   NA
8      8   NA
9      9   NA
10    10   NA
11    11   NA
12    12   NA
13     1 2001
14     2   NA
15     3   NA
16     4   NA
17     5   NA
18     6   NA
19     7   NA
20     8   NA
21     9   NA
22    10   NA
23    11   NA
24    12   NA
> 
```

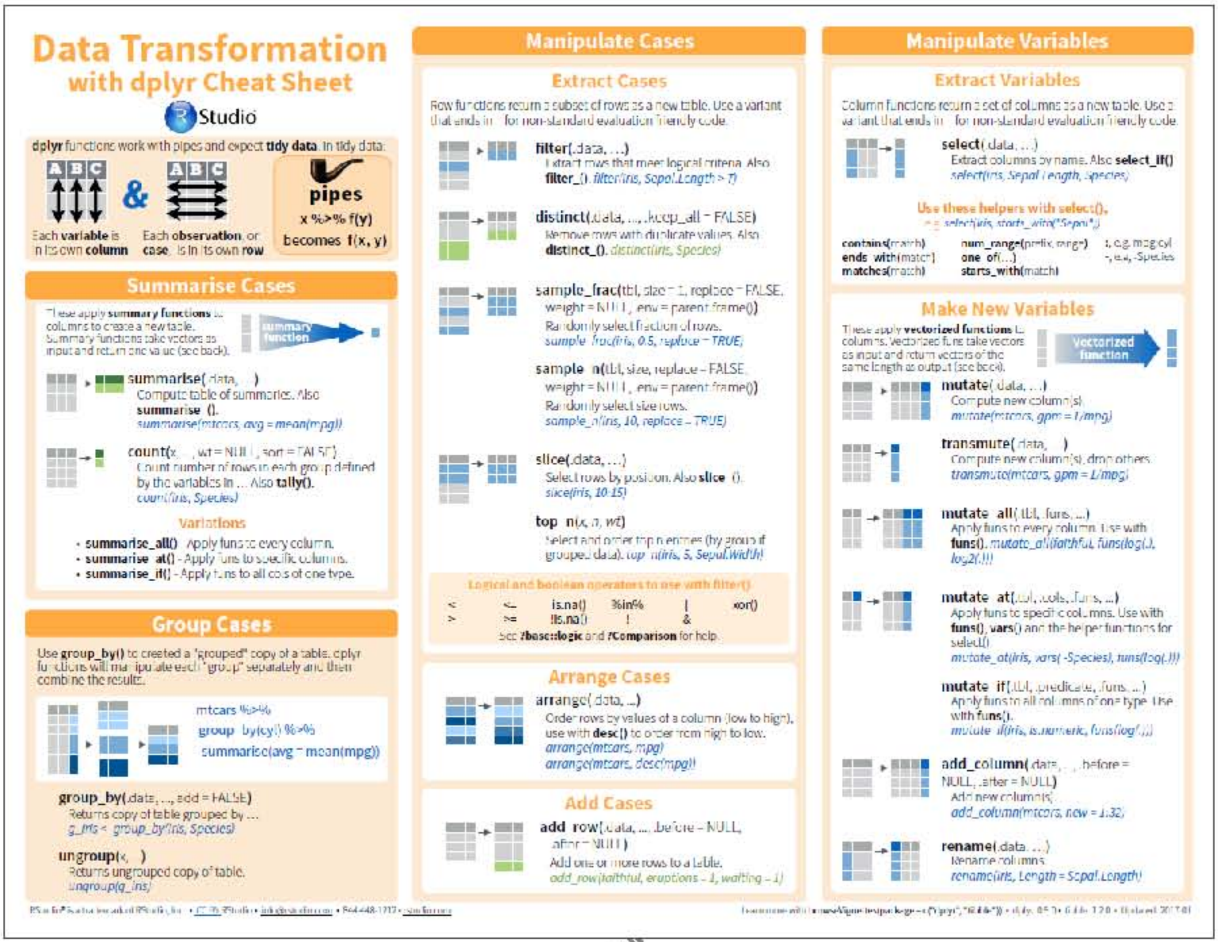
Et donc on applique la fonction fill( ) pour obtenir:



Pour garder tout les colonnes qui commencent par une certaine chaîne de caractères:

```
> (mydata <- data.frame( ABC_1 = runif(5),
+              ABC_2 = runif(5),
+              XYZ_1 = runif(5),
+              XYZ_2 = runif(5)))
      ABC_1     ABC_2     XYZ_1     XYZ_2
1 0.7133016 0.9130878 0.7442772 0.2834595
2 0.1007690 0.9458531 0.9159742 0.2515457
3 0.9503049 0.2791562 0.9945982 0.5032552
4 0.1218178 0.1234711 0.9423607 0.4969662
5 0.2196566 0.7971605 0.4861354 0.3184458
> mydata <- mydata %>% select(starts_with("ABC"))
> mydata
      ABC_1     ABC_2
1 0.7133016 0.9130878
2 0.1007690 0.9458531
3 0.9503049 0.2791562
4 0.1218178 0.1234711
5 0.2196566 0.7971605
> 
```

Enfin, pour ceux qui aiment les Cheatsheet, voici la dernière en date de dplyr:


Data Transformation
with dplyr Cheat Sheet
Studio
dplyr functions work with pipes and expect tidy data. In tidy data:
Each variable is in its own column
Each observation, or case, is in its own row
pipes
x %>% f(y) becomes f(x, y)
Summarise Cases
summarise(.data, ...)
count(x, ..., wt = NULL, sort = FALSE)
Variations
summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.
Group Cases
group_by(.data, ..., add = FALSE)
ungroup(x, ...)
Manipulate Cases
Extract Cases
filter(.data, ...)
distinct(.data, ..., .keep_all = FALSE)
sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame())
sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame())
slice(.data, ...)
top_n(x, n, wt)
Logical and boolean operators to use with filter()
Arrange Cases
arrange(.data, ...)
Add Cases
add_row(.data, ..., .before = NULL, .after = NULL)
Manipulate Variables
Extract Variables
select(.data, ...)
Use these helpers with select()
contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(...)
starts_with(match)
Make New Variables
mutate(.data, ...)
transmute(.data, ...)
mutate_all(.tbl, .funs, ...)
mutate_at(.tbl, .cols, .funs, ...)
mutate_if(.tbl, .predicate, .funs, ...)
add_column(.data, ..., .before = NULL, .after = NULL)
rename(.data, ...)

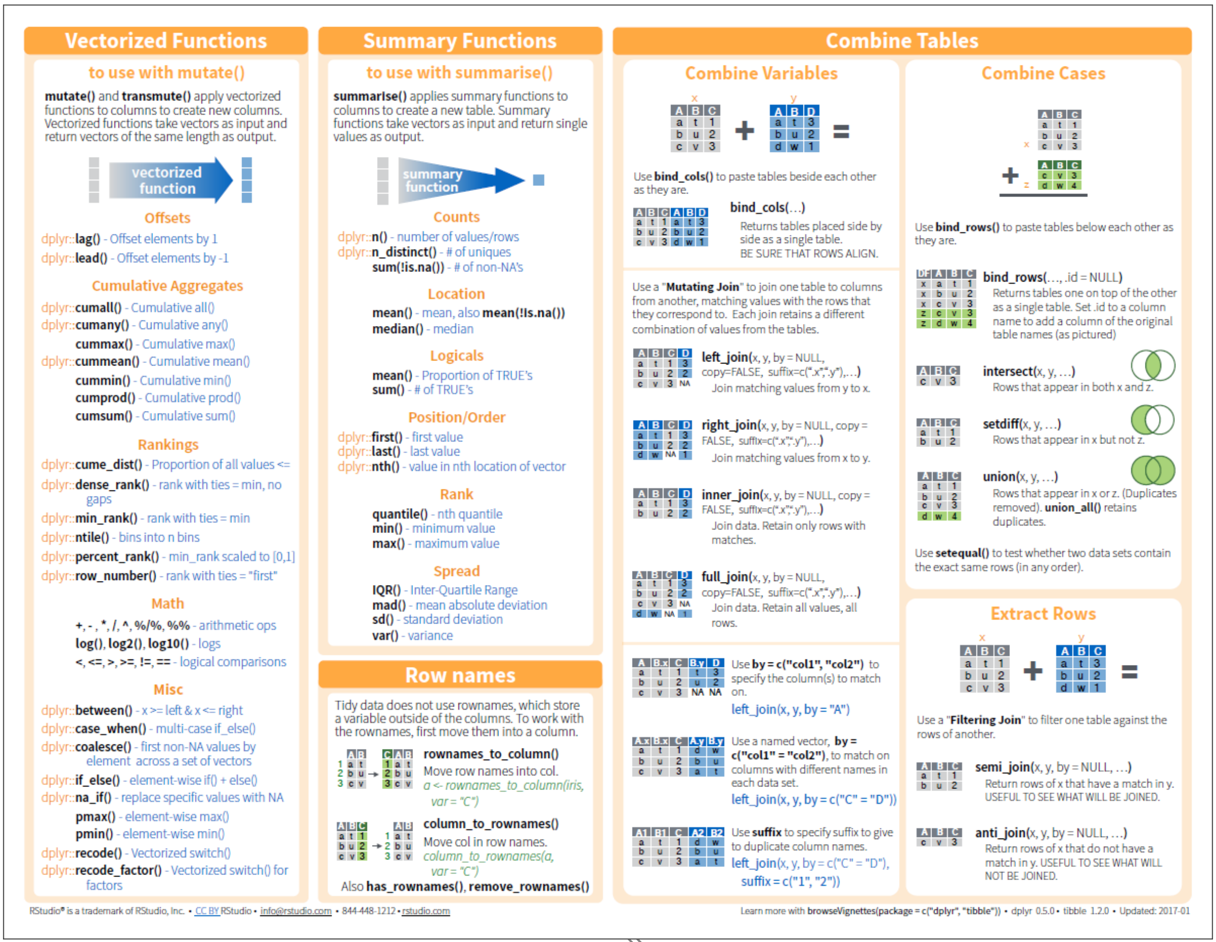
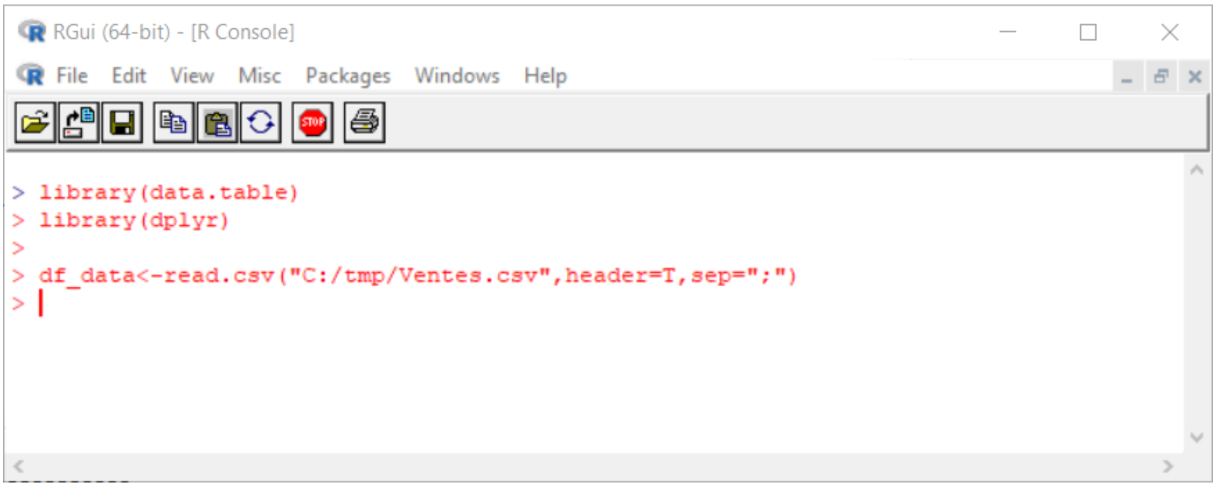


Figure 33 Cheatsheet dplyr

Package data.table

Voyons maintenant quelques classiques **de Data.Table** avec à chaque fois son équivalent en **Dplyr** (cela ne fait pas de mal de répéter ce dernier).

D'abord, nous chargeons les éléments nécessaires:

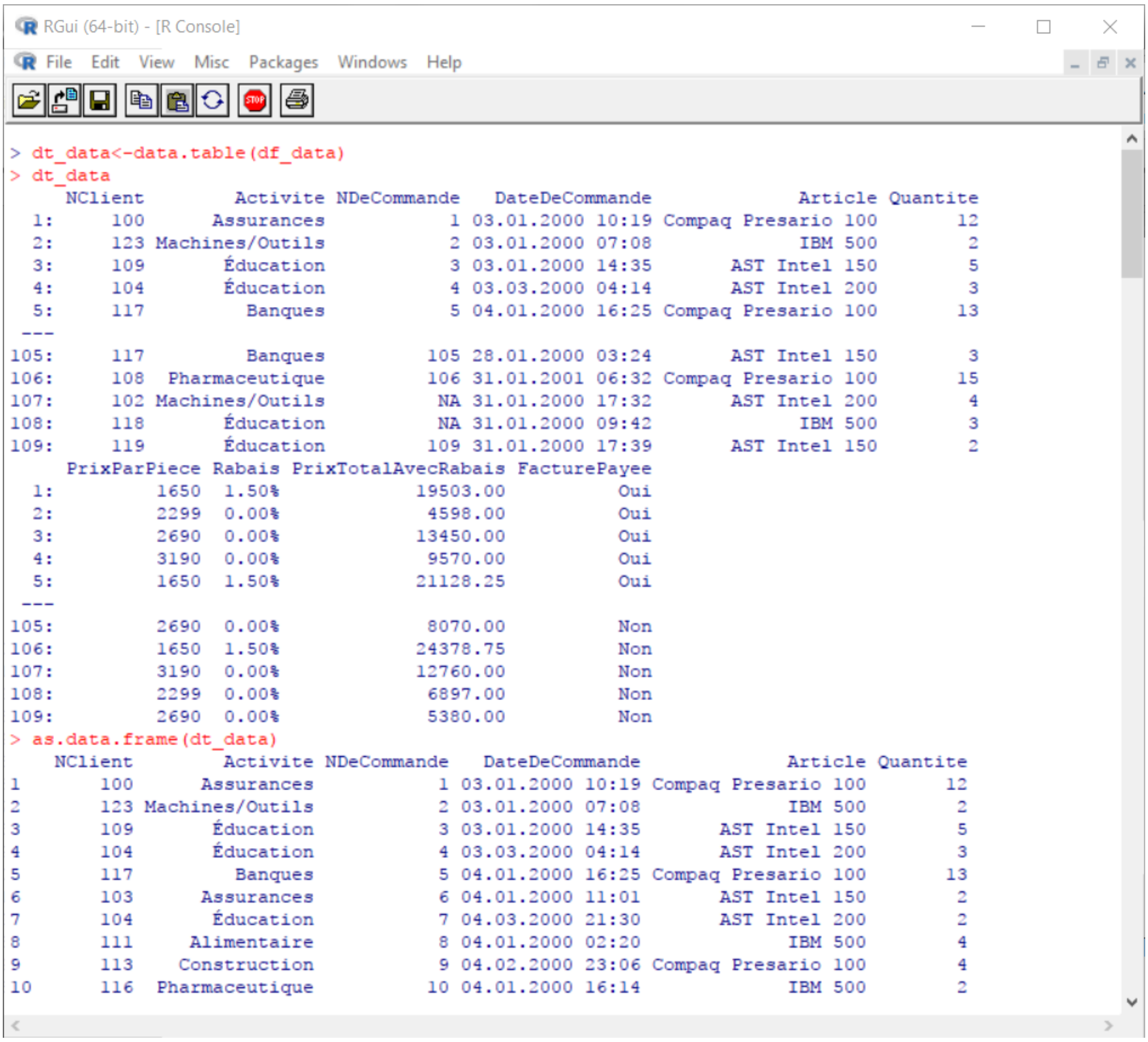


Remarque: Pour importer de gros jeux de données, nous travaillerons plutôt avec la fonction **fread( )** de **data.table** (voir page 2313) plutôt que **read.csv( )** du package base!

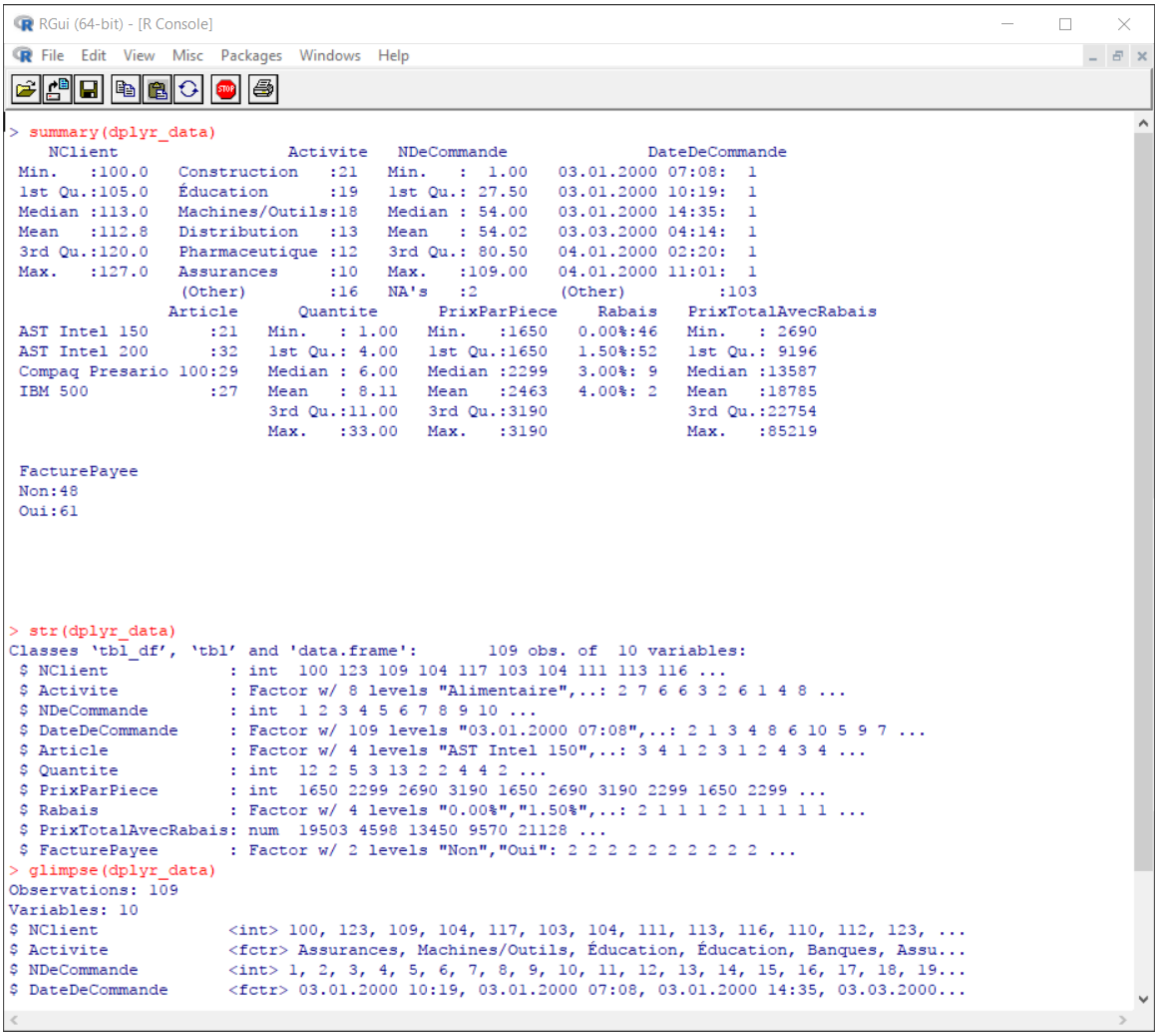
Voici d'abord comment nous transformons un dataframe en objet dplyr (rappel):

```
> dplyr_data<-tbl_df(df_data)
> head(dplyr_data)
# A tibble: 6 x 10
  NClient        Activite NDeCommande   DateDeCommande            Article Quantite
    <int>          <fctr>       <int>           <fctr>             <fctr>    <int>
1     100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
2     123 Machines/Outils           2 03.01.2000 07:08            IBM 500        2
3     109       Éducation           3 03.01.2000 14:35      AST Intel 150        5
4     104       Éducation           4 03.03.2000 04:14      AST Intel 200        3
5     117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13
6     103      Assurances           6 04.01.2000 11:01      AST Intel 150        2
# ... with 4 more variables: PrixParPiece <int>, Rabais <fctr>,
#   PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
> as.data.frame(dplyr_data)
   NClient        Activite NDeCommande   DateDeCommande             Article
1      100      Assurances           1 03.01.2000 10:19 Compaq Presario 100
2      123 Machines/Outils           2 03.01.2000 07:08             IBM 500
3      109       Éducation           3 03.01.2000 14:35       AST Intel 150
4      104       Éducation           4 03.03.2000 04:14       AST Intel 200
5      117         Banques           5 04.01.2000 16:25 Compaq Presario 100
6      103      Assurances           6 04.01.2000 11:01       AST Intel 150
7      104       Éducation           7 04.03.2000 21:30       AST Intel 200
8      111     Alimentaire           8 04.01.2000 02:20             IBM 500
9      113    Construction           9 04.02.2000 23:06 Compaq Presario 100
10     116  Pharmaceutique          10 04.01.2000 16:14             IBM 500
11     110    Distribution          11 05.01.2000 14:21       AST Intel 200
12     112 Machines/Outils          12 05.01.2000 03:29 Compaq Presario 100
13     123 Machines/Outils          13 05.01.2000 16:22             IBM 500
14     113    Construction          14 05.01.2000 16:09       AST Intel 150
15     115    Distribution          15 05.02.2000 01:47 Compaq Presario 100
16     124       Éducation          16 05.03.2000 21:13       AST Intel 200
17     124       Éducation          17 05.01.2000 01:48 Compaq Presario 100
18     106    Construction          18 05.01.2000 14:16       AST Intel 200
19     101    Construction          19 05.01.2000 17:34 Compaq Presario 100
20     116  Pharmaceutique          20 06.02.2000 23:05             IBM 500
21     112 Machines/Outils          21 06.01.2000 12:40       AST Intel 150
22     125    Construction          22 06.03.2000 23:18 Compaq Presario 100
23     100      Assurances          23 06.01.2000 06:48             IBM 500
```

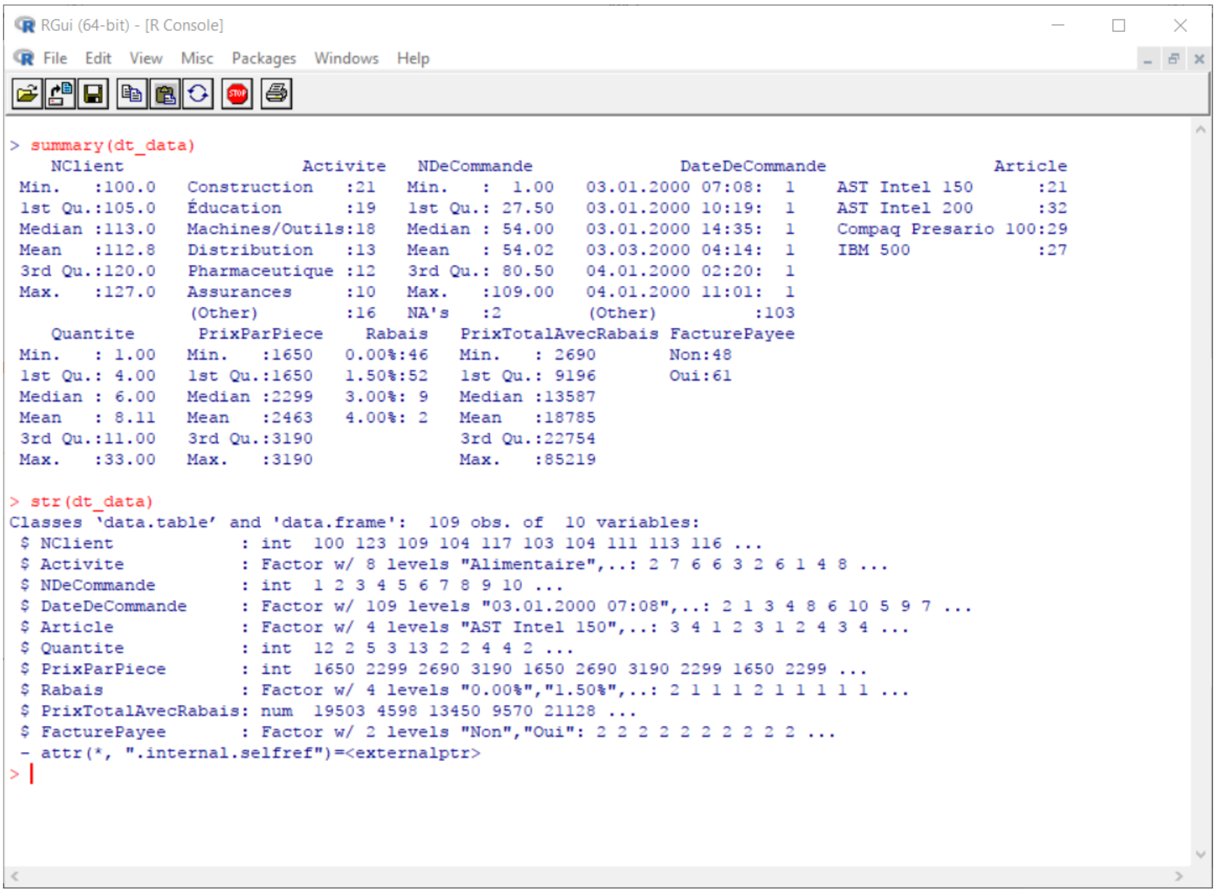
et enfin en objet data.table avec la fonction… **data.table( )** :



Voyons maintenant les différentes façons d'avoir un résumé d'un objet dplyr (rappel):



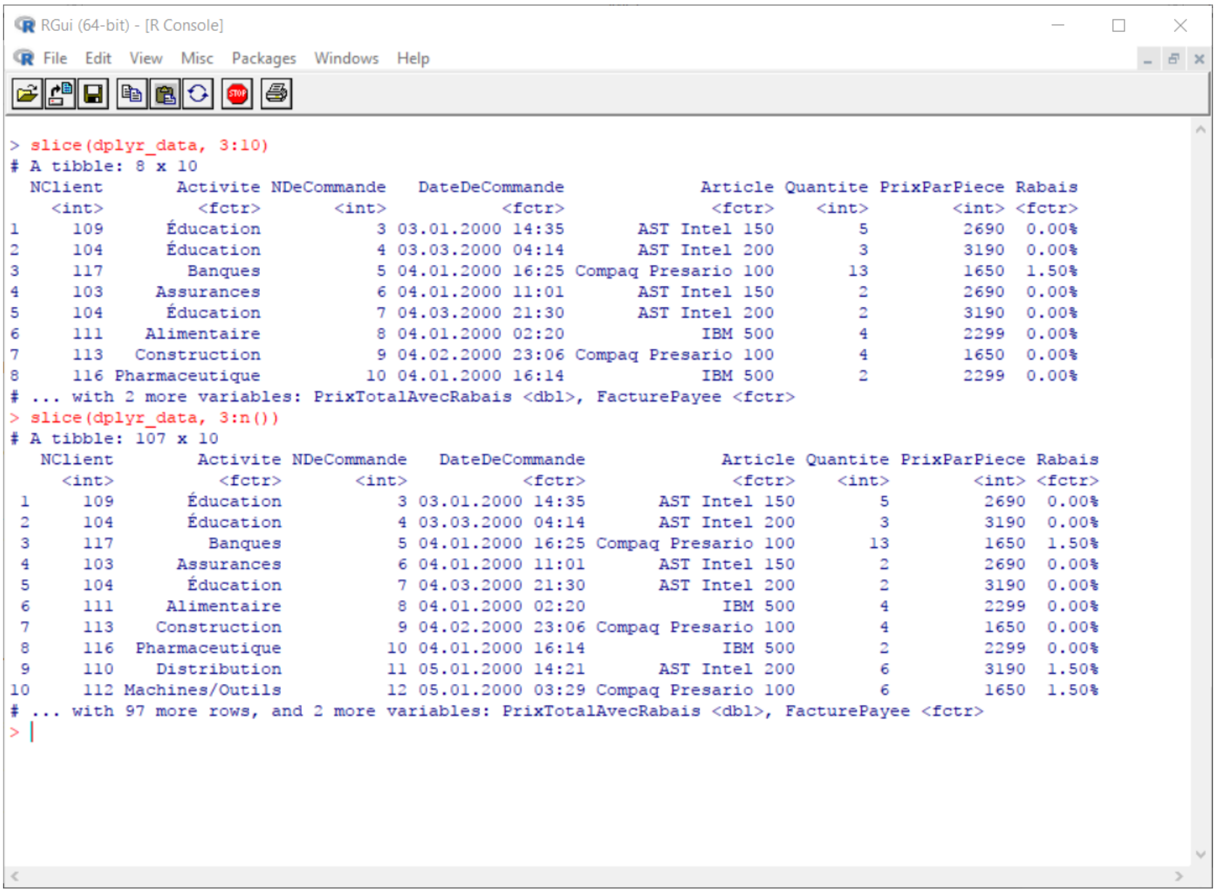
et avec data.table (la fonction **glimpse( )** étant spécifique à dplyr!):



```
> summary(dt_data)
    NClient                   Activite    NDeCommande               DateDeCommande              Article
 Min.   :100.0   Construction    :21   Min.   :  1.00   03.01.2000 07:08:  1   AST Intel 150       :21
 1st Qu.:105.0   Éducation       :19   1st Qu.: 27.50   03.01.2000 10:19:  1   AST Intel 200       :32
 Median :113.0   Machines/Outils:18   Median : 54.00   03.01.2000 14:35:  1   Compaq Presario 100:29
 Mean   :112.8   Distribution    :13   Mean   : 54.02   03.03.2000 04:14:  1   IBM 500             :27
 3rd Qu.:120.0   Pharmaceutique :12   3rd Qu.: 80.50   04.01.2000 02:20:  1
 Max.   :127.0   Assurances      :10   Max.   :109.00   04.01.2000 11:01:  1
                 (Other)         :16   NA's   :2        (Other)         :103
    Quantite       PrixParPiece     Rabais    PrixTotalAvecRabais FacturePayee
 Min.   : 1.00   Min.   :1650   0.00%:46   Min.   : 2690       Non:48
 1st Qu.: 4.00   1st Qu.:1650   1.50%:52   1st Qu.: 9196       Oui:61
 Median : 6.00   Median :2299   3.00%: 9   Median :13587
 Mean   : 8.11   Mean   :2463   4.00%: 2   Mean   :18785
 3rd Qu.:11.00   3rd Qu.:3190              3rd Qu.:22754
 Max.   :33.00   Max.   :3190              Max.   :85219

> str(dt_data)
Classes 'data.table' and 'data.frame':  109 obs. of  10 variables:
 $ NClient            : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Activite           : Factor w/ 8 levels "Alimentaire",..: 2 7 6 6 3 2 6 1 4 8 ...
 $ NDeCommande        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande     : Factor w/ 109 levels "03.01.2000 07:08",..: 2 1 3 4 8 6 10 5 9 7 ...
 $ Article            : Factor w/ 4 levels "AST Intel 150",..: 3 4 1 2 3 1 2 4 3 4 ...
 $ Quantite           : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece       : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais             : Factor w/ 4 levels "0.00%","1.50%",..: 2 1 1 1 2 1 1 1 1 1 ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21128 ...
 $ FacturePayee       : Factor w/ 2 levels "Non","Oui": 2 2 2 2 2 2 2 2 2 2 ...
 - attr(*, ".internal.selfref")=<externalptr>
>
```

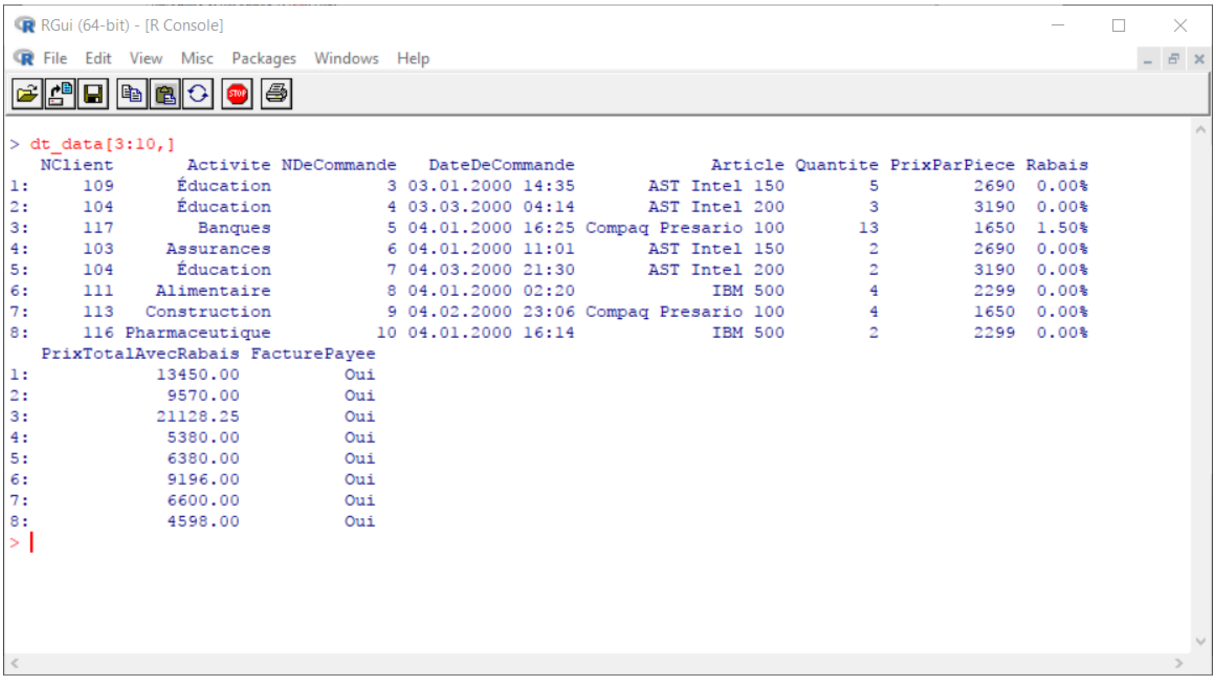
Ou extraire quelques lignes avec dplyr (rappel):



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> slice(dplyr_data, 3:10)
# A tibble: 8 x 10
  NClient        Activite NDeCommande   DateDeCommande              Article Quantite PrixParPiece Rabais
    <int>          <fctr>       <int>           <fctr>               <fctr>    <int>        <int> <fctr>
1     109       Éducation           3 03.01.2000 14:35        AST Intel 150        5         2690  0.00%
2     104       Éducation           4 03.03.2000 04:14        AST Intel 200        3         3190  0.00%
3     117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
4     103      Assurances           6 04.01.2000 11:01        AST Intel 150        2         2690  0.00%
5     104       Éducation           7 04.03.2000 21:30        AST Intel 200        2         3190  0.00%
6     111     Alimentaire           8 04.01.2000 02:20              IBM 500        4         2299  0.00%
7     113    Construction           9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%
8     116 Pharmaceutique           10 04.01.2000 16:14              IBM 500        2         2299  0.00%
# ... with 2 more variables: PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
> slice(dplyr_data, 3:n())
# A tibble: 107 x 10
   NClient        Activite NDeCommande   DateDeCommande              Article Quantite PrixParPiece Rabais
     <int>          <fctr>       <int>           <fctr>               <fctr>    <int>        <int> <fctr>
 1     109       Éducation           3 03.01.2000 14:35        AST Intel 150        5         2690  0.00%
 2     104       Éducation           4 03.03.2000 04:14        AST Intel 200        3         3190  0.00%
 3     117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
 4     103      Assurances           6 04.01.2000 11:01        AST Intel 150        2         2690  0.00%
 5     104       Éducation           7 04.03.2000 21:30        AST Intel 200        2         3190  0.00%
 6     111     Alimentaire           8 04.01.2000 02:20              IBM 500        4         2299  0.00%
 7     113    Construction           9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%
 8     116  Pharmaceutique          10 04.01.2000 16:14              IBM 500        2         2299  0.00%
 9     110    Distribution          11 05.01.2000 14:21        AST Intel 200        6         3190  1.50%
10     112 Machines/Outils          12 05.01.2000 03:29 Compaq Presario 100        6         1650  1.50%
# ... with 97 more rows, and 2 more variables: PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
>
```

et avec data.table:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> dt_data[3:10,]
   NClient       Activite NDeCommande   DateDeCommande              Article Quantite PrixParPiece Rabais
1:     109      Éducation           3 03.01.2000 14:35        AST Intel 150        5         2690  0.00%
2:     104      Éducation           4 03.03.2000 04:14        AST Intel 200        3         3190  0.00%
3:     117        Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
4:     103     Assurances           6 04.01.2000 11:01        AST Intel 150        2         2690  0.00%
5:     104      Éducation           7 04.03.2000 21:30        AST Intel 200        2         3190  0.00%
6:     111    Alimentaire           8 04.01.2000 02:20              IBM 500        4         2299  0.00%
7:     113   Construction           9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%
8:     116 Pharmaceutique          10 04.01.2000 16:14              IBM 500        2         2299  0.00%
   PrixTotalAvecRabais FacturePayee
1:            13450.00          Oui
2:             9570.00          Oui
3:            21128.25          Oui
4:             5380.00          Oui
5:             6380.00          Oui
6:             9196.00          Oui
7:             6600.00          Oui
8:             4598.00          Oui
>
```

On sélectionne ou retire seulement quelques colonnes d'abord avec dplyr (pour rappel):

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

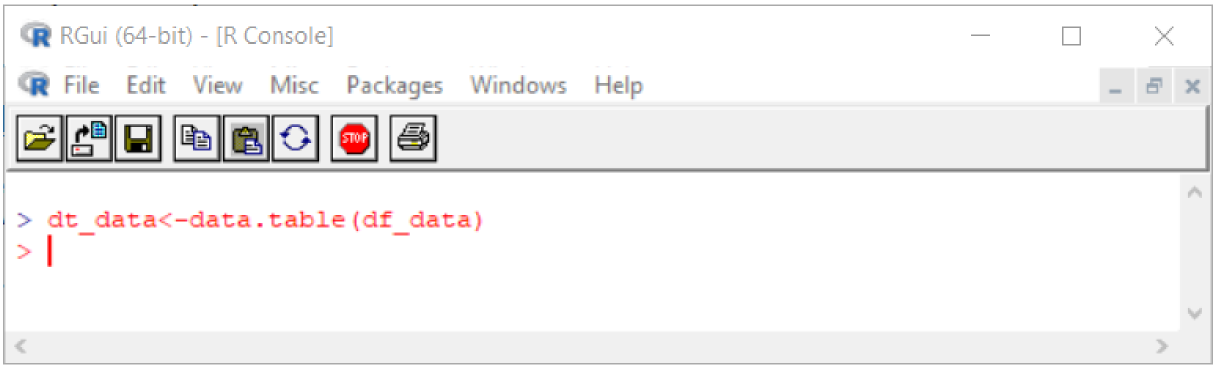
> select(dplyr_data,Activite,NClient)
# A tibble: 109 x 2
          Activite NClient
            <fctr>   <int>
 1      Assurances     100
 2 Machines/Outils     123
 3       Éducation     109
 4       Éducation     104
 5         Banques     117
 6      Assurances     103
 7       Éducation     104
 8     Alimentaire     111
 9    Construction     113
10  Pharmaceutique     116
# ... with 99 more rows
> select(dplyr_data,-NClient,-Activite)
# A tibble: 109 x 8
   NDeCommande    DateDeCommande             Article Quantite PrixParPiece Rabais PrixTotalAvecRabais
         <int>            <fctr>              <fctr>    <int>        <int> <fctr>               <dbl>
 1           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%            19503.00
 2           2 03.01.2000 07:08             IBM 500        2         2299  0.00%             4598.00
 3           3 03.01.2000 14:35       AST Intel 150        5         2690  0.00%            13450.00
 4           4 03.03.2000 04:14       AST Intel 200        3         3190  0.00%             9570.00
 5           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%            21128.25
 6           6 04.01.2000 11:01       AST Intel 150        2         2690  0.00%             5380.00
 7           7 04.03.2000 21:30       AST Intel 200        2         3190  0.00%             6380.00
 8           8 04.01.2000 02:20             IBM 500        4         2299  0.00%             9196.00
 9           9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%             6600.00
10          10 04.01.2000 16:14             IBM 500        2         2299  0.00%             4598.00
# ... with 99 more rows, and 1 more variables: FacturePayee <fctr>
> 
```

ensuite avec data.table:

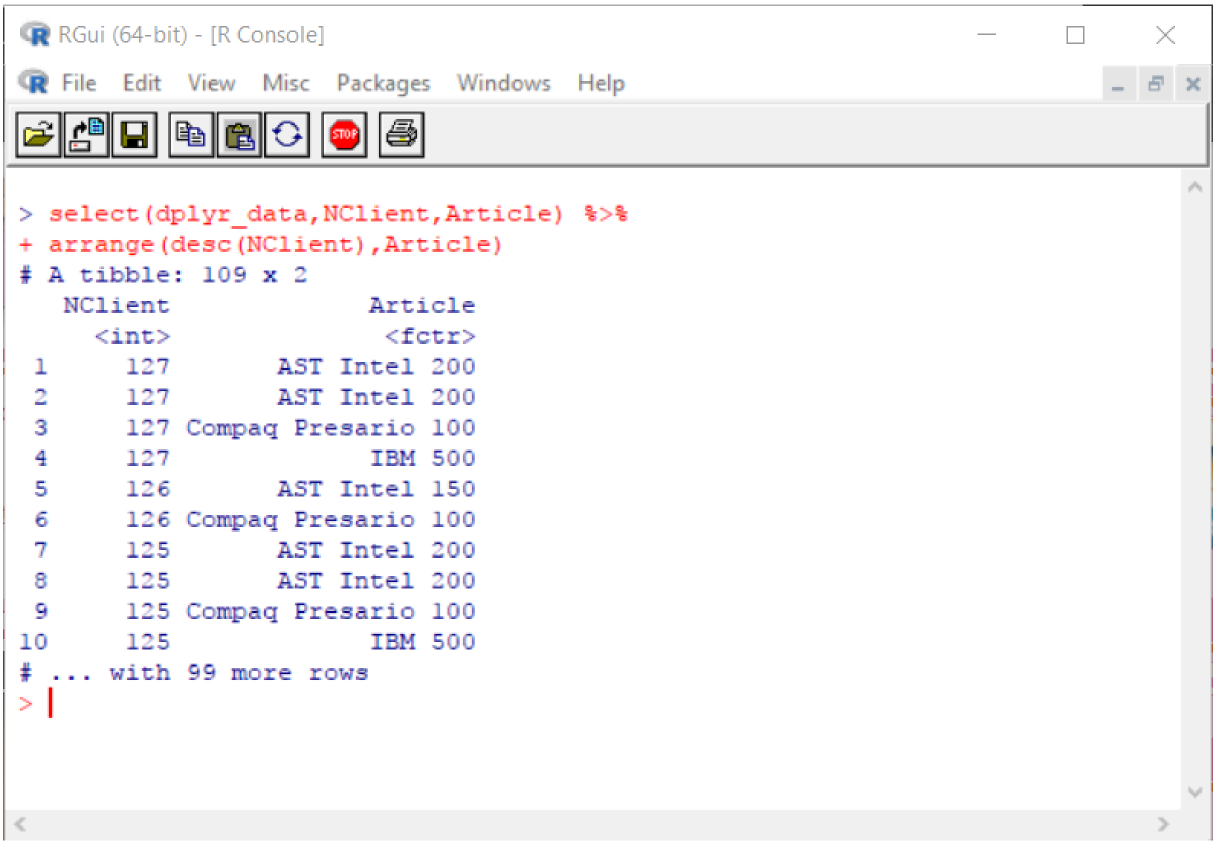
```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> dt_data[,.(Activite,NClient)]
            Activite NClient
  1:      Assurances     100
  2: Machines/Outils     123
  3:       Éducation     109
  4:       Éducation     104
  5:         Banques     117
 ---
105:         Banques     117
106:  Pharmaceutique     108
107: Machines/Outils     102
108:       Éducation     118
109:       Éducation     119
> dt_data[,c("Activite","NClient"):=NULL]
> dt_data
     NDeCommande    DateDeCommande             Article Quantite PrixParPiece Rabais PrixTotalAvecRabais
  1:           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%            19503.00
  2:           2 03.01.2000 07:08             IBM 500        2         2299  0.00%             4598.00
  3:           3 03.01.2000 14:35       AST Intel 150        5         2690  0.00%            13450.00
  4:           4 03.03.2000 04:14       AST Intel 200        3         3190  0.00%             9570.00
  5:           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%            21128.25
 ---
105:         105 28.01.2000 03:24       AST Intel 150        3         2690  0.00%             8070.00
106:         106 31.01.2001 06:32 Compaq Presario 100       15         1650  1.50%            24378.75
107:          NA 31.01.2000 17:32       AST Intel 200        4         3190  0.00%            12760.00
108:          NA 31.01.2000 09:42             IBM 500        3         2299  0.00%             6897.00
109:         109 31.01.2000 17:39       AST Intel 150        2         2690  0.00%             5380.00
     FacturePayee
  1:          Oui
  2:          Oui
  3:          Oui
  4:          Oui
```

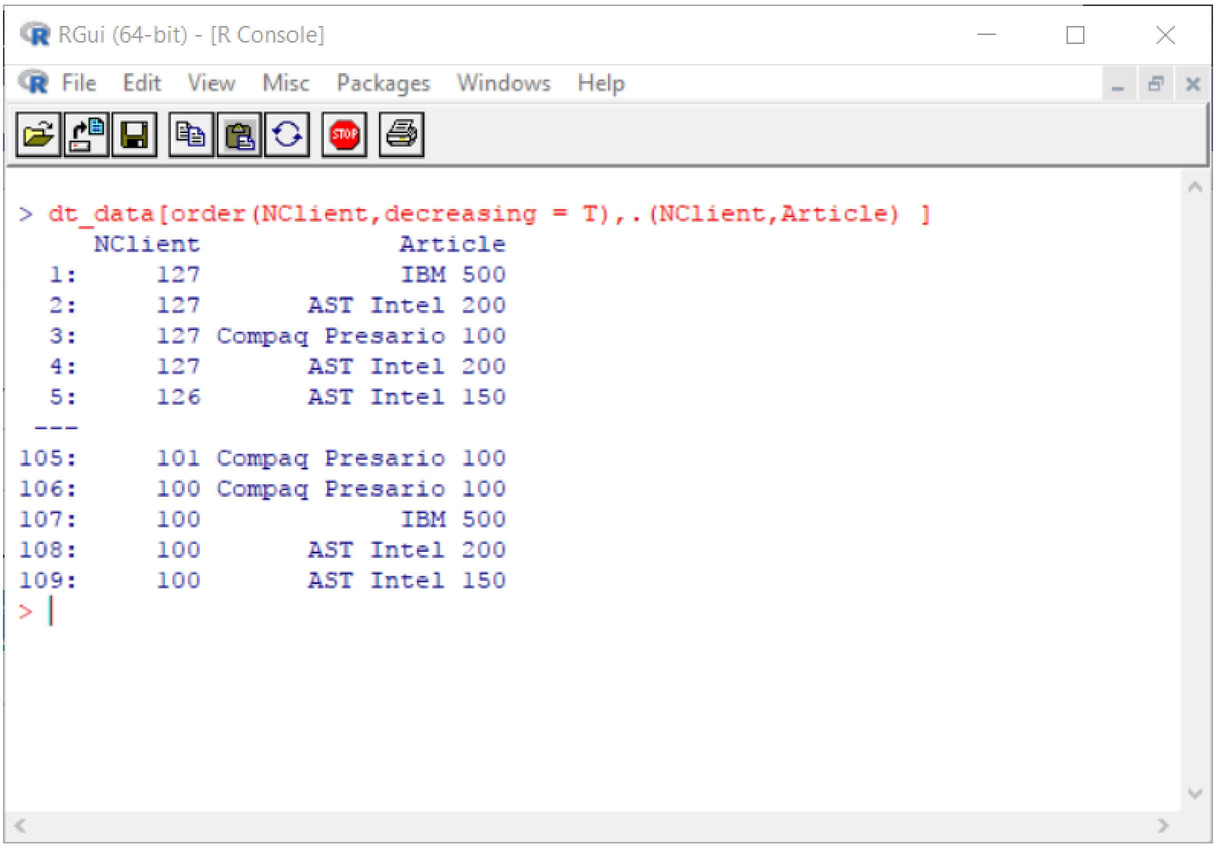
Avant de poursuivre on réinitialise l'objet data.table:



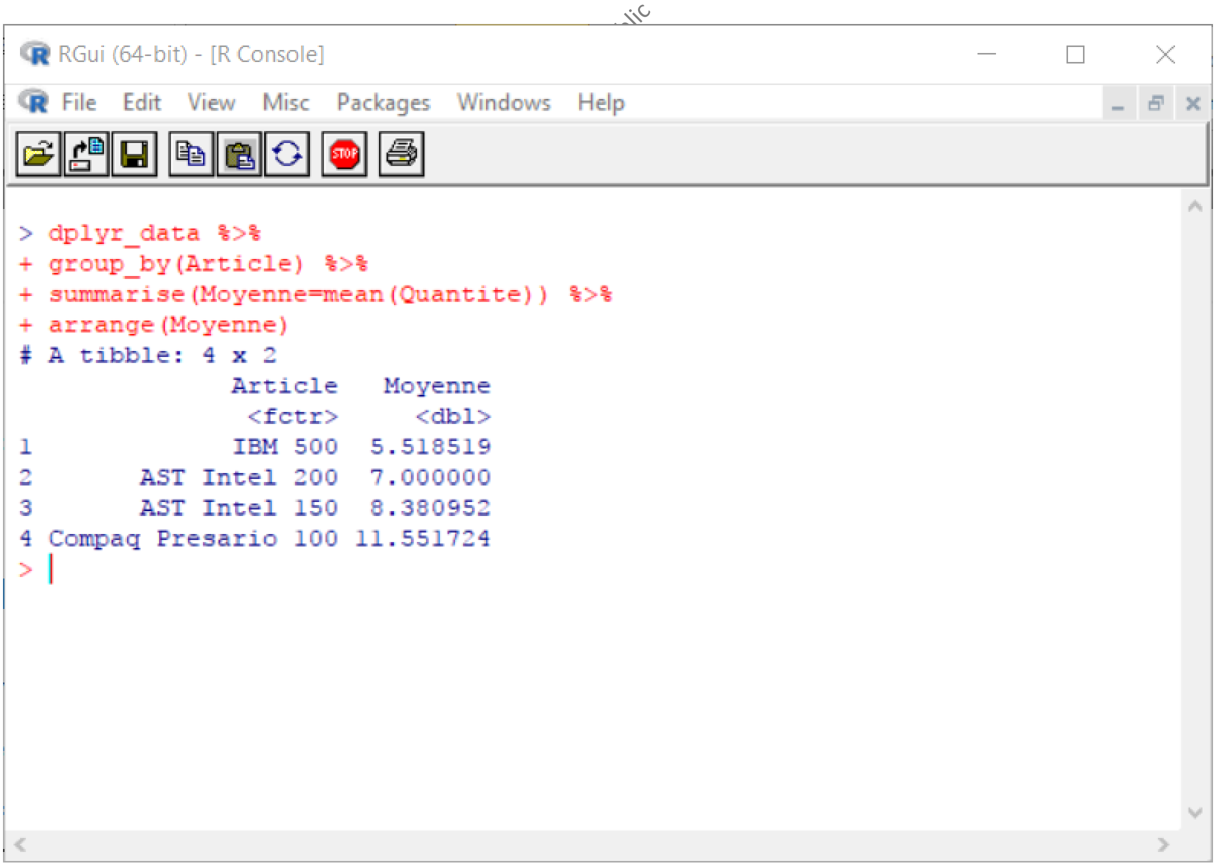
Nous poursuivons maintenant avec un tri simple avec quelques colonnes seulement avec dplyr (pour rappel):



et son équivalent avec data.table:



Maintenant un rappel de groupement en dplyr:



et son équivalent en data.table:

```
> dt_data[,.(Quantite.Moyenne=mean(Quantite)),by=Article]
              Article Quantite.Moyenne
1: Compaq Presario 100        11.551724
2:        IBM 500              5.518519
3:     AST Intel 150           8.380952
4:     AST Intel 200           7.000000
>
```

Pour renommer une colonne avec dplyr (rappel):

```
> rename(dplyr_data, NumeroClient = NClient)
# A tibble: 109 x 10
   NumeroClient        Activite NDeCommande  DateDeCommande             Article Quantite
          <int>          <fctr>       <int>          <fctr>              <fctr>    <int>
 1          100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
 2          123 Machines/Outils           2 03.01.2000 07:08         IBM 500        2
 3          109       Éducation           3 03.01.2000 14:35     AST Intel 150        5
 4          104       Éducation           4 03.03.2000 04:14     AST Intel 200        3
 5          117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13
 6          103      Assurances           6 04.01.2000 11:01     AST Intel 150        2
 7          104       Éducation           7 04.03.2000 21:30     AST Intel 200        2
 8          111     Alimentaire           8 04.01.2000 02:20         IBM 500        4
 9          113    Construction           9 04.02.2000 23:06 Compaq Presario 100        4
10          116  Pharmaceutique          10 04.01.2000 16:14         IBM 500        2
# ... with 99 more rows, and 4 more variables: PrixParPiece <int>, Rabais <fctr>,
#   PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
```

et son équivalent en data.table:

```
> setnames(dt_data, old = "NClient", new = "NumeroClient")
> dt_data
    NumeroClient        Activite NDeCommande   DateDeCommande            Article Quantite
  1:         100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
  2:         123 Machines/Outils           2 03.01.2000 07:08            IBM 500        2
  3:         109       Éducation           3 03.01.2000 14:35      AST Intel 150        5
  4:         104       Éducation           4 03.03.2000 04:14      AST Intel 200        3
  5:         117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13
 ---
105:         117         Banques         105 28.01.2000 03:24      AST Intel 150        3
106:         108  Pharmaceutique         106 31.01.2001 06:32 Compaq Presario 100       15
107:         102 Machines/Outils          NA 31.01.2000 17:32      AST Intel 200        4
108:         118       Éducation          NA 31.01.2000 09:42            IBM 500        3
109:         119       Éducation         109 31.01.2000 17:39      AST Intel 150        2
    PrixParPiece Rabais PrixTotalAvecRabais FacturePayee
  1:        1650  1.50%            19503.00          Oui
```

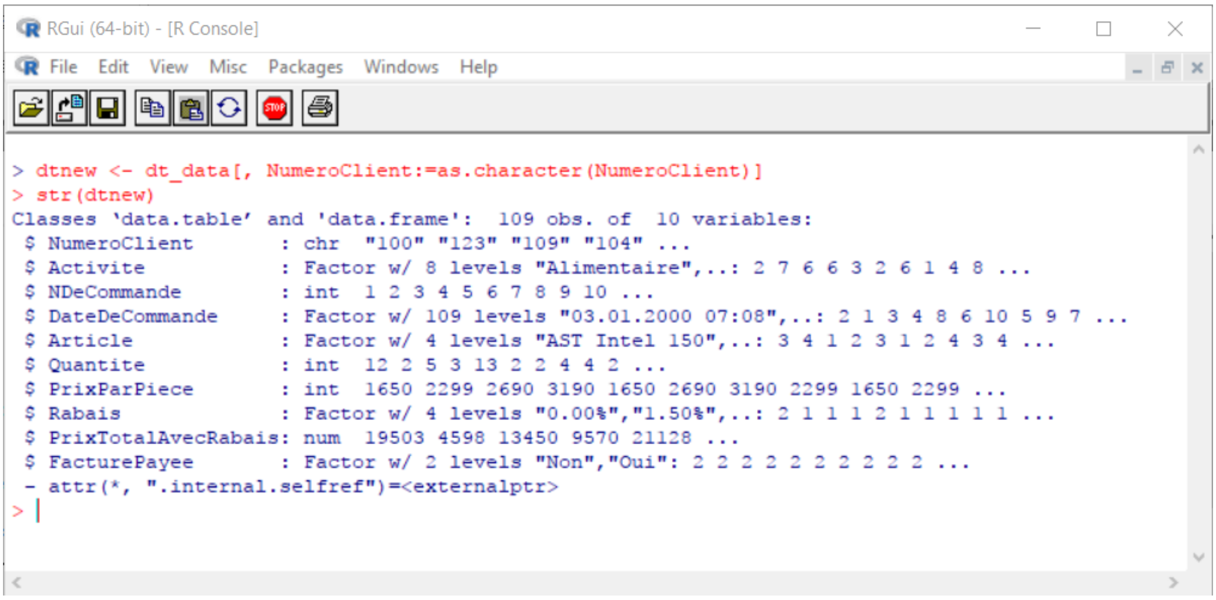
Avant de poursuivre on réinitialise l'objet data.table:

```
> dt_data<-data.table(df_data)
>
```

Changer le typage d'une colonne avec dplyr (rappel):

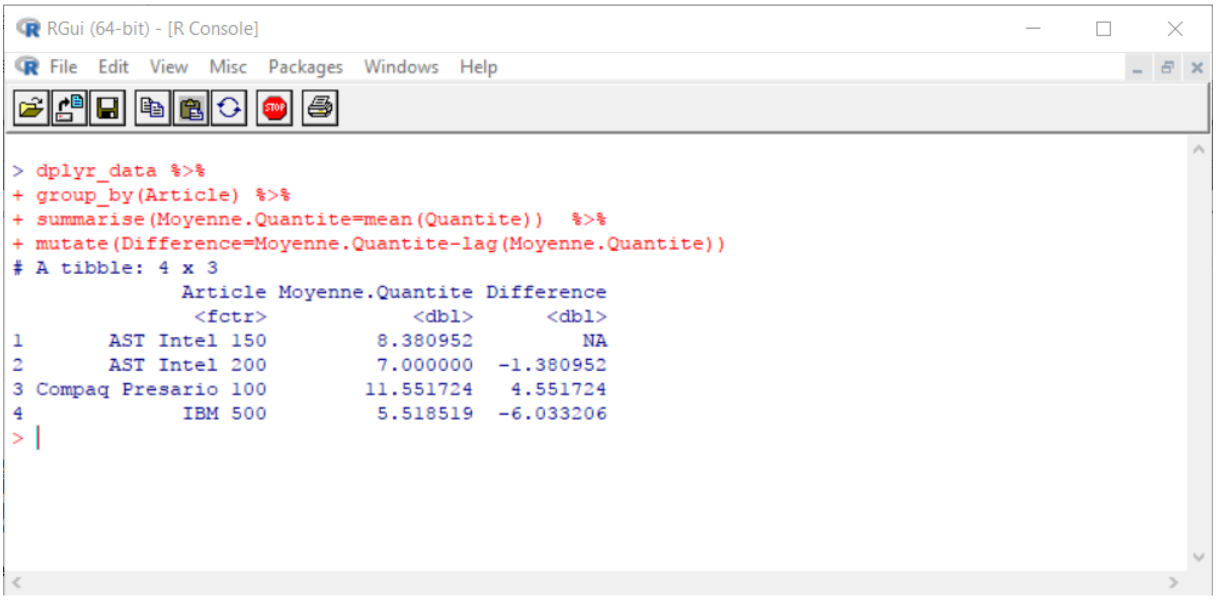
```
> dplyr_data %>%
+ mutate(NClient = as.character(NClient))
# A tibble: 109 x 10
   NClient        Activite NDeCommande   DateDeCommande            Article Quantite
     <chr>          <fctr>       <int>           <fctr>             <fctr>    <int>
 1     100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12
 2     123 Machines/Outils           2 03.01.2000 07:08            IBM 500        2
 3     109       Éducation           3 03.01.2000 14:35      AST Intel 150        5
 4     104       Éducation           4 03.03.2000 04:14      AST Intel 200        3
 5     117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13
 6     103      Assurances           6 04.01.2000 11:01      AST Intel 150        2
 7     104       Éducation           7 04.03.2000 21:30      AST Intel 200        2
 8     111     Alimentaire           8 04.01.2000 02:20            IBM 500        4
 9     113    Construction           9 04.02.2000 23:06 Compaq Presario 100        4
10     116  Pharmaceutique          10 04.01.2000 16:14            IBM 500        2
# ... with 99 more rows, and 4 more variables: PrixParPiece <int>, Rabais <fctr>,
```

et son équivalent avec data.table:



```
> dtnew <- dt_data[, NumeroClient:=as.character(NumeroClient)]
> str(dtnew)
Classes 'data.table' and 'data.frame':  109 obs. of  10 variables:
 $ NumeroClient       : chr  "100" "123" "109" "104" ...
 $ Activite           : Factor w/ 8 levels "Alimentaire",..: 2 7 6 6 3 2 6 1 4 8 ...
 $ NDeCommande        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande     : Factor w/ 109 levels "03.01.2000 07:08",..: 2 1 3 4 8 6 10 5 9 7 ...
 $ Article            : Factor w/ 4 levels "AST Intel 150",..: 3 4 1 2 3 1 2 4 3 4 ...
 $ Quantite           : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece       : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais             : Factor w/ 4 levels "0.00%","1.50%",..: 2 1 1 1 2 1 1 1 1 1 ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21128 ...
 $ FacturePayee       : Factor w/ 2 levels "Non","Oui": 2 2 2 2 2 2 2 2 2 2 ...
 - attr(*, ".internal.selfref")=<externalptr>
> 
```

Maintenant une création d'une colonne avec une statistique et un lag avec dplyr (rappel):



```
> dplyr_data %>%
+ group_by(Article) %>%
+ summarise(Moyenne.Quantite=mean(Quantite))  %>%
+ mutate(Difference=Moyenne.Quantite-lag(Moyenne.Quantite))
# A tibble: 4 x 3
              Article Moyenne.Quantite Difference
               <fctr>            <dbl>      <dbl>
1       AST Intel 150         8.380952         NA
2       AST Intel 200         7.000000  -1.380952
3 Compaq Presario 100        11.551724   4.551724
4             IBM 500         5.518519  -6.033206
> 
```

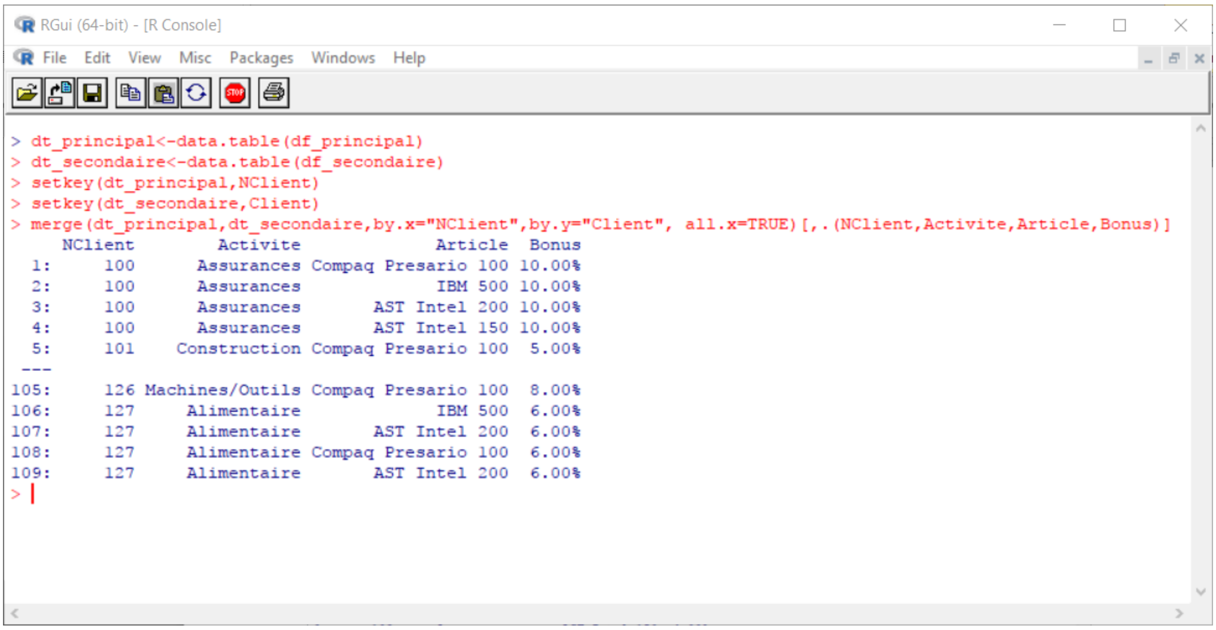
et le même avec data.table:

```
> dt_data[order(Article),.(Moyenne.Quantite = mean(Quantite)),by=Article][,Difference:=Moyenne.Quantite-lag(Moyenne.Quantite)][]
               Article Moyenne.Quantite Difference
1:       AST Intel 150         8.380952         NA
2:       AST Intel 200         7.000000  -1.380952
3: Compaq Presario 100        11.551724   4.551724
4:             IBM 500         5.518519  -6.033206
>
```

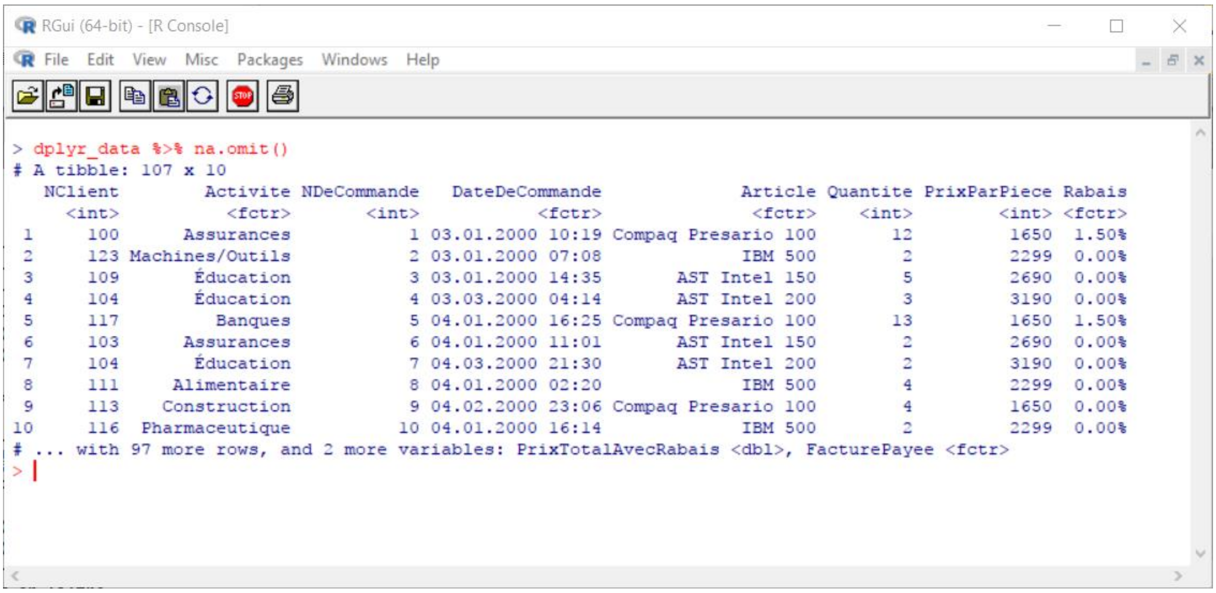
Jointure gauche entre deux tables avec une seule clé de liaison avec dplyr (rappel):

```
> df_principal<-read.csv("c:/tmp/Ventes.csv",header=T,sep=";")
> df_secondaire<-read.csv("c:/tmp/Bonus.csv",header=T,sep=",")
> select(left_join(df_principal,df_secondaire,by=c("NClient"="Client")),NClient,Activite,Article,Bonus)
   NClient        Activite             Article  Bonus
1      100      Assurances Compaq Presario 100 10.00%
2      123 Machines/Outils             IBM 500  6.00%
3      109       Éducation       AST Intel 150  8.00%
4      104       Éducation       AST Intel 200   <NA>
5      117         Banques Compaq Presario 100  7.00%
6      103      Assurances       AST Intel 150  4.00%
7      104       Éducation       AST Intel 200   <NA>
8      111     Alimentaire             IBM 500  1.00%
9      113    Construction Compaq Presario 100   <NA>
10     116  Pharmaceutique             IBM 500  3.00%
11     110    Distribution       AST Intel 200  9.00%
12     112 Machines/Outils Compaq Presario 100 10.00%
13     123 Machines/Outils             IBM 500  6.00%
14     113    Construction       AST Intel 150   <NA>
15     115    Distribution Compaq Presario 100  8.00%
16     124       Éducation       AST Intel 200 10.00%
17     124       Éducation Compaq Presario 100 10.00%
```

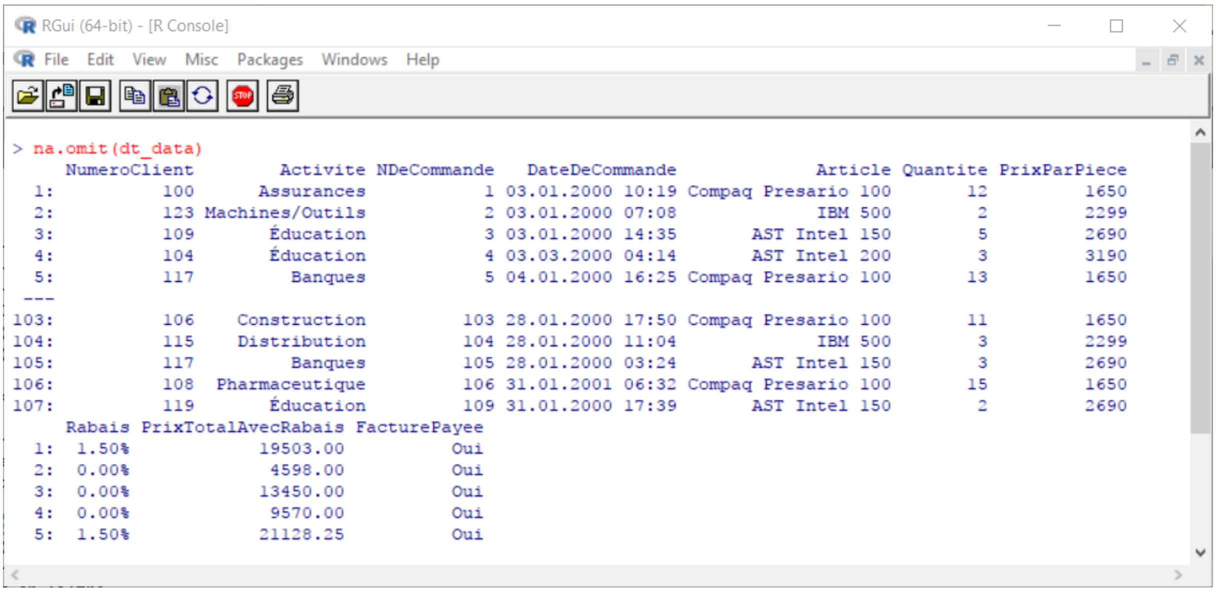
La même chose avec data.table:



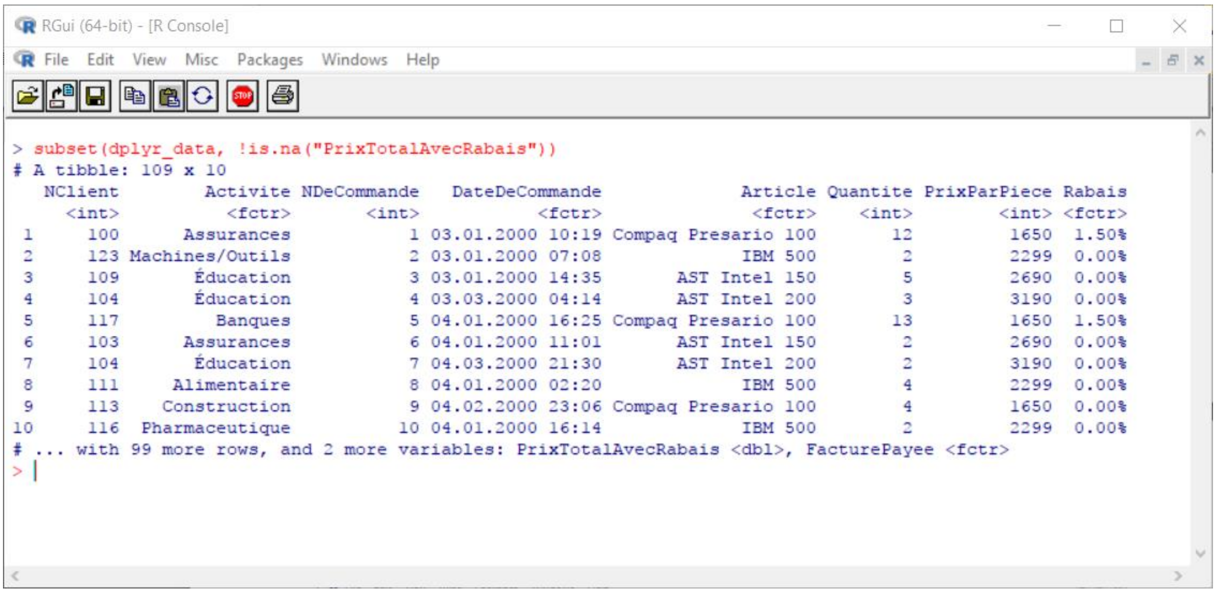
On exclut les lignes avec des valeurs NULL peut importe la colonne avec dplyr (rappel):



La même chose avec data.table:



On conserve que les lignes qui sont non-vides mais par rapport à une colonne précise avec dlplyr (rappel):



La même chose avec data.table donne:

```
> na.omit(dt_data,"PrixTotalAvecRabais")
    NumeroClient          Activite NDeCommande    DateDeCommande           Article Quantite PrixParPiece
  1:         100        Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650
  2:         123 Machines/Outils           2 03.01.2000 07:08           IBM 500        2         2299
  3:         109         Éducation           3 03.01.2000 14:35     AST Intel 150        5         2690
  4:         104         Éducation           4 03.03.2000 04:14     AST Intel 200        3         3190
  5:         117           Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650
 ---
105:         117           Banques         105 28.01.2000 03:24     AST Intel 150        3         2690
106:         108   Pharmaceutique         106 31.01.2001 06:32 Compaq Presario 100       15         1650
107:         102 Machines/Outils          NA 31.01.2000 17:32     AST Intel 200        4         3190
108:         118         Éducation          NA 31.01.2000 09:42           IBM 500        3         2299
109:         119         Éducation         109 31.01.2000 17:39     AST Intel 150        2         2690
    Rabais PrixTotalAvecRabais FacturePayee
  1:  1.50%            19503.00          Oui
  2:  0.00%             4598.00          Oui
  3:  0.00%            13450.00          Oui
  4:  0.00%             9570.00          Oui
  5:  1.50%            21128.25          Oui
```

Uniquement les lignes avec la colonne *NDeCommande* ayant des *Null* ET avec la *Quantite* supérieure à zéro et on remplace les valeurs *Null* de la colonne *NDeCommande* par des *0* avec dplyr (rappel):

```
> dplyr_data %>%
+ filter(is.na(NDeCommande),Quantite>0)
# A tibble: 2 x 10
  NClient        Activite NDeCommande   DateDeCommande       Article Quantite PrixParPiece Rabais
    <int>          <fctr>       <int>           <fctr>        <fctr>    <int>        <int> <fctr>
1     102 Machines/Outils          NA 31.01.2000 17:32 AST Intel 200        4         3190  0.00%
2     118       Éducation          NA 31.01.2000 09:42       IBM 500        3         2299  0.00%
# ... with 2 more variables: PrixTotalAvecRabais <dbl>, FacturePayee <fctr>
> dplyr_data %>% mutate(NDeCommande=ifelse(is.na(NDeCommande),0,NDeCommande))
# A tibble: 109 x 10
   NClient        Activite NDeCommande   DateDeCommande             Article Quantite PrixParPiece Rabais
     <int>          <fctr>       <dbl>           <fctr>              <fctr>    <int>        <int> <fctr>
 1     100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%
 2     123 Machines/Outils           2 03.01.2000 07:08             IBM 500        2         2299  0.00%
 3     109       Éducation           3 03.01.2000 14:35       AST Intel 150        5         2690  0.00%
 4     104       Éducation           4 03.03.2000 04:14       AST Intel 200        3         3190  0.00%
 5     117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
 6     103      Assurances           6 04.01.2000 11:01       AST Intel 150        2         2690  0.00%
 7     104       Éducation           7 04.03.2000 21:30       AST Intel 200        2         3190  0.00%
 8     111     Alimentaire           8 04.01.2000 02:20             IBM 500        4         2299  0.00%
 9     113    Construction           9 04.02.2000 23:06 Compaq Presario 100        4         1650  0.00%
10     116  Pharmaceutique          10 04.01.2000 16:14             IBM 500        2         2299  0.00%
```

La même chose avec data.table:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> dt_data[is.na(NDeCommande) & Quantite>0,]
   NumeroClient         Activite NDeCommande    DateDeCommande       Article Quantite PrixParPiece Rabais
1:          102 Machines/Outils          NA 31.01.2000 17:32 AST Intel 200        4         3190  0.00%
2:          118        Éducation          NA 31.01.2000 09:42       IBM 500        3         2299  0.00%
   PrixTotalAvecRabais FacturePayee
1:               12760          Non
2:                6897          Non
> dt_data[,NDeCommande:=ifelse(is.na(NDeCommande),0,NDeCommande)]
> |
```

Pour fusionner deux jeux de données en dplyr (rappel):

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> df_partie1<-read.csv("c:/tmp/Ventes_01.csv",header=T,sep=";")
> df_partie2<-read.csv("c:/tmp/Ventes_02.csv",header=T,sep=";")
> bind_rows(df_partie1,df_partie2)
    NClient        Activite NDeCommande    DateDeCommande              Article Quantite PrixParPiece Rabais
1       100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%
2       123 Machines/Outils           2 03.01.2000 07:08             IBM 500        2         2299  0.00%
3       109       Éducation           3 03.01.2000 14:35       AST Intel 150        5         2690  0.00%
4       104       Éducation           4 03.01.2000 04:14       AST Intel 200        3         3190  0.00%
5       117         Banques           5 04.01.2000 16:25 Compaq Presario 100       13         1650  1.50%
6       103      Assurances           6 04.01.2000 11:01       AST Intel 150        2         2690  0.00%
7       104       Éducation           7 04.01.2000 21:30       AST Intel 200        2         3190  0.00%
```


Et l'équivalent en data.table:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> dt_partie1<-data.table(df_partie1)
> dt_partie2<-data.table(df_partie2)
> merge(dt_partie1,dt_partie2,all=TRUE)
     NClient        Activite NDeCommande    DateDeCommande              Article Quantite PrixParPiece Rabais
  1:     100      Assurances           1 03.01.2000 10:19 Compaq Presario 100       12         1650  1.50%
  2:     100      Assurances          23 06.01.2000 06:48             IBM 500        3         2299  0.00%
  3:     100      Assurances          45 12.01.2000 17:05       AST Intel 200        4         3190  0.00%
  4:     100      Assurances          50 13.01.2000 11:02       AST Intel 150       13         2690  1.50%
  5:     101    Construction          19 05.01.2000 17:34 Compaq Presario 100       14         1650  1.50%
 ---
105:     126 Machines/Outils          63 19.01.2000 03:24 Compaq Presario 100       12         1650  1.50%
```

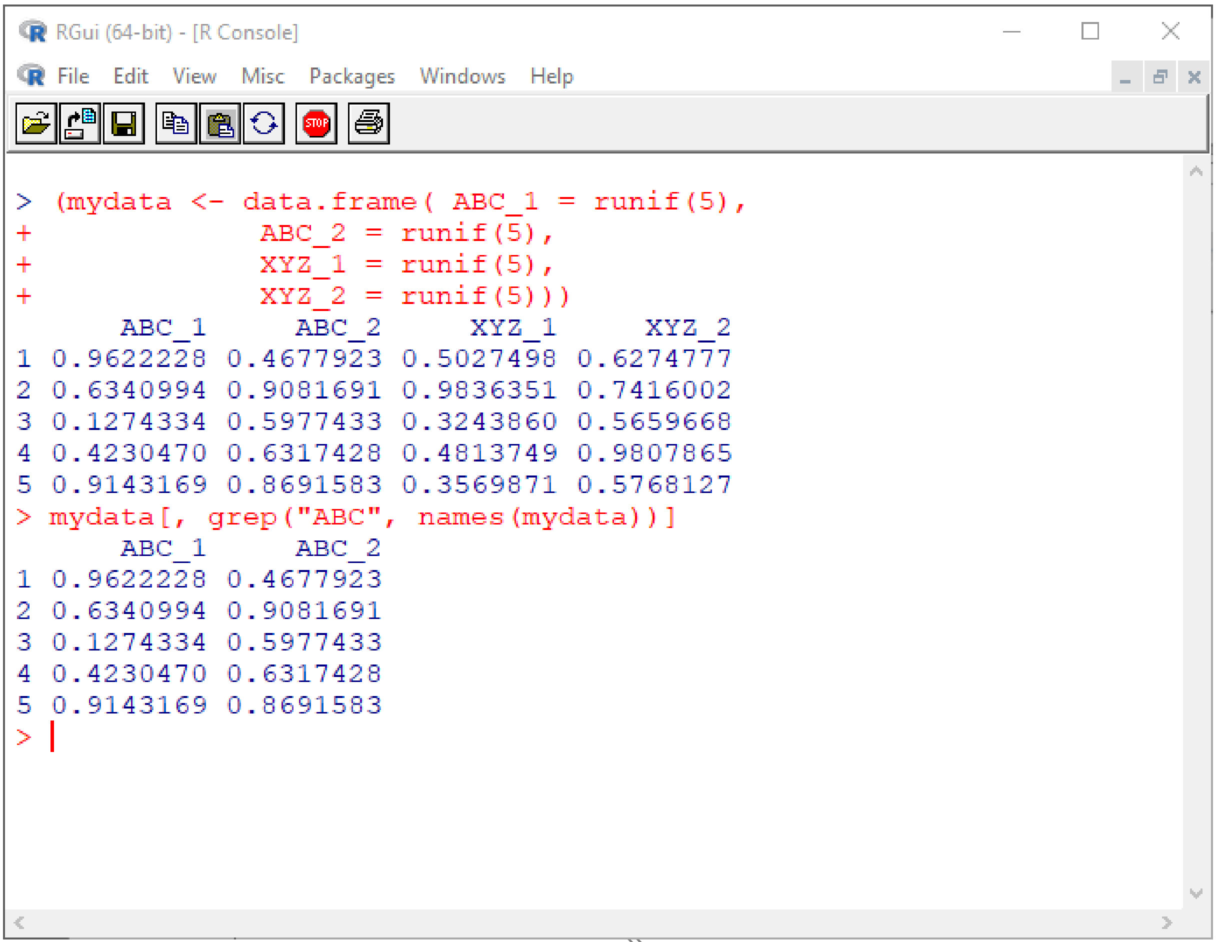
Enfin une petite compilation sympa :

```
> library("data.table")
>
> mydata<-fread("c:/tmp/Ventes.csv",header=T,sep=";")
> #on renomme les colonnes (plus joli)
> colnames(mydata)<-c("Client","Secteur","NumCommande", "Date","Article","Quantite","PrixUnitaire",
+ "Rabais","PrixTotal","FacturePayee")
> #vecteur avec toutes les colonnes de dates
> date_cols <- c("Date")
> #on convertit les colonnes de dates en vraies dates...
> mydata[, (date_cols) := lapply(.SD, function(x) as.Date(x,"%d.%m.%Y")), .SDcols = date_cols]
>
> str(mydata)
Classes 'data.table' and 'data.frame':  109 obs. of  10 variables:
 $ Client      : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Secteur     : chr  "Assurances" "Machines/Outils" "Éducation" "Éducation" ...
 $ NumCommande : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Date        : Date, format: "2000-01-03" "2000-01-03" ...
 $ Article     : chr  "Compaq Presario 100" "IBM 500" "AST Intel 150" "AST Intel 200" ...
 $ Quantite    : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixUnitaire: int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais      : chr  "1.50%" "0.00%" "0.00%" "0.00%" ...
 $ PrixTotal   : num  19503 4598 13450 9570 21128 ...
 $ FacturePayee: chr  "Oui" "Oui" "Oui" "Oui" ...
 - attr(*, ".internal.selfref")=<externalptr>
> #on groupe
> mydata<-setDT(mydata)[, .(SommeQuantite = sum(Quantite)), by = .(yr = year(Date), mon = month(Date))]
> #on trie
> mydata<- mydata[order(yr, mon)]
> #on ajoute une clé
> mydata[, ID := .I]
> #on groupe les mois et années
> mydata[, AnneeMois:=(paste0(yr,"-", mon))]
> #on supprime deux colonnes non-désirées
> mydata[,c('yr','mon') := NULL]
> mydata
   SommeQuantite ID AnneeMois
1:           739  1    2000-1
2:            49  2    2000-2
3:            36  3    2000-3
4:            45  4    2001-1
5:             6  5    2001-3
6:             9  6    2001-4
>
```

Enfin pour garder toutes les colonnes qui commencent par une certaine chaîne en dplyr (rappel):

```
> (mydata <- data.frame( ABC_1 = runif(5),
+               ABC_2 = runif(5),
+               XYZ_1 = runif(5),
+               XYZ_2 = runif(5)))
      ABC_1     ABC_2     XYZ_1     XYZ_2
1 0.7133016 0.9130878 0.7442772 0.2834595
2 0.1007690 0.9458531 0.9159742 0.2515457
3 0.9503049 0.2791562 0.9945982 0.5032552
4 0.1218178 0.1234711 0.9423607 0.4969662
5 0.2196566 0.7971605 0.4861354 0.3184458
> mydata <- mydata %>% select(starts_with("ABC"))
> mydata
      ABC_1     ABC_2
1 0.7133016 0.9130878
2 0.1007690 0.9458531
3 0.9503049 0.2791562
4 0.1218178 0.1234711
5 0.2196566 0.7971605
>
```

Et l'équivalent en data.table:



Enfin, voyons encore un autre exemple relativement courant, qui consiste à pivoter des données, et qui pose problème dans le monde Microsoft Excel et Microsoft Power BI, mais qui s'avère très facile avec R en utilisant les fonctions **dcast( )**, **setDT( )** et **rowid( )** du package data.table:

```
> library("data.table")
> df <- iris
> #nous voulons pivoter les noms des espèces relativement à la longuer de sépales
> head(iris,10)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1:          5.1         3.5          1.4         0.2  setosa
 2:          4.9         3.0          1.4         0.2  setosa
 3:          4.7         3.2          1.3         0.2  setosa
 4:          4.6         3.1          1.5         0.2  setosa
 5:          5.0         3.6          1.4         0.2  setosa
 6:          5.4         3.9          1.7         0.4  setosa
 7:          4.6         3.4          1.4         0.3  setosa
 8:          5.0         3.4          1.5         0.2  setosa
 9:          4.4         2.9          1.4         0.2  setosa
10:          4.9         3.1          1.5         0.1  setosa
> dcast(setDT(df), rowid(Species)~Species, value.var = "Sepal.Length")
    Species setosa versicolor virginica
 1:       1    5.1        7.0       6.3
 2:       2    4.9        6.4       5.8
 3:       3    4.7        6.9       7.1
 4:       4    4.6        5.5       6.3
 5:       5    5.0        6.5       6.5
 6:       6    5.4        5.7       7.6
 7:       7    4.6        6.3       4.9
 8:       8    5.0        4.9       7.3
 9:       9    4.4        6.6       6.7
10:      10    4.9        5.2       7.2
11:      11    5.4        5.0       6.5
12:      12    4.8        5.9       6.4
13:      13    4.8        6.0       6.8
14:      14    4.3        6.1       5.7
15:      15    5.8        5.6       5.8
16:      16    5.7        6.7       6.4
17:      17    5.4        5.6       6.5
18:      18    5.1        5.8       7.7
19:      19    5.7        6.2       7.7
20:      20    5.1        5.6       6.0
21:      21    5.4        5.9       6.9
22:      22    5.1        6.1       5.6
23:      23    4.6        6.3       7.7
24:      24    5.1        6.1       6.3
25:      25    4.8        6.4       6.7
26:      26    5.0        6.6       7.2
27:      27    5.0        6.8       6.2
28:      28    5.2        6.7       6.1
29:      29    5.2        6.0       6.4
30:      30    4.7        5.7       7.2
31:      31    4.8        5.5       7.4
32:      32    5.4        5.5       7.9
33:      33    5.2        5.8       6.4
34:      34    5.5        6.0       6.3
35:      35    4.9        5.4       6.1
36:      36    5.0        6.0       7.7
```

Bingo! Opération réussie!

Nous n'avons pas trouvé d'équivalent avec dplyr à ce jour!

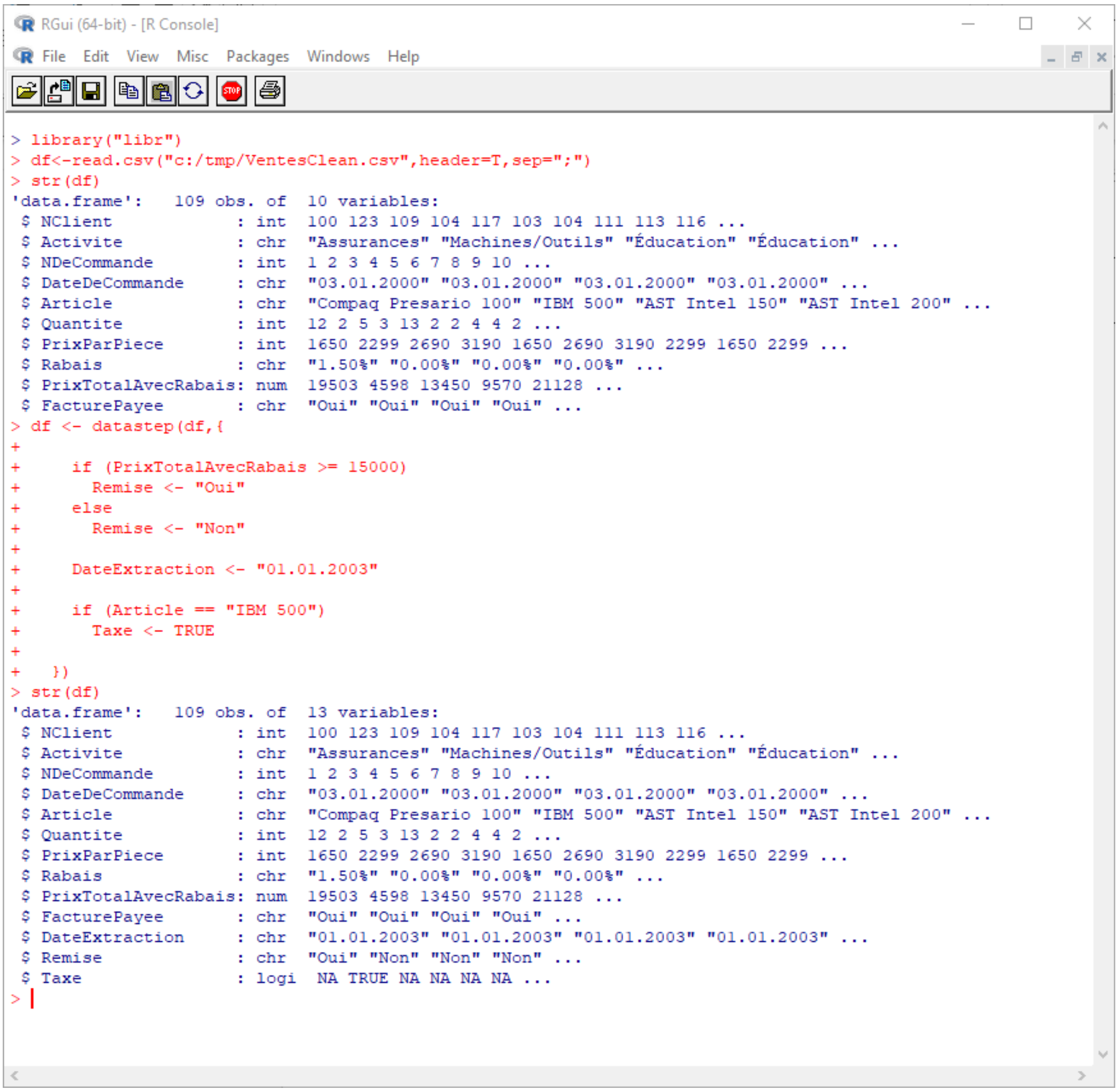
Package libr

Ce page de manipulationd de données plairera particulièrement à ceux qui viennent du monde SAS.

Les quelques exemples que nous allons donner ci-dessous sont adaptés de la vignette du package **libr** disponible ici:

https://cran.rstudio.com/web/packages/libr/vignettes/libr-datastep.html

Commençons par un ajout de quelques colonnes à notre jeu de données habituel:




```
> library("libr")
> df<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=";")
> str(df)
'data.frame':   109 obs. of  10 variables:
 $ NClient           : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Activite          : chr  "Assurances" "Machines/Outils" "Éducation" "Éducation" ...
 $ NDeCommande       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande    : chr  "03.01.2000" "03.01.2000" "03.01.2000" "03.01.2000" ...
 $ Article           : chr  "Compaq Presario 100" "IBM 500" "AST Intel 150" "AST Intel 200" ...
 $ Quantite          : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece      : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais            : chr  "1.50%" "0.00%" "0.00%" "0.00%" ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21128 ...
 $ FacturePayee      : chr  "Oui" "Oui" "Oui" "Oui" ...
> df <- datastep(df,{
+
+     if (PrixTotalAvecRabais >= 15000)
+       Remise <- "Oui"
+     else
+       Remise <- "Non"
+
+     DateExtraction <- "01.01.2003"
+
+     if (Article == "IBM 500")
+       Taxe <- TRUE
+
+   })
> str(df)
'data.frame':   109 obs. of  13 variables:
 $ NClient           : int  100 123 109 104 117 103 104 111 113 116 ...
 $ Activite          : chr  "Assurances" "Machines/Outils" "Éducation" "Éducation" ...
 $ NDeCommande       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DateDeCommande    : chr  "03.01.2000" "03.01.2000" "03.01.2000" "03.01.2000" ...
 $ Article           : chr  "Compaq Presario 100" "IBM 500" "AST Intel 150" "AST Intel 200" ...
 $ Quantite          : int  12 2 5 3 13 2 2 4 4 2 ...
 $ PrixParPiece      : int  1650 2299 2690 3190 1650 2690 3190 2299 1650 2299 ...
 $ Rabais            : chr  "1.50%" "0.00%" "0.00%" "0.00%" ...
 $ PrixTotalAvecRabais: num  19503 4598 13450 9570 21128 ...
 $ FacturePayee      : chr  "Oui" "Oui" "Oui" "Oui" ...
 $ DateExtraction    : chr  "01.01.2003" "01.01.2003" "01.01.2003" "01.01.2003" ...
 $ Remise            : chr  "Oui" "Non" "Non" "Non" ...
 $ Taxe              : logi  NA TRUE NA NA NA NA ...
>
```

Etc. Pour plus d'infos sur le groupement et autres options, voir la documentation du package!

Exemple: Comparaison de data frames

Nous allons voir ici un survol du package **diffdf** qui peut s'avérer très utile dans certaines situations (renvoi régulier de data frame par des clients externe/collègue d'un autre département avec modifications régulières).

Ce qui va suivre est simplement un copier/coller traduit en français de la vignette disponible ici:

https://cran.r-project.org/web/packages/diffdf/vignettes/diffdf-basic.html

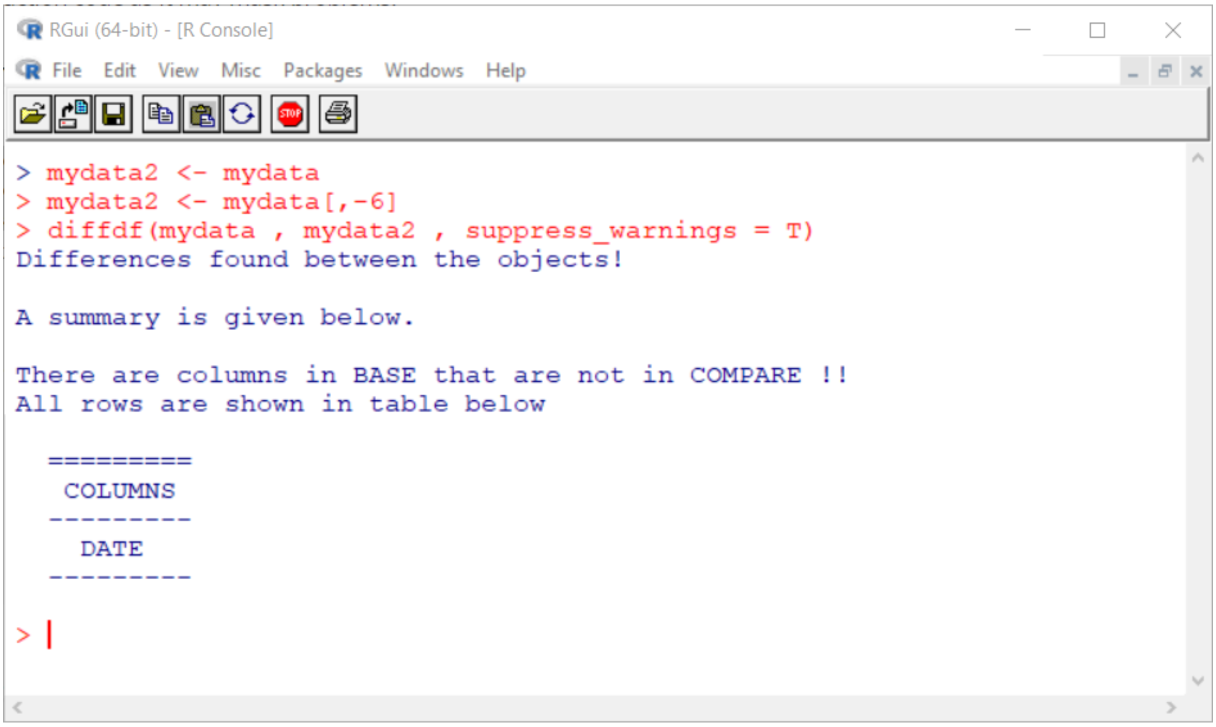
On va d'abord charger les données de l'exemple ainsi que le package:

```
> mydata<-read.csv("c:/tmp/CompareDataframe.csv",header=T,sep=";")
> mydata
   ID GROUP1 GROUP2 INTEGER BINARY       DATE            DATETIME CONTINUOUS CATEGORICAL LOGICAL               CHARACTER
1   1      1      1      41      M 2003-06-22 2000-11-28 20:40:53   21.62687           C   FALSE             czNCZ58ZzIvM
2   2      1      2      41      F 2016-12-03 1994-08-30 19:05:02   26.47256           C    TRUE YithbnzPt6P3xRXmL1IF8w3z
3   3      1      3      41      M 2016-05-08 1992-09-11 11:30:18   16.12255           A    TRUE          Wsixhy7Vp894V9b
4   4      1      4      32      M 2015-06-02 2007-11-12 11:28:29   23.47854           C    TRUE               uatNHn9wSNV
5   5      1      5      55      F 1986-04-15 1998-08-04 01:27:49   21.20108           A    TRUE              BPcEVZhU2xL6
6   6      1      6      33      M 1994-05-25 2001-12-05 08:24:35   46.92355           A    TRUE               0iCTNtAfNC8
7   7      1      7      40      F 2009-02-08 1986-11-02 18:13:03   28.21110           A    TRUE                q4YhlIweU1
8   8      1      8      44      F 2020-07-21 1998-08-22 05:23:24   27.65611           A   FALSE             3QcjumCbGjgsg
9   9      1      9      51      F 1967-05-25 2003-01-03 22:09:29   22.02877           C    TRUE                cMo87KmHAI0
10 10      1     10      40      M 2044-03-11 1996-04-19 11:10:12   40.85163           B   FALSE                pvnxMdEP0FS
11 11      1     11      40      M 2001-02-02 2001-05-25 09:25:26   33.85712           A   FALSE              KDx7126vJZVqP
12 12      1     12      45      F 2021-09-18 1999-12-25 02:20:59   36.37379           A   FALSE               Khe741OWgFTr
13 13      1     13      48      M 2020-06-17 1978-12-06 07:43:56   31.94585           B    TRUE                  IHMYsSB4w
14 14      1     14      30      M 2035-11-26 1995-06-25 23:07:52   21.92185           C   FALSE           XHkT4wevVvf4Fs3dW
15 15      1     15      44      F 2021-10-14 2001-04-04 11:19:43   31.78516           C   FALSE                 Oged9XWMr
16 16      2      1      39      M 2004-05-29 1995-09-05 08:49:41   15.64697           C   FALSE              S1VkcOidBLVRs
17 17      2      2      35      F 2016-09-11 2005-12-01 06:46:44   15.68732           B   FALSE                   NWUvAS5
18 18      2      3      40      F 1986-08-08 2000-03-12 12:47:12   41.36808           B    TRUE                tlvd1rs38YX
19 19      2      4      50      F 1978-09-20 1992-12-24 14:24:30   30.54506           C    TRUE       v7wrPk2vA99LRI5aY01L
20 20      2      5      51      F 2006-03-05 2007-02-21 23:21:50   42.59986           B   FALSE                LN2Xnoo85If
21 21      2      6      45      M 2031-01-25 1995-07-11 17:13:39   34.02535           B    TRUE            eaL9r8K7Np4GeVNv
22 22      2      7      44      M 1998-08-26 2001-02-09 20:49:32   50.85575           B    TRUE              OgHM3RDaECPE
23 23      2      8      42      M 2005-02-22 1999-03-30 08:12:24   23.52028           C    TRUE                0kQtq6v6ntq
24 24      2      9      34      M 2002-03-25 1997-07-12 12:26:53   31.01492           B   FALSE              w9BYbGGi0oU55I
25 25      2     10      37      F 2022-02-01 1994-09-13 10:38:53   34.13827           B    TRUE                   Lxn56Eu
26 26      2     11      37      M 1998-01-01 2012-12-02 21:49:49   39.23846           B    TRUE                lmcKQSK0KvA
27 27      2     12      44      F 2024-01-05 1998-04-06 04:42:27   48.79333           B    TRUE                 AYZnijBM6
28 28      2     13      47      F 2001-08-26 2000-09-25 22:43:09   43.25825           A    TRUE              Z6AijFc9Mw5KIx
29 29      2     14      41      F 1994-08-20 2007-05-20 12:17:27   37.41091           C    TRUE               5ATsKW8MlP7h
30 30      2     15      41      F 1951-11-22 2007-12-30 19:37:24   26.73231           C    TRUE                  GcklrCCs
>
```

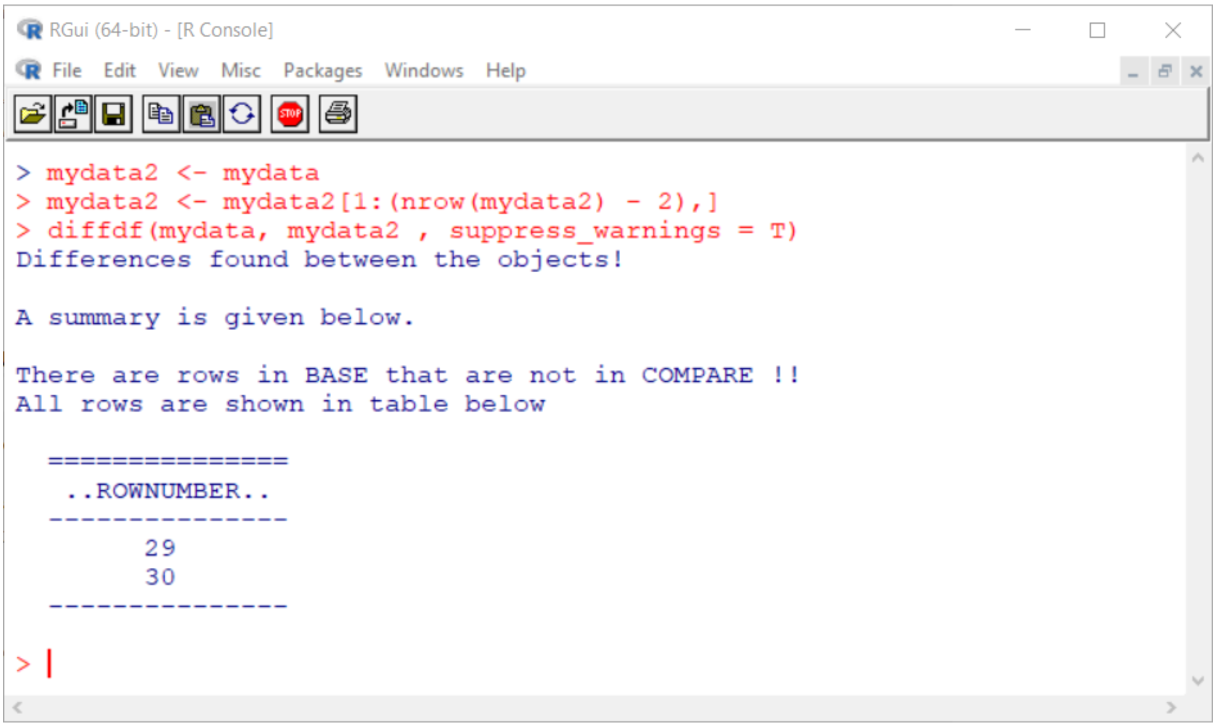
Commençons à comparer le data frame à lui-même avec la fonction **diffdf( )** :

```
> diffdf(mydata,mydata)
No issues were found!
>
```

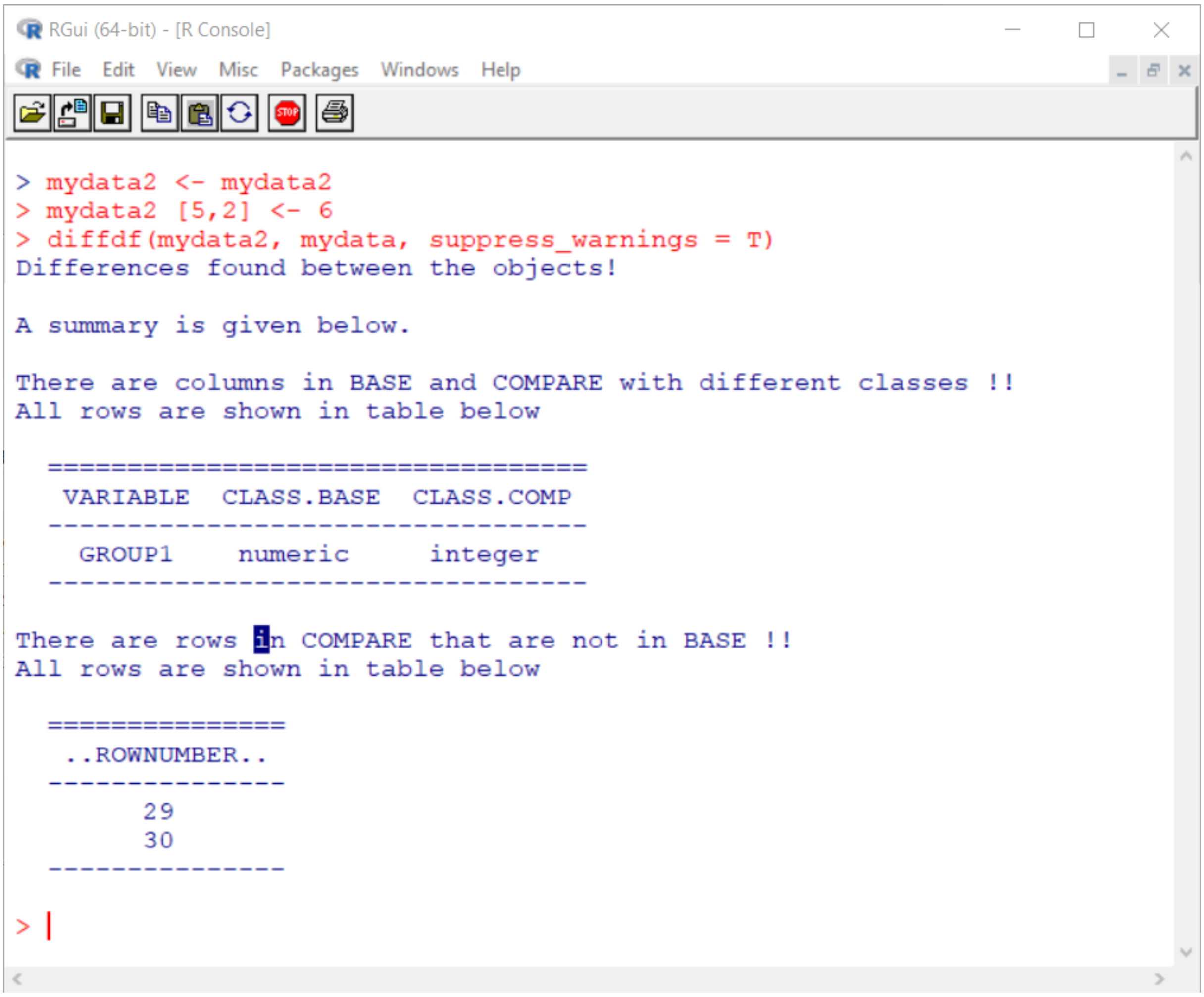
Comparons les 2 data frames avec une colonne en moins:



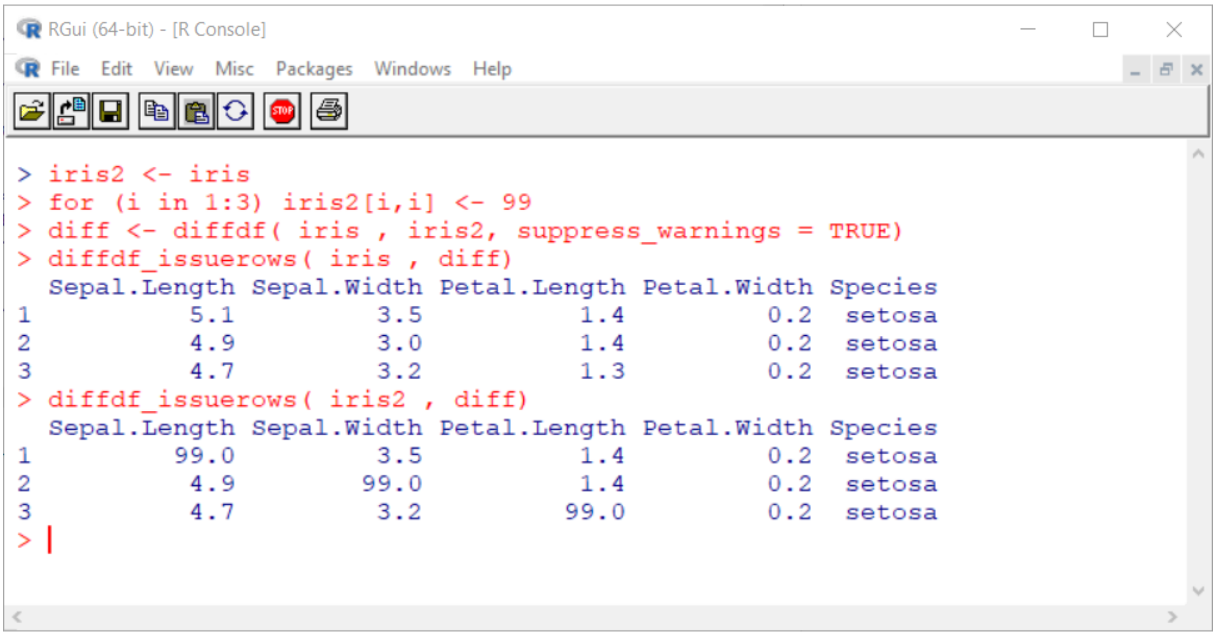
Avec différentes lignes:



Différentes valeurs:



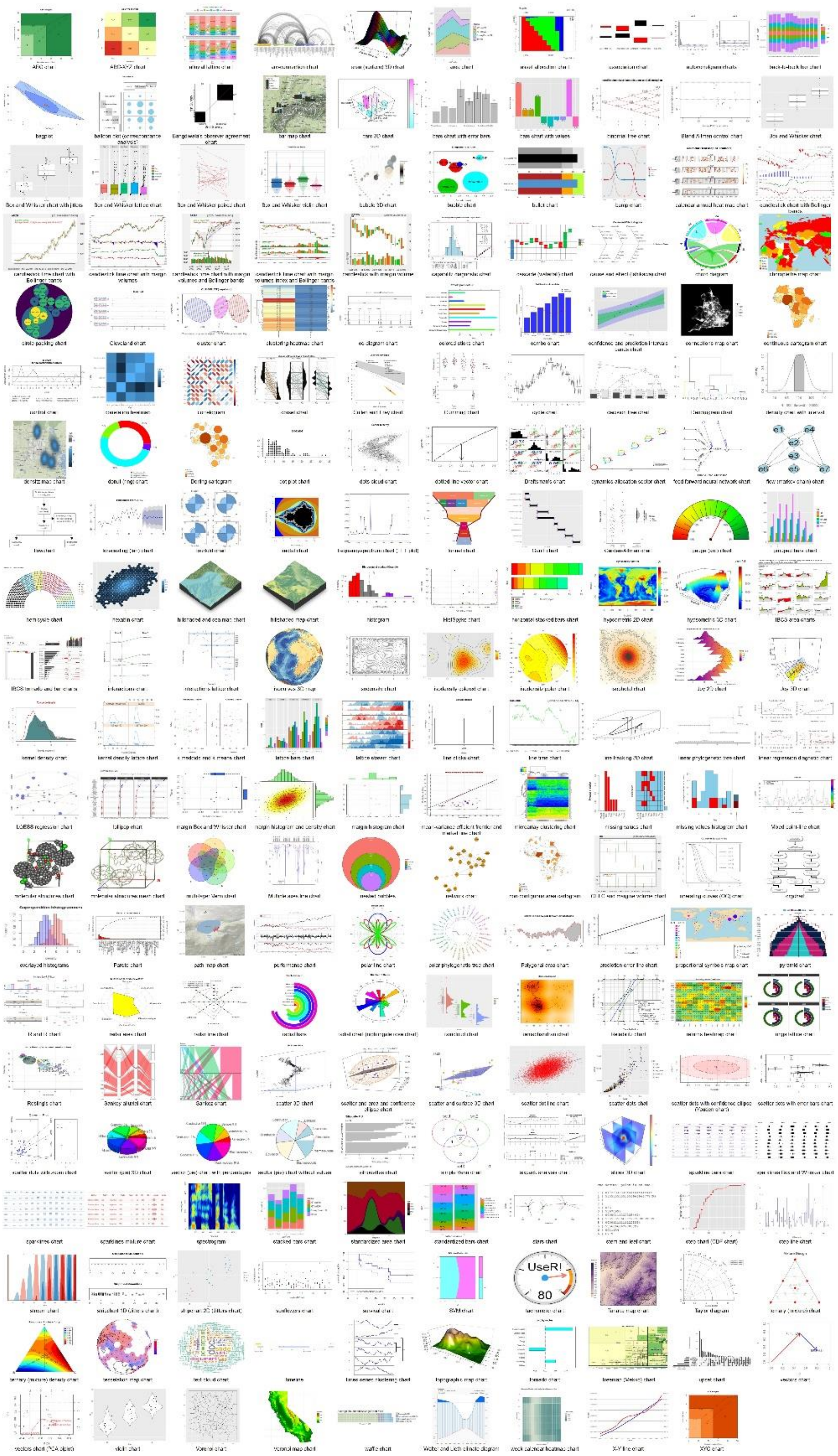
On peut mettre en évidence les lignes qui ont aussi des différences avec la fonction **diffdf_issuerows( )** :



11. Créer des diagrammes/graphiques (Data Viz)

Nous allons dans ce chapitre voir seulement des diagrammes de ce qui est typiquement demandé pendant les études obligatoires (les graphiques allant au-delà seront vus dans les chapitres thématiques de chaque domaine de spécialisation).

Voici d'abord une gallerie de tous les types de graphiques (205) existants à ce jour et faisables avec R et dont l'élaboration est présentée dans ce livre:



Je ne suis personnellement pas un fan de l'utilisation des graphiques en statistiques (on passe trop de temps à bricoler le design plutôt que de réfléchir à la robustesse du modèle ou de chercher pour de meilleurs modèles existants). Donc si jamais j'améliorerai les graphiques donnés ci-dessous en fonction des questions des participants et étudiants qui suivent mes cours.

Au niveau esthétique (du moins à l'écran!) vous verrez que sommes très loin de ce qu'un tableur comme Microsoft Excel 2007 ou ultérieur peut faire (ce qui maîtrisent totalement Microsoft Excel me comprendront). De plus le comportement de R est simplement catastrophique lorsque l'utilisateur redimensionne la fenêtre contenant un graphique. Cependant, techniquement, R est plus puissant et plus flexible (dommage que Microsoft n'ajoute pas de nouveaux graphiques à Excel depuis ces 20 dernières années...).

Avant de commencer ce chapitre, nous recommandons fortement au lecteur de visiter le site suivant:

https://www.r-graph-gallery.com/

Pour commencer à faire des graphiques divers élémentaires nous allons essayer d'utiliser toujours dans la mesure du possible la même source de données, à savoir:

RGui (64-bit) - [R Console]

Fichier Edition Voir Misc Packages Fenêtres Aide

```
> setwd("C:/")
> mydata=read.csv("Ventes.csv",header=T,sep=";")
> mydata
   N..Client        Activité N..de.Commande Date.de.commande          Article Quantité Prix.par.pièce Rabais. Prix.total.avec.rabais Facture.payée
1        100      Assurances              1       03.01.2000 Compaq Presario 100       12           1650   1.50%               19503.00           Oui
2        123 Machines/Outils              2       03.01.2000          IBM 500        2           2299   0.00%                4598.00           Oui
3        109       Éducation              3       03.01.2000    AST Intel 150        5           2690   0.00%               13450.00           Oui
4        104       Éducation              4       03.01.2000    AST Intel 200        3           3190   0.00%                9570.00           Oui
5        117         Banques              5       04.01.2000 Compaq Presario 100       13           1650   1.50%               21128.25           Oui
6        103      Assurances              6       04.01.2000    AST Intel 150        2           2690   0.00%                5380.00           Oui
7        104       Éducation              7       04.01.2000    AST Intel 200        2           3190   0.00%                6380.00           Oui
8        111     Alimentaire              8       04.01.2000          IBM 500        4           2299   0.00%                9196.00           Oui
9        113    Construction              9       04.01.2000 Compaq Presario 100        4           1650   0.00%                6600.00           Oui
10       116  Pharmaceutique             10       04.01.2000          IBM 500        2           2299   0.00%                4598.00           Oui
11       110    Distribution             11       05.01.2000    AST Intel 200        6           3190   1.50%               18852.90           Oui
12       112 Machines/Outils             12       05.01.2000 Compaq Presario 100        6           1650   1.50%                9751.50           Oui
13       123 Machines/Outils             13       05.01.2000          IBM 500        6           2299   1.50%               13587.09           Oui
14       113    Construction             14       05.01.2000    AST Intel 150        3           2690   0.00%                8070.00           Oui
15       115    Distribution             15       05.01.2000 Compaq Presario 100        8           1650   1.50%               13002.00           Oui
16       124       Éducation             16       05.01.2000    AST Intel 200        8           3190   1.50%               25137.20           Oui
17       124       Éducation             17       05.01.2000 Compaq Presario 100       11           1650   1.50%               17877.75           Oui
18       106    Construction             18       05.01.2000    AST Intel 200       11           3190   1.50%               34563.65           Oui
19       101    Construction             19       05.01.2000 Compaq Presario 100       14           1650   1.50%               22753.50           Non
20       116  Pharmaceutique             20       06.01.2000          IBM 500        7           2299   1.50%               15851.60           Non
21       112 Machines/Outils             21       06.01.2000    AST Intel 150        6           2690   1.50%               15897.90           Oui
22       125    Construction             22       06.01.2000 Compaq Presario 100       23           1650   3.00%               36811.50           Oui
23       100      Assurances             23       06.01.2000          IBM 500        3           2299   0.00%                6897.00           Oui
24       125    Construction             24       06.01.2000    AST Intel 200        2           3190   0.00%                6380.00           Oui
25       104       Éducation             25       07.01.2000    AST Intel 150       12           2690   1.50%               31795.80           Oui
26       126 Machines/Outils             26       07.01.2000    AST Intel 150       24           2690   3.00%               62623.20           Oui
27       121  Pharmaceutique             27       07.01.2000          IBM 500        8           2299   1.50%               18116.12           Non
28       114    Distribution             28       07.01.2000    AST Intel 200        9           3190   1.50%               28279.35           Oui
```

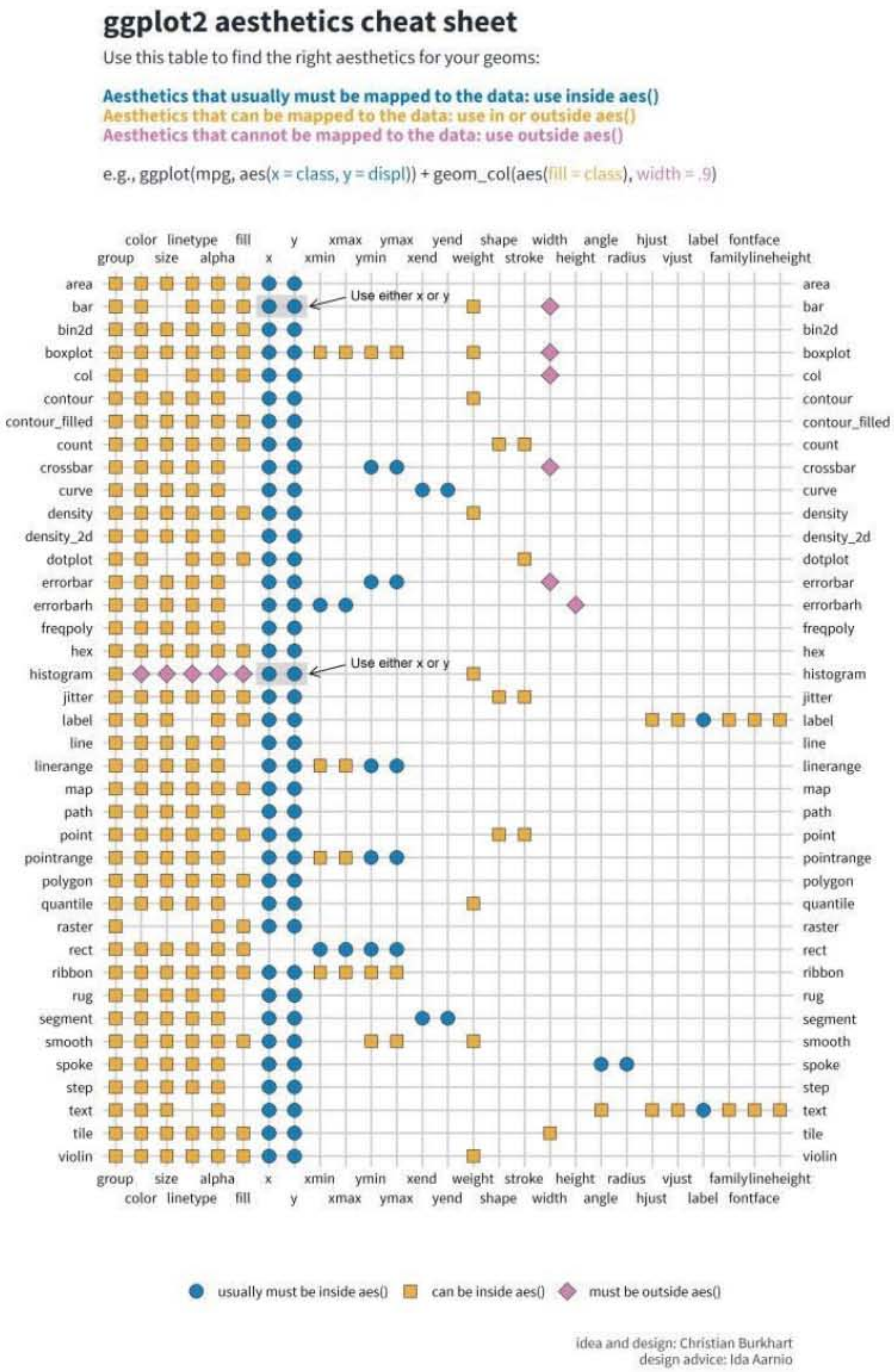
Dans le but d'effectuer quelques graphiques classiques de ce tableau mais que Microsoft Excel par exemple ne saurait pas effectuer sans un traitement préalable ou sans passer par un tableau croisé dynamique.

Si jamais comme couleurs dans les graphiques sont souvent utilisées, voici une image qui peut être utile qui donne seulement un échantillon des premières lettres de l'alphabet:

1	white	#FFFFFF	255	255	255
2	aliceblue	#F0F8FF	240	248	255
3	antiquewhite	#FAEBD7	250	235	215
4	antiquewhite1	#FFEFDB	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8378	139	131	120
8	aquamarine	#7FFFD4	127	255	212
9	aquamarine1	#7FFFD4	127	255	212
10	aquamarine2	#76EEC6	118	238	198
11	aquamarine3	#66CDAA	102	205	170
12	aquamarine4	#458B74	69	139	116
13	azure	#F0FFFF	240	255	255
14	azure1	#F0FFFF	240	255	255
15	azure2	#E0EEEE	224	238	238
16	azure3	#C1CDCD	193	205	205
17	azure4	#838B8B	131	139	139
18	beige	#F5F5DC	245	245	220
19	bisque	#FFE4C4	255	228	196
20	bisque1	#FFE4C4	255	228	196
21	bisque2	#EED5B7	238	213	183
22	bisque3	#CDB79E	205	183	158
23	bisque4	#8B7D6B	139	125	107
24	black	#000000	0	0	0
25	blanchedalmond	#FFEBCD	255	235	205
26	blue	#0000FF	0	0	255
27	blue1	#0000FF	0	0	255
28	blue2	#0000EE	0	0	238
29	blue3	#0000CD	0	0	205
30	blue4	#00008B	0	0	139
31	blueviolet	#8A2BE2	138	43	226
32	brown	#A52A2A	165	42	42
33	brown1	#FF4040	255	64	64
34	brown2	#EE3B3B	238	59	59
35	brown3	#CD3333	205	51	51
36	brown4	#8B2323	139	35	35
37	burlywood	#DEB887	222	184	135
38	burlywood1	#FFD39B	255	211	155
39	burlywood2	#EEC591	238	197	145
40	burlywood3	#CDAA7D	205	170	125
41	burlywood4	#8B7355	139	115	85
42	cadetblue	#5F9EA0	95	158	160
43	cadetblue1	#98F5FF	152	245	255
44	cadetblue2	#8EE5EE	142	229	238
45	cadetblue3	#7AC5CD	122	197	205
46	cadetblue4	#53868B	83	134	139
47	chartreuse	#7FFF00	127	255	0
48	chartreuse1	#7FFF00	127	255	0
49	chartreuse2	#76EE00	118	238	0
50	chartreuse3	#66CD00	102	205	0
51	chartreuse4	#458B00	69	139	0
52	chocolate	#D2691E	210	105	30
53	chocolate1	#FF7F24	255	127	36
54	chocolate2	#EE7621	238	118	33
55	chocolate3	#CD661D	205	102	29
56	chocolate4	#8B4513	139	69	19
57	coral	#FF7F50	255	127	80
58	coral1	#FF7256	255	114	86
59	coral2	#EE6A50	238	106	80
60	coral3	#CD5B45	205	91	69
61	coral4	#8B3E2F	139	62	47
62	cornflowerblue	#6495ED	100	149	237
63	cornsilk	#FFF8DC	255	248	220
64	cornsilk1	#FFF8DC	255	248	220
65	cornsilk2	#EEE8CD	238	232	205
66	cornsilk3	#CDC8B1	205	200	177
67	cornsilk4	#8B8878	139	136	120
68	cyan	#00FFFF	0	255	255
69	cyan1	#00FFFF	0	255	255
70	cyan2	#00EEEE	0	238	238
71	cyan3	#00CDCD	0	205	205
72	cyan4	#008B8B	0	139	139
73	darkblue	#00008B	0	0	139
74	darkcyan	#008B8B	0	139	139
75	darkgoldenrod	#B8860B	184	134	11
76	darkgoldenrod1	#FFB90F	255	185	15
77	darkgoldenrod2	#EEAD0E	238	173	14
78	darkgoldenrod3	#CD950C	205	149	12
79	darkgoldenrod4	#8B6508	139	101	8
80	darkgray	#A9A9A9	169	169	169
81	darkgreen	#006400	0	100	0
82	darkgrey	#A9A9A9	169	169	169
83	darkkhaki	#BDB76B	189	183	107
84	darkmagenta	#8B008B	139	0	139
85	darkolivegreen	#556B2F	85	107	47
86	darkolivegreen1	#CAFF70	202	255	112
87	darkolivegreen2	#BCEE68	188	238	104
88	darkolivegreen3	#A2CD5A	162	205	90
89	darkolivegreen4	#6E8B3D	110	139	61
90	darkorange	#FF8C00	255	140	0
91	darkorange1	#FF7F00	255	127	0
92	darkorange2	#EE7600	238	118	0
93	darkorange3	#CD6600	205	102	0
94	darkorange4	#8B4500	139	69	0
95	darkorchid	#9932CC	153	50	204
96	darkorchid1	#BF3EFF	191	62	255
97	darkorchid2	#B23AEE	178	58	238
98	darkorchid3	#9A32CD	154	50	205
99	darkorchid4	#68228B	104	34	139
100	darkred	#8B0000	139	0	0

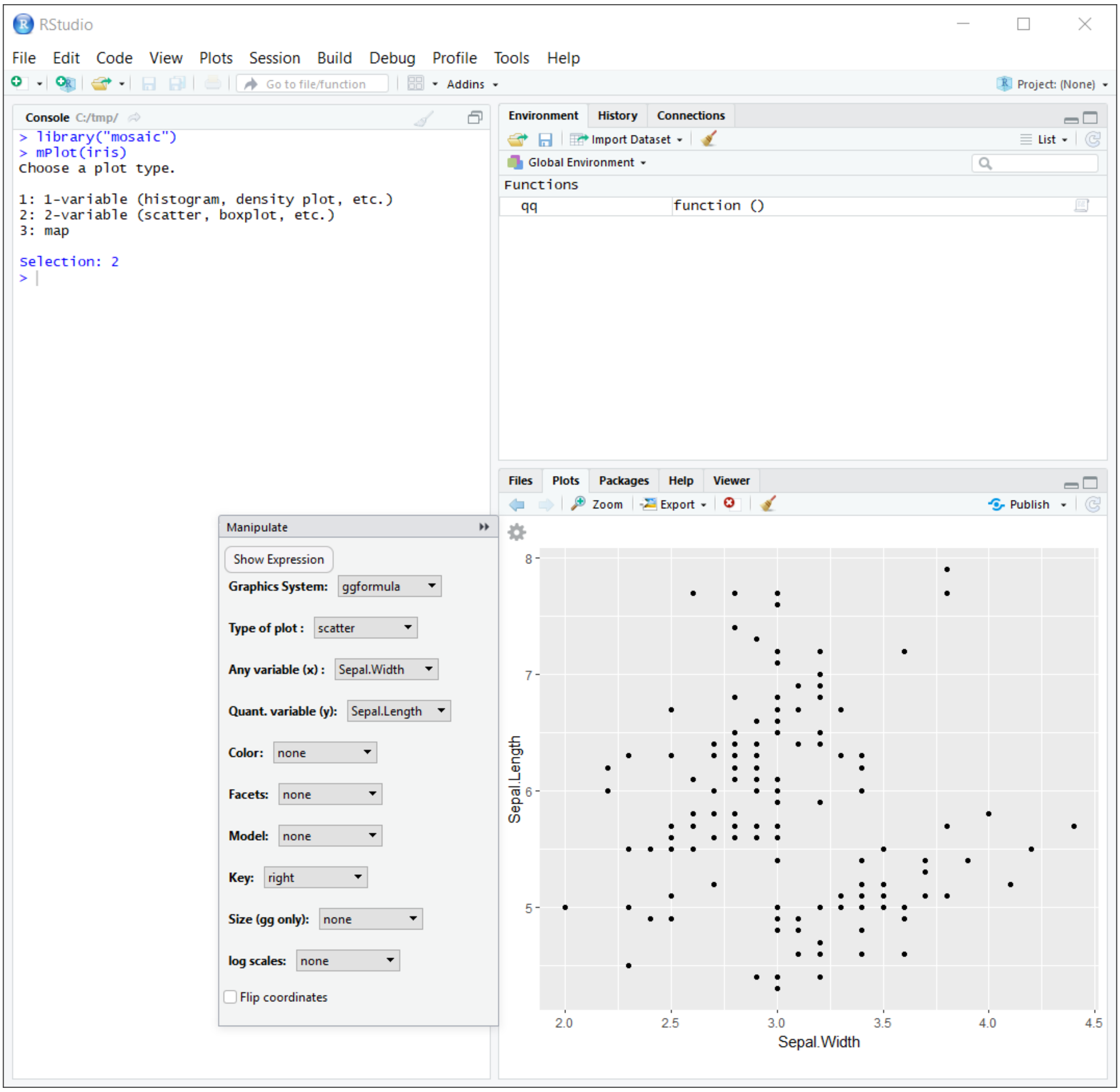
Interfaces d'aide à la construction de graphiques (basés sur ggplot2)

Avant de voir les quelques outils qui permettent de construire des graphiques à l'aide d'IDE, voici un petite cheat sheet bien utile sur ggplot2:



Package "Mosaic"

Si vous êtes un utilisateur R Studio et ggplot2, vous allez probablement adorer le package **mosaic** et sa fonction **mPlot( )** qui permet un peu à la façon de Microsoft Excel et Matlab, de personnaliser des graphiques avec un assistant, et d'en obtenir automatique le code équivalent:



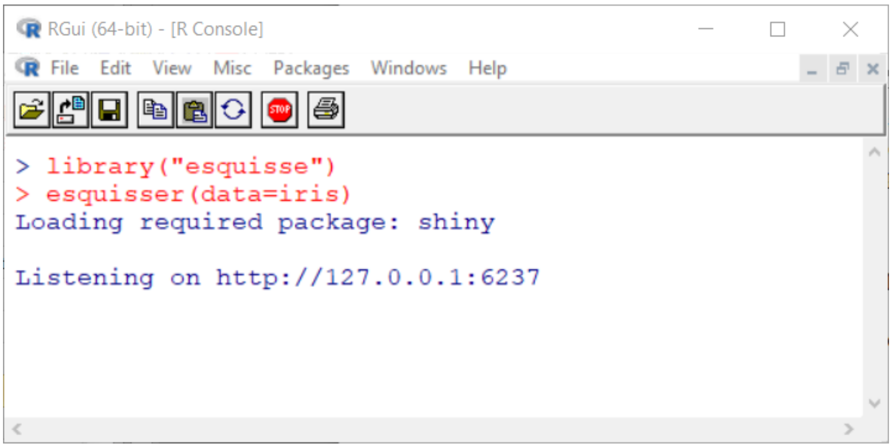
Une fois qu'on connait ce package, il n'y plus qu'à jouer avec l'engrenage visible dans le coin supérieur gauche du graphique et les options qui s'y trouvent!!!

Si jamais on pourrait dire que son son pseudo-équivalent en ligne est:

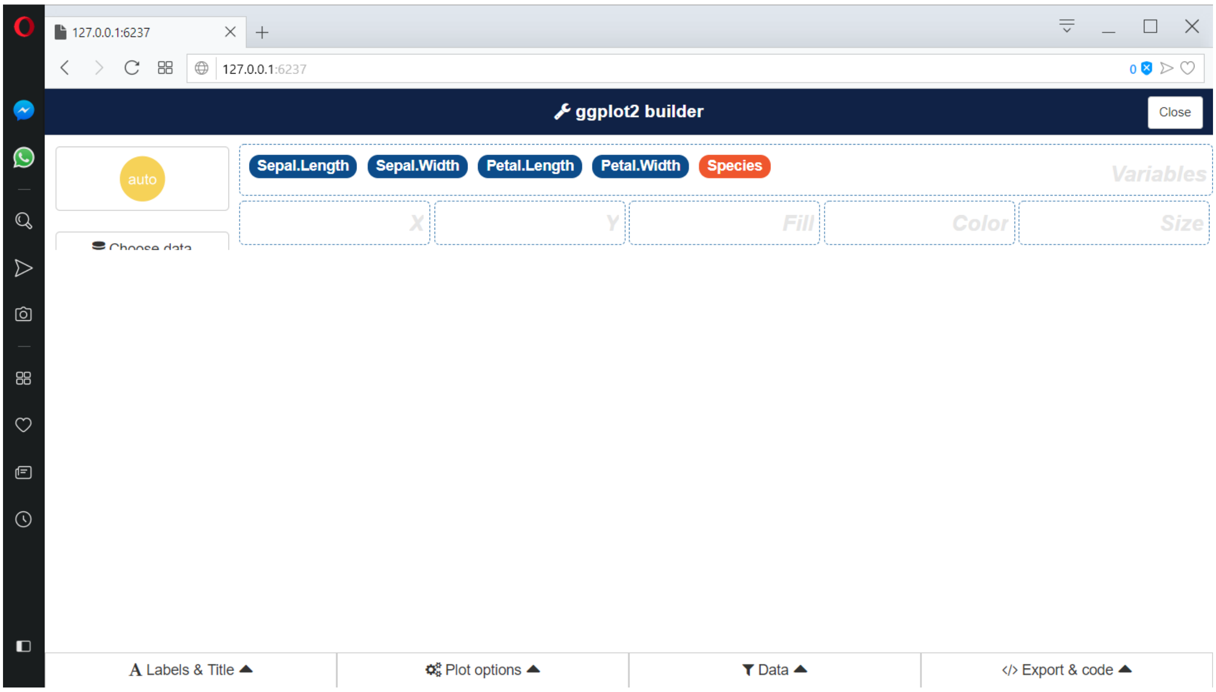
https://shiny.gmw.rug.nl/ggplotgui/

Package "esquisse"

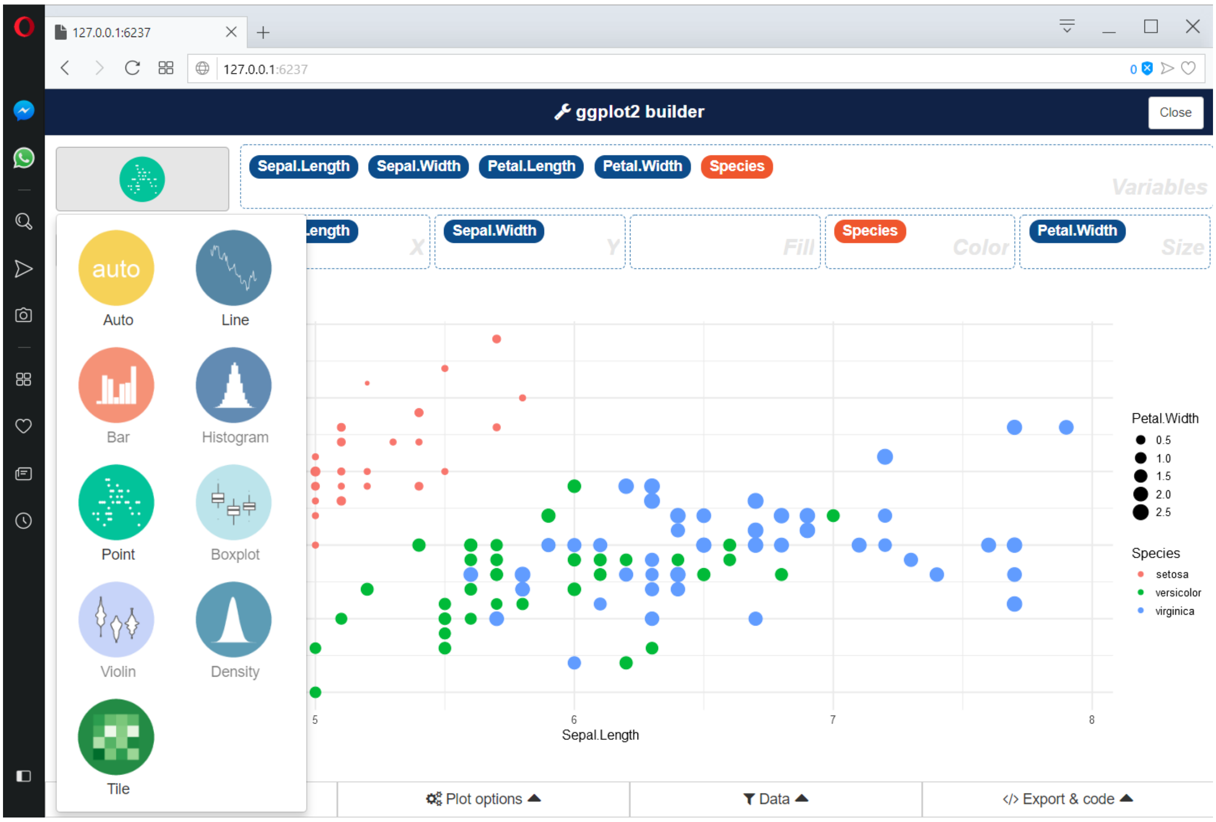
Enfin sachez qu'il existe à ce jour un package nommé **esquisse** (existant depuis 2017) qui permet de construire (à l'aide de RShiny sur Mac, Linux ou Microsoft Windows) ses graphiques visuellement un peu comme dans Microsoft Excel et ce à l'aide de la fonction **esquisse( )**:



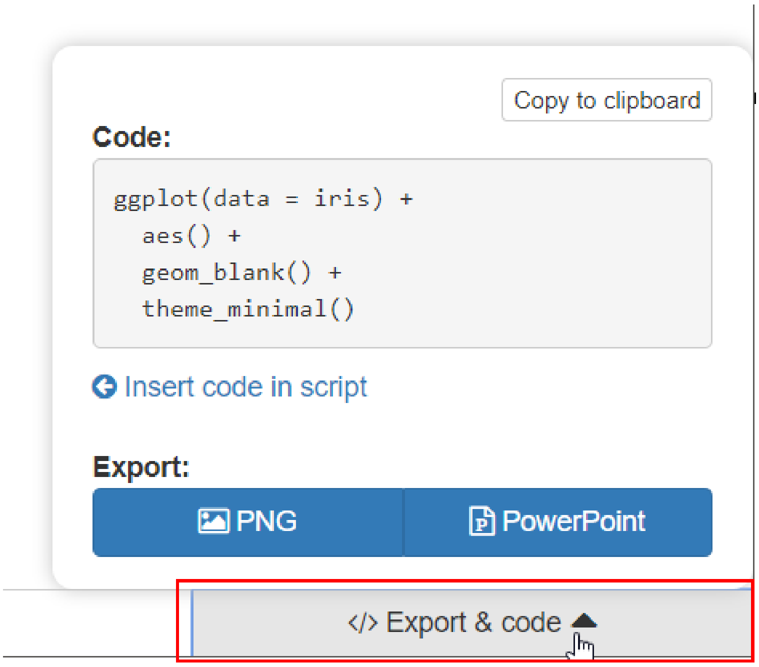
Ce qui va lancer une interface RShiny, intitulée *ggplot2 builder* dans votre navigateur:



Avec un certain nombre d'options très intuitives:



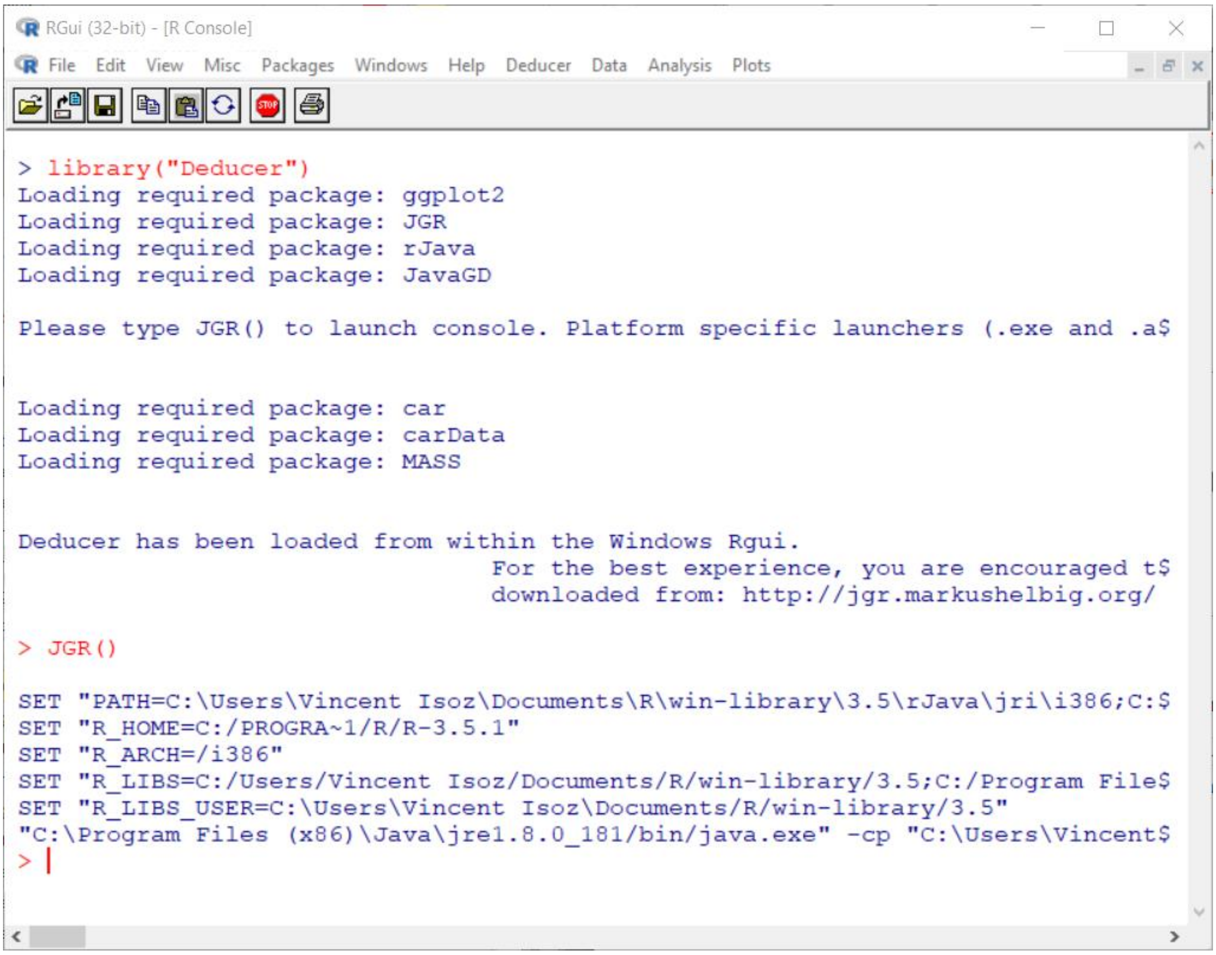
N'espérez toutefois pas, à ce jour (!), toutes les possibilités que l'on peut atteindre en écrivant le script soi-même…! Cependant il est intéressant de noter que cela permet de gagner beaucoup de temps puisqu'on a un bouton d'export (génération) du script qui prémâche une grande partie du travail:



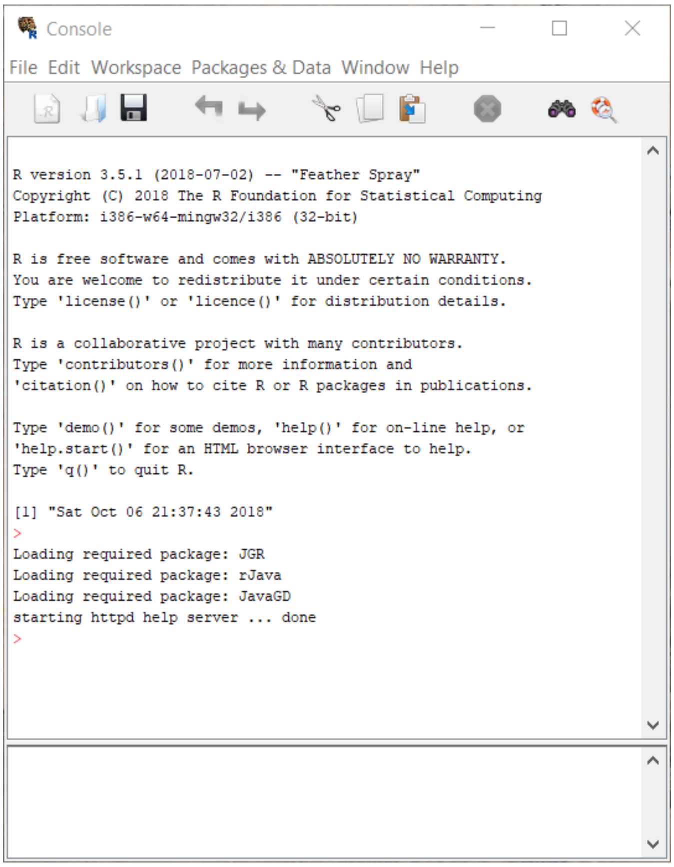
Package "Deducer"

Un autre package aidant à la construction de graphique est le package **Deducer** avec son complément qui est la package **DeducerExtras** (il faut absolument installer ce dernier pour avoir accès aux options des graphiques). Cependant il fonction avec le runtime Java, doit il faudra que ce dernier soit installé sur votre machine…

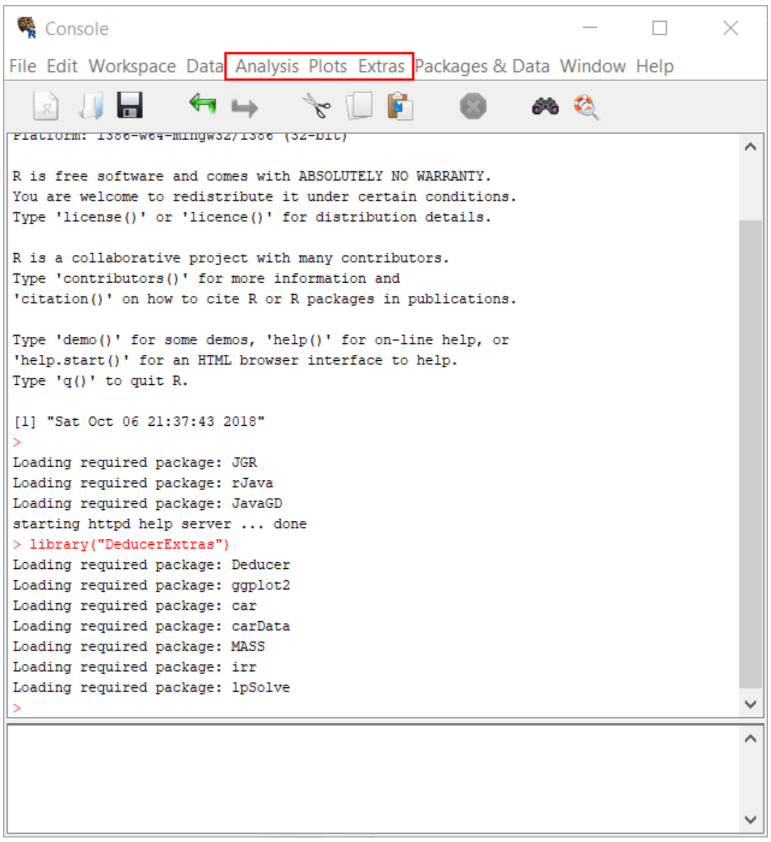
Une fois le package **Deducer** chargé, on lance l'interface Java à l'aide de la commande **JGR( )**:



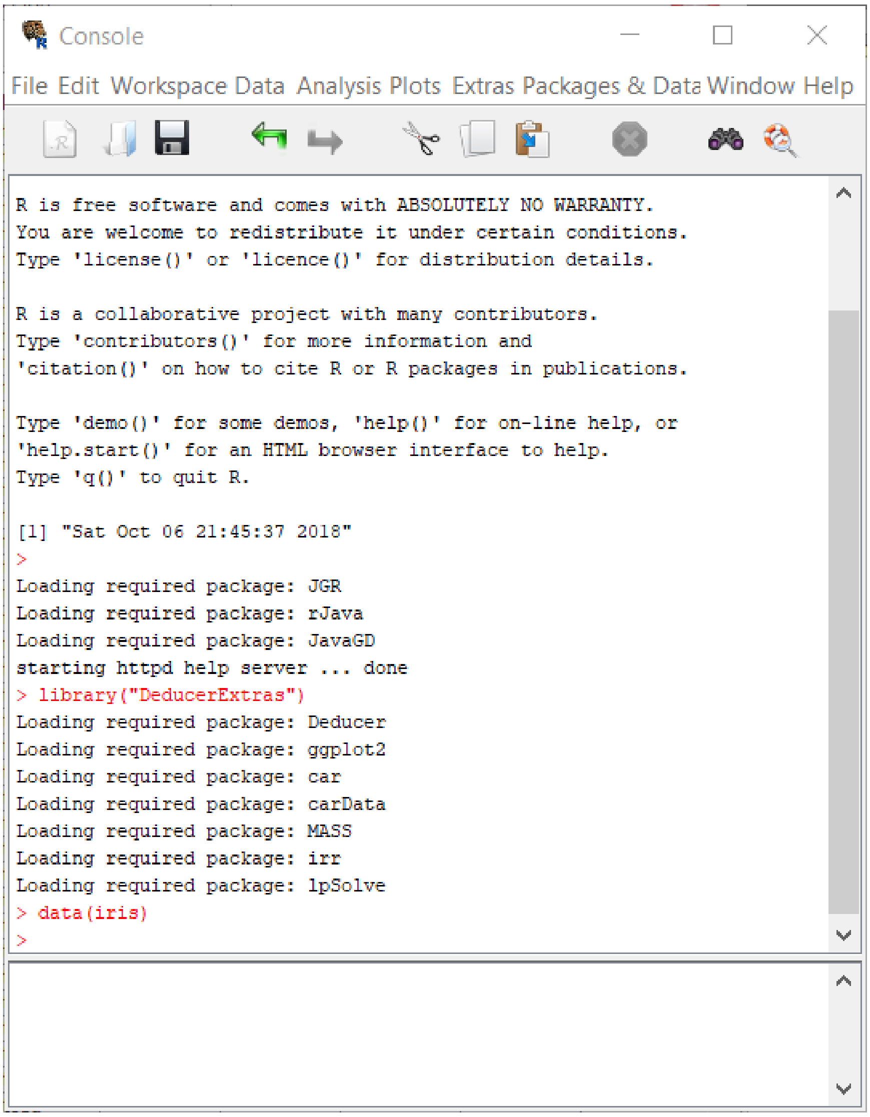
Ce qui ouvrira à l'écran:



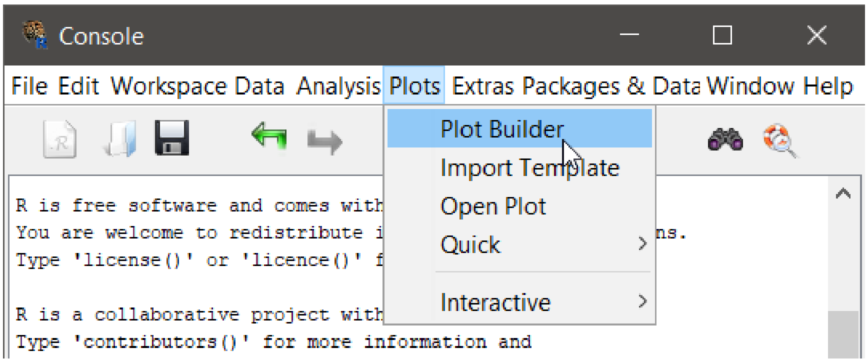
Ensuite, on charge le package **DeducerExtras** (en ayant pris soin de l'installer auparavant évidemment…), ce qui aura pour effet d'ajouter de nouveaux menus à la console comme visible ci-dessous:



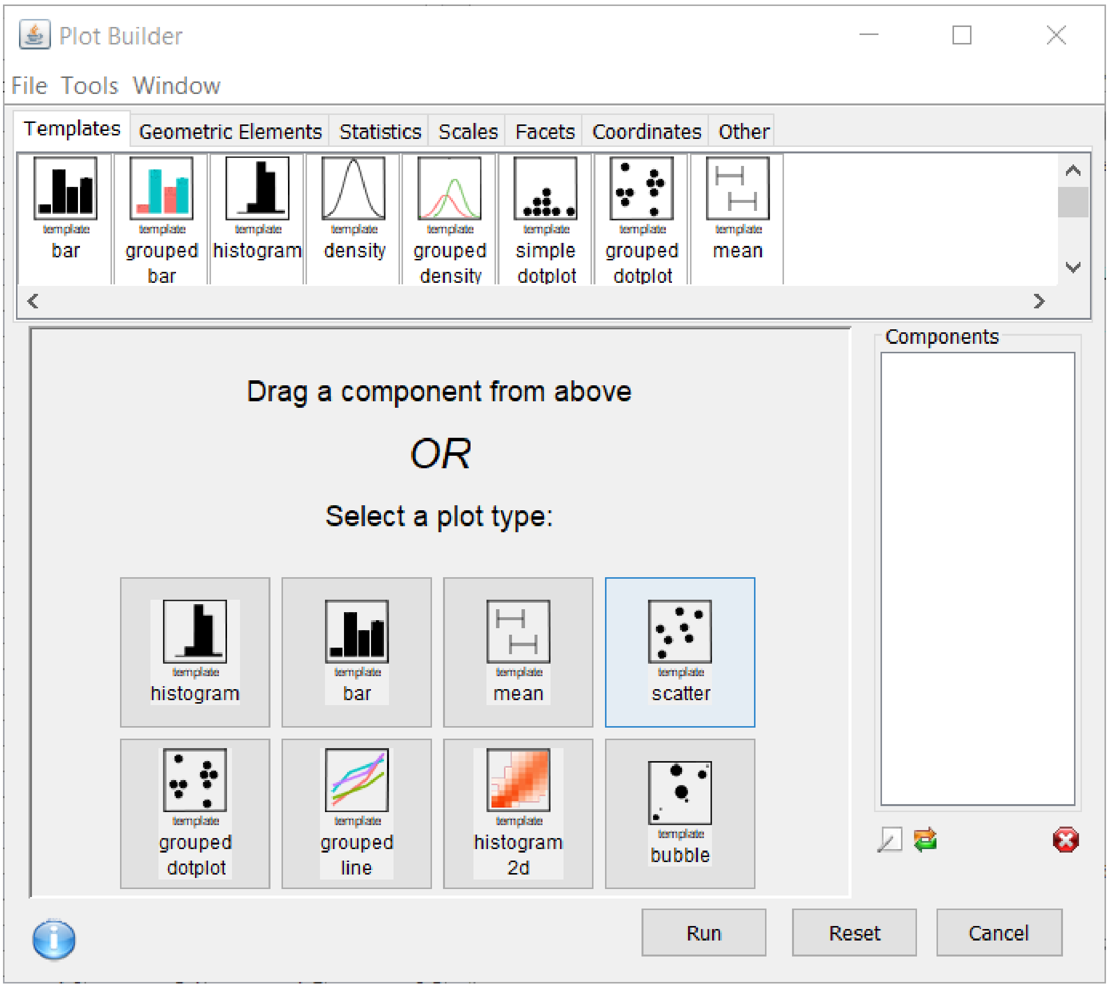
Ensuite on charge un jeu de données comme par exemple (pour faire original…) le jeu *iris*:



Et on va dans le menu **Plots/Plot Builder**:



Ce qui va ouvrir:



Et ensuite y'a plus qu'à…

Pour information cette interface génère aussi le code **ggplot2** dans la console qui est en arrière plan!!!!

Pour ceux qui veulent en savoir plus sur ggplot2, ouvre le livre de Daname Kolani sur le sujet, voici un cheatsheet sur ce package qui le résume assez bien de manière non exhaustive:

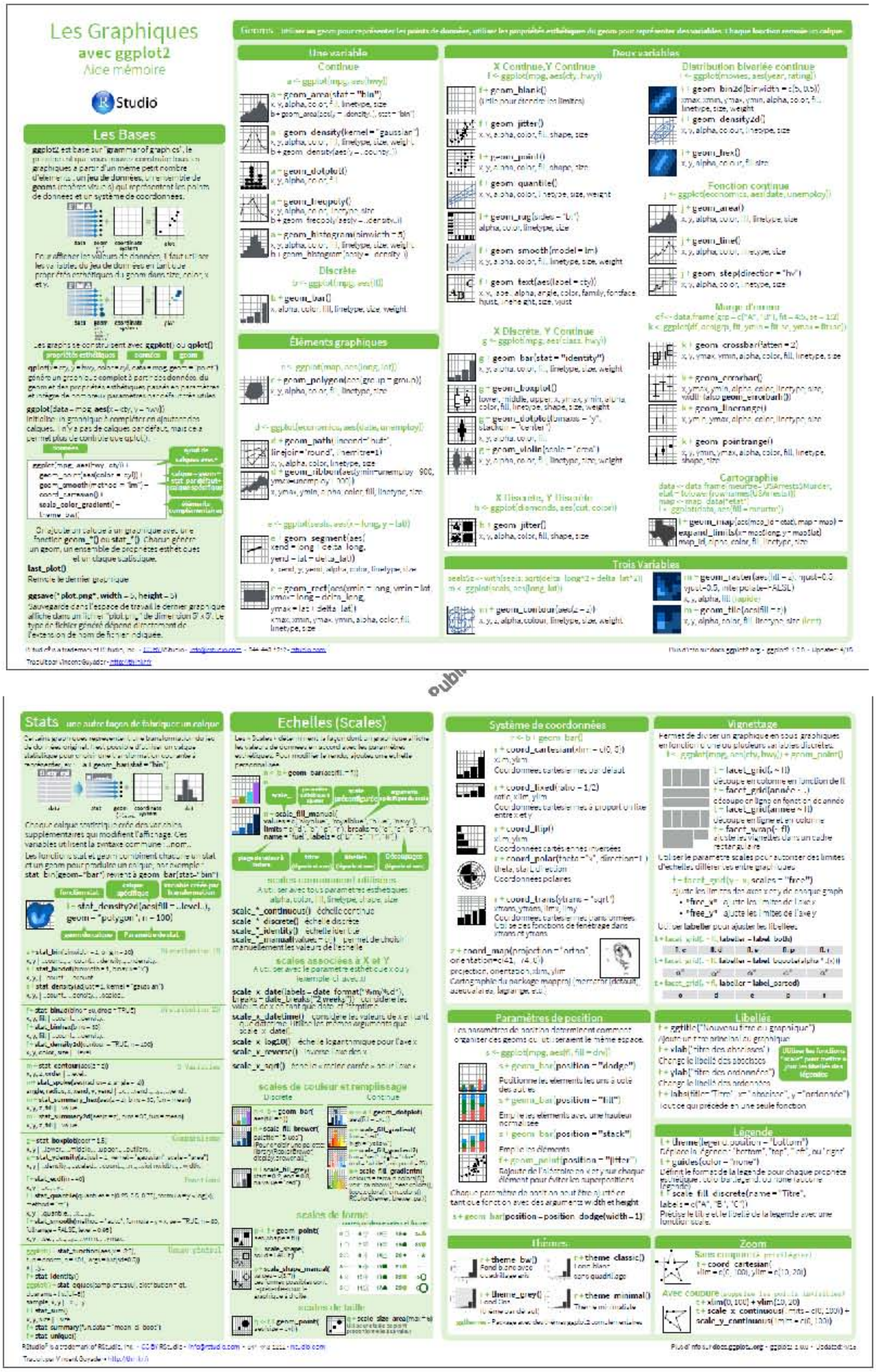
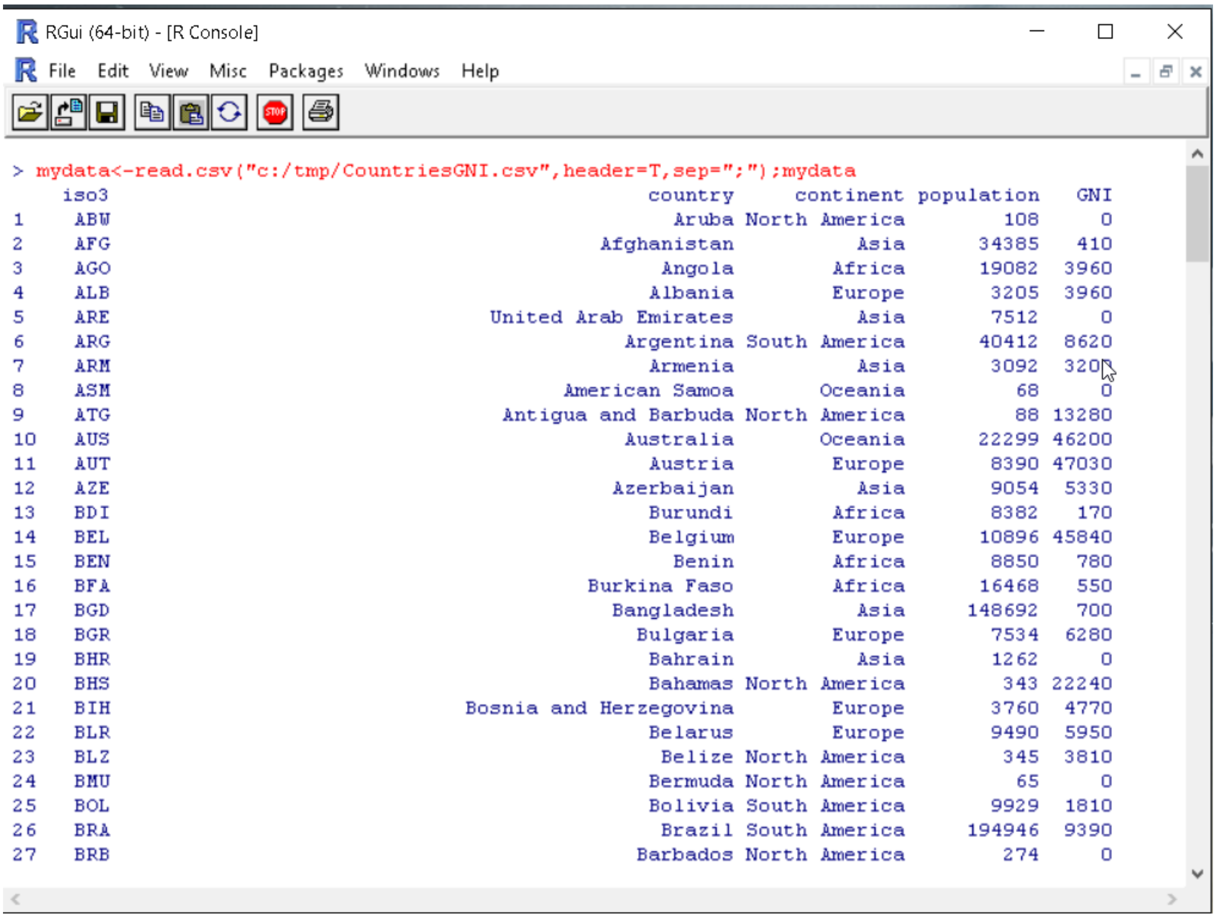


Figure 34 Cheatsheet ggplot2

Exemple 103.: Diagrammes à textes

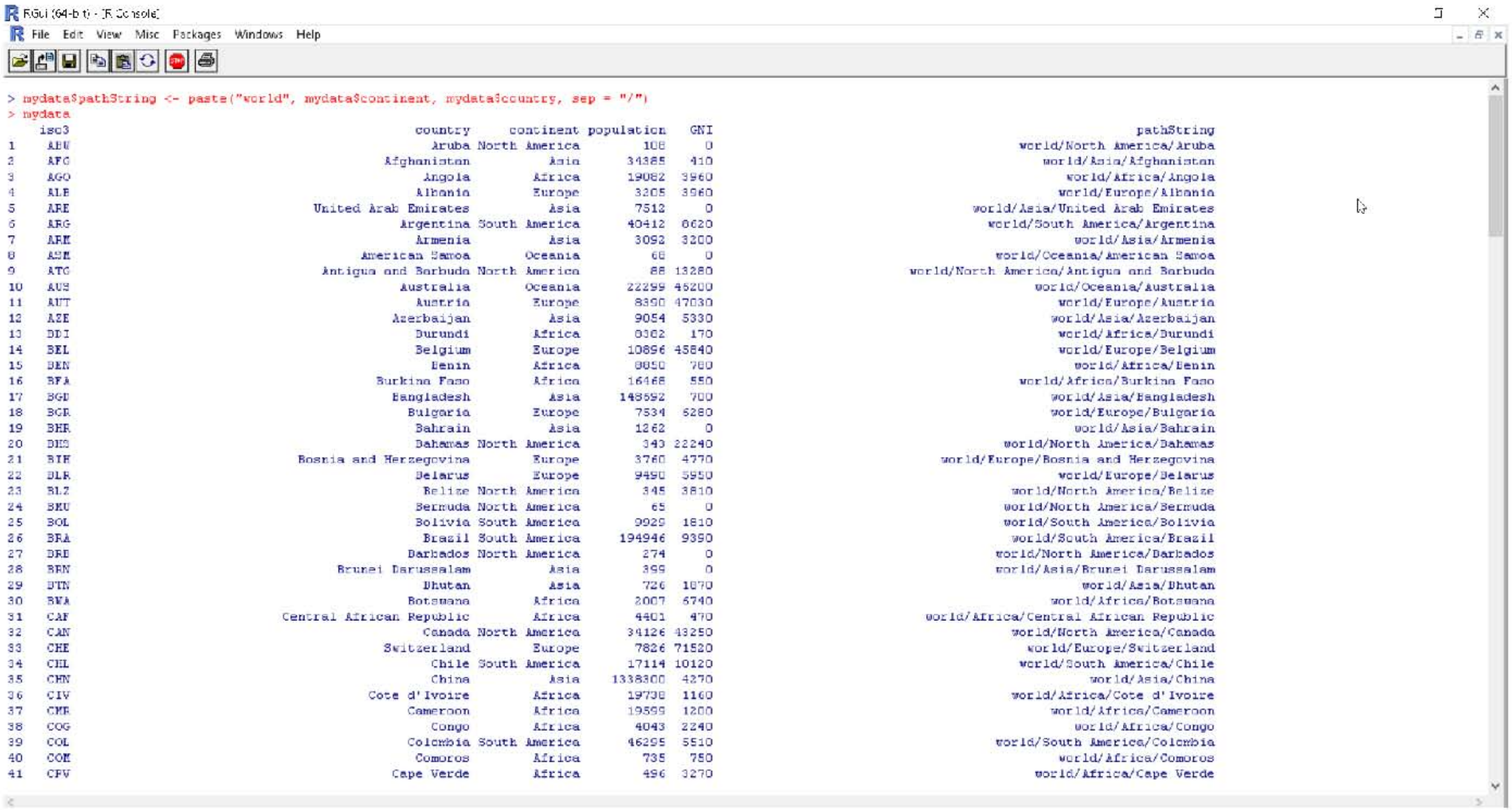
R 3.2.1

Les diagrammes de types textes sont les diagrammes les plus simples et les plus brutes que l'on puisse probablement concevoir. Pour voir de quoi il s'agit nous allons travailler avec le jeu suivant:



```
> mydata<-read.csv("c:/tmp/CountriesGNI.csv",header=T,sep=";");mydata
   iso3                                        country     continent population   GNI
1   ABW                                          Aruba North America        108     0
2   AFG                                    Afghanistan          Asia      34385   410
3   AGO                                         Angola        Africa      19082  3960
4   ALB                                        Albania        Europe       3205  3960
5   ARE                           United Arab Emirates          Asia       7512     0
6   ARG                                      Argentina South America      40412  8620
7   ARM                                        Armenia          Asia       3092  3200
8   ASM                                 American Samoa       Oceania         68     0
9   ATG                            Antigua and Barbuda North America         88 13280
10  AUS                                      Australia       Oceania      22299 46200
11  AUT                                        Austria        Europe       8390 47030
12  AZE                                     Azerbaijan          Asia       9054  5330
13  BDI                                        Burundi        Africa       8382   170
14  BEL                                        Belgium        Europe      10896 45840
15  BEN                                          Benin        Africa       8850   780
16  BFA                                   Burkina Faso        Africa      16468   550
17  BGD                                     Bangladesh          Asia     148692   700
18  BGR                                       Bulgaria        Europe       7534  6280
19  BHR                                        Bahrain          Asia       1262     0
20  BHS                                        Bahamas North America        343 22240
21  BIH                         Bosnia and Herzegovina        Europe       3760  4770
22  BLR                                        Belarus        Europe       9490  5950
23  BLZ                                         Belize North America        345  3810
24  BMU                                        Bermuda North America         65     0
25  BOL                                        Bolivia South America       9929  1810
26  BRA                                         Brazil South America     194946  9390
27  BRB                                       Barbados North America        274     0
```

D'abord, nous allons structurer cela sous forme d'arbre en construisant une colonne concaténée:

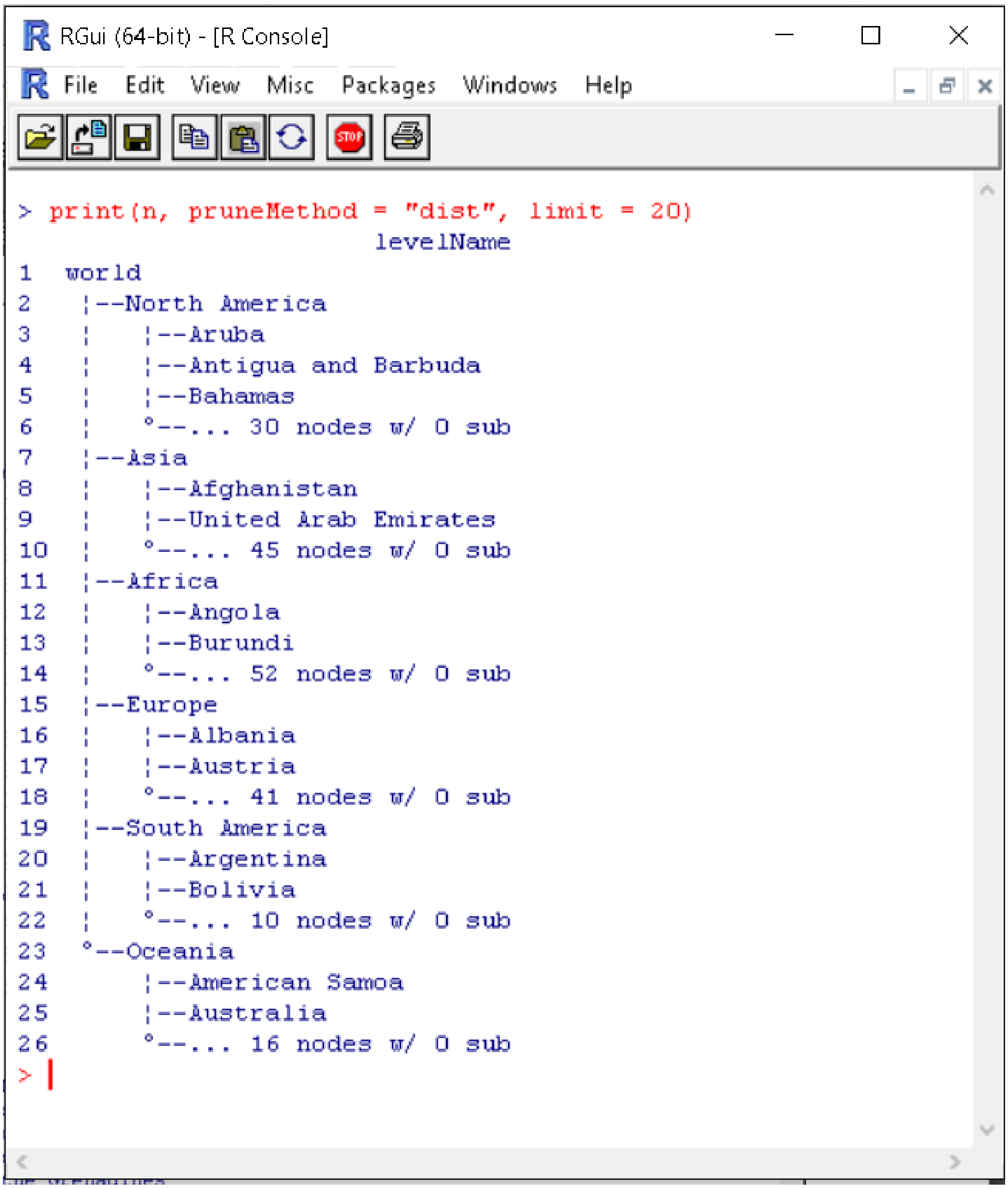

```
> mydata$pathString <- paste("world", mydata$continent, mydata$country, sep = "/")
> mydata
   iso3                          country     continent population   GNI                                pathString
1   ABW                            Aruba North America        108     0                 world/North America/Aruba
2   AFG                      Afghanistan          Asia      34385   410                  world/Asia/Afghanistan
3   AGO                           Angola        Africa      19082  3960                      world/Africa/Angola
4   ALB                          Albania        Europe       3205  3960                     world/Europe/Albania
5   ARE             United Arab Emirates          Asia       7512     0          world/Asia/United Arab Emirates
6   ARG                        Argentina South America      40412  8620           world/South America/Argentina
7   ARM                          Armenia          Asia       3092  3200                      world/Asia/Armenia
8   ASM                   American Samoa       Oceania         68     0           world/Oceania/American Samoa
9   ATG              Antigua and Barbuda North America         88 13280 world/North America/Antigua and Barbuda
10  AUS                        Australia       Oceania      22299 46200                  world/Oceania/Australia
11  AUT                          Austria        Europe       8390 47030                     world/Europe/Austria
12  AZE                       Azerbaijan          Asia       9054  5330                   world/Asia/Azerbaijan
13  BDI                          Burundi        Africa       8382   170                     world/Africa/Burundi
14  BEL                          Belgium        Europe      10896 45840                     world/Europe/Belgium
15  BEN                            Benin        Africa       8850   780                       world/Africa/Benin
16  BFA                     Burkina Faso        Africa      16468   550                world/Africa/Burkina Faso
17  BGD                       Bangladesh          Asia     148692   700                   world/Asia/Bangladesh
18  BGR                         Bulgaria        Europe       7534  6280                    world/Europe/Bulgaria
19  BHR                          Bahrain          Asia       1262     0                      world/Asia/Bahrain
20  BHS                          Bahamas North America        343 22240               world/North America/Bahamas
21  BIH           Bosnia and Herzegovina        Europe       3760  4770      world/Europe/Bosnia and Herzegovina
22  BLR                          Belarus        Europe       9490  5950                     world/Europe/Belarus
23  BLZ                           Belize North America        345  3810                world/North America/Belize
24  BMU                          Bermuda North America         65     0               world/North America/Bermuda
25  BOL                          Bolivia South America       9929  1810               world/South America/Bolivia
26  BRA                           Brazil South America     194946  9390                world/South America/Brazil
27  BRB                         Barbados North America        274     0              world/North America/Barbados
28  BRN                Brunei Darussalam          Asia        399     0            world/Asia/Brunei Darussalam
29  BTN                           Bhutan          Asia        726  1870                       world/Asia/Bhutan
30  BWA                         Botswana        Africa       2007  6740                    world/Africa/Botswana
31  CAF         Central African Republic        Africa       4401   470    world/Africa/Central African Republic
32  CAN                           Canada North America      34126 43250                world/North America/Canada
33  CHE                      Switzerland        Europe       7826 71520                 world/Europe/Switzerland
34  CHL                            Chile South America      17114 10120                 world/South America/Chile
35  CHN                            China          Asia    1338300  4270                        world/Asia/China
36  CIV                    Cote d'Ivoire        Africa      19738  1160               world/Africa/Cote d'Ivoire
37  CMR                         Cameroon        Africa      19599  1200                    world/Africa/Cameroon
38  COG                            Congo        Africa       4043  2240                       world/Africa/Congo
39  COL                         Colombia South America      46295  5510              world/South America/Colombia
40  COM                          Comoros        Africa        735   750                     world/Africa/Comoros
41  CPV                       Cape Verde        Africa        496  3270                  world/Africa/Cape Verde
```

Ensuite avec la fonction **as.Node( )** du package **data.tree** nous construisons un arbre:

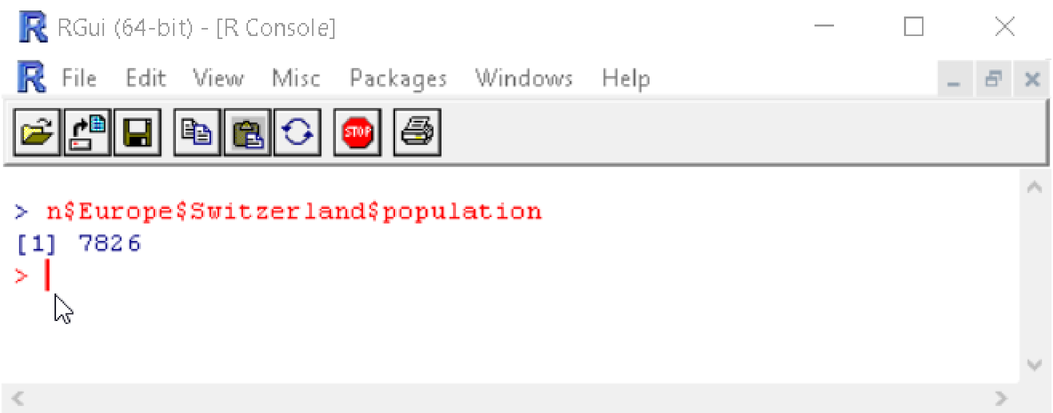
```
> library(data.tree)
> n<-as.Node(mydata[,])
> n
                                            levelName
1   world
2    ¦--North America
3    ¦   ¦--Aruba
4    ¦   ¦--Antigua and Barbuda
5    ¦   ¦--Bahamas
6    ¦   ¦--Belize
7    ¦   ¦--Bermuda
8    ¦   ¦--Barbados
9    ¦   ¦--Canada
10   ¦   ¦--Costa Rica
11   ¦   ¦--Cuba
12   ¦   ¦--Curacao
13   ¦   ¦--Cayman Islands
14   ¦   ¦--Dominica
15   ¦   ¦--Dominican Republic
16   ¦   ¦--Grenada
17   ¦   ¦--Greenland
18   ¦   ¦--Guatemala
19   ¦   ¦--Honduras
20   ¦   ¦--Haiti
21   ¦   ¦--Jamaica
22   ¦   ¦--Saint Kitts and Nevis
23   ¦   ¦--Saint Lucia
24   ¦   ¦--Saint Martin
25   ¦   ¦--Mexico
26   ¦   ¦--Nicaragua
27   ¦   ¦--Panama
28   ¦   ¦--Puerto Rico
29   ¦   ¦--El Salvador
30   ¦   ¦--Sint Maarten (Dutch part)
31   ¦   ¦--Turks and Caicos Islands
32   ¦   ¦--Trinidad and Tobago
33   ¦   ¦--United States of America
34   ¦   ¦--Saint Vincent and the Grenadines
35   ¦   °--United States Virgin Islands
36   ¦--Asia
37   ¦   ¦--Afghanistan
```

Figure 35 Graphique d'arborescence textuelle

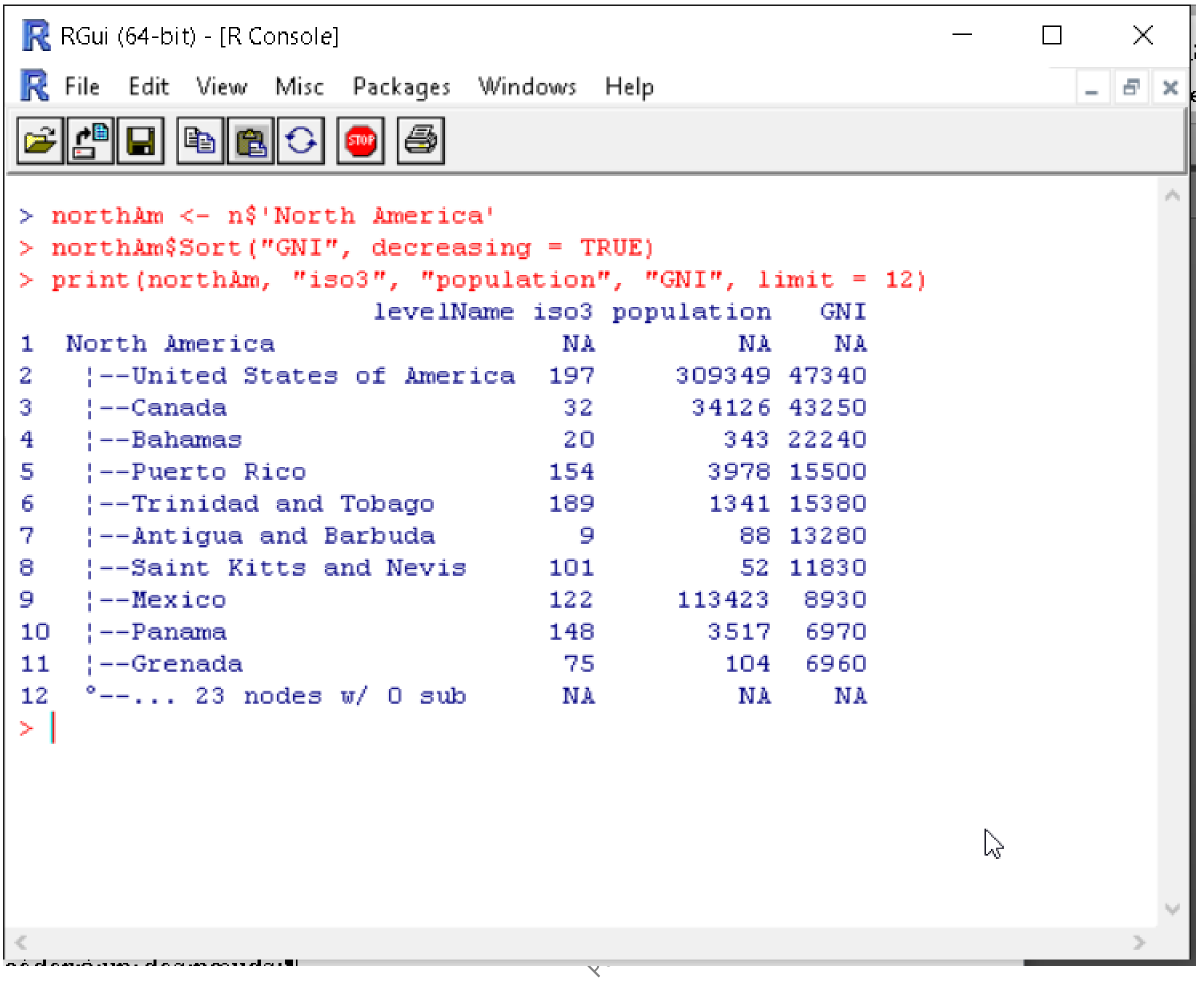
Et nous allons limiter la taille de l'arbre en prenant une distance alphabétique:



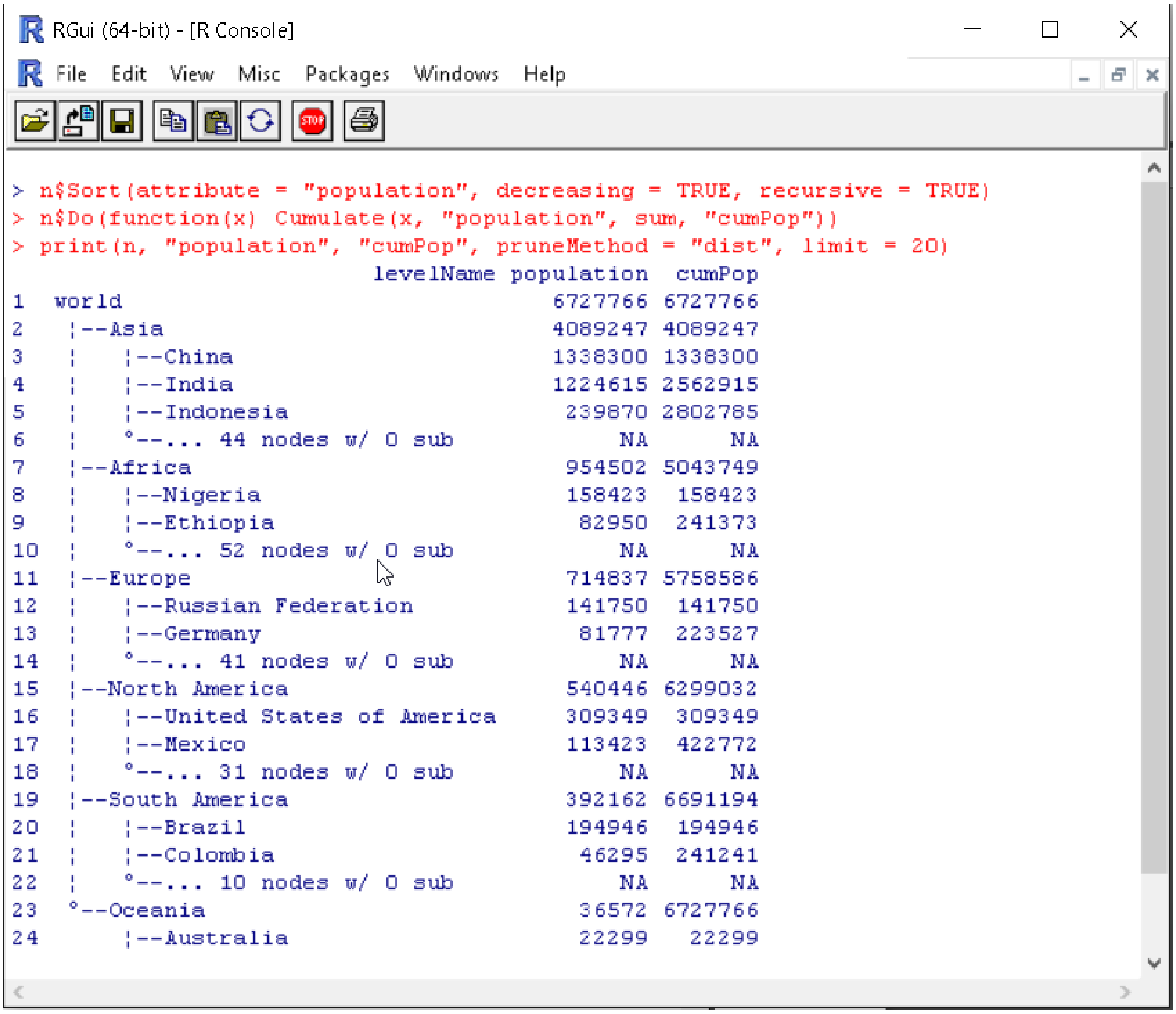
Nous pouvons directement accéder à un des nœuds:



On peut jouer à ajouter des statistiques (on reconnaîtra une structure assez ressemblante aux tableaux croisés dynamiques dans les tableurs):



Nous pouvons aussi agréger des sous-totaux et trier les résultats:

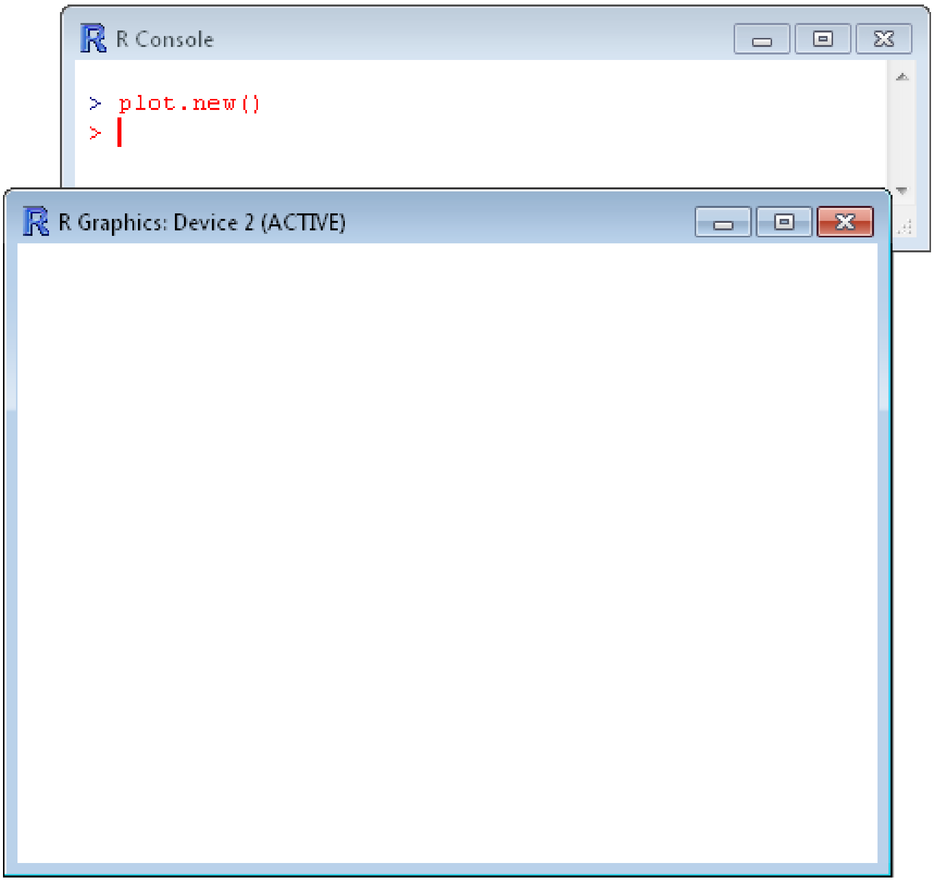


Sinon pour ce qui est d'afficher le résultat d'une analyse statistique en tant que texte dans une fenêtre graphique, nous renvoyons le lecteur à la page 378.

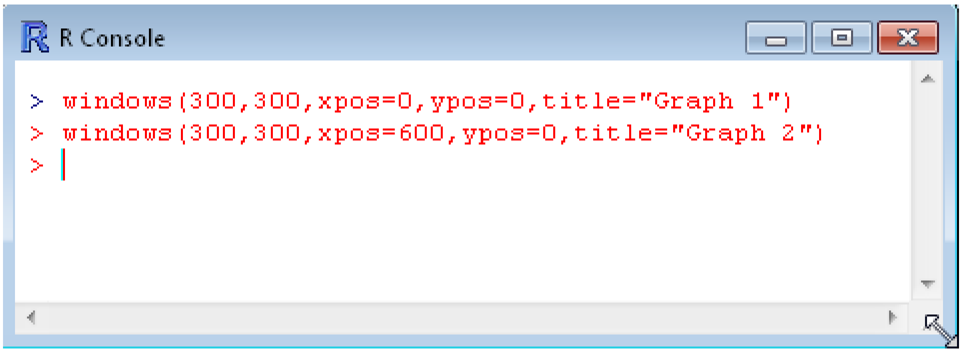
Exemple 104.: Contrôler la fenêtre graphique

R 3.2.1

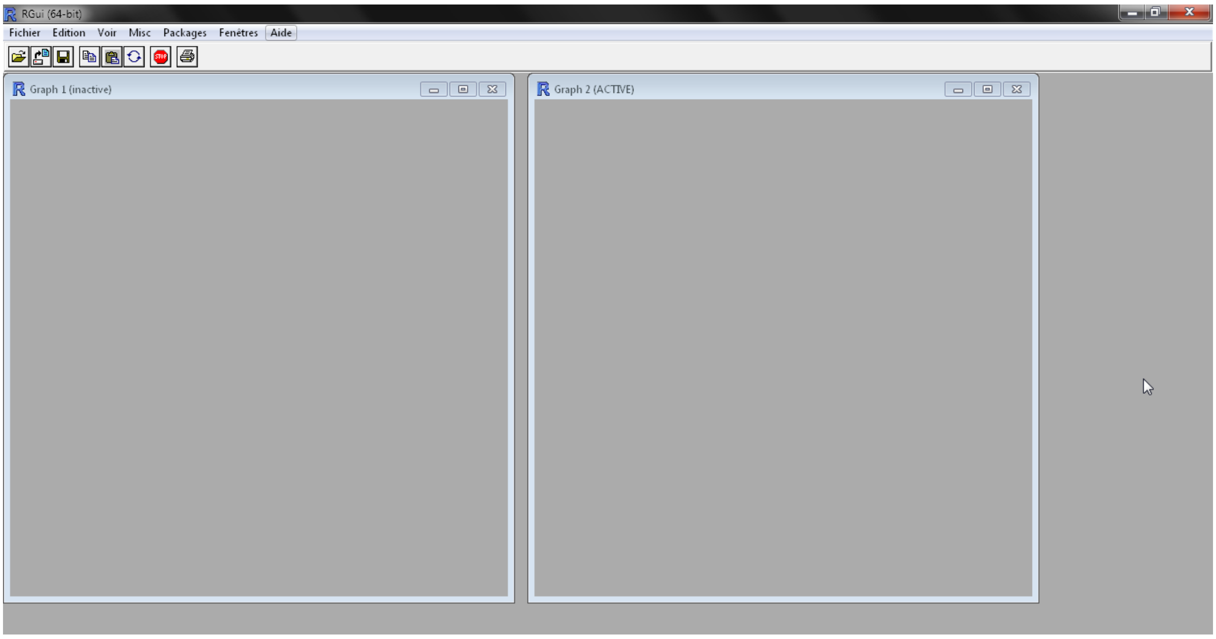
Nous pouvons ouvrir aussi très rapidement une fenêtre toute vide avec la commande **plot.new( )** (cette commande efface le graphique actif en cours!!!)



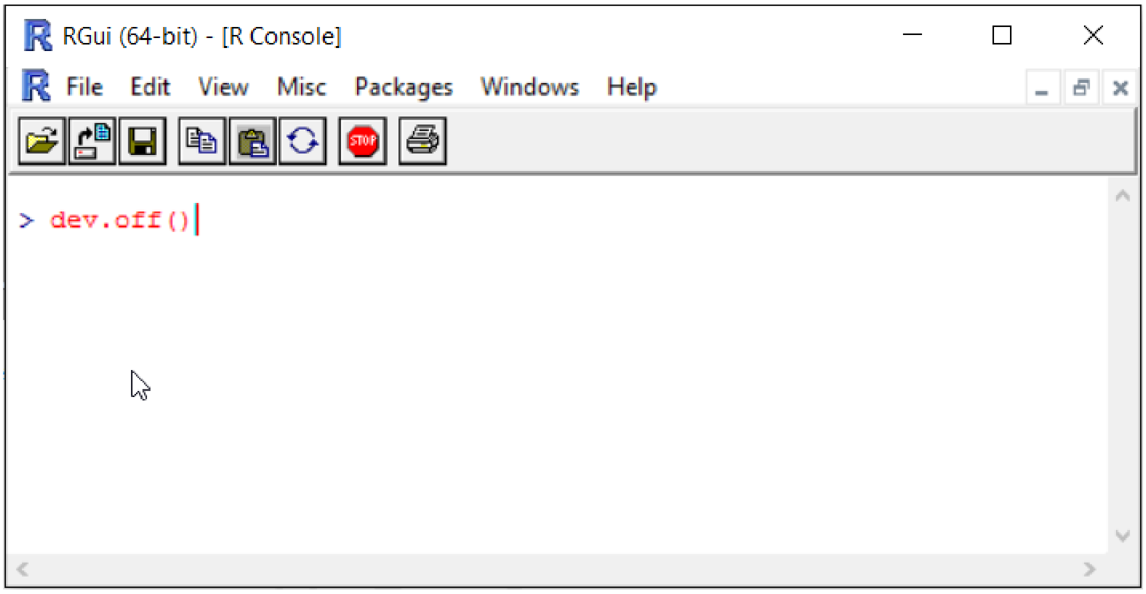
Mais il existe une autre méthode beaucoup plus intéressant et que nous retrouvons dans des cas pratiques relativement complexes qui consistent à pouvoir ouvrir autant de fenêtre graphique que l'on désire avec un nom et une taille au choix:



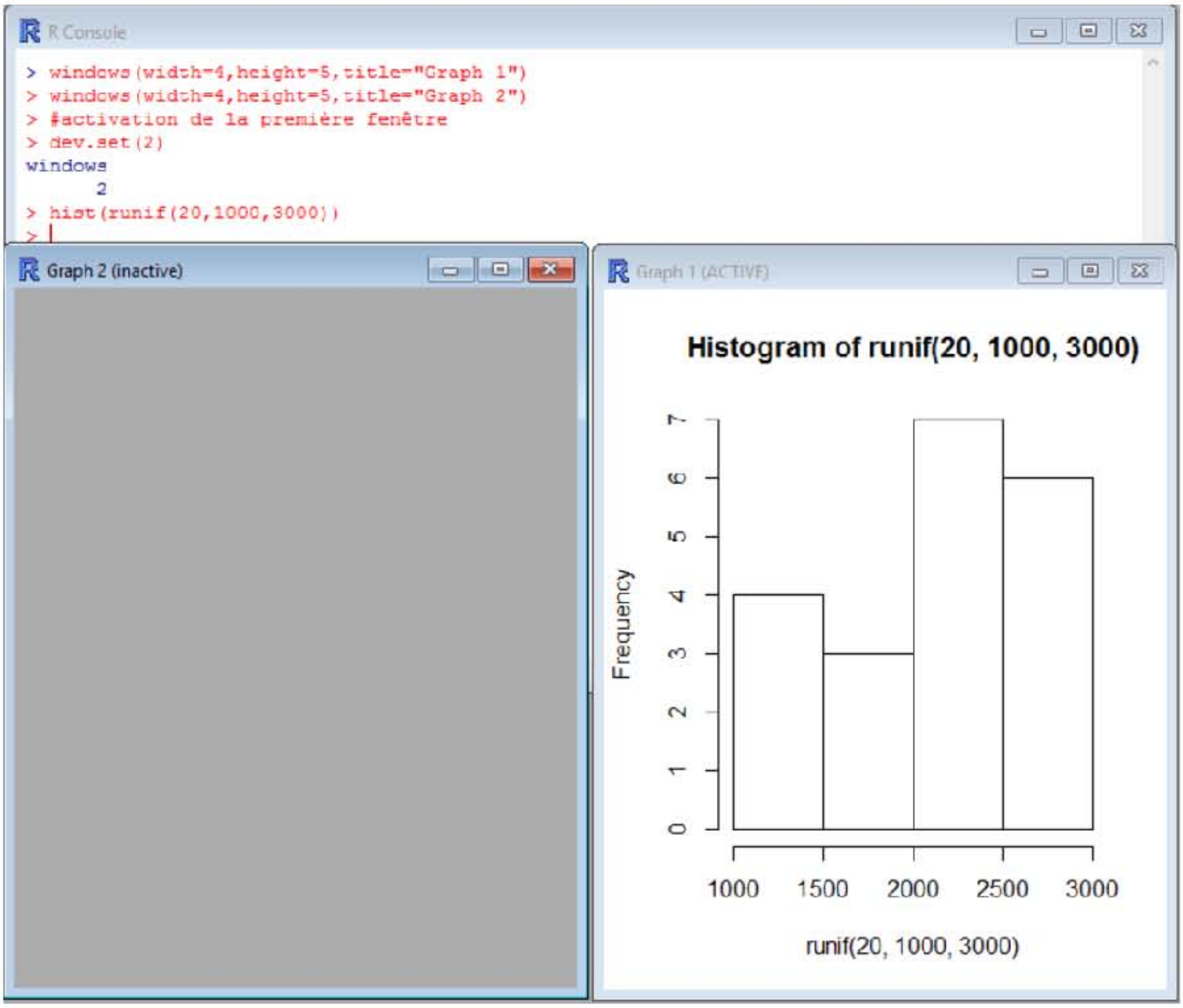
Ce qui donnera:



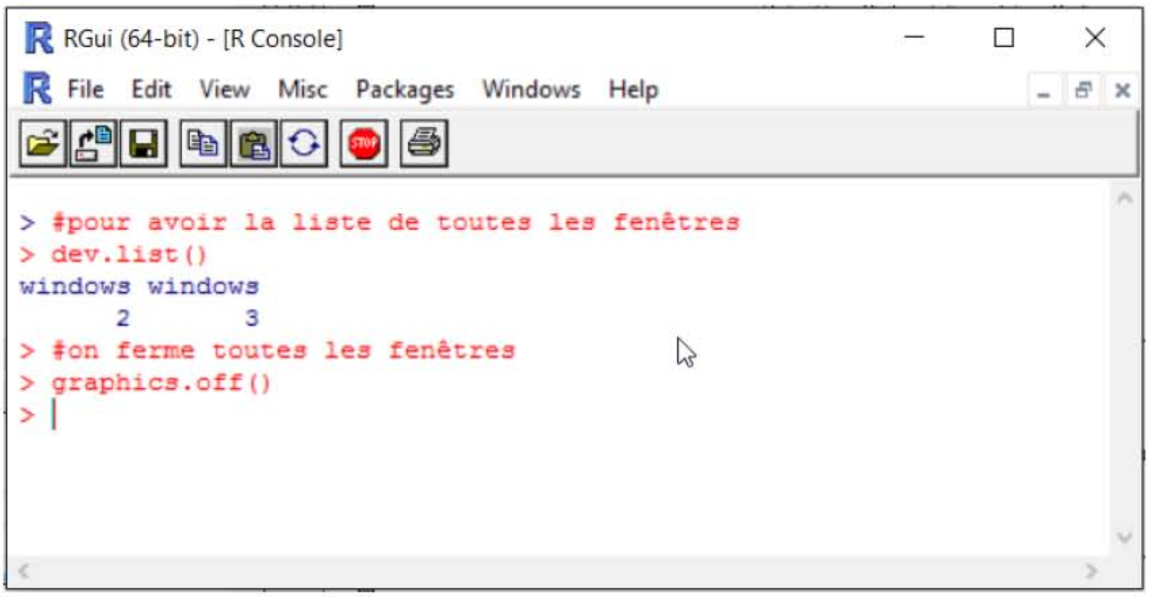
et on ferme la fenêtre active:



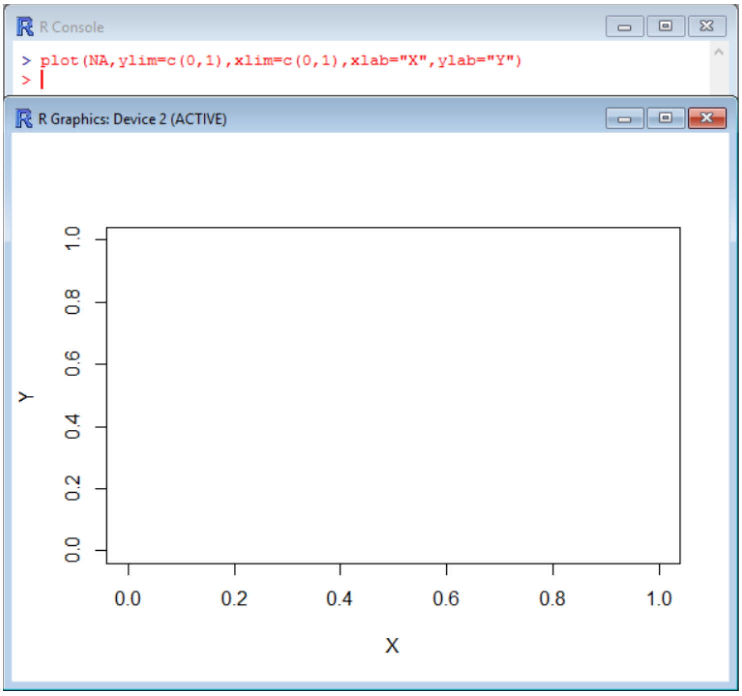
Voici un autre exemple:



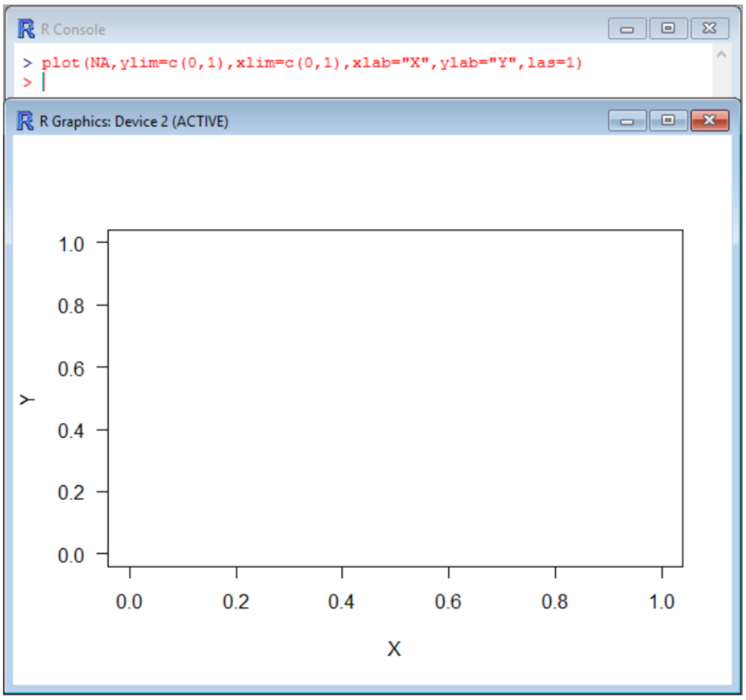
et:



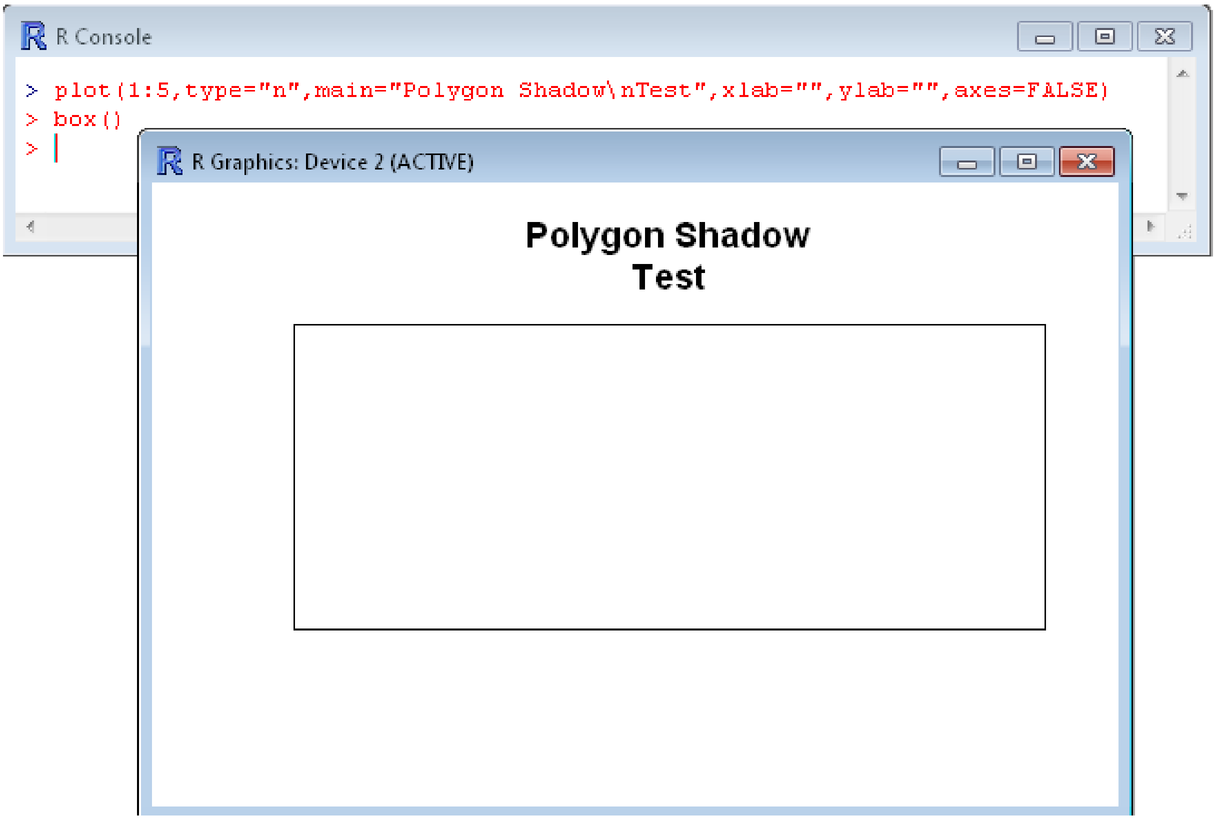
Si nous faisons un simple plot vide avec la fonction **plot( )** :



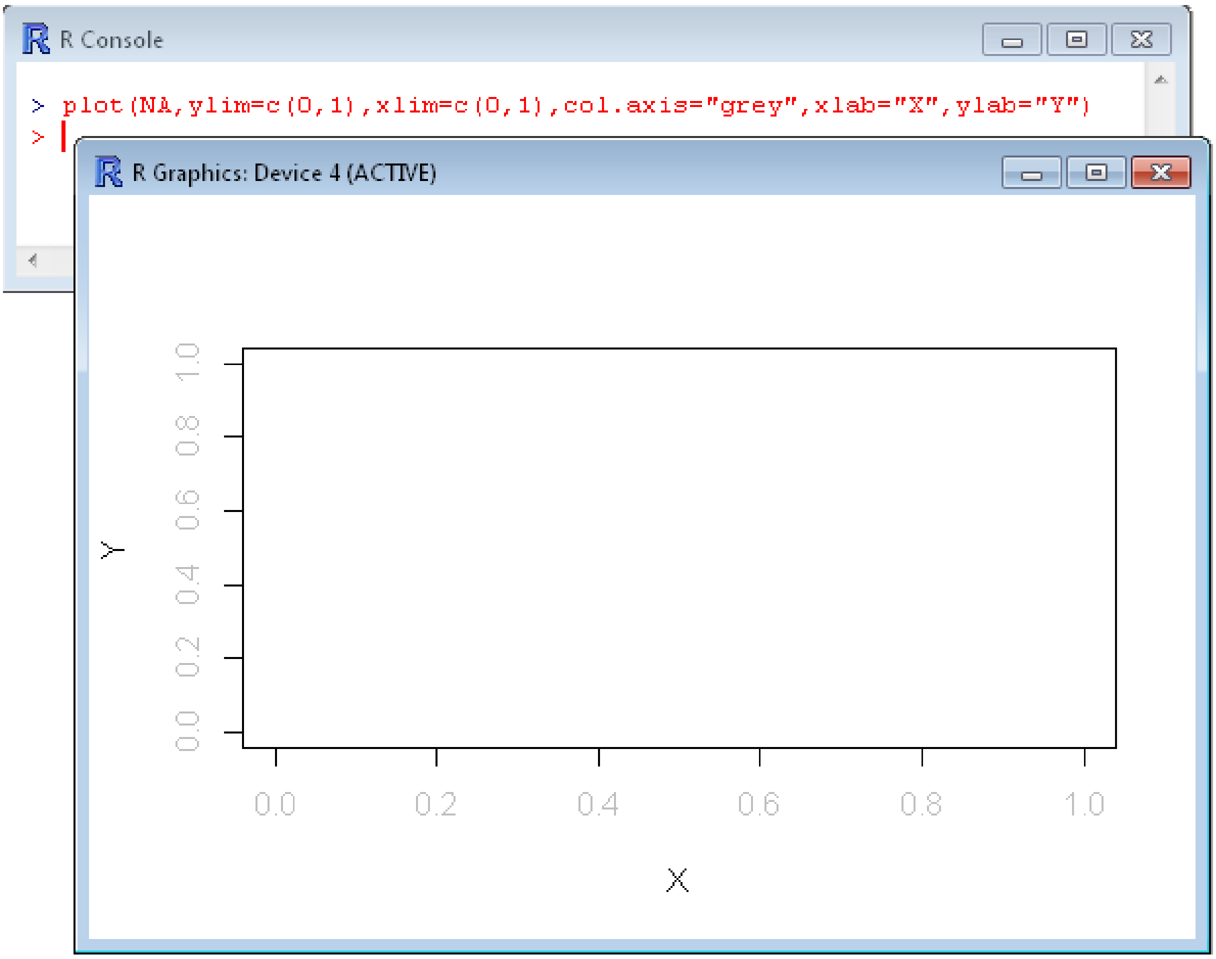
Au passage, le paramètre pour mettre les nombres de l'axe des ordonnées à l'horizontale est *las*:



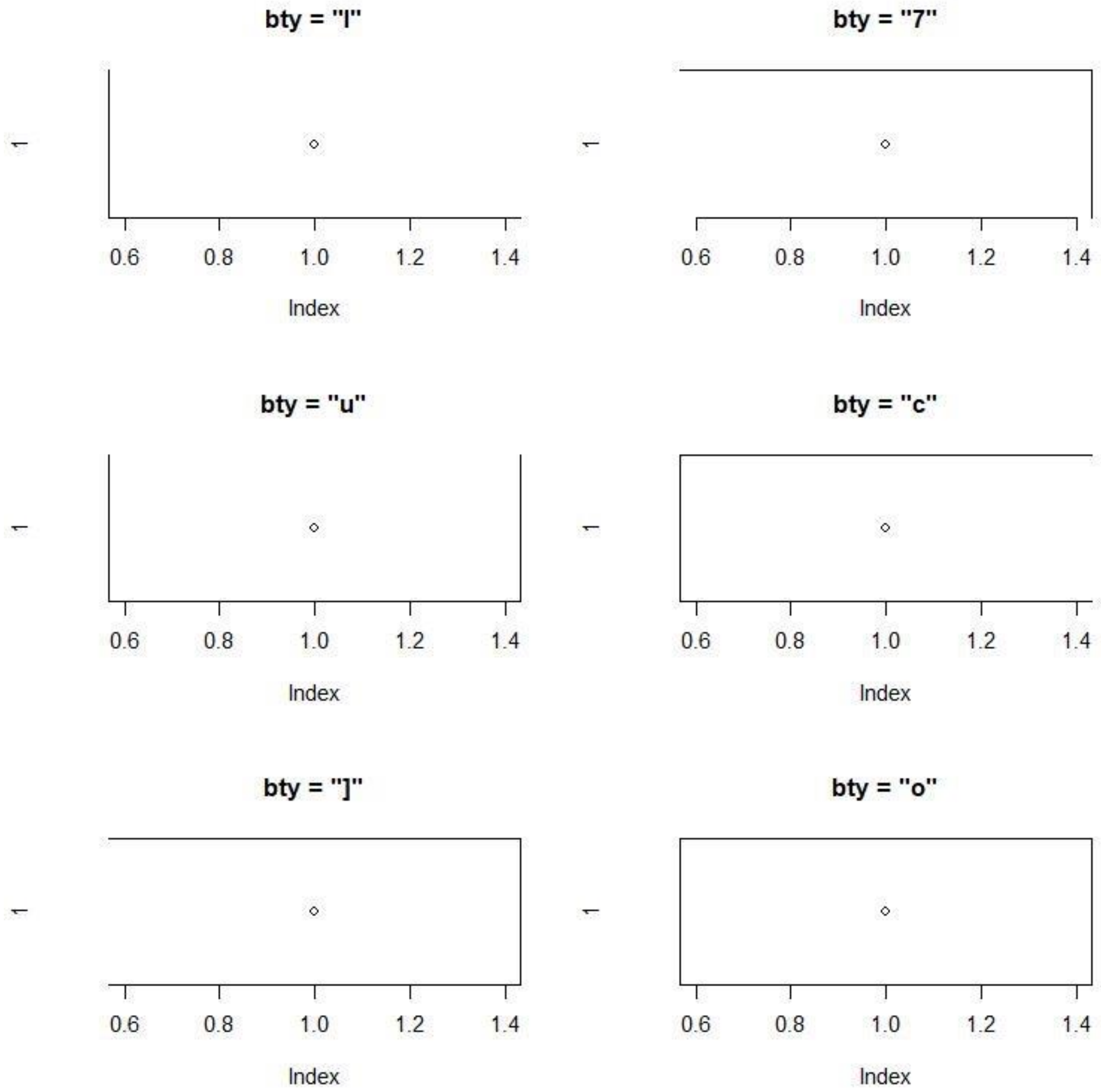
et si nous voulons la zone de traçage mais sans les axes (remarquez au passage la méthode pour faire un retour à la ligne dans une légende quelconque en utilisant le **\n**):



ou avec des axes dans un couleur plus claire que les axes:



On peut choisir différentes mantières d'agencer les axes avec le paramètre *bty* de la fonction **plot( )** comme l'illustra la figure suivante:



Au niveau de la grille, nous avons avec la fonction **grid( )** :

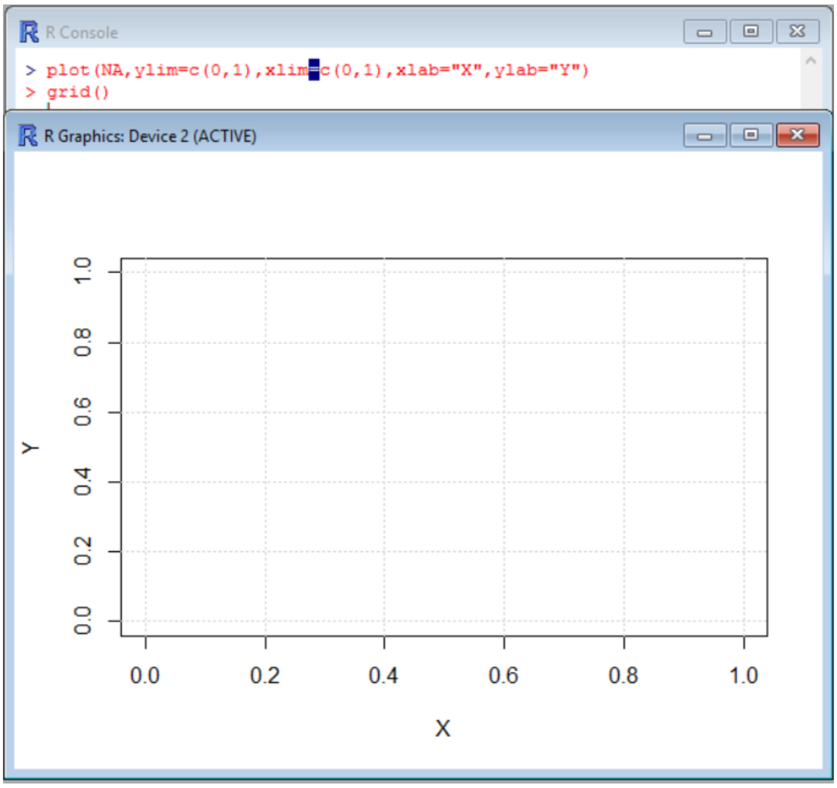
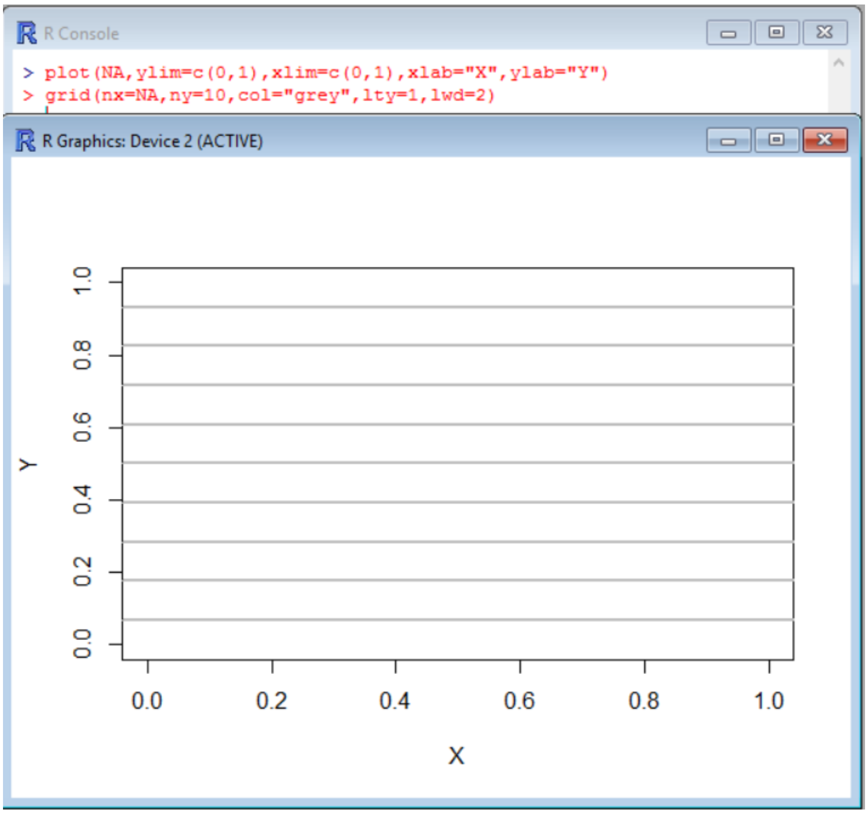


Figure 36 Graph avec option d'affichage de la grille

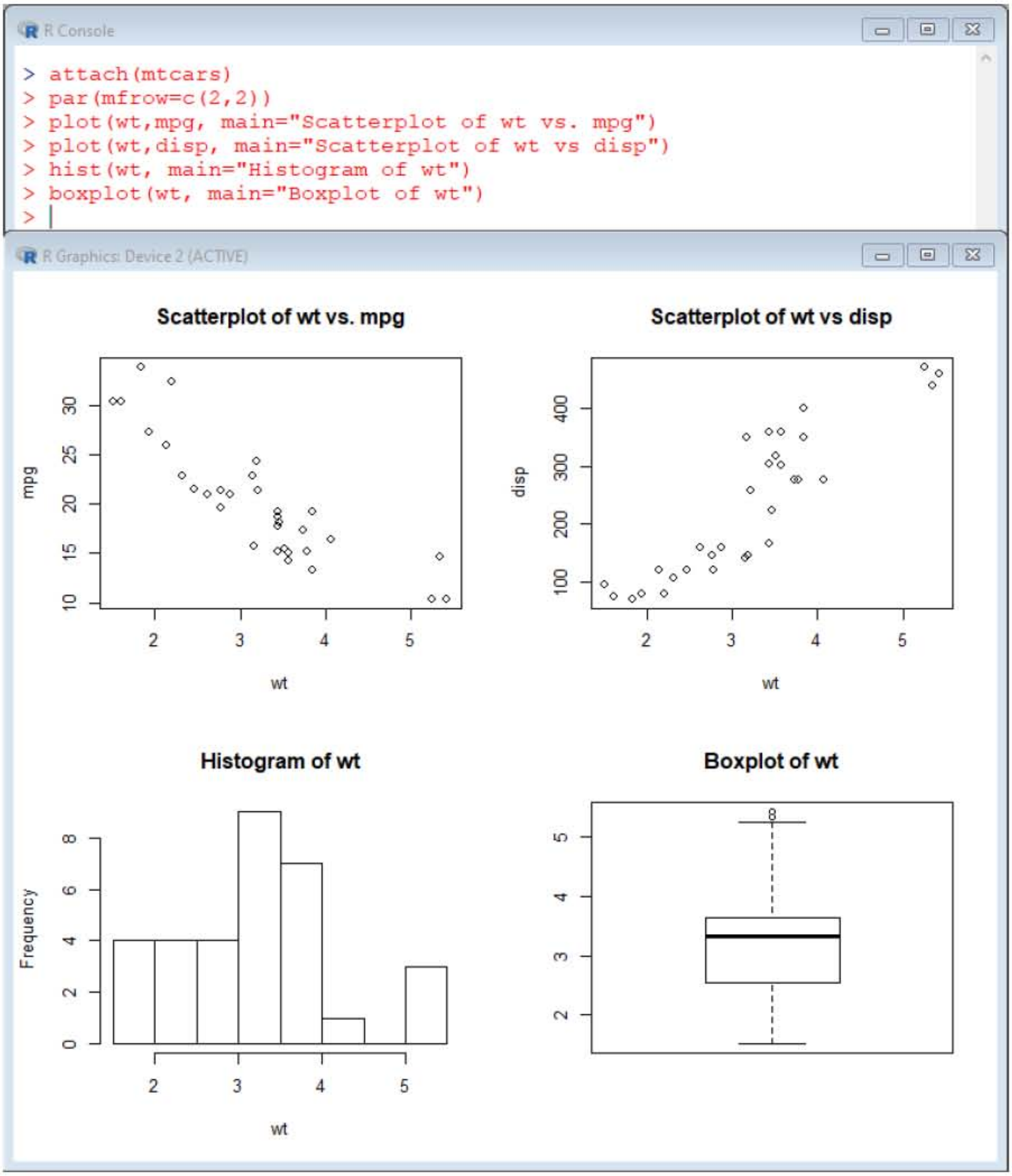
ou:



Plots multiples dans une fenêtre graphique

Il existe plusieurs manières de combiner des plots dans une fenêtre graphique.

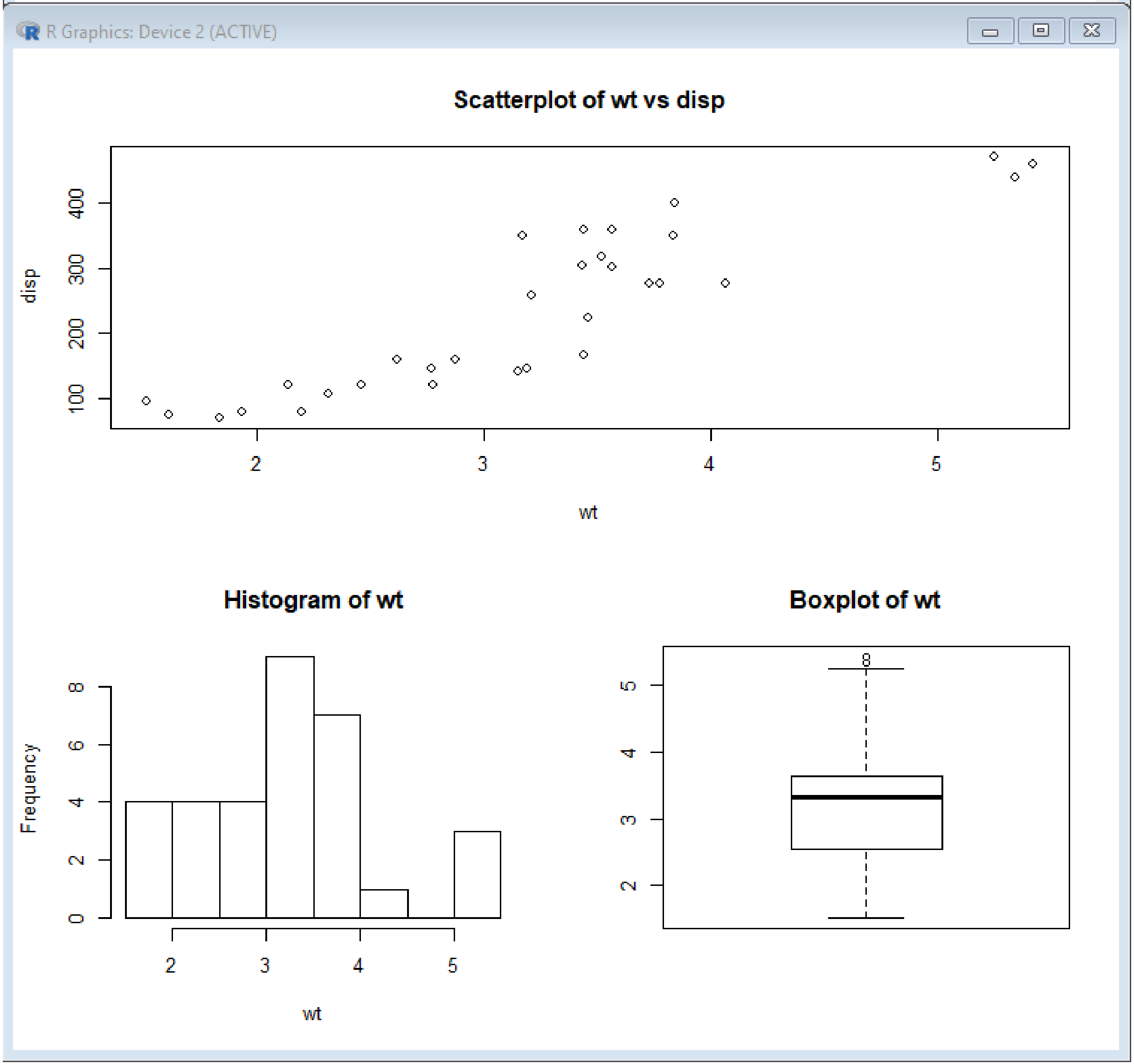
Voyons d'abord la méthode de base utilisant par et mfrow:



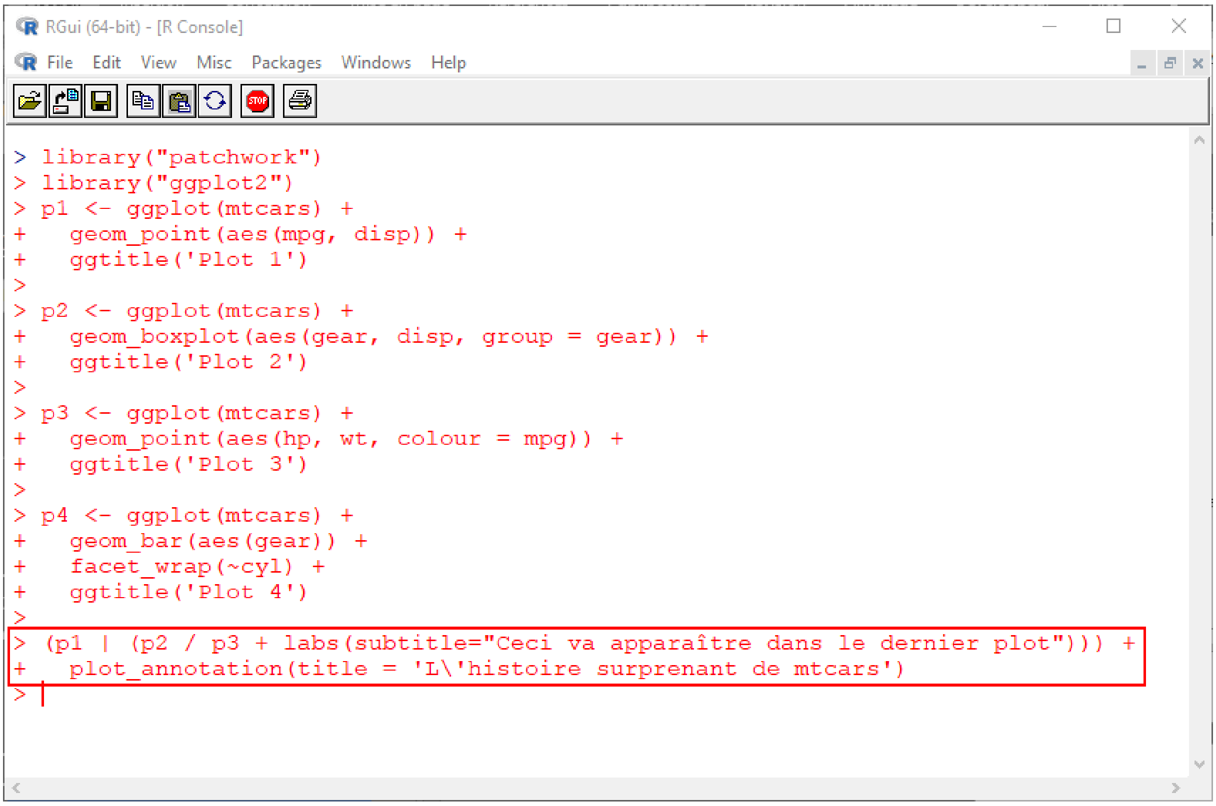
Ou pour des configuratiosn plus exotiques:

R Console

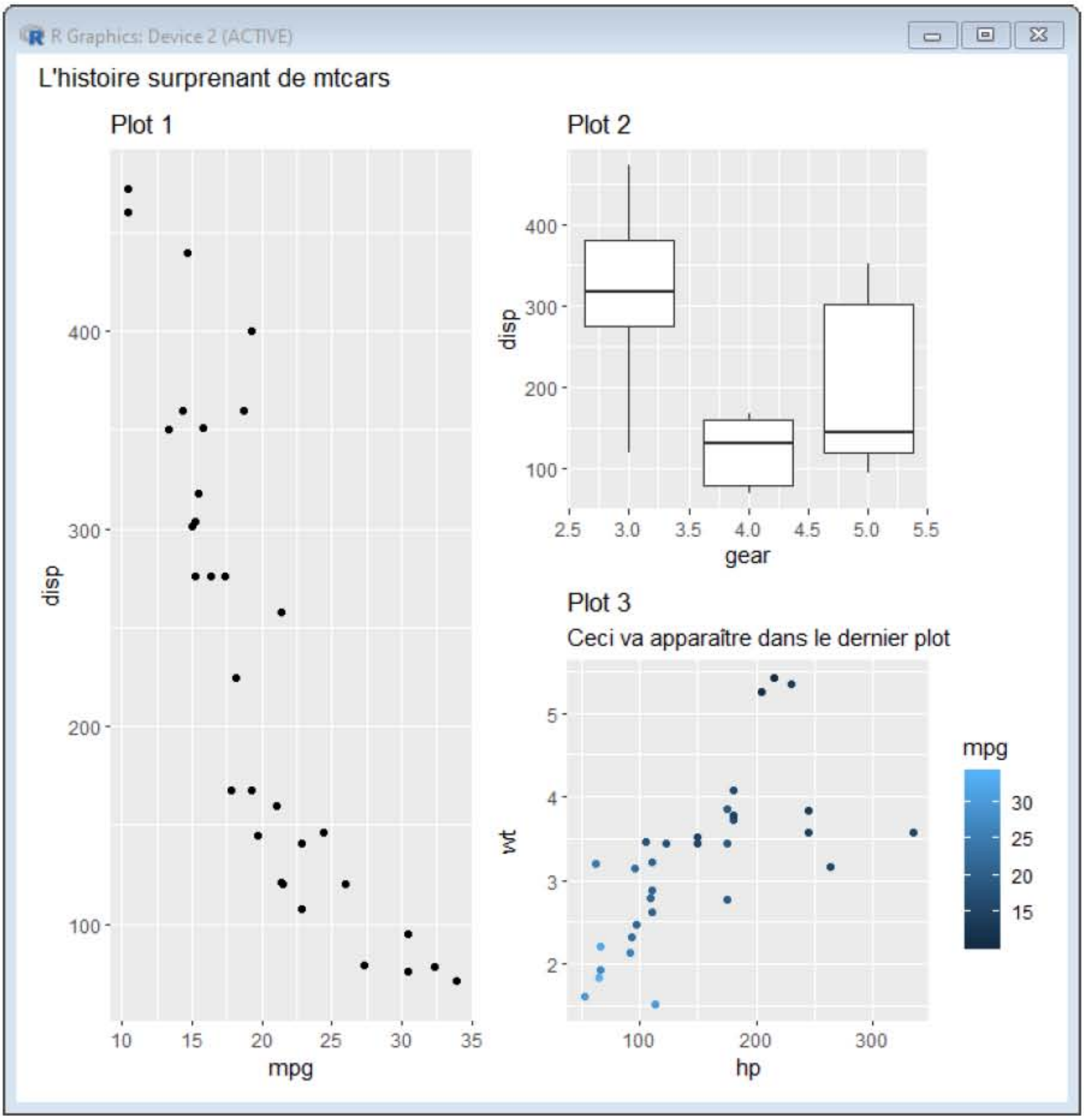
```
> layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
> plot(wt,disp, main="Scatterplot of wt vs disp")
> hist(wt, main="Histogram of wt")
> boxplot(wt, main="Boxplot of wt")
>
```



Et si nous chargeons le package patchwork, il y a une manière beaucoup plus ludique de gérer la disposition de graphiques (voir la vignette correspondant pour plusd'exemples: https://cran.r-project.org/web/packages/patchwork/vignettes/patchwork.html):


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("patchwork")
> library("ggplot2")
> p1 <- ggplot(mtcars) +
+   geom_point(aes(mpg, disp)) +
+   ggtitle('Plot 1')
>
> p2 <- ggplot(mtcars) +
+   geom_boxplot(aes(gear, disp, group = gear)) +
+   ggtitle('Plot 2')
>
> p3 <- ggplot(mtcars) +
+   geom_point(aes(hp, wt, colour = mpg)) +
+   ggtitle('Plot 3')
>
> p4 <- ggplot(mtcars) +
+   geom_bar(aes(gear)) +
+   facet_wrap(~cyl) +
+   ggtitle('Plot 4')
>
> (p1 | (p2 / p3 + labs(subtitle="Ceci va apparaître dans le dernier plot"))) +
+   plot_annotation(title = 'L\'histoire surprenant de mtcars')
>

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
L'histoire surprenant de mtcars
Plot 1
disp
mpg
Plot 2
disp
gear
Plot 3
Ceci va apparaître dans le dernier plot
wt
hp
mpg

Exemple 105.: Thèmes de couleurs des graphes de base de R

R 3.6.1

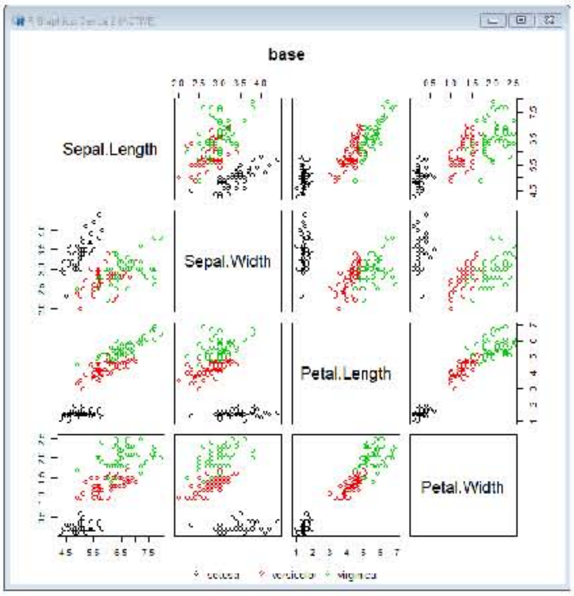
Nous verrons plus tard qu'il existe un package fameux pour les graphs qui se nomme ggplot2 et qui possède des thèmes de couleurs que l'on peut définir de manière centralisée dans une entreprise ou université.

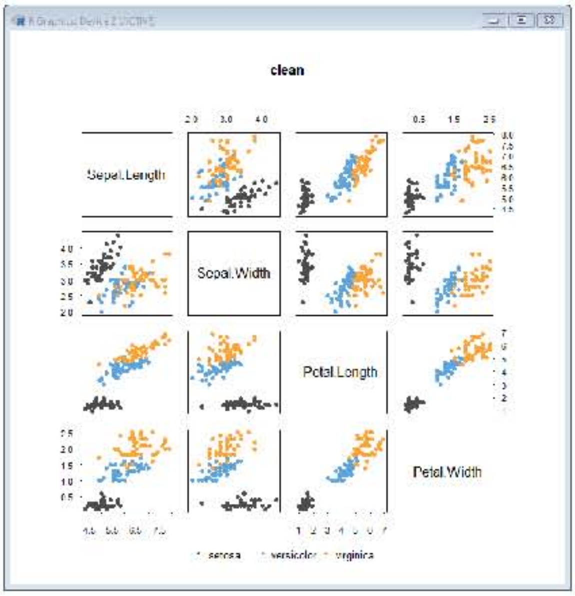
Cependant les graphiques de base de R ont aussi des modèles de couleurs grâce au package **basetheme** et sa fonction basetheme( ).

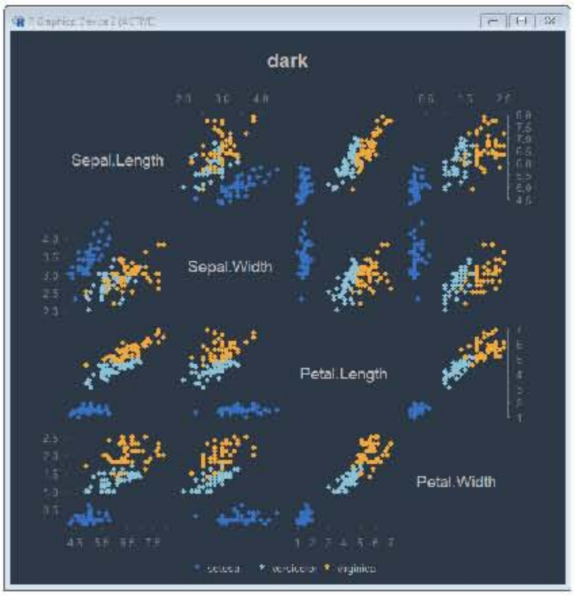
Voyons quelques exemples

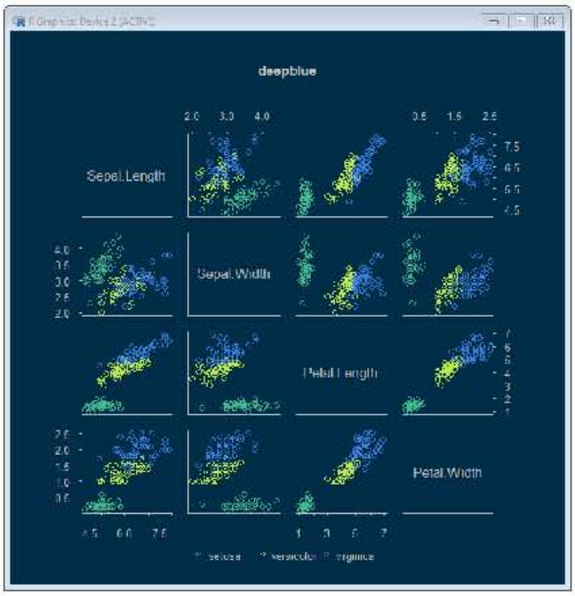
```
> library("basetheme")
>
> #pour enlever d'abord tout thème appliqué éventuellement avant
> basetheme(NULL)
>
> #maintenant allons-y
> pairs(iris[,1:4], main="base", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
> basetheme("clean")
> pairs(iris[,1:4], main="clean", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
> basetheme("dark")
> pairs(iris[,1:4], main="dark", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
> basetheme("deepblue")
> pairs(iris[,1:4],  main="deepblue",col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
> basetheme("minimal")
> pairs(iris[,1:4], main="minimal", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
> basetheme("void")
> pairs(iris[,1:4], main="void", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
```

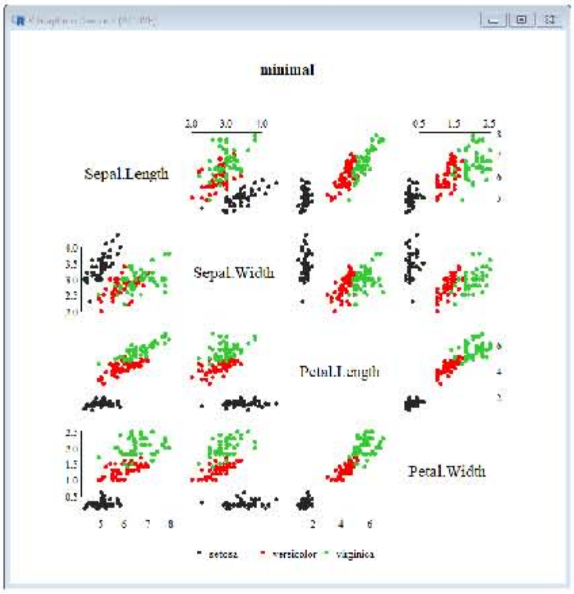
Ce qui donne dans l'ordre:

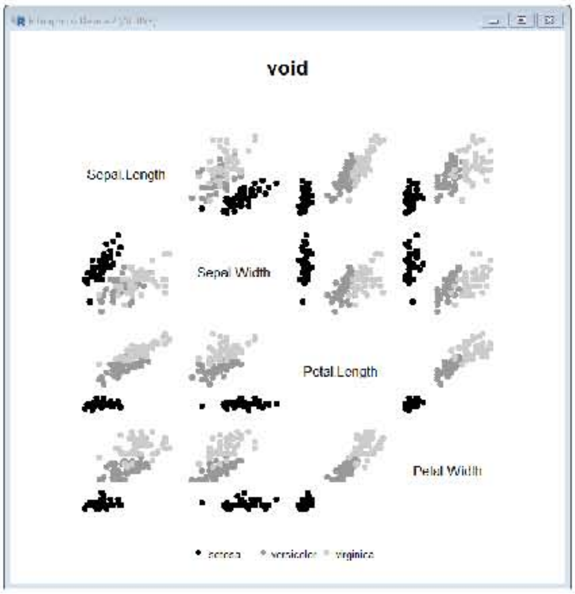

base
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa versicolor virginica


clean
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa versicolor virginica


dark
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width


deepblue
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa versicolor virginica

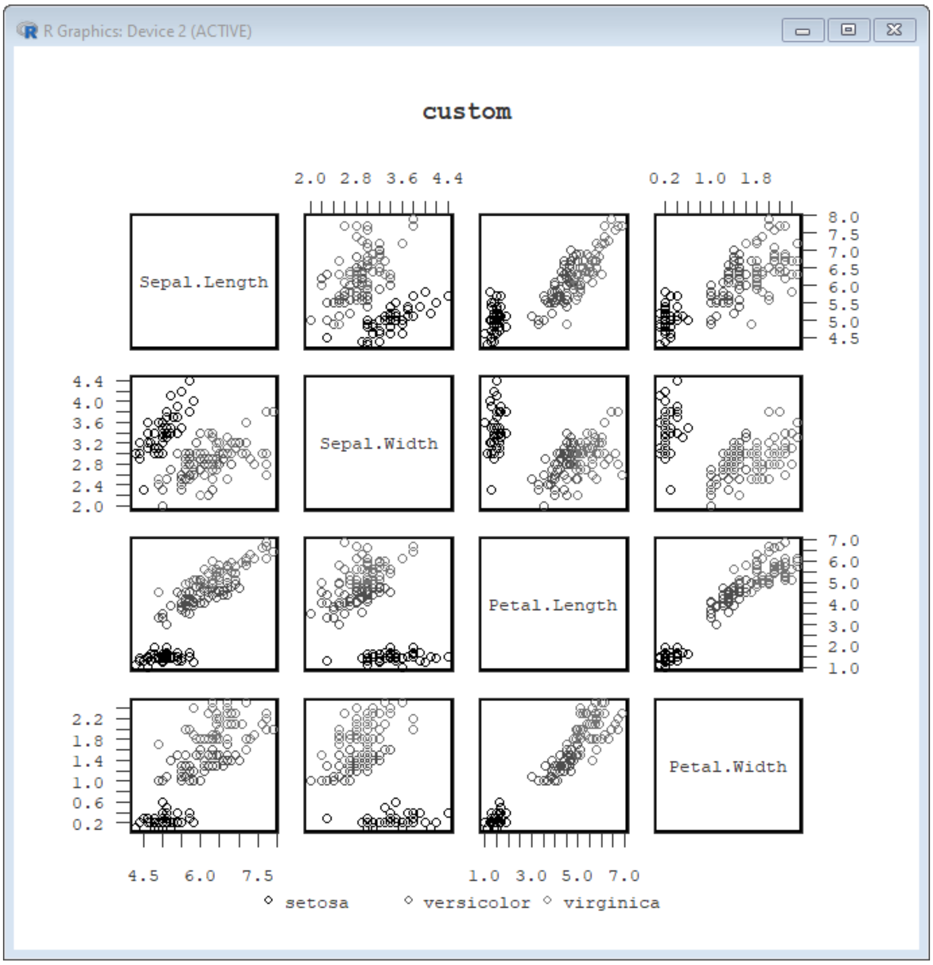

minimal
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa versicolor virginica


void
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa versicolor virginica

Sinon on peut créer ses propres thèmes comme l'illustre l'exemple suivant:

```
> pars <- basetheme("default")
> pars$palette <- c("black", grey.colors(8))  # numbered colors - shades of grey
> pars$bg  <- "white"                         # some colors
> pars$fg  <- "gray20"                        # some colors
> pars$col <- "gray20"                        # some colors
> pars$col.main <- "black"                    # some colors
> pars$col.axis <- "gray20"                   # some colors
> pars$col.lab  <- "gray20"                   # some colors
> pars$family   <-  "mono"                    # change font
> pars$lab      <-  c(10,10,7)                # more ticks on axes
> pars$cex.axis <-  0.8                       # smaller axis labels
> pars$las      <-  1                         # always horizontal axis labels
> pars$rect.border <- "black"                 # box around the plot
> pars$rect.lwd    <- 4                       # ticker border
>
> #on charge le thème
> basetheme(pars)
> pairs(iris[,1:4], main="custom", col=iris$Species)
> legend("top", legend=unique(iris$Species), col=unique(iris$Species),
+        pch=par("pch"), cex=0.8, horiz=TRUE, bty="n", inset=c(0,1), xpd=TRUE)
>
```

Ce qui donne:

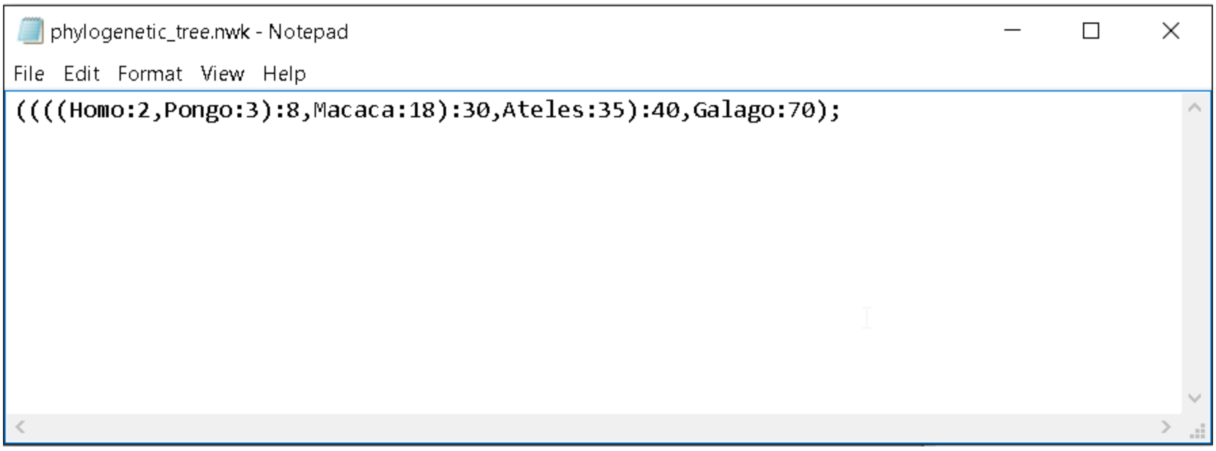


Exemple 106.: Arbre phylogénétique (phylogrammes, cladogrammes)

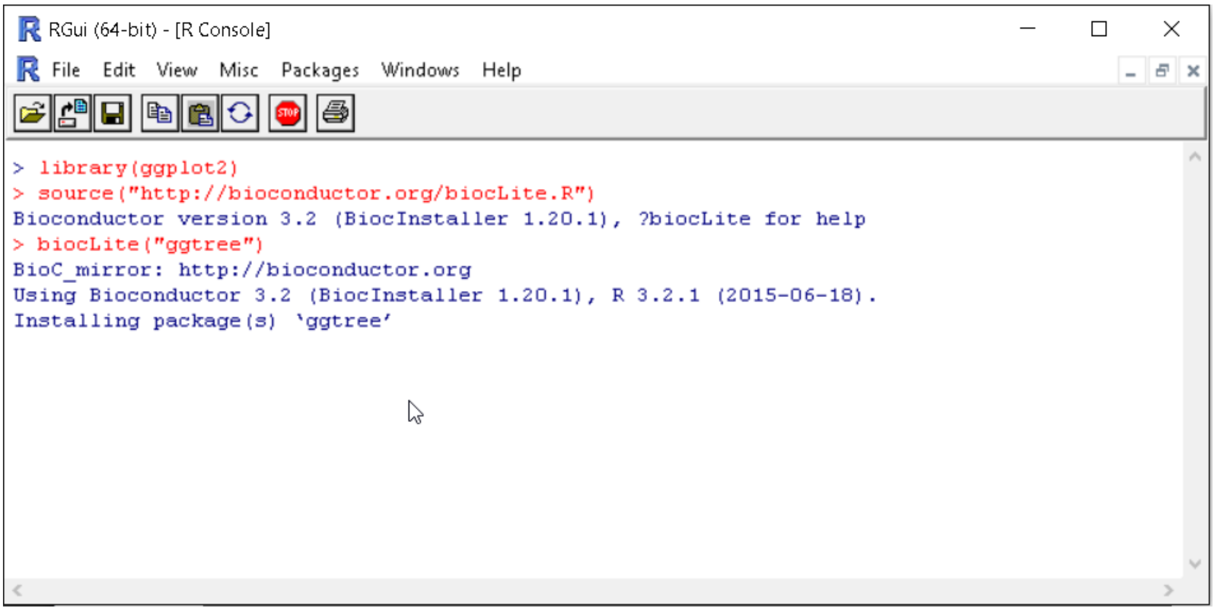
R 3.2.1

Nous allons construire un arbre phylogénétique non radial basé sur une structure nwk (structure créée par le non moins fameux mathématicien Arthur Cayley pour rappel…).

Le fichier que nous allons utiliser est simplement avec une unité de distance euclidienne qui sera prise pour chaque nœud et chaque élément depuis l'extrémité:



D'abord nous installons les packages nécessaires car **ggplot** appartient à **bioconductor** et nécessite **ggplot2**:



Après nous y allons avec un exemple:

```
R Console
> require(ggtree)
>
> mytree <- read.tree("c:/tmp/phylogenetic_tree.nwk");mytree

Phylogenetic tree with 5 tips and 4 internal nodes.

Tip labels:
[1] "Homo"   "Pongo"  "Macaca" "Ateles" "Galago"

Rooted; includes branch lengths.
> ggtree(mytree,aes(x, y),linetype="dotted",ladderize=TRUE)+geom_point(aes(shape=isTip, color=isTip), size=3)+
+ geom_text(aes(x=branch, label=label), size=3, color="purple", vjust=-0.3)+ theme_tree2()
Warning message:
Removed 4 rows containing missing values (geom_text).
>
```

Ce qui donne un phylogramme:

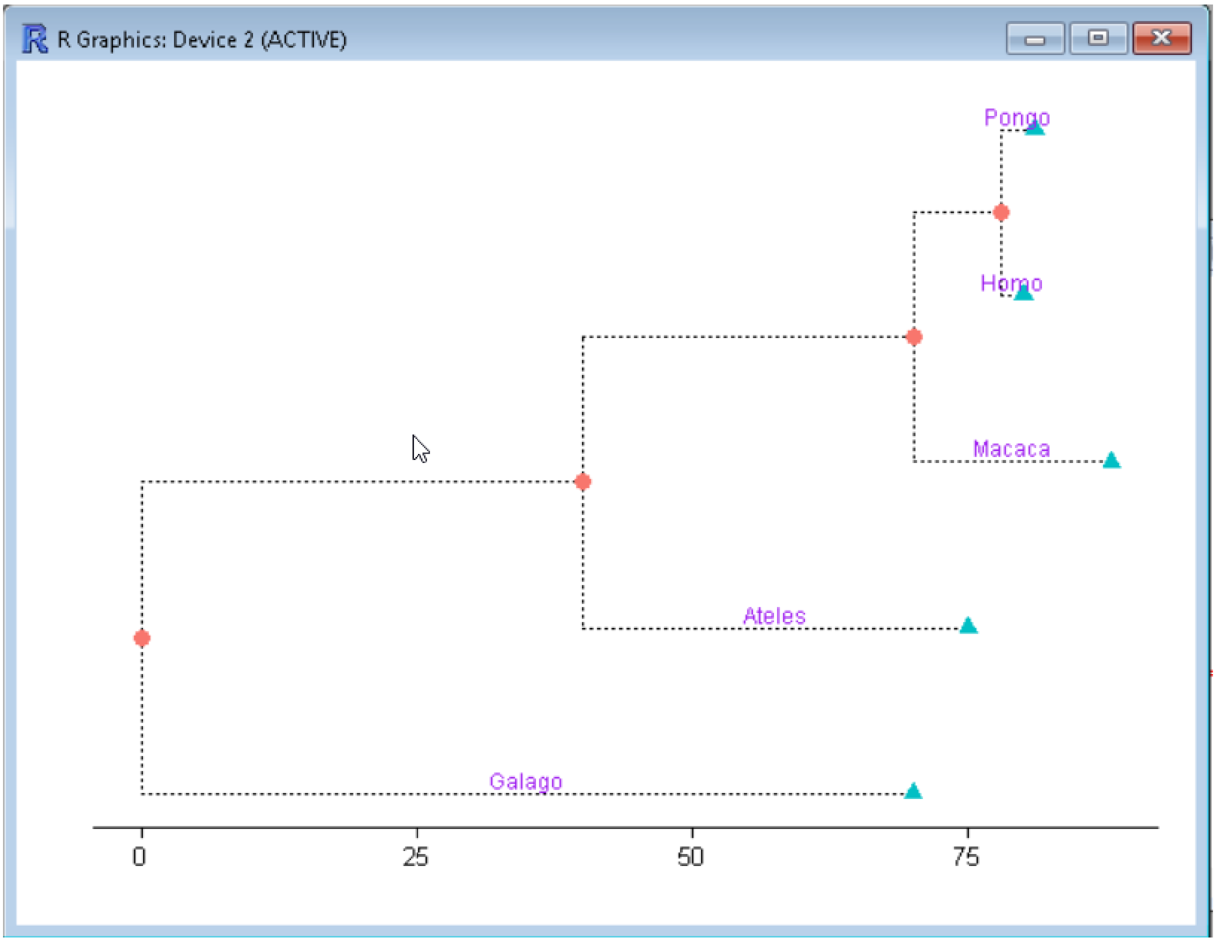


Figure 37 Graph de type phylogramme

En rajoutant *branch.length='none'* nous obtenons un cladogramme:

```
R Console
> ggtree(mytree,aes(x, y),linetype="dotted",ladderize=TRUE,branch.length='none')+
+ geom_point(aes(shape=isTip, color=isTip), size=3)+
+ geom_text(aes(x=branch, label=label), size=3, color="purple", vjust=-0.3)+
+ theme_tree2()
Warning message:
Removed 4 rows containing missing values (geom_text).
>
```

Ce qui donne:

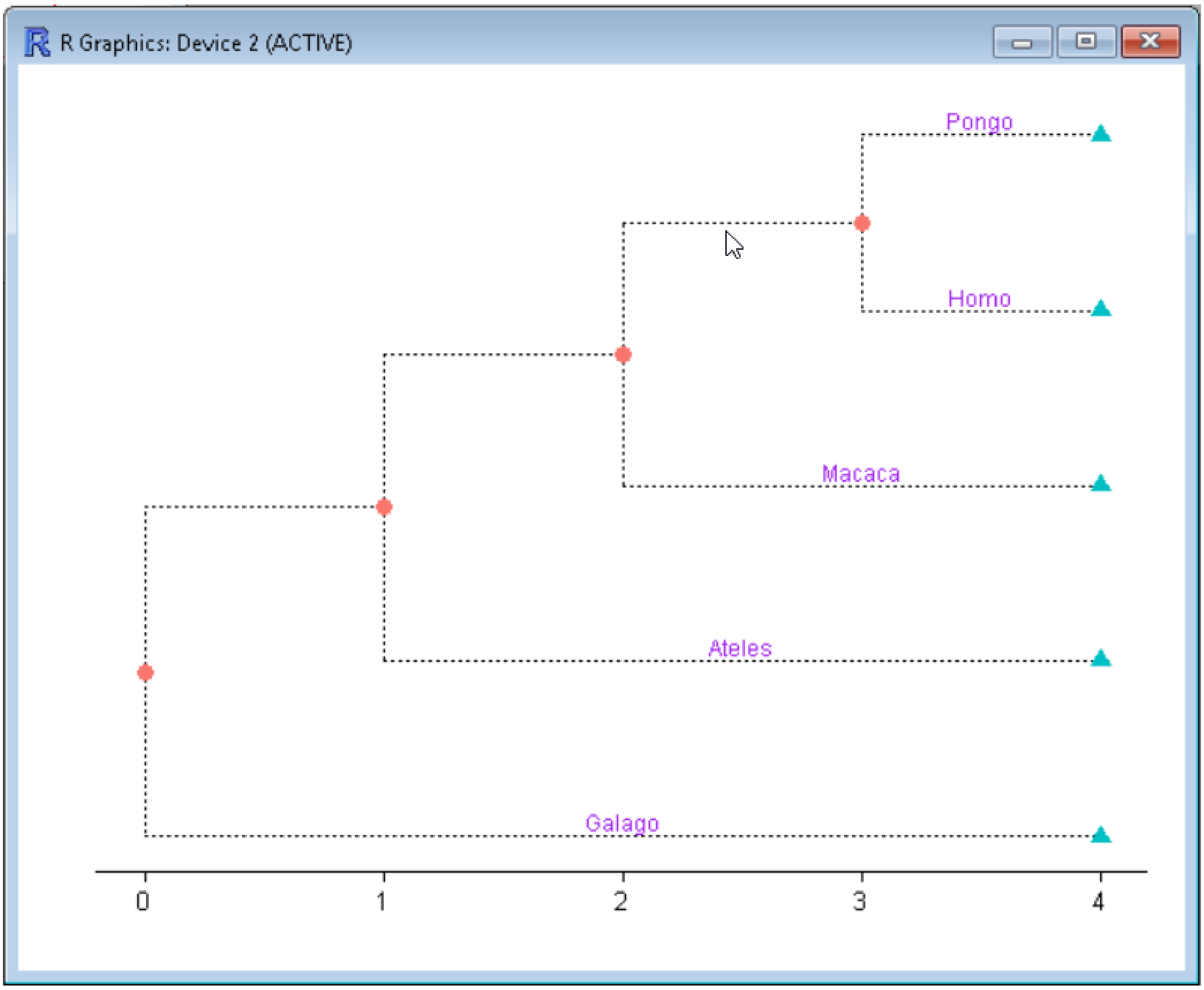
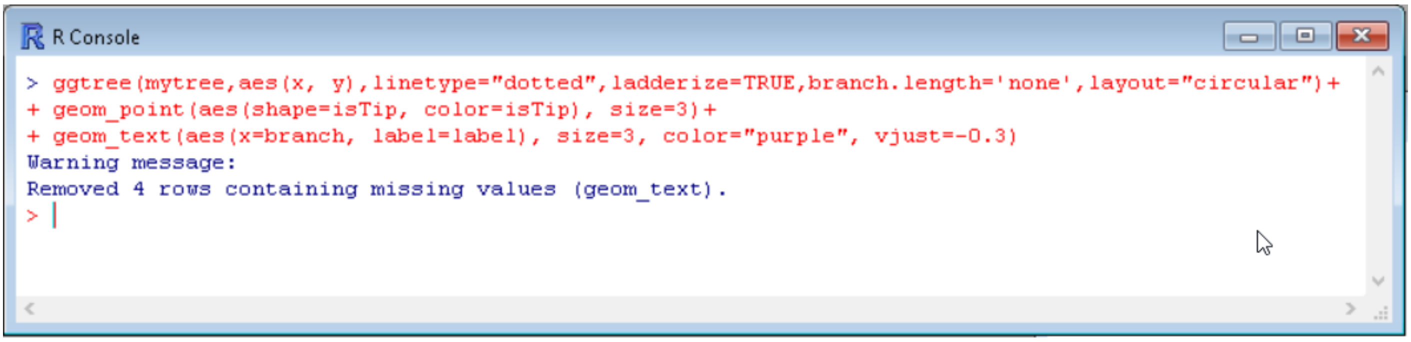


Figure 38 Graph de type cladogramme

On peut faire un cladogramme circulaire aussi:



Ce qui donne:

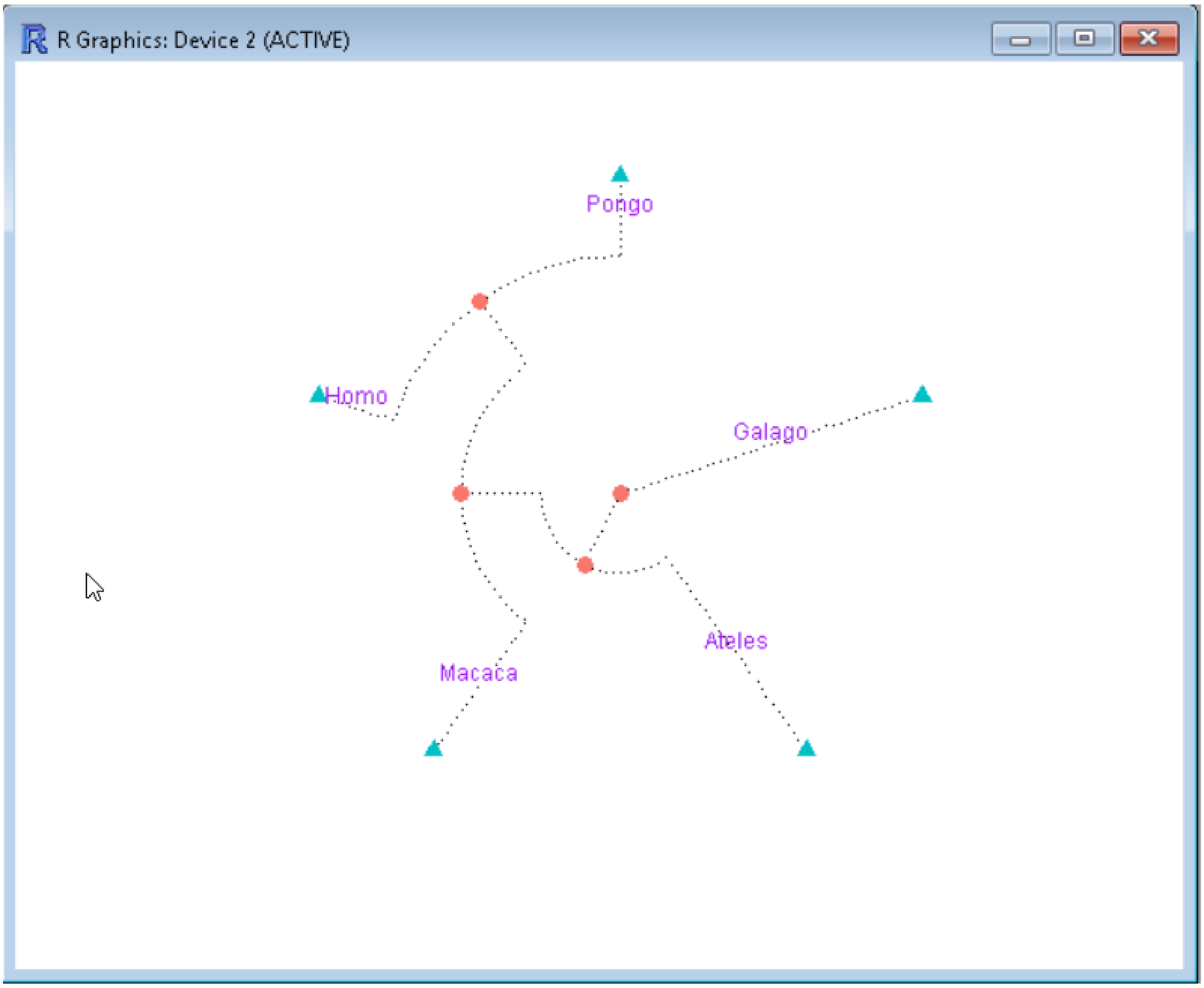
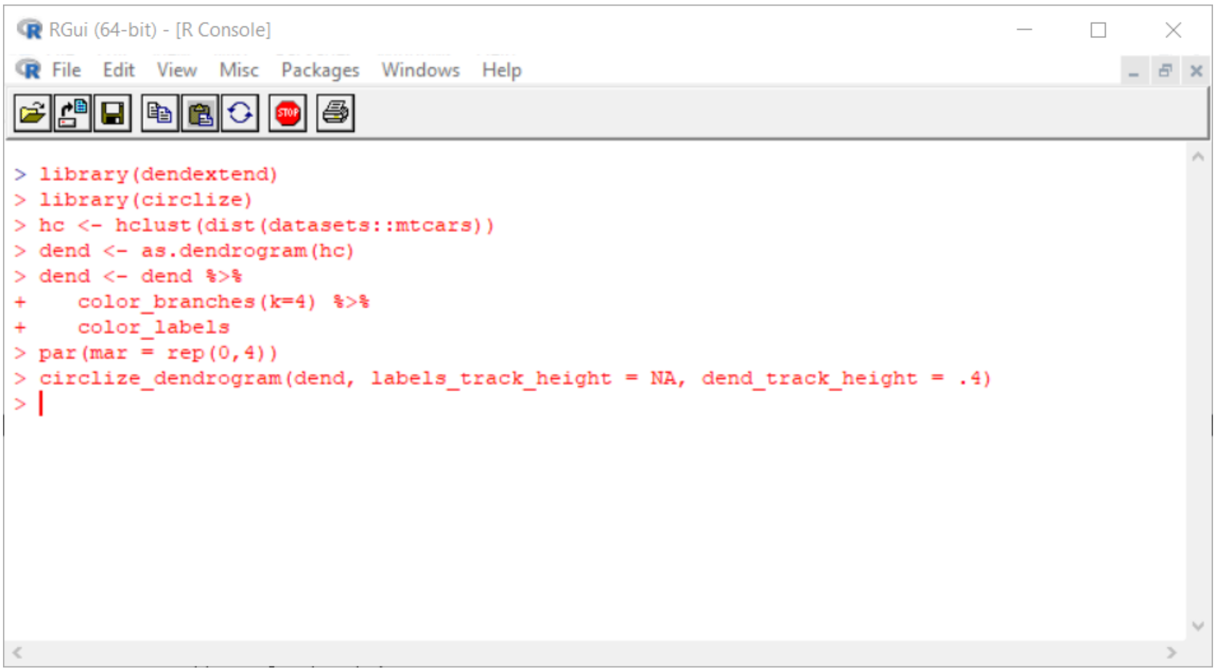


Figure 39 Graph cladogramme circulaire

ou une version plus sympa:



qui donne (c'est joli!):

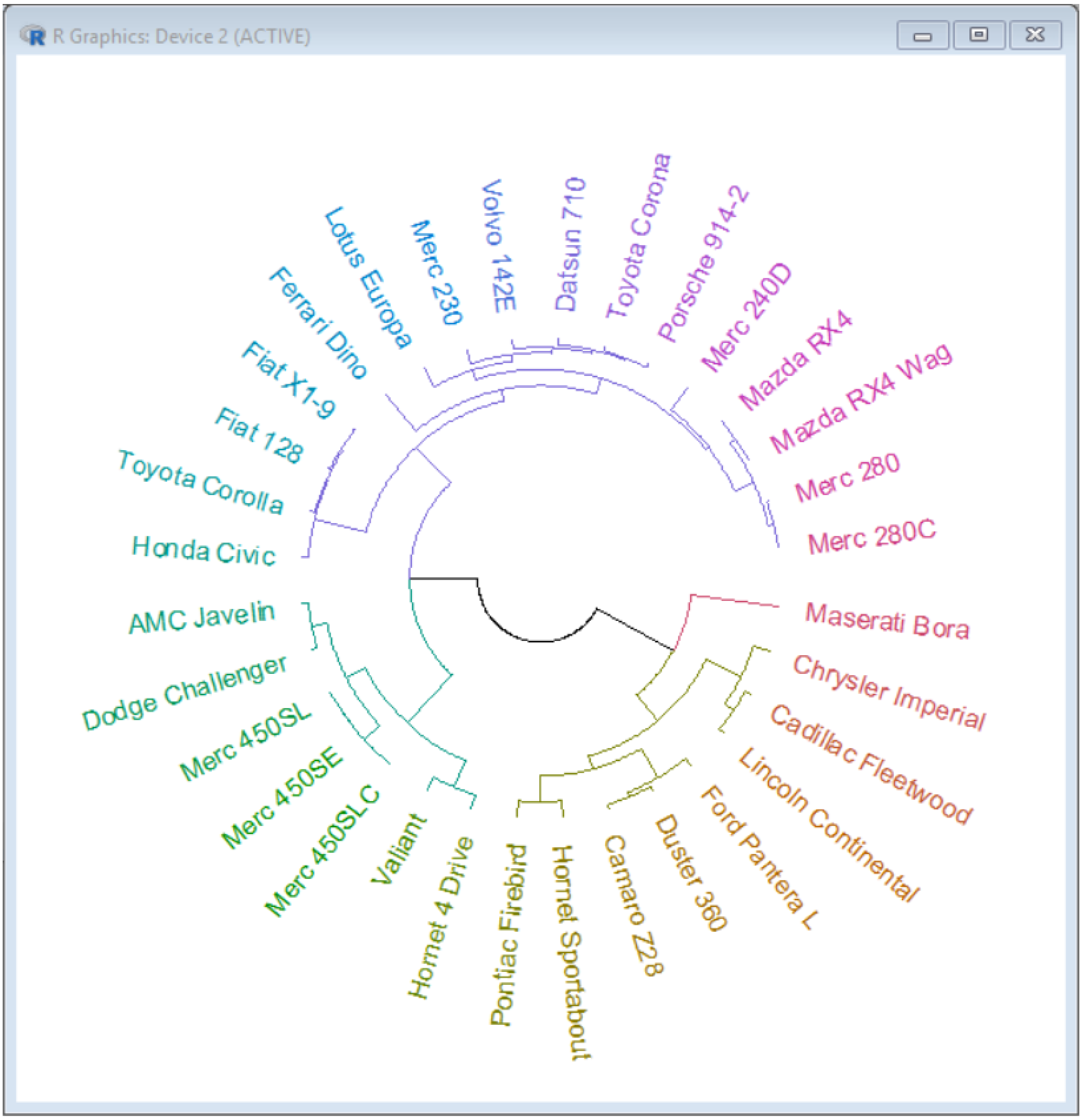
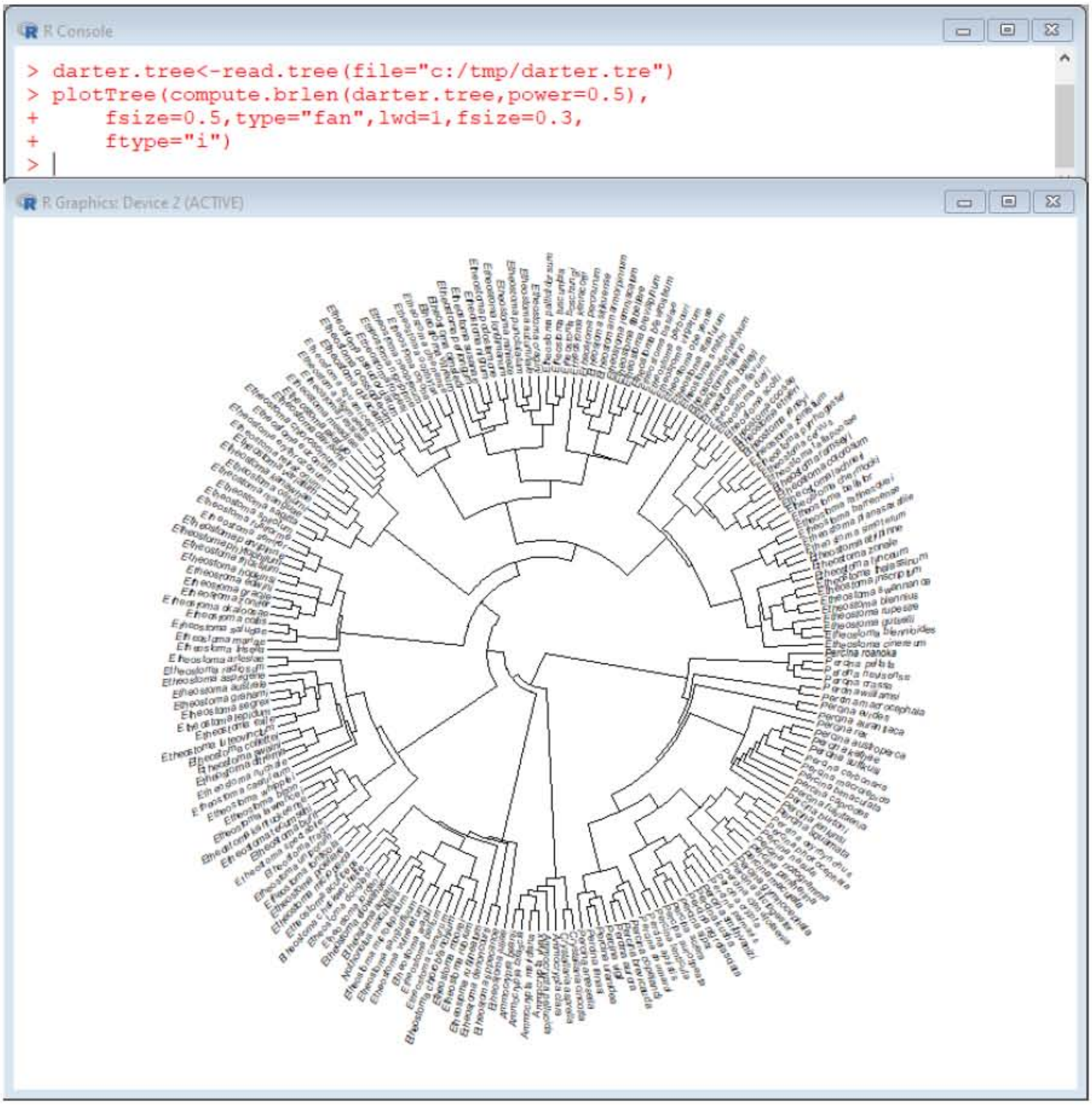


Figure 40 Graph dendrogramme circulaire

Sinon de manière plus complète on a à l'aide du package **phytools** et sa fonction **plotTree( )**:


R Console
> darter.tree<-read.tree(file="c:/tmp/darter.tre")
> plotTree(compute.brlen(darter.tree,power=0.5),
+   fsize=0.5,type="fan",lwd=1,fsize=0.3,
+   ftype="i")
>
R Graphics: Device 2 (ACTIVE)

Exemple 107.: Diagrammes vectoriels (dessiner des flèches et mettre du texte)

R 3.0.2

Bon là il n'y a pas grand chose à dire. C'est à usage principalement scolaire mais sinon sachez que c'est très utile de savoir dessiner des flèches et de mettre des textes sur de plots quels qu'ils soient avec la commande **arrows( )** :

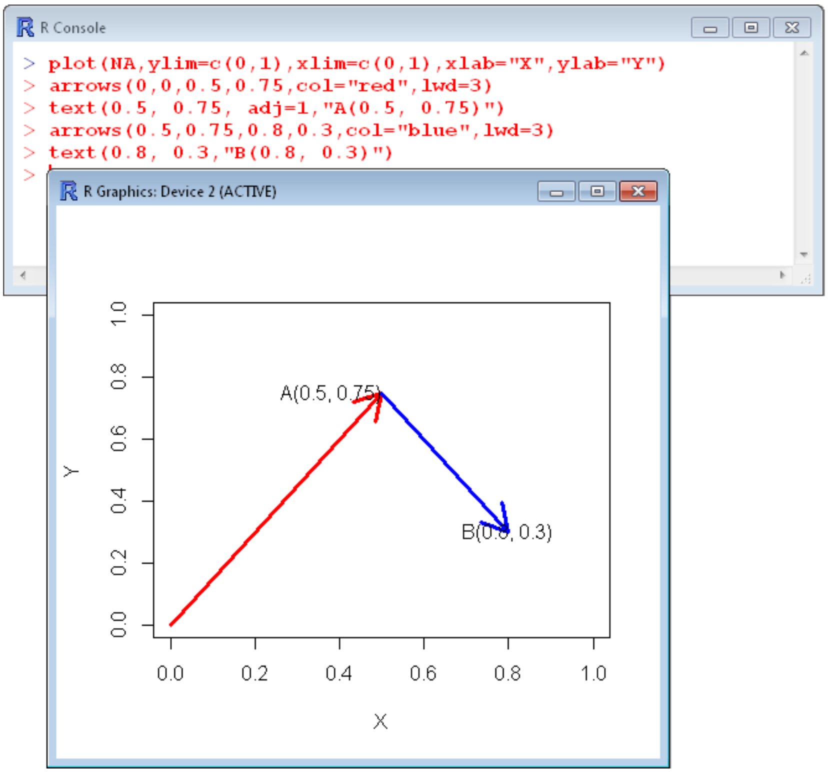
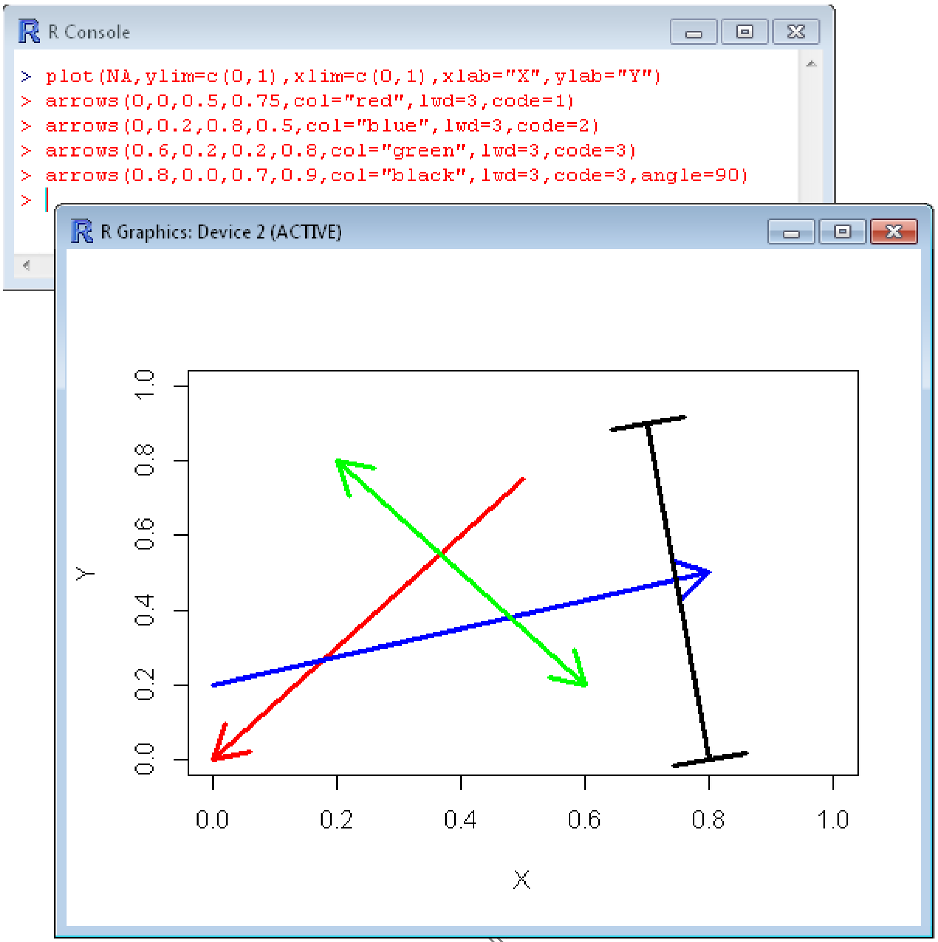


Figure 41 Graph avec vecteurs (et textes)

Ou nous pouvons jouer avec différents styles de flèches:



Voyons un cas très souvent utilisé en probabilité des flèches (mais simplifié dans le cas présent):

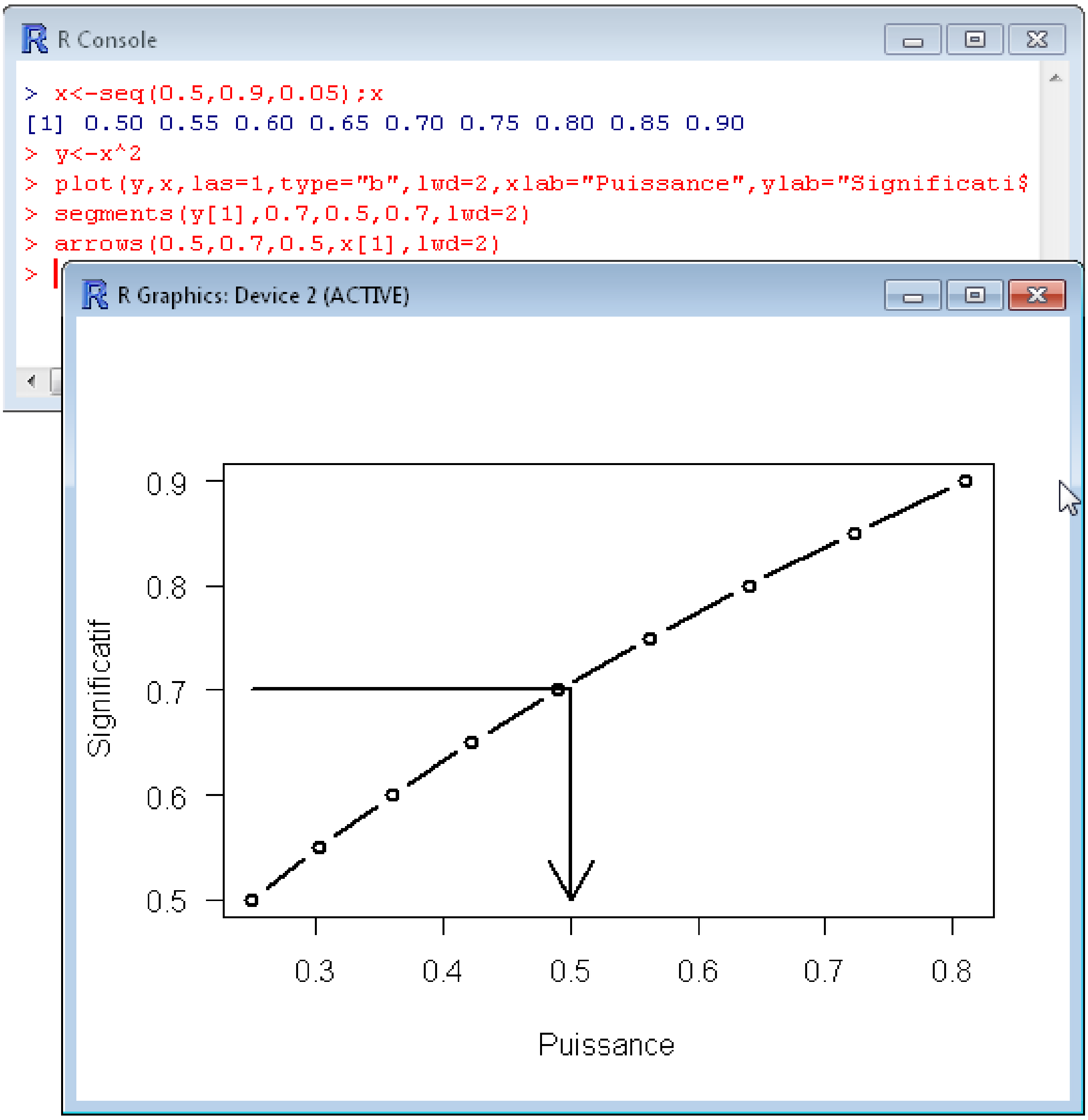
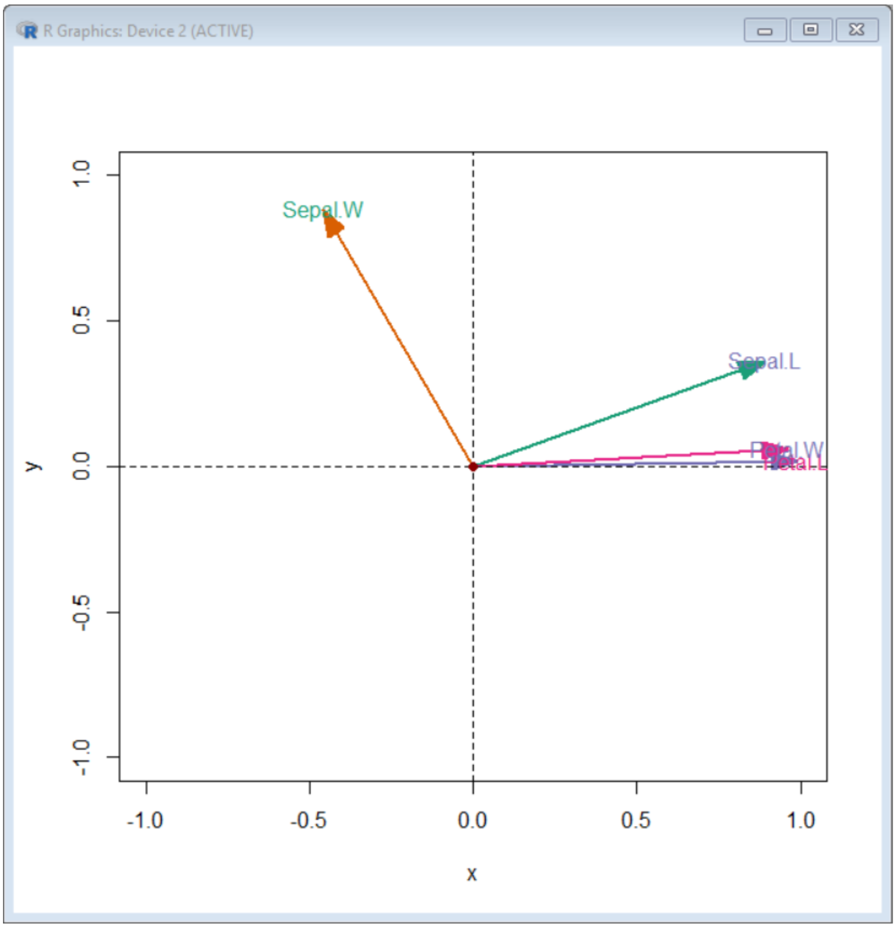


Figure 42 Graph avec fonction f(x) et ligne et flèches

Nous pouvons faire des diagrammes vectoriels beaucoup plus beaux avec le package **plot3D**. Par exemple, d'abord avec les vecteurs sur un plan en utilisant la fonction **arrows2D( )** :

```
> library("plot3D")
> #points de départs des 4 vecteurs
> x0 <- c(0, 0, 0, 0)
> y0 <- c(0, 0, 0, 0)
> #points d'arrivées des 4 vecteus
> x1 <- c(0.89, -0.46, 0.99, 0.96)
> y1 <- c(0.36,  0.88, 0.02, 0.06)
> cols <- c("#1B9E77", "#D95F02", "#7570B3", "#E7298A")
>
> arrows2D(x0, y0,  x1, y1, col = cols,
+          lwd = 2, xlim = c(-1, 1), ylim = c(-1, 1))
> # ajouter une ligne horizontale et verticale à c(0,0)
> abline(h =0, v = 0, lty = 2)
> # ajout d'un point en (0,0)
> points2D(x0, y0, add = TRUE, col="darkred",
+           colkey = FALSE, pch = 19, cex = 1)
> # ajout de légendes aux vecteurs
> text2D(x1, y1, c("Sepal.L", "Sepal.W", "Petal.L", "Petal.W"),
+         colvar = x1^2, col = cols, add=TRUE, colkey = FALSE)
>
```

Ce qui donne:



Nous verrons plus loin comment faire des diagrammes vectoriels en 3D (voir page 998).

Signalons qu'en installant le package **plotrix** il est possible à tout hasard... de faire des textes circulaires avec la commande **arctext( )** :

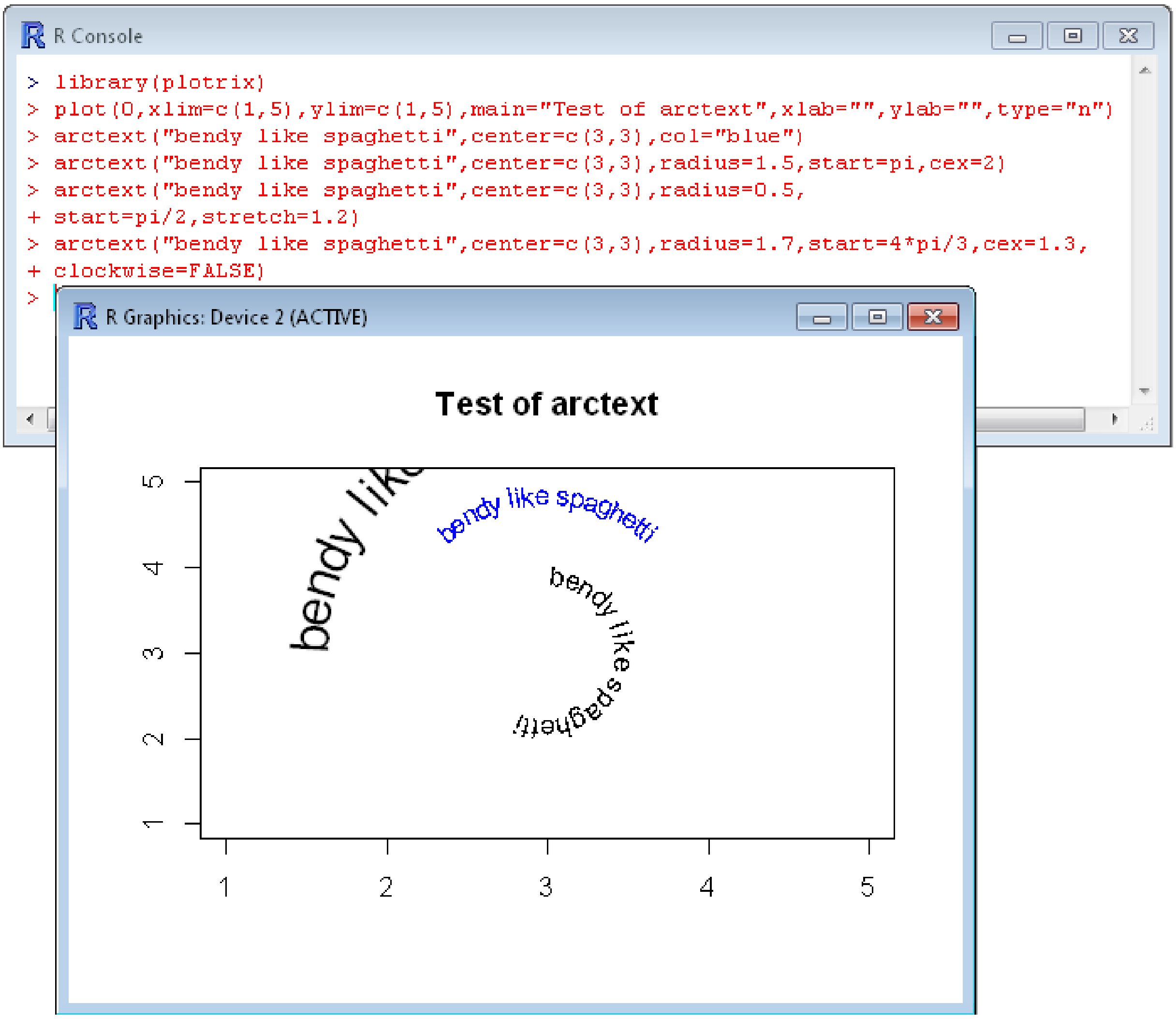
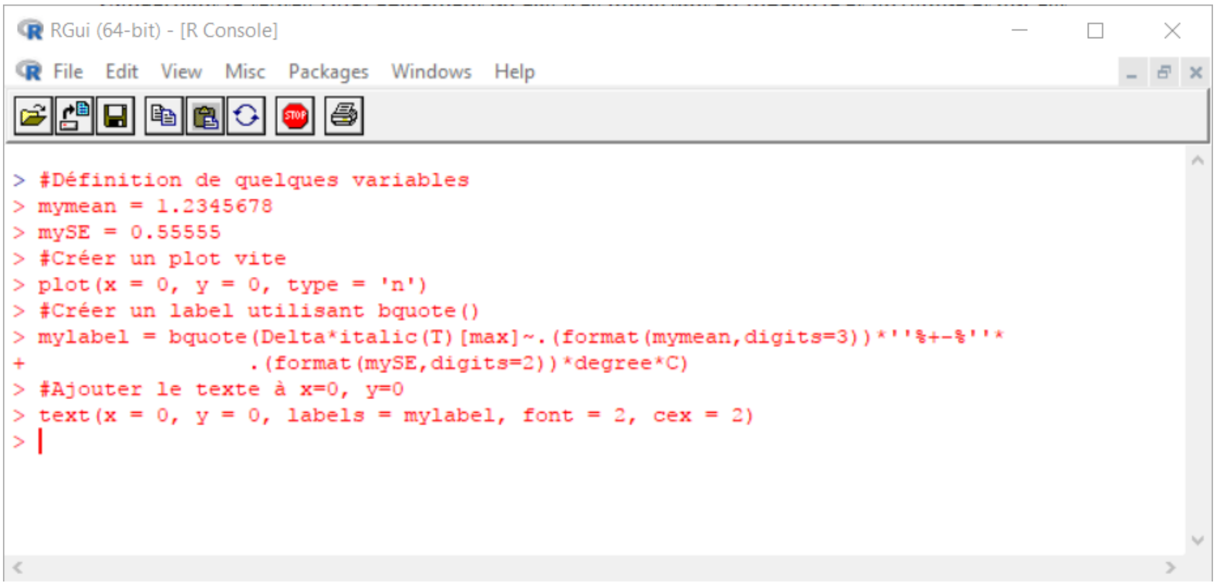


Figure 43 Graph avec textes circulaires

Concernant les textes voici également un cas très important en ingénierie et physique et qui est loin d'être trivial:



ce qui donne:

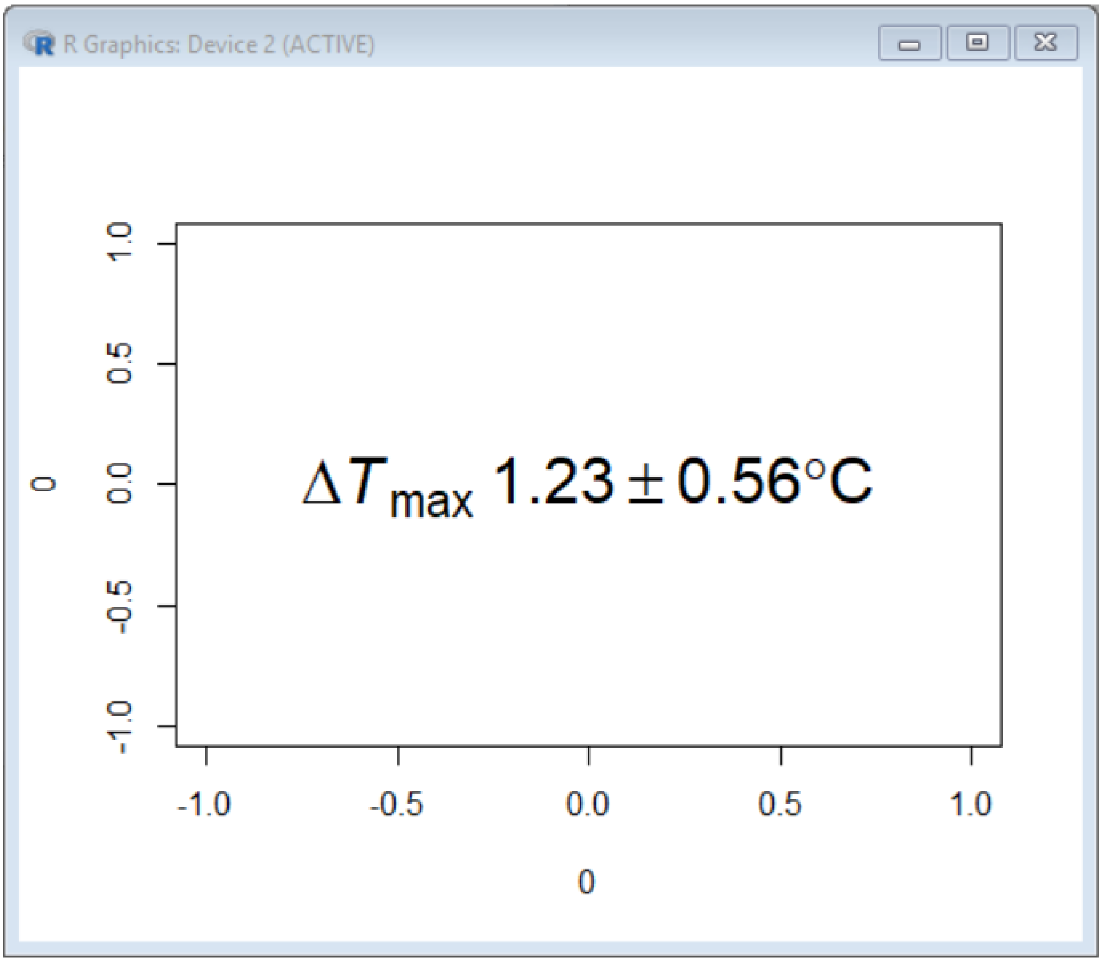


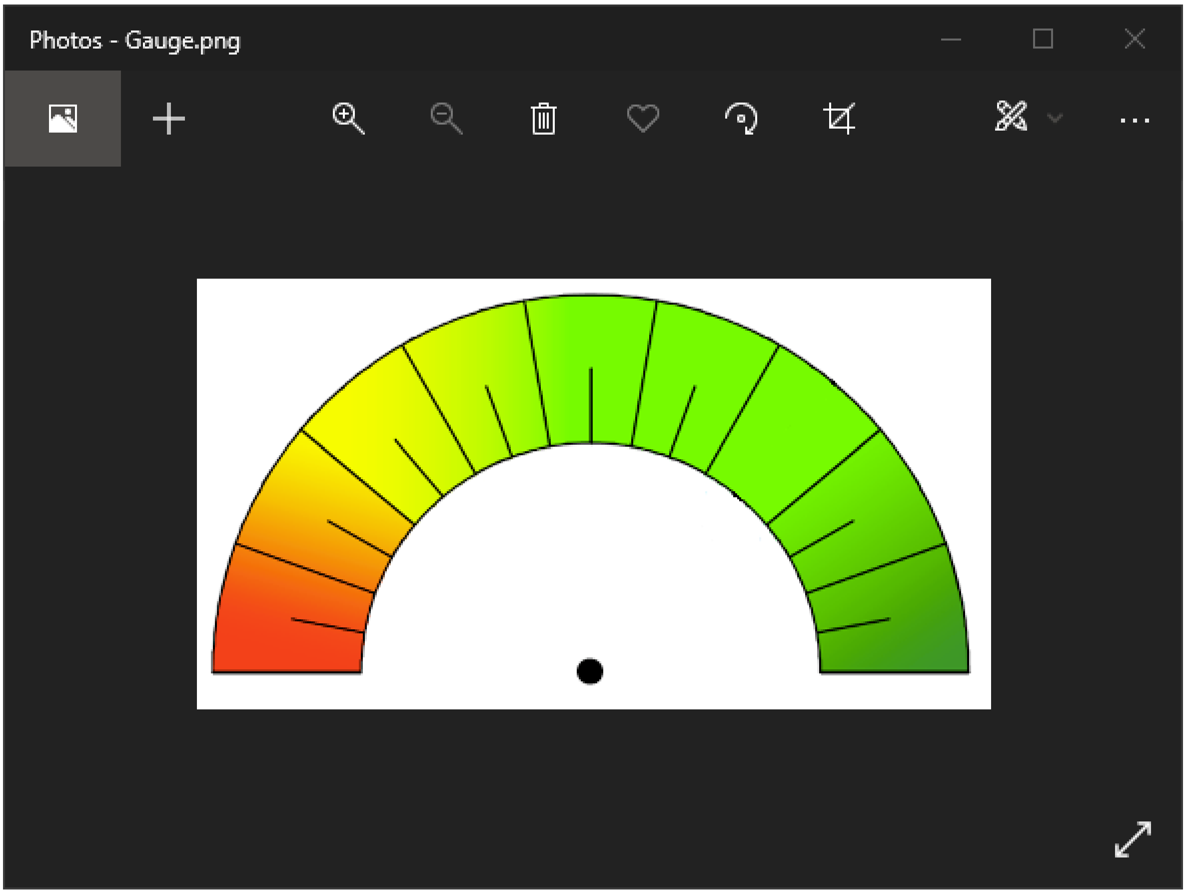
Figure 44 Graph avec texte et variables et symboles mathématiques

Exemple 108.: Mettre une image de fond

R 3.0.2

En entreprise on cherchera à avoir de multiples graphes dans une même fenêtre de R (voir plus loin) avec des gauges (speedomètres/tachymètres) pour montrer la performance d'un processus, d'un département ou d'une chaîne de production. Une idée simple est d'utiliser alors une image de fond comme-ci après en utilisant la commande **rasterImage( )** du package **png**.

Par exemple sur la base de l'image suivante:



Nous pouvons créer dans R:



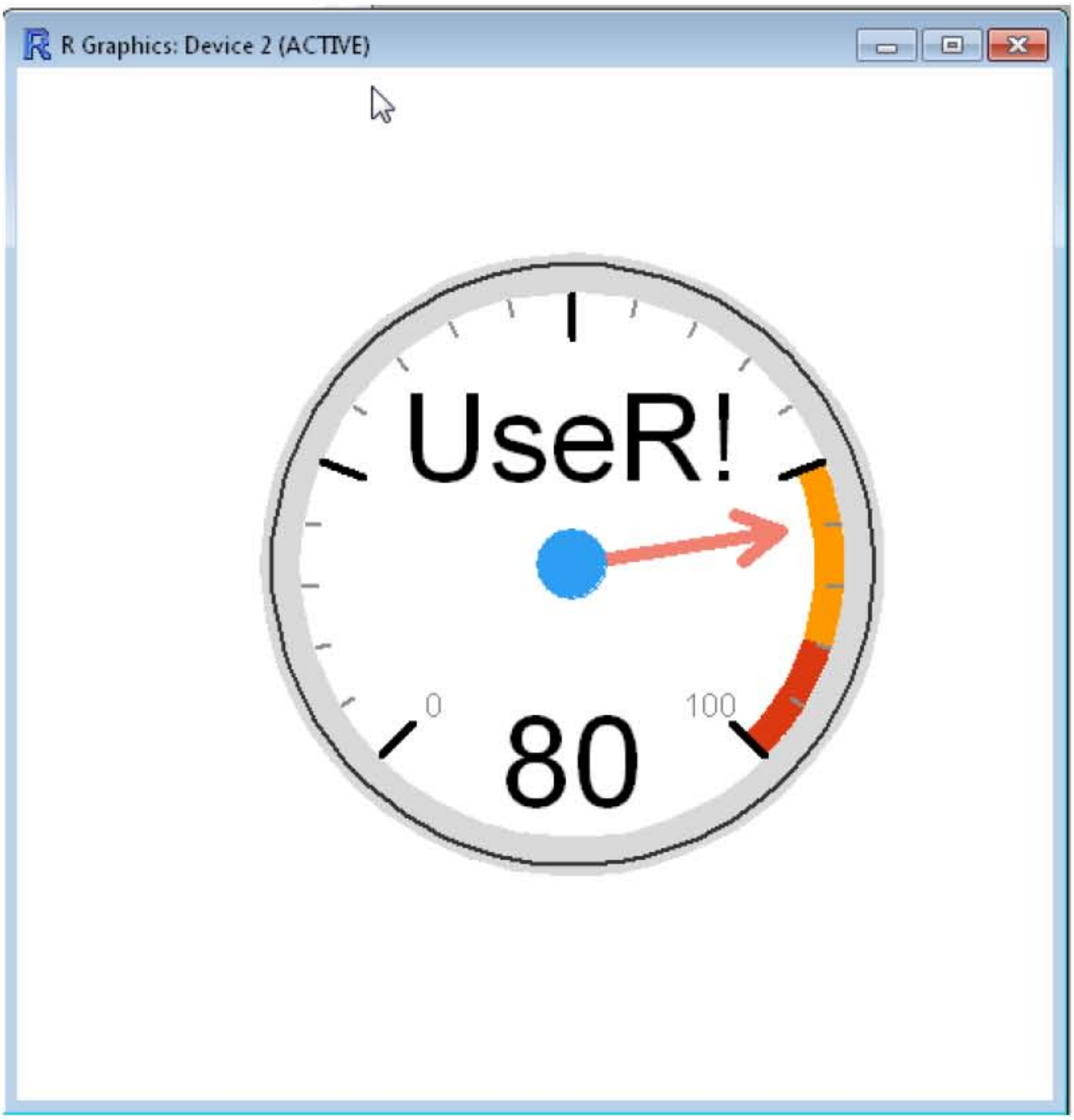
Figure 45 Graph avec image de fond

Évidemment l'exemple a un potentiel d'amélioration considérable...

Pour un exemple plus concret le lecteur pourra se rendre sur:

http://gastonsanchez.com/blog/how-to/2013/01/10/Gauge-Chart-in-R.html

où il pourra obtenir le résultat visible à la page suivante:


R Graphics: Device 2 (ACTIVE)
UseR!
0
100
80

Exemple 109.: Dessiner des rectangles

R 3.0.2

Lors de la création de scripts vous créerez probablement vos propres graphiques. Dès lors, dans certains cas il est bien utile de savoir dessiner des rectangles.

Voici un exemple pris de l'aide de R:

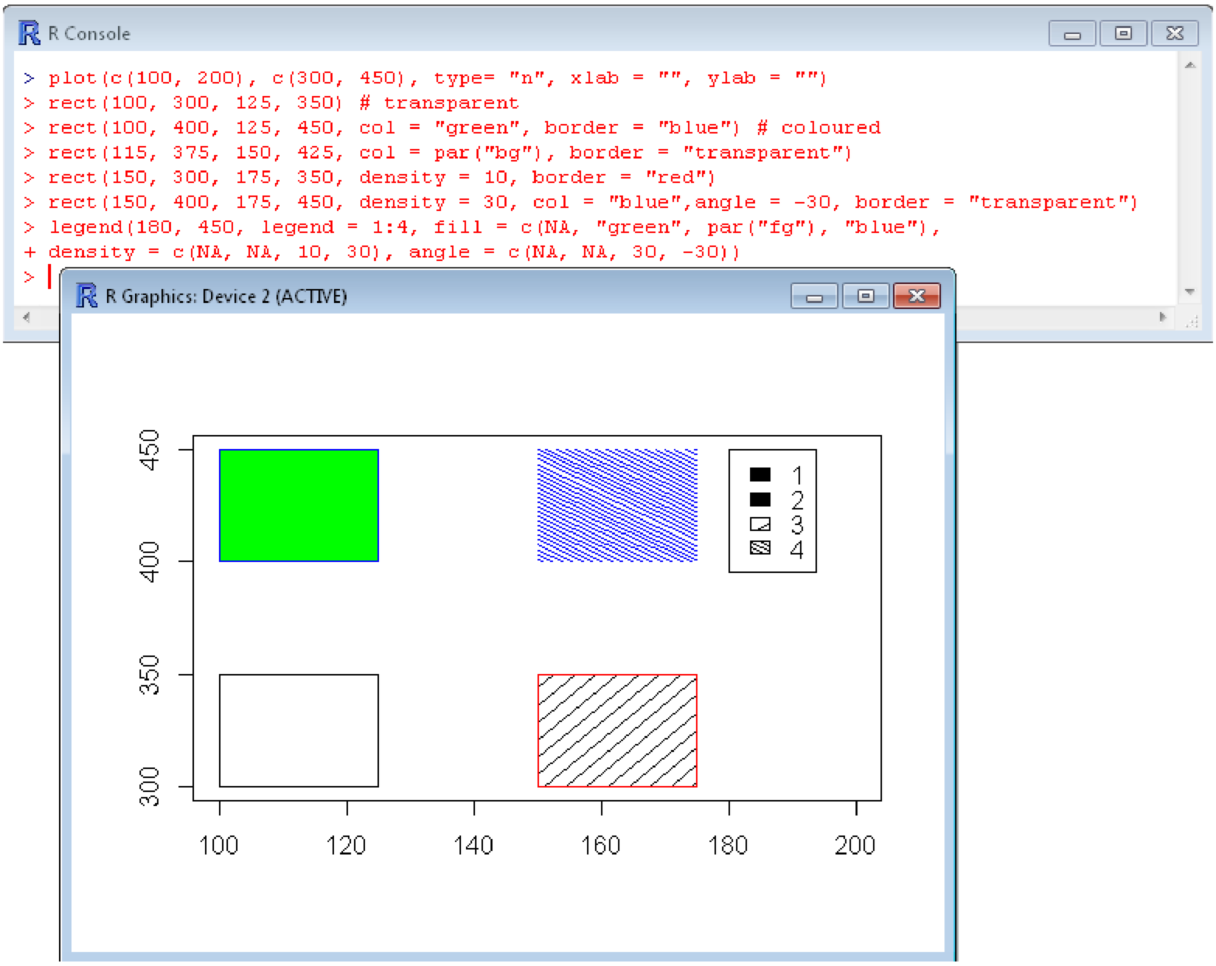


Figure 46Graph avec rectangles dessinés

Exemple 110.: Dessiner des cercles

R 3.0.2

Il faudra faire appel au package **plotrix** pour dessiner des cercles ou des disques avec la commande **draw.circle( )** :

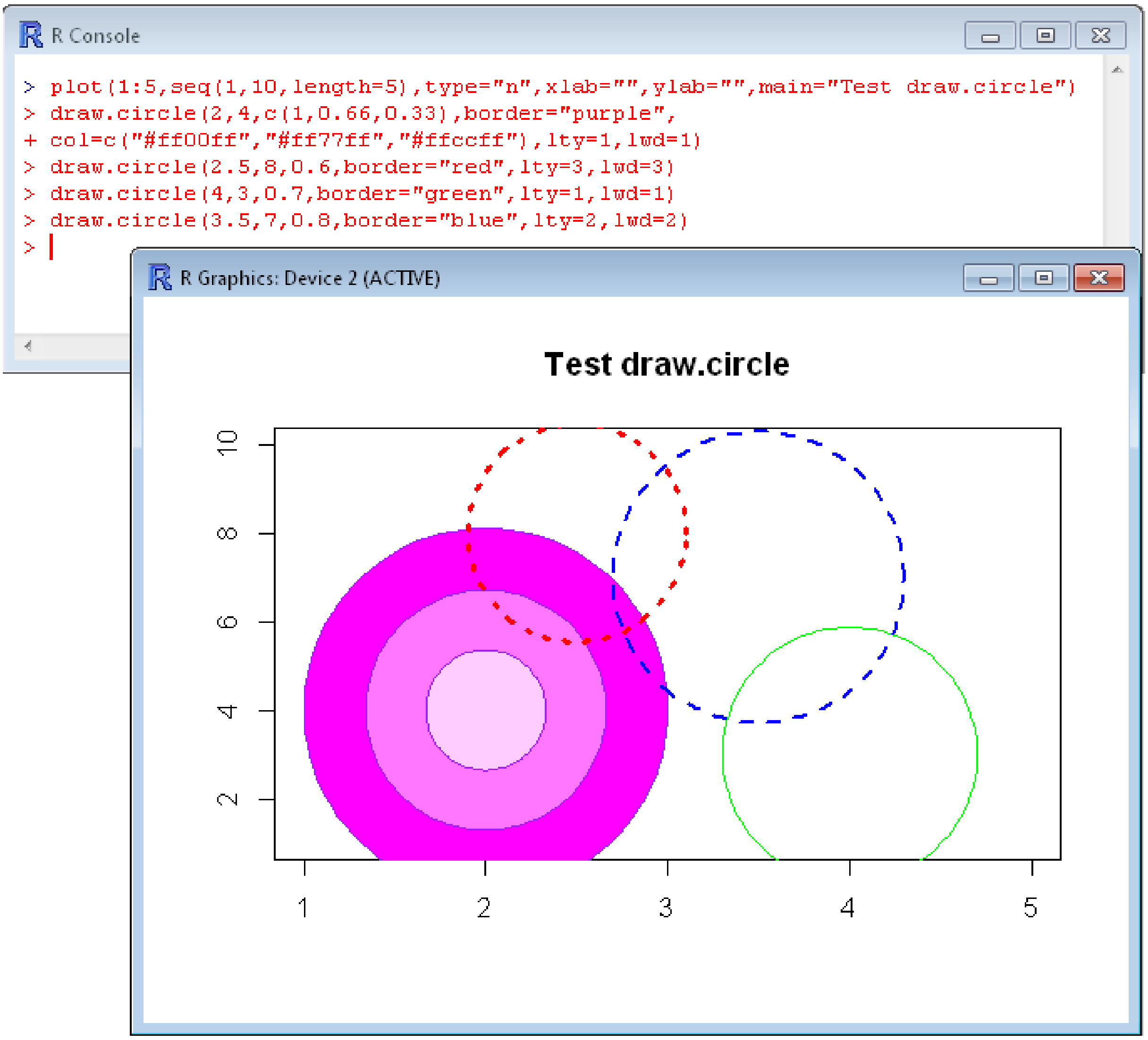
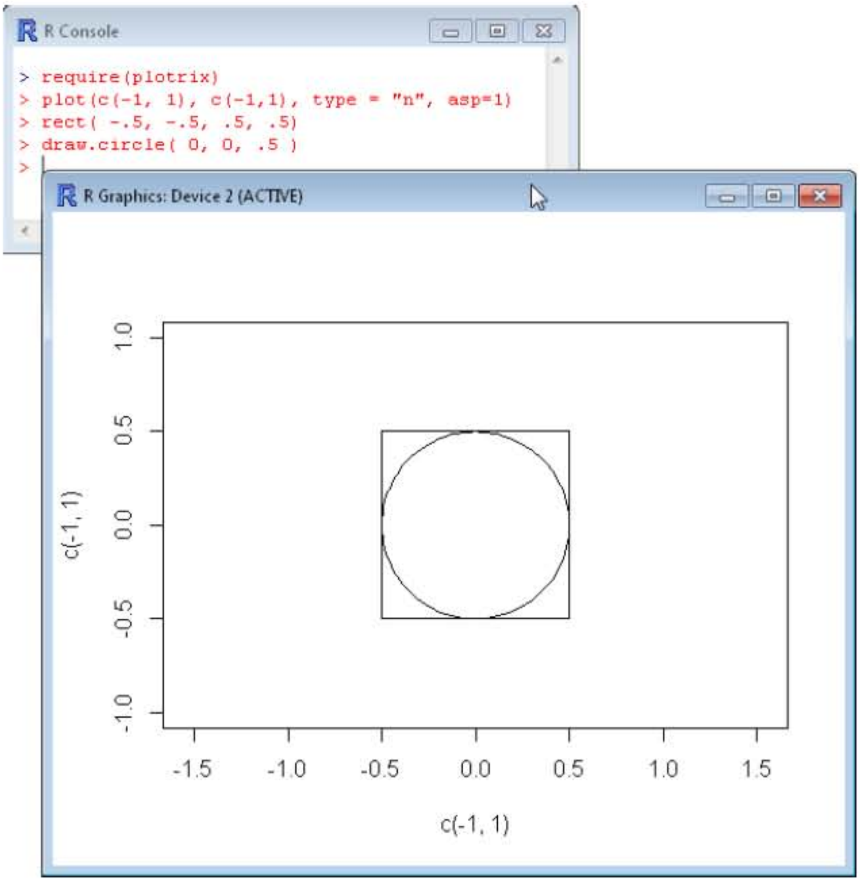


Figure 47 Graph avec cercles et disques dessinés

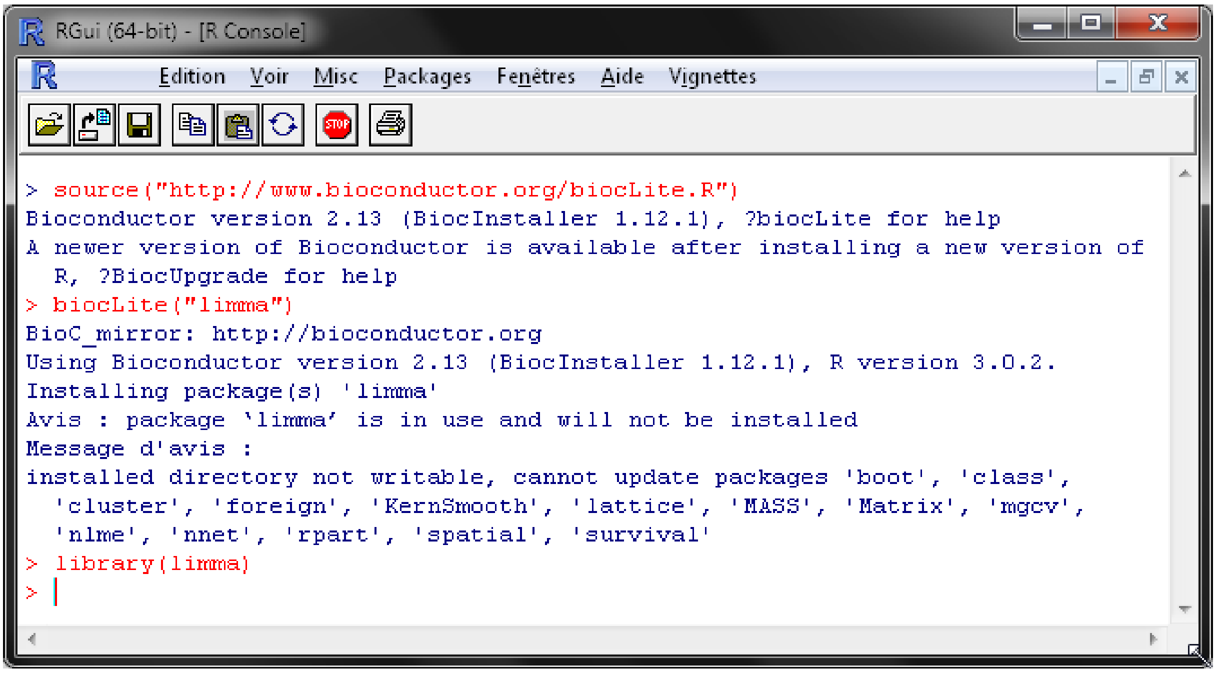
Libre à nous ensuite de mélanger rectangles/carrés et cercles sur de mêmes diagrammes:


R Console
> require(plotrix)
> plot(c(-1, 1), c(-1,1), type = "n", asp=1)
> rect(-.5, -.5, .5, .5)
> draw.circle(0, 0, .5)
R Graphics: Device 2 (ACTIVE)
c(-1, 1)
c(-1, 1)

Diagrammes de Venn

En bioinformatique nous avons régulièrement des vecteurs dont les composantes sont des séquençages. L'idée est de voir par exemple entre trois bactéries quels sont les dénominateurs communs au niveau des gènes. Nous pouvons alors typiquement représenter cela avec un diagramme de Venn. Comme il s'agit le plus souvent d'une représentation de cercles, j'ai jugé bon de le mettre ici. Petit couac cependant... il doit y avoir une méthode plus simple que ce qui est proposé ci-dessous... mais à chercher (n'hésitez pas à me dire si vous avez une astuce).

Nous allons devoir d'abord utiliser le package **Bioconductor** qui n'est pas dans CRAN et une sous libraire qui se nomme **limma**:



Ensuite, nous faisons quelques manipulations de nos ensembles qui servent d'exemples (exemple très scolaire dans le cas présent):

```
> # Jeux d'exemple
> set1<-c("A","G","B","C","D")
> set2<-c("B","F","G","W","O","Y")
> set3<-c("B","X","Y","G","A","M")
>
> # Lettre uniques
> universe <- sort(unique(c(set1, set2, set3)));universe
 [1] "A" "B" "C" "D" "F" "G" "M" "O" "W" "X" "Y"
>
> # On génère une matrice avec les ensembles en colonnes et les éléments en ligne
> Counts <- matrix(0, nrow=length(universe), ncol=3)
> # On remplit la matrice
> for (i in 1:length(universe)) {
+    Counts[i,1] <- universe[i] %in% set1
+    Counts[i,2] <- universe[i] %in% set2
+    Counts[i,3] <- universe[i] %in% set3
+ }
>
> # On nomme les colonnes
> colnames(Counts) <- c("set1","set2","set3");Counts
      set1 set2 set3
 [1,]    1    0    1
 [2,]    1    1    1
 [3,]    1    0    0
 [4,]    1    0    0
 [5,]    0    1    0
 [6,]    1    1    1
 [7,]    0    0    1
 [8,]    0    1    0
 [9,]    0    1    0
[10,]    0    0    1
[11,]    0    1    1
>
```

Et ensuite il nous reste plus qu'à utiliser les commandes **vennDiagram( )** et **vennCounts( )** de la librairie **limma** pour arriver au résultat attendu:

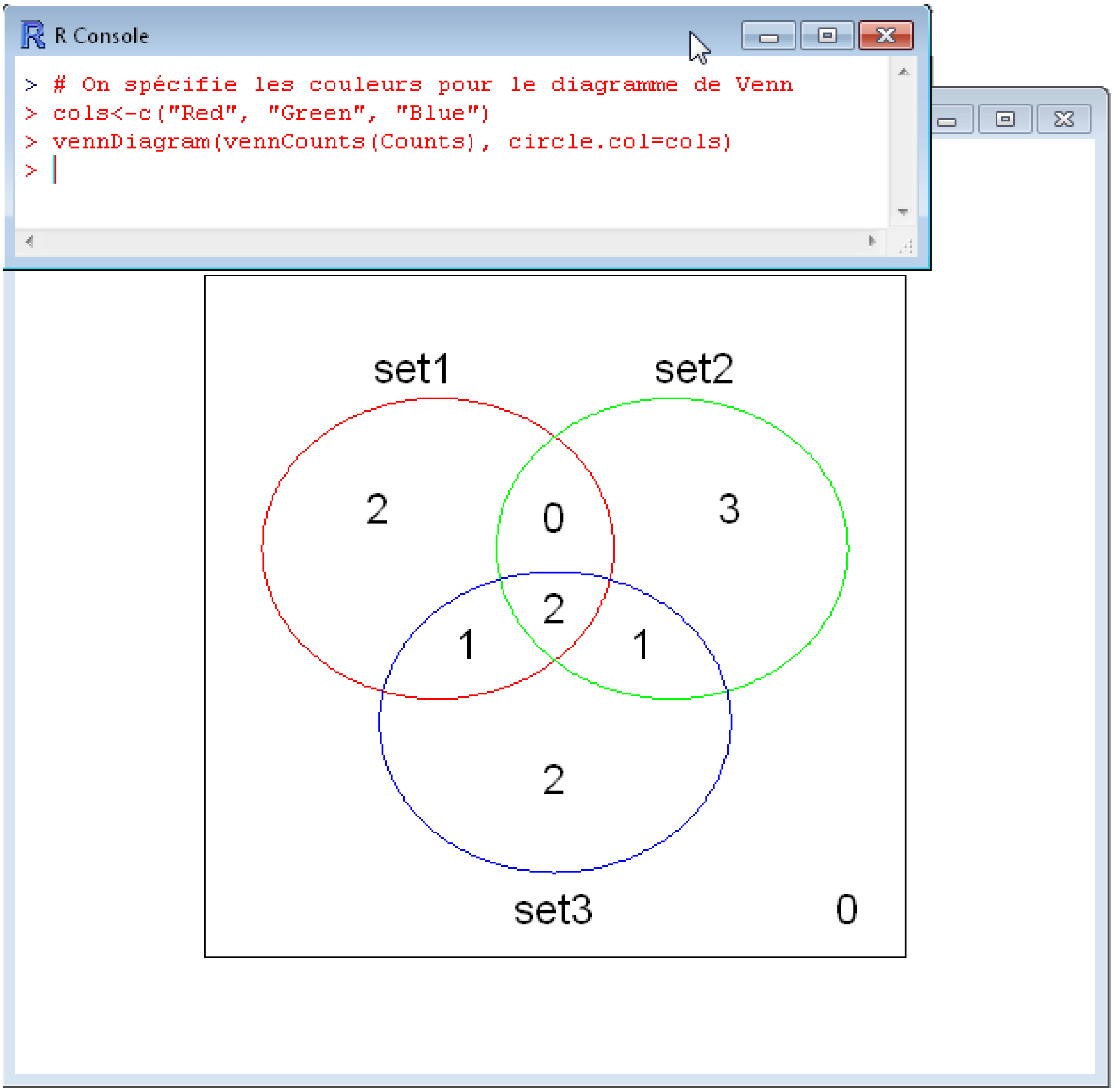
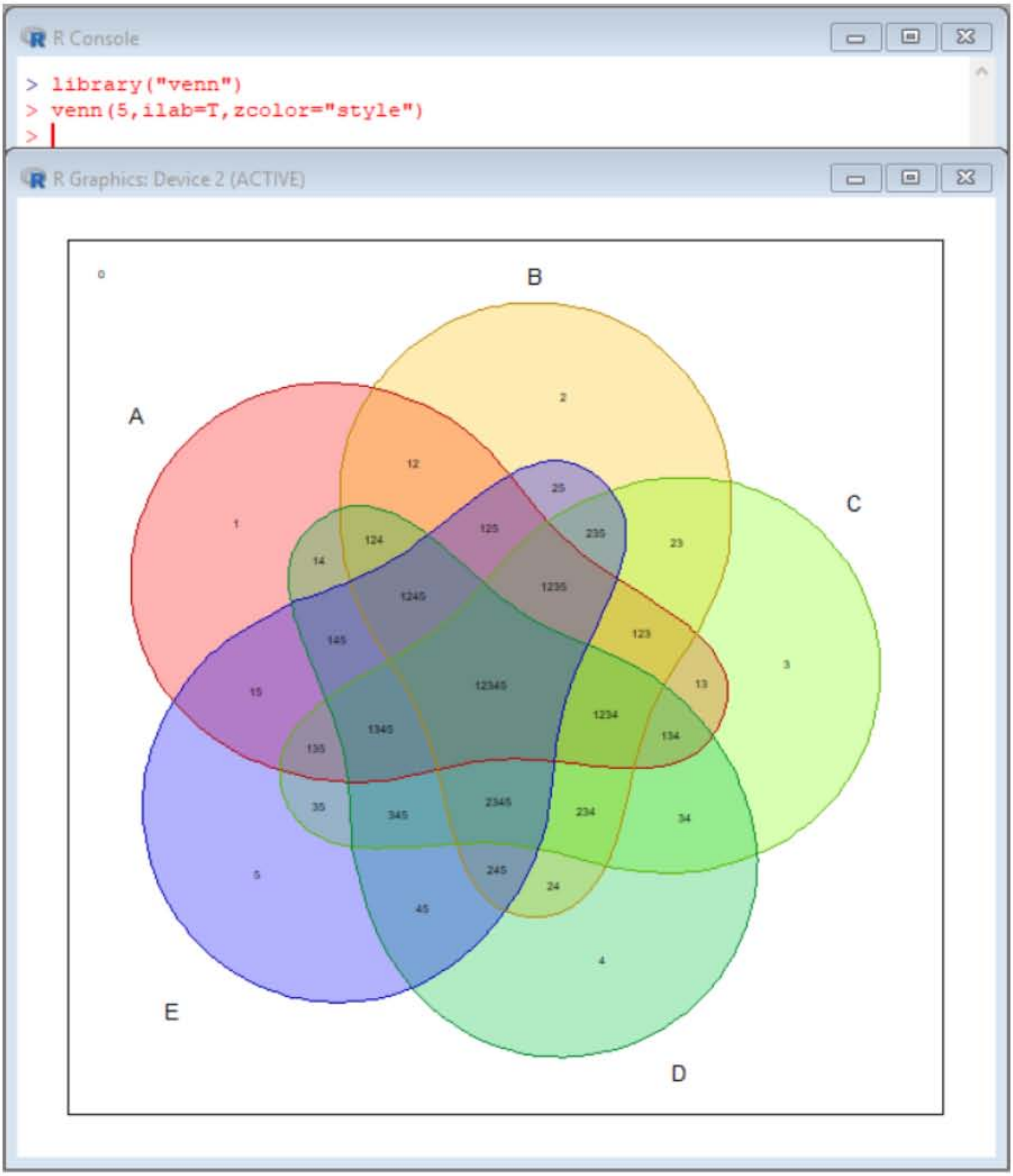
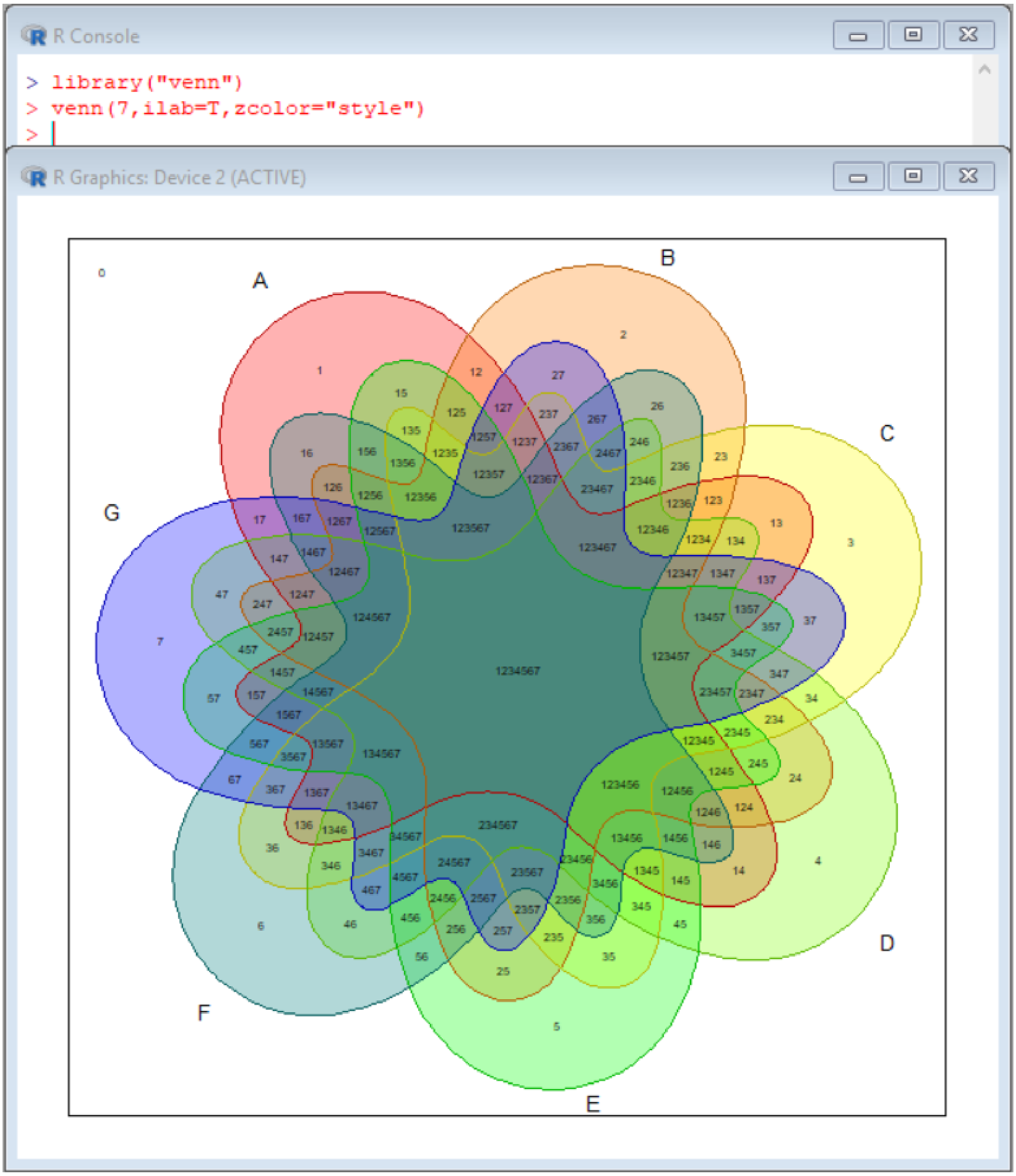


Figure 48 Graph de type diagramme de Venn

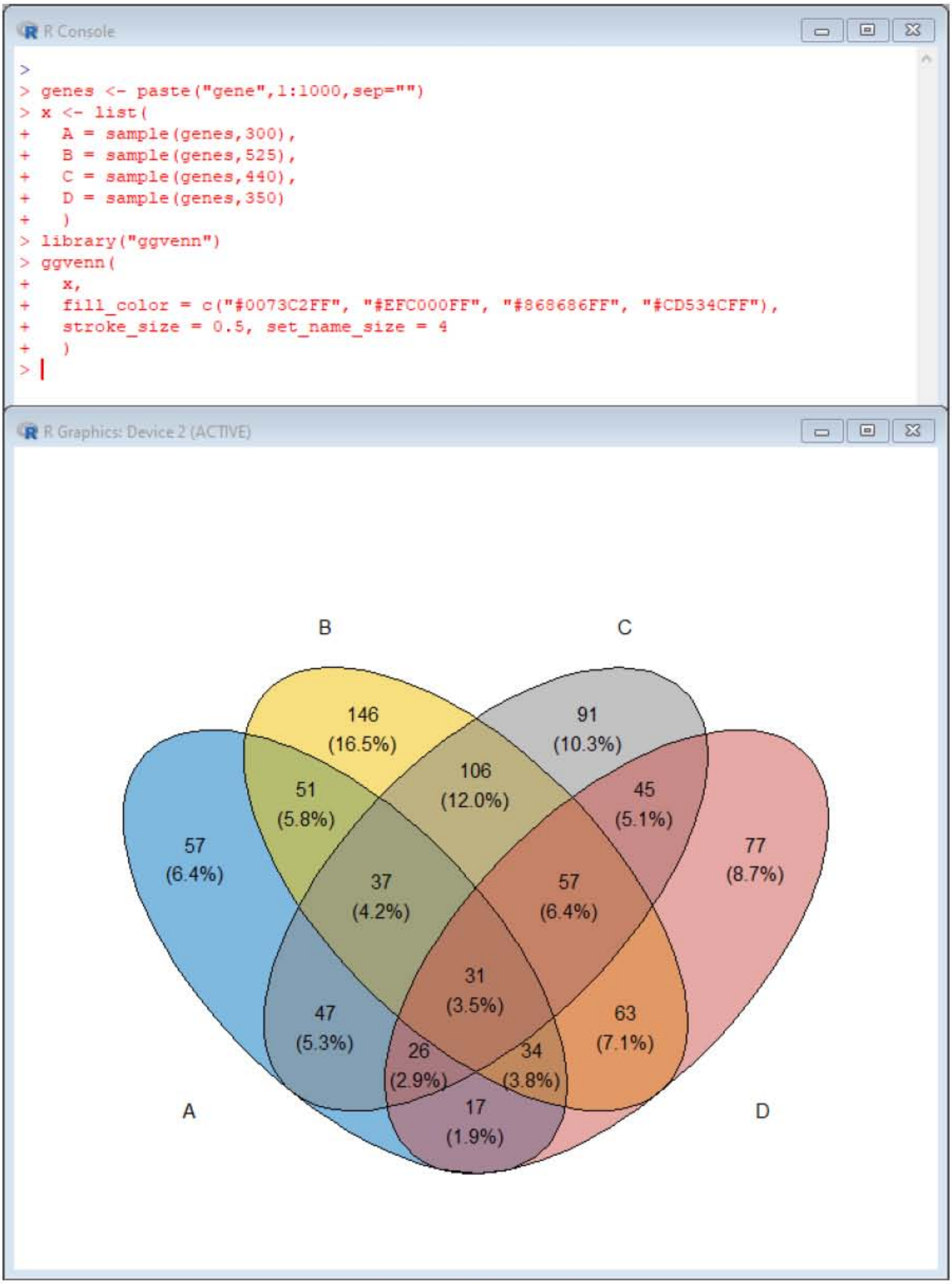
Laborieux...

Indiquons aussi le package **venn** qui avec la commande **venn( )** permet d'obtenir des diagrammes de Venn plus que satisfaisant esthétiquement parlant:


R Console
> library("venn")
> venn(5,ilab=T,zcolor="style")
>
R Graphics: Device 2 (ACTIVE)
A
B
C
D
E



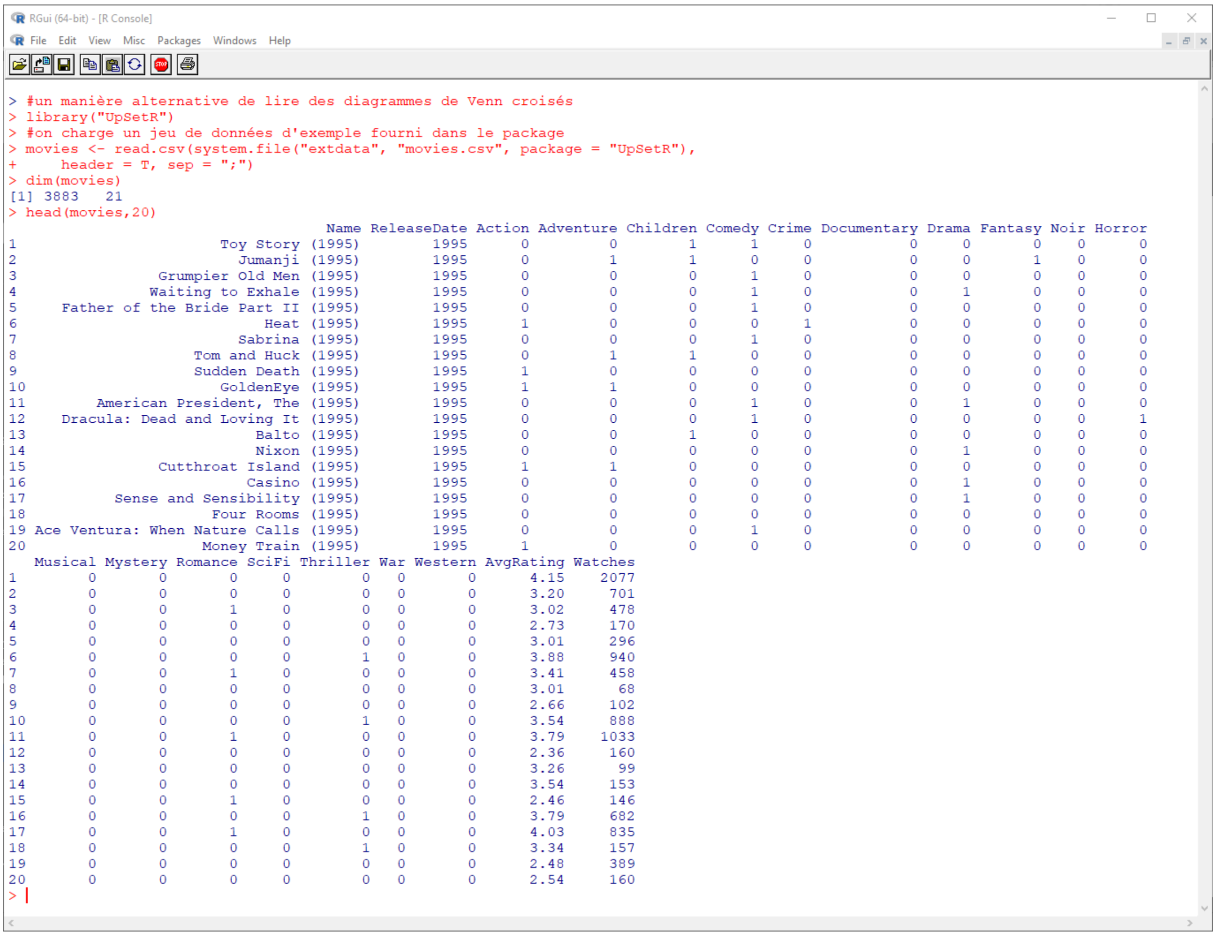
On a aussi sinon avec le package **ggvenn** et ça fonction **ggvenn( )**:


R Console
>
> genes <- paste("gene",1:1000,sep="")
> x <- list(
+   A = sample(genes,300),
+   B = sample(genes,525),
+   C = sample(genes,440),
+   D = sample(genes,350)
+   )
> library("ggvenn")
> ggvenn(
+   x,
+   fill_color = c("#0073C2FF", "#EFC000FF", "#868686FF", "#CD534CFF"),
+   stroke_size = 0.5, set_name_size = 4
+   )
>
R Graphics: Device 2 (ACTIVE)
B
C
A
D
146 (16.5%)
91 (10.3%)
106 (12.0%)
51 (5.8%)
45 (5.1%)
57 (6.4%)
77 (8.7%)
37 (4.2%)
57 (6.4%)
31 (3.5%)
47 (5.3%)
63 (7.1%)
26 (2.9%)
34 (3.8%)
17 (1.9%)

Diagrammes UpSet

Nous savons que les diagrammes de Venn ne sont pas ce qu'il y a de plus efficace à lire surtout lorsque leur nombre devient grand. Pour remédier à cela, un solution consiste à utiliser les diagrammes upset, disponible dans R à l'aide du package **upset** et de sa fonction **upset( )**.

Voyons cela avec l'exemple inclus dans le package et en chargeant d'abord le jeu de données qui y est inclus:



```
> #un manière alternative de lire des diagrammes de Venn croisés
> library("UpSetR")
> #on charge un jeu de données d'exemple fourni dans le package
> movies <- read.csv(system.file("extdata", "movies.csv", package = "UpSetR"),
+     header = T, sep = ";")
> dim(movies)
[1] 3883   21
> head(movies,20)
                                   Name ReleaseDate Action Adventure Children Comedy Crime Documentary Drama Fantasy Noir Horror
1                      Toy Story (1995)        1995      0         0        1      1     0           0     0       0    0      0
2                        Jumanji (1995)        1995      0         1        1      0     0           0     0       1    0      0
3               Grumpier Old Men (1995)        1995      0         0        0      1     0           0     0       0    0      0
4              Waiting to Exhale (1995)        1995      0         0        0      1     0           0     1       0    0      0
5    Father of the Bride Part II (1995)        1995      0         0        0      1     0           0     0       0    0      0
6                           Heat (1995)        1995      1         0        0      0     1           0     0       0    0      0
7                        Sabrina (1995)        1995      0         0        0      1     0           0     0       0    0      0
8                   Tom and Huck (1995)        1995      0         1        1      0     0           0     0       0    0      0
9                   Sudden Death (1995)        1995      1         0        0      0     0           0     0       0    0      0
10                     GoldenEye (1995)        1995      1         1        0      0     0           0     0       0    0      0
11       American President, The (1995)        1995      0         0        0      1     0           0     1       0    0      0
12   Dracula: Dead and Loving It (1995)        1995      0         0        0      1     0           0     0       0    0      1
13                         Balto (1995)        1995      0         0        1      0     0           0     0       0    0      0
14                         Nixon (1995)        1995      0         0        0      0     0           0     1       0    0      0
15              Cutthroat Island (1995)        1995      1         1        0      0     0           0     0       0    0      0
16                        Casino (1995)        1995      0         0        0      0     0           0     1       0    0      0
17         Sense and Sensibility (1995)        1995      0         0        0      0     0           0     1       0    0      0
18                    Four Rooms (1995)        1995      0         0        0      0     0           0     0       0    0      0
19 Ace Ventura: When Nature Calls (1995)        1995      0         0        0      1     0           0     0       0    0      0
20                   Money Train (1995)        1995      1         0        0      0     0           0     0       0    0      0
   Musical Mystery Romance SciFi Thriller War Western AvgRating Watches
1        0       0       0     0        0   0       0      4.15    2077
2        0       0       0     0        0   0       0      3.20     701
3        0       0       1     0        0   0       0      3.02     478
4        0       0       0     0        0   0       0      2.73     170
5        0       0       0     0        0   0       0      3.01     296
6        0       0       0     0        1   0       0      3.88     940
7        0       0       1     0        0   0       0      3.41     458
8        0       0       0     0        0   0       0      3.01      68
9        0       0       0     0        0   0       0      2.66     102
10       0       0       0     0        1   0       0      3.54     888
11       0       0       1     0        0   0       0      3.79    1033
12       0       0       0     0        0   0       0      2.36     160
13       0       0       0     0        0   0       0      3.26      99
14       0       0       0     0        0   0       0      3.54     153
15       0       0       1     0        0   0       0      2.46     146
16       0       0       0     0        1   0       0      3.79     682
17       0       0       1     0        0   0       0      4.03     835
18       0       0       0     0        1   0       0      3.34     157
19       0       0       0     0        0   0       0      2.48     389
20       0       0       0     0        0   0       0      2.54     160
>
```

Et le graphique correspondant:


R Console
> upset(movies, nsets = 6, point.size = 3.5, line.size = 2,
+   mainbar.y.label = "Genre Intersections", sets.x.label = "Movies Per Genre",
+   text.scale = c(1.3, 1.3, 1, 1, 2, 0.75))
>
R Graphics: Device 2 (ACTIVE)
Genre Intersections
900
600
300
0
Horror
Romance
Thriller
Action
Comedy
Drama
1500
1000
500
0
Movies Per Genre

Exemple 111.: Dessiner des ellipses

R 3.0.2

Évidemment... Nous ne pouvons pas ne pas montrer comme dessiner des ellipses (pensez ne serait qu'à cause des fonctions bivariées Normales ou des copulas gaussiens).

Pour cela, nous allons aussi devoir utiliser la package **plotrix** avec la commande **draw.ellipse( )** :

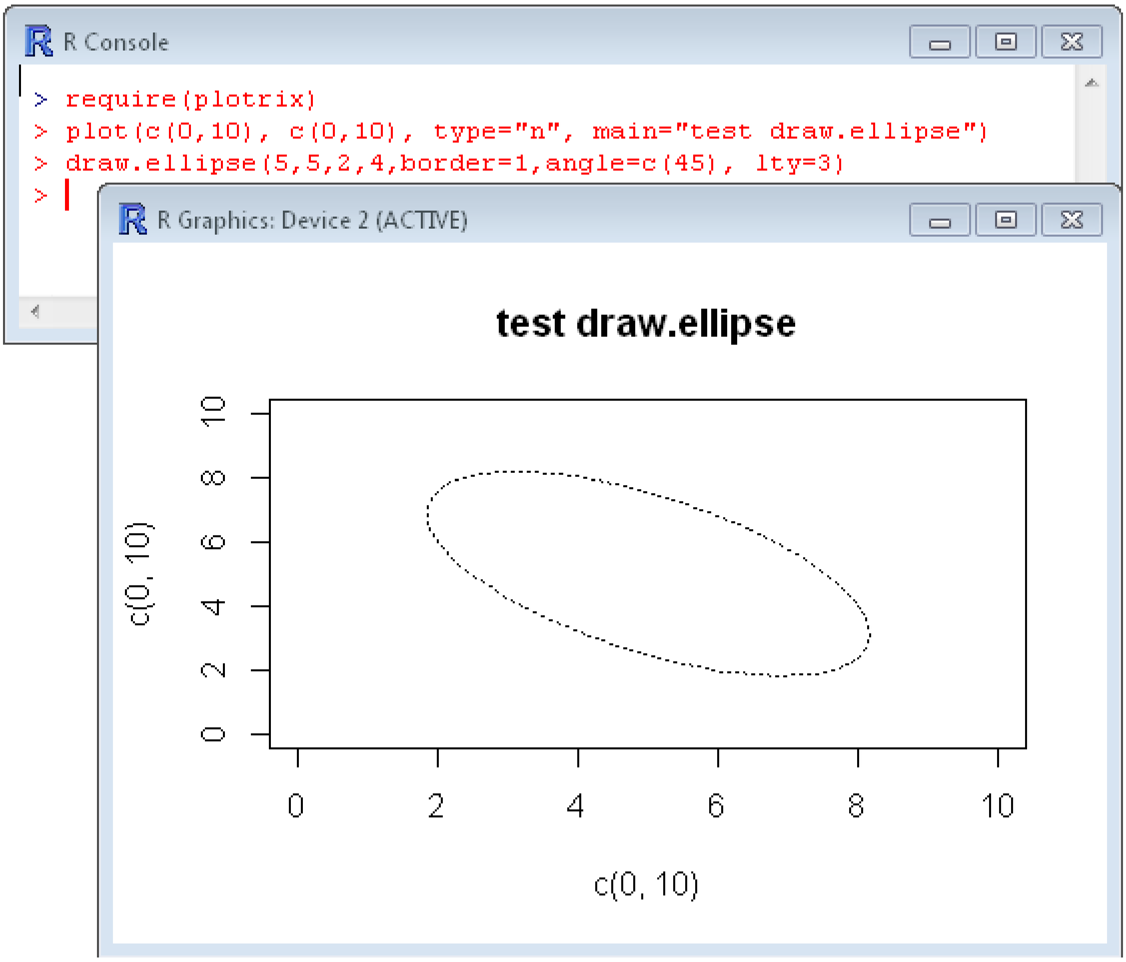
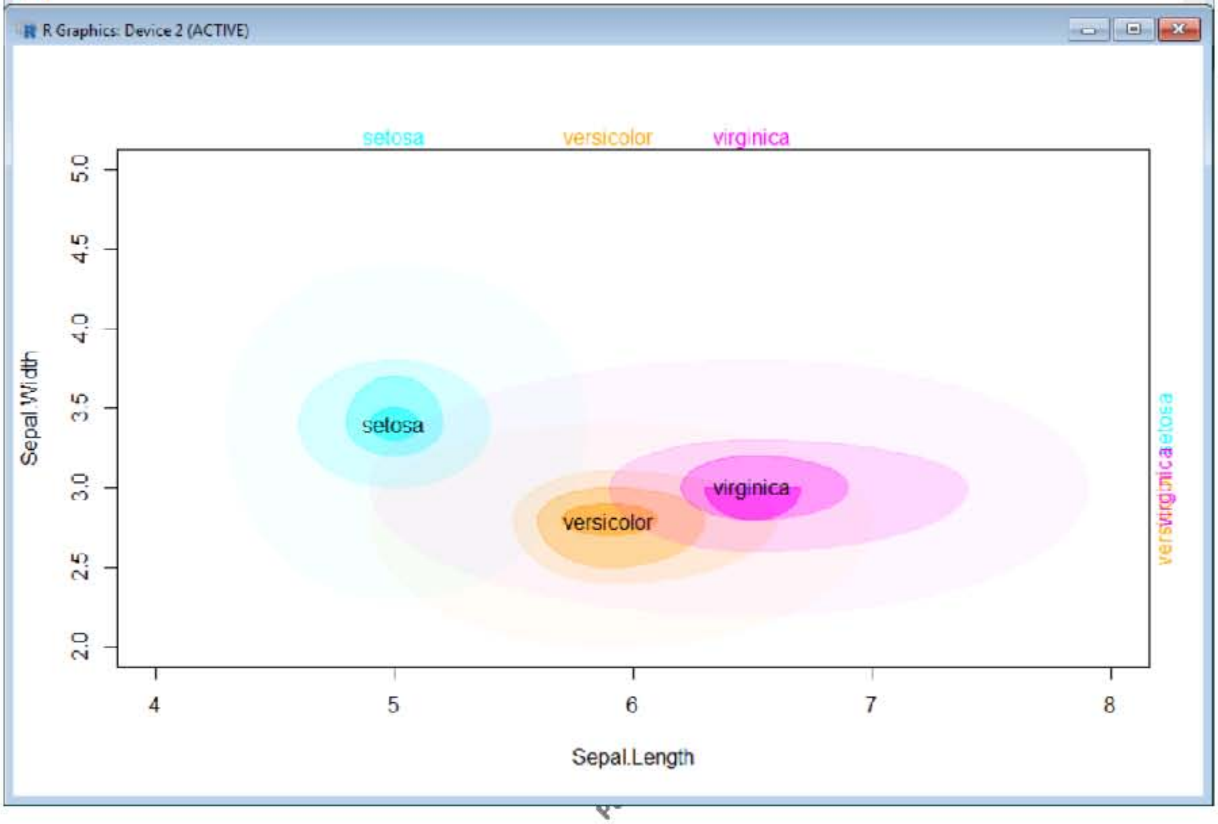


Figure 49 Grap avec ellipses dessinées

Plus sérieusement et scientifiquement avec des ellipses, nous avons aussi la fonction **ellipseplot( )** du package **elliplot** qui fait des merveilles:

```
> library("elliplot")
> ellipseplot(iris[,c("Species","Sepal.Length")],iris[,c("Species", "Sepal.Width")],
+ xlim=c(4,8), ylim=c(2,5),col=c("cyan", "orange", "magenta"))
>
```



Exemple 112.: Dessiner des polygones

R 3.0.2

Savoir dessiner des polygones est extrêmement important dans R car dans de nombreux cas pratiques nous remplissons des surfaces se trouvant en-dessous de la courbe d'une fonction (particulièrement dans les domaines des probabilités ou des statistiques géographiques où les territoires sont définis par des polygones).

Voyons un exemple simple et trivial pour commencer:

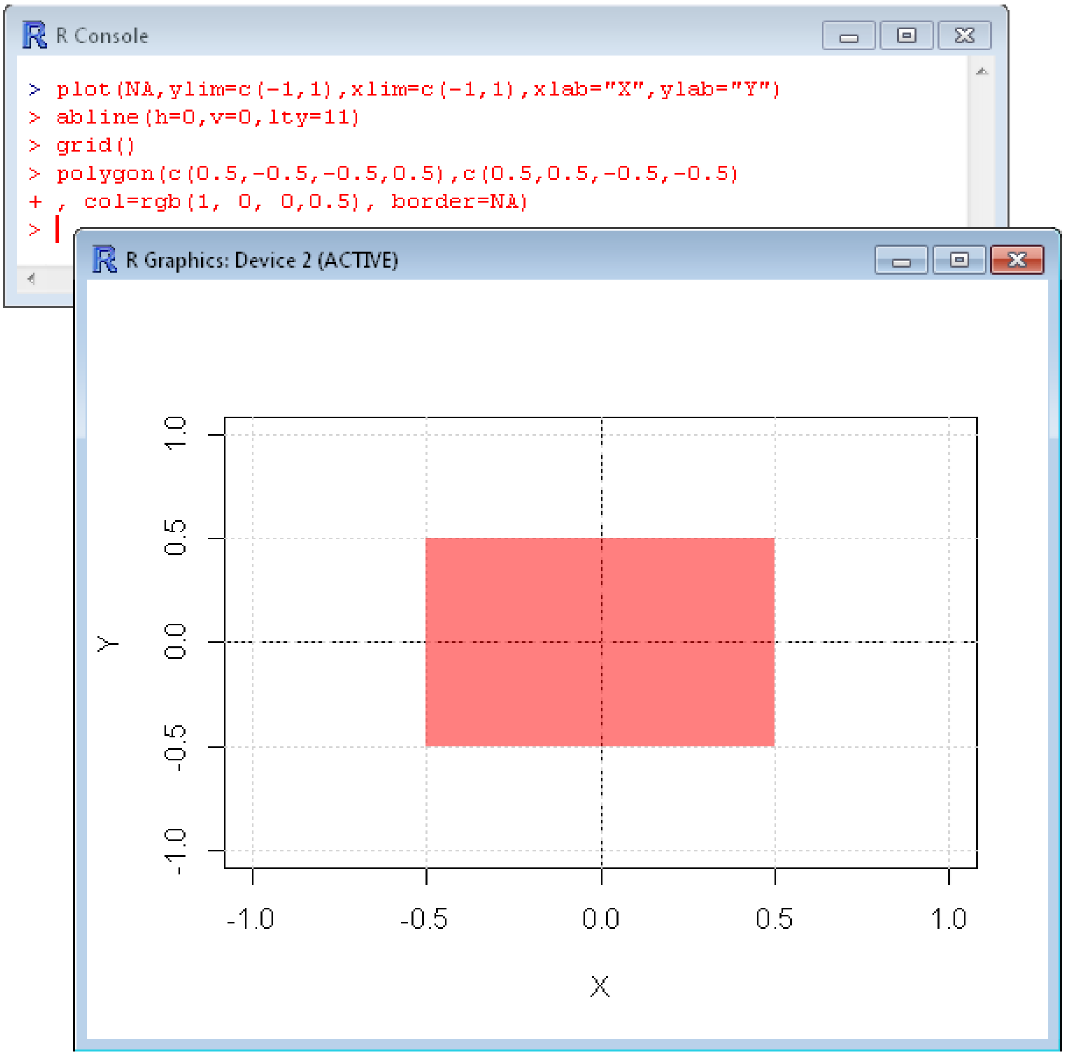
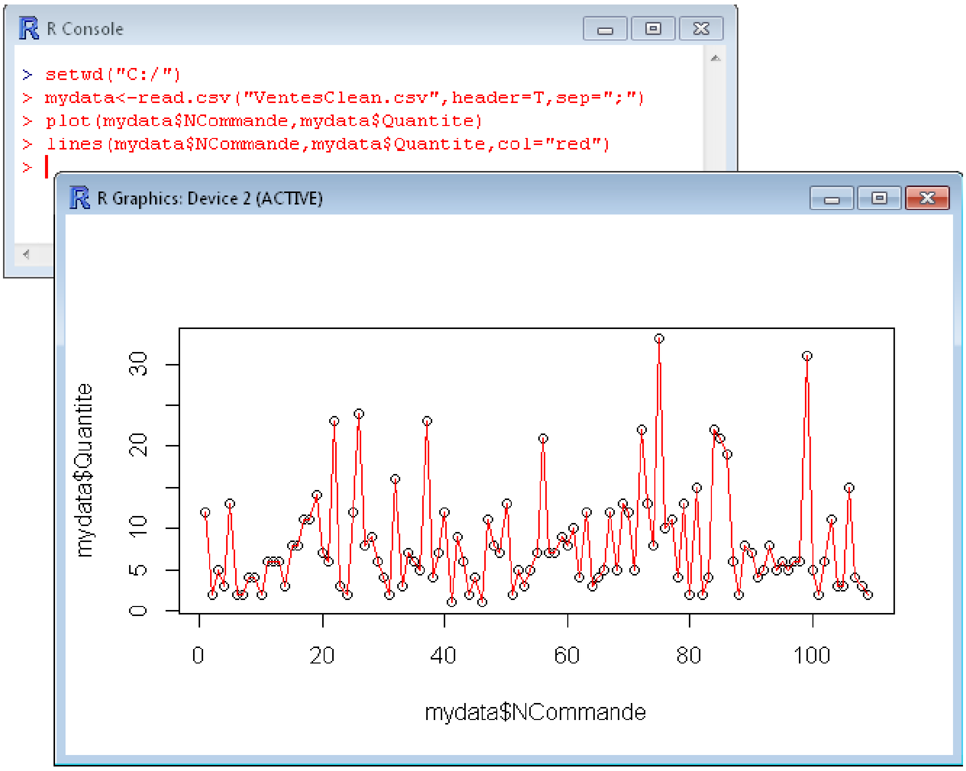
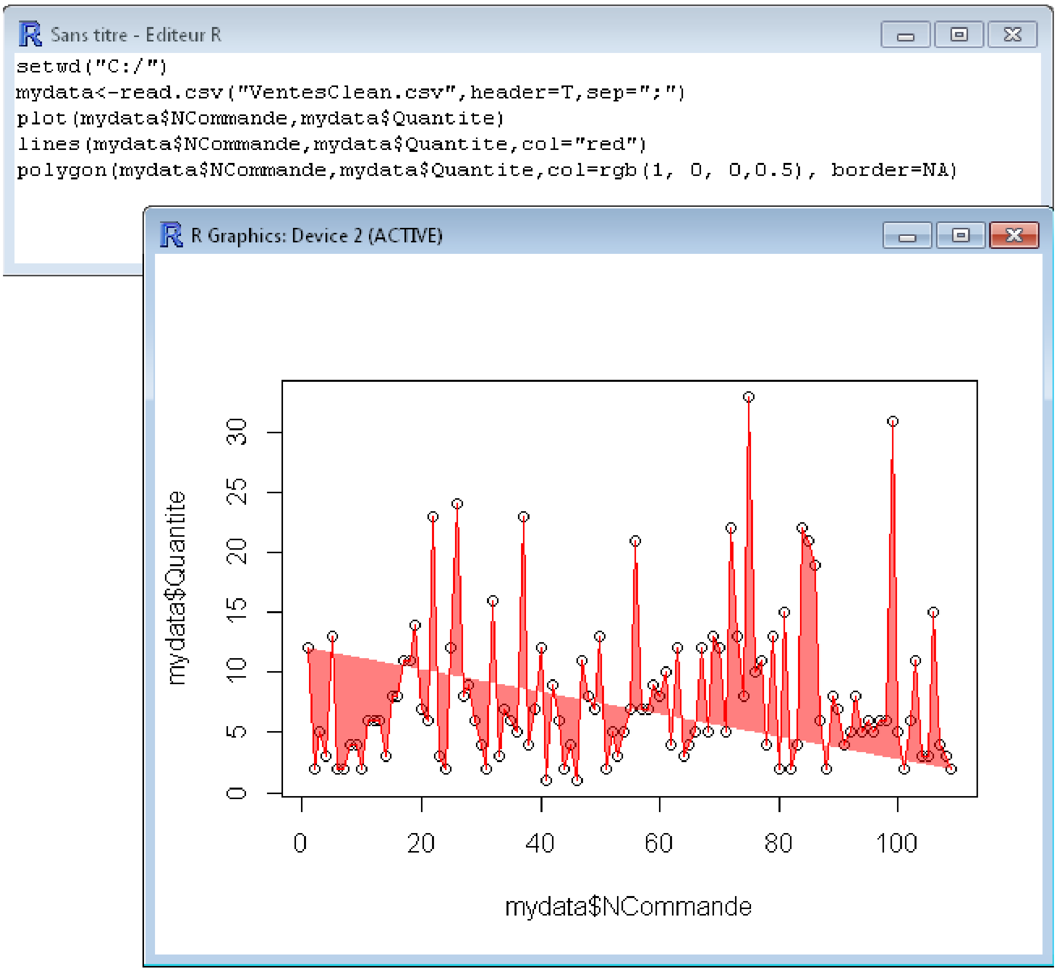


Figure 50 Graph avec polygones dessinés

Donc à partir de là on devine assez aisément la suite (ou voir les exemples dans l'aide de la commande **polygon( )**). Par exemple, si nous partons de:



Et avec les polygones en plus:



Mais ce n'est évidemment pas le résultant que nous voulons! Il faut rajouter artificiellement des points au début et à la fin tel que:

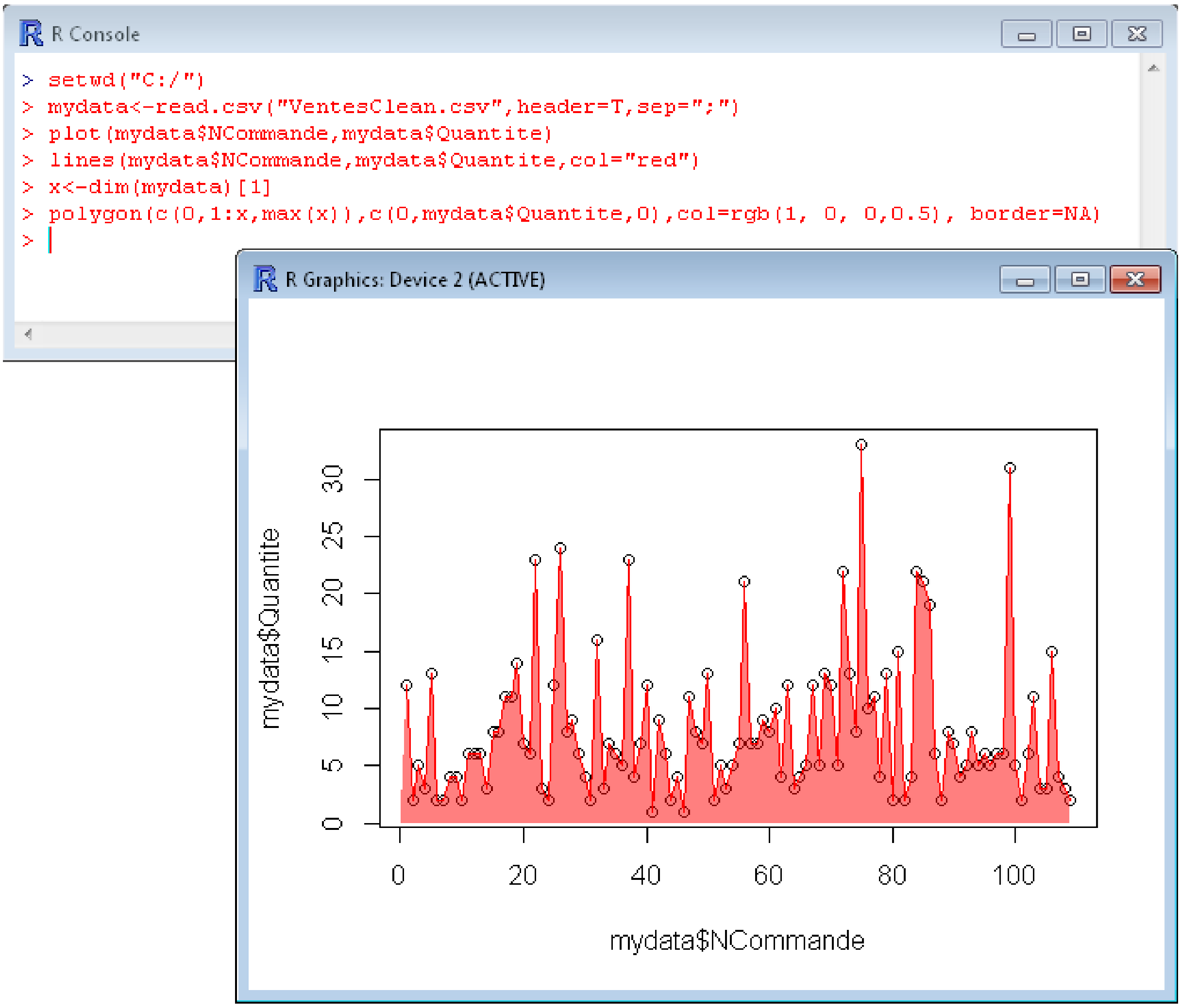


Figure 51 Graph de type aire (basé sur l'usage de polygones)

Ou dans un autre style super important dans la pratique:

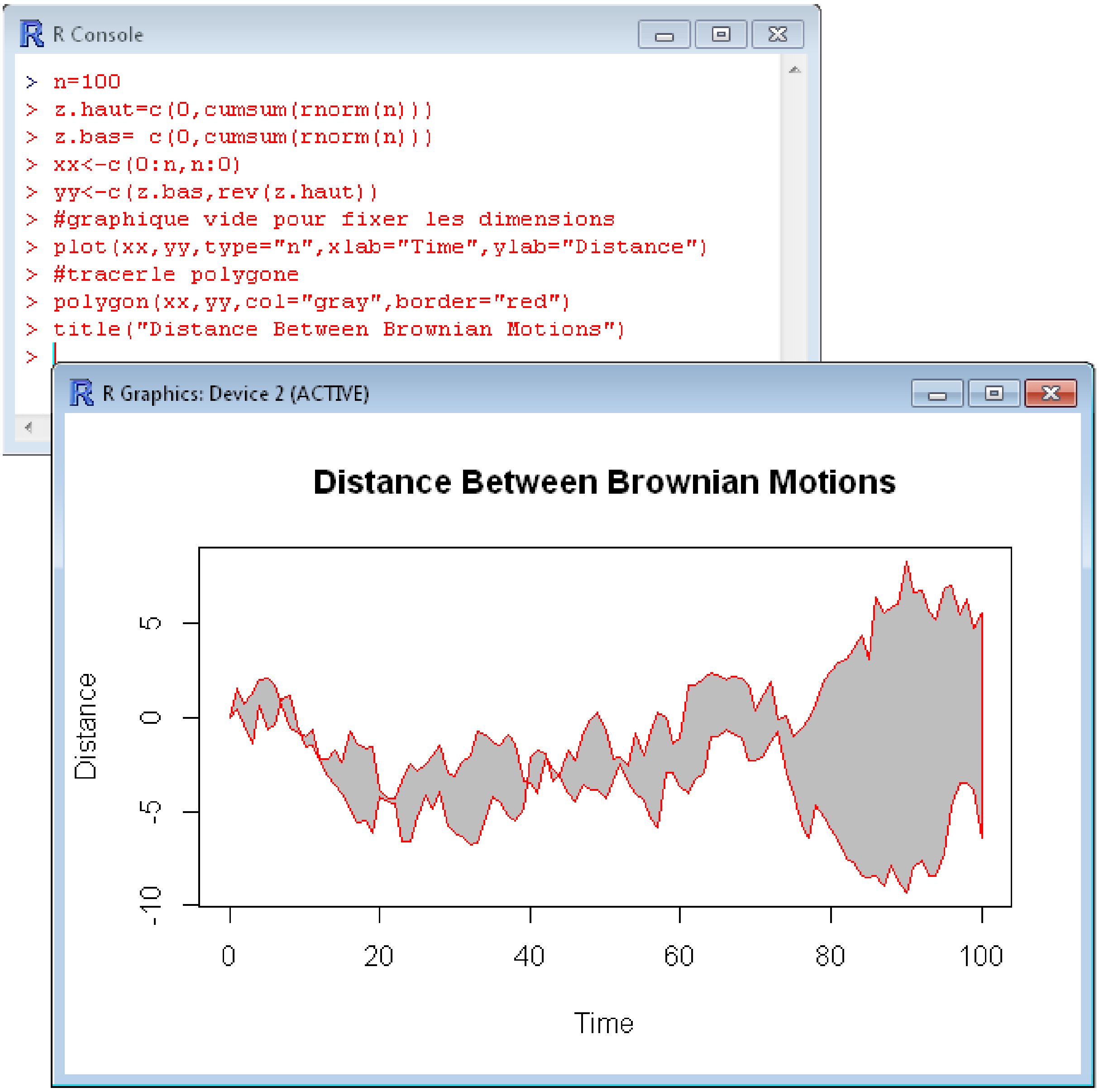
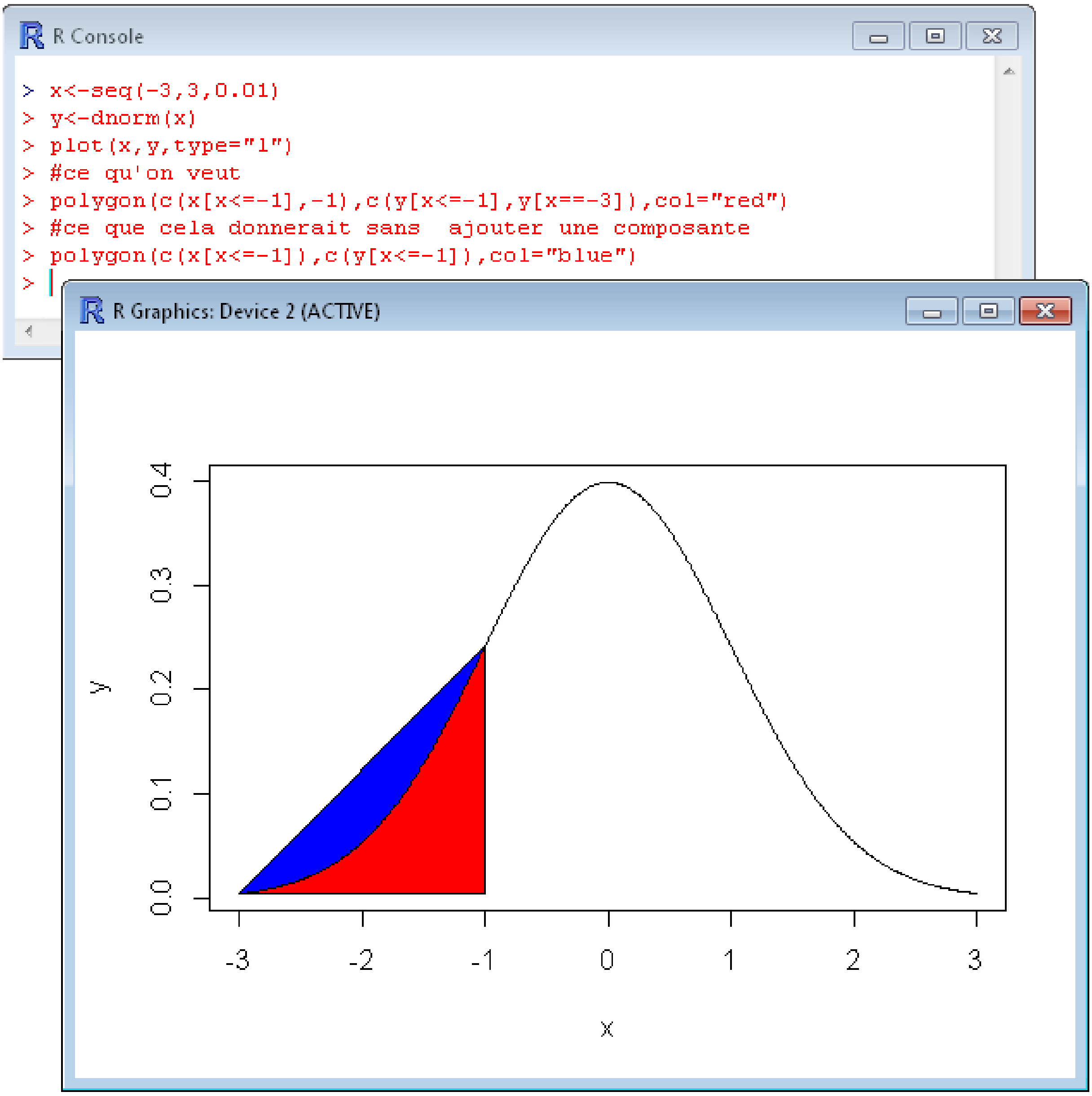


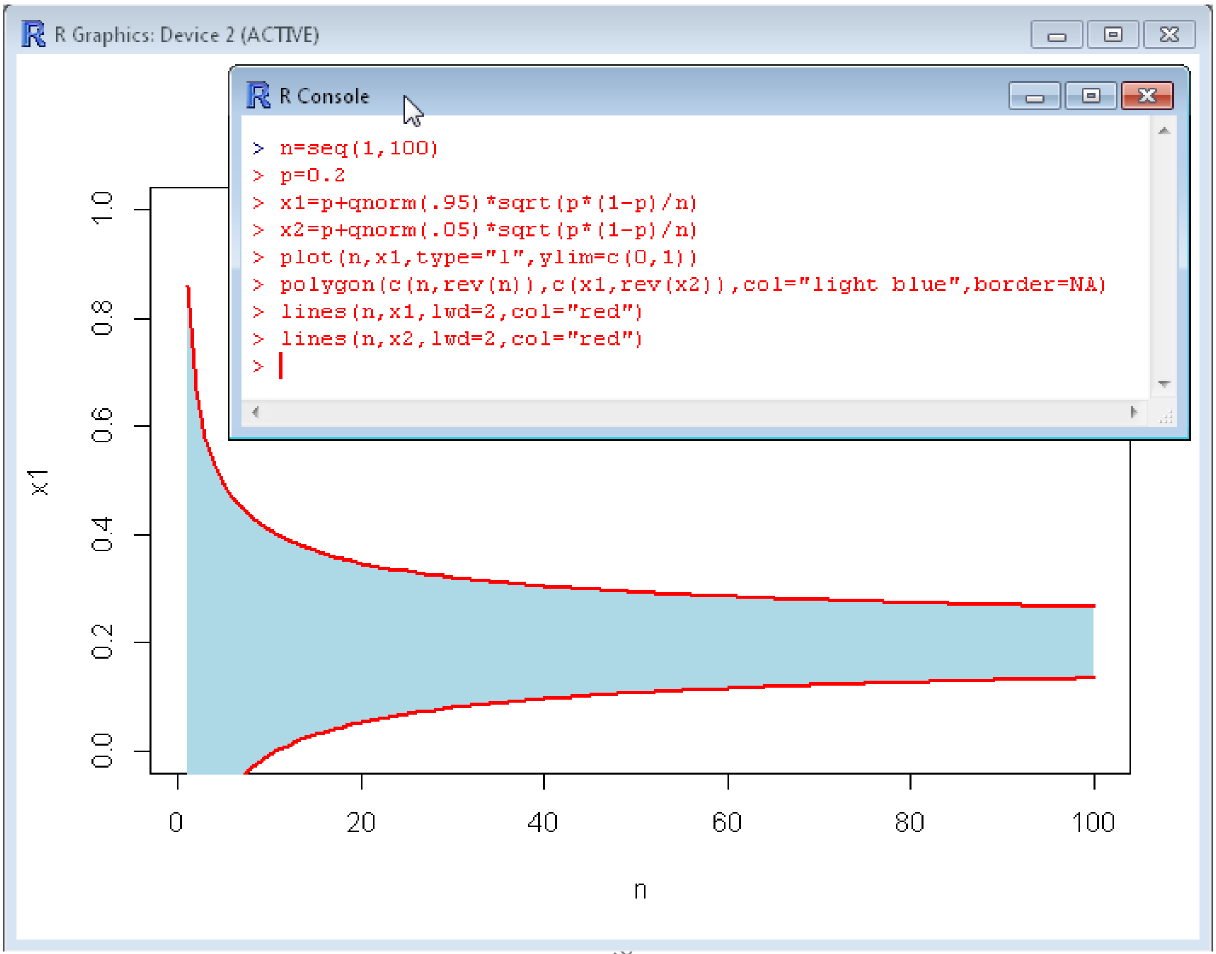
Figure 52 Graph d'intervalle (basé sur l'usage de polygones)

Après, et toujours sur la même idée, on peut superposer plusieurs tracés de lignes remplies les unes par dessus les autres.

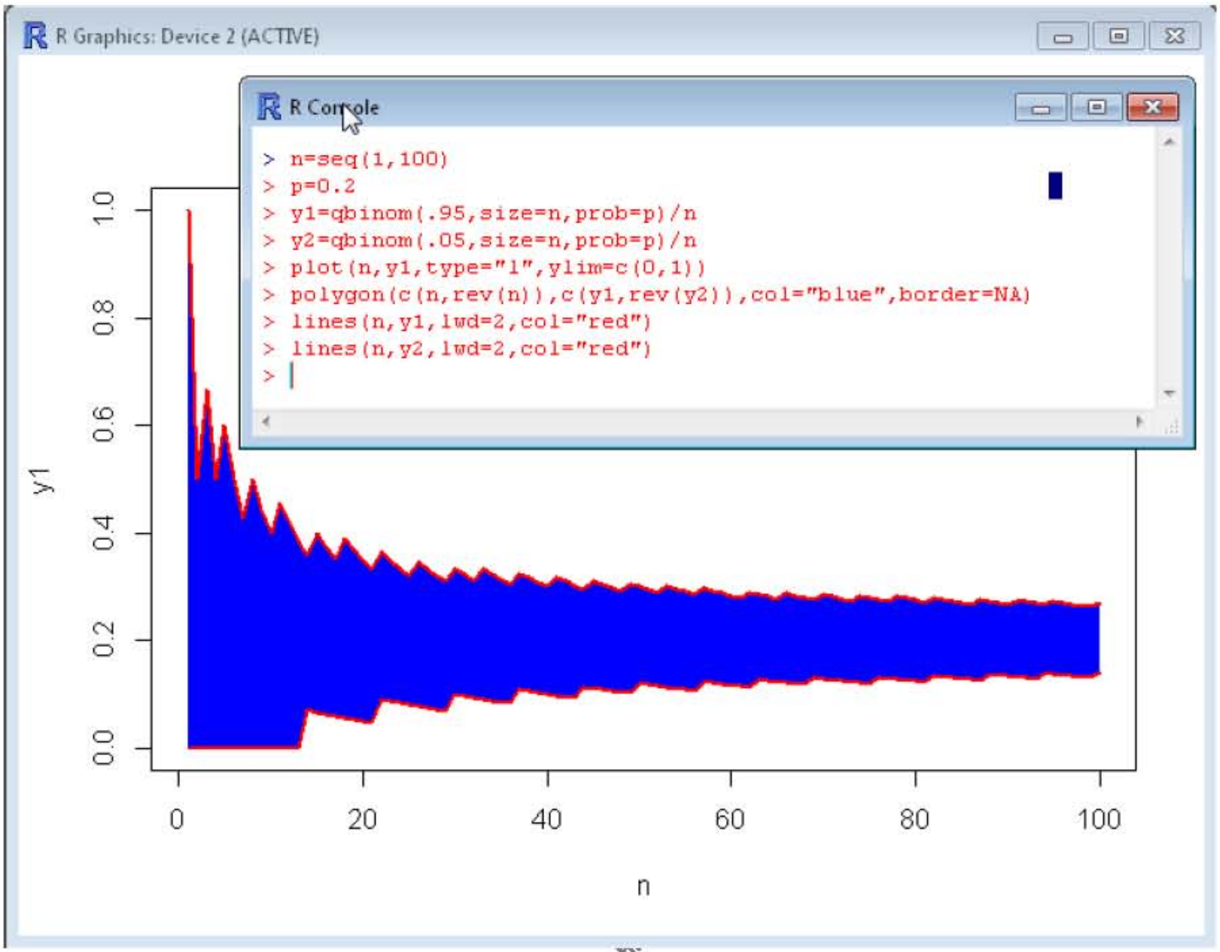
Ou toujours dans les cas simples (nous reviendrons sur les distributions de probabilités plus loin):



Ou encore pour représenter l'évolution d'un intervalle de confiance (dans le cas présent celui d'une proportion):



Ou en étant plus exacte puisque ci-dessus nous avons utilisé l'approximation par une loi Normale:


R Graphics: Device 2 (ACTIVE)
R Console
> n=seq(1,100)
> p=0.2
> y1=qbinom(.95,size=n,prob=p)/n
> y2=qbinom(.05,size=n,prob=p)/n
> plot(n,y1,type="l",ylim=c(0,1))
> polygon(c(n,rev(n)),c(y1,rev(y2)),col="blue",border=NA)
> lines(n,y1,lwd=2,col="red")
> lines(n,y2,lwd=2,col="red")
y1
n
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

Exemple 113.: Diagrammes de matrices

R 3.5.1

Voici comment on peut faire un graph du contenu d'une matrice, ce qui est assez courant dans le domaine du traitement du signal mais aussi dans l'analyse visuelle des matrices de corrélations (même si nous verrons pour ces dernières des techniques plus adaptées) et commençons par le package **lattice** et la fonction **levelplot( )** mais aussi la fonction de base **colorRampPalette( )**:

R Console

```
> library("lattice")
> #matrice 10x10 avec loi normale centrée réduite
> my_matrix<-replicate(10, rnorm(10))
> colors<-colorRampPalette(c("blue","white","red"))(256)
> levelplot(my_matrix,col.regions=colors)
>
```

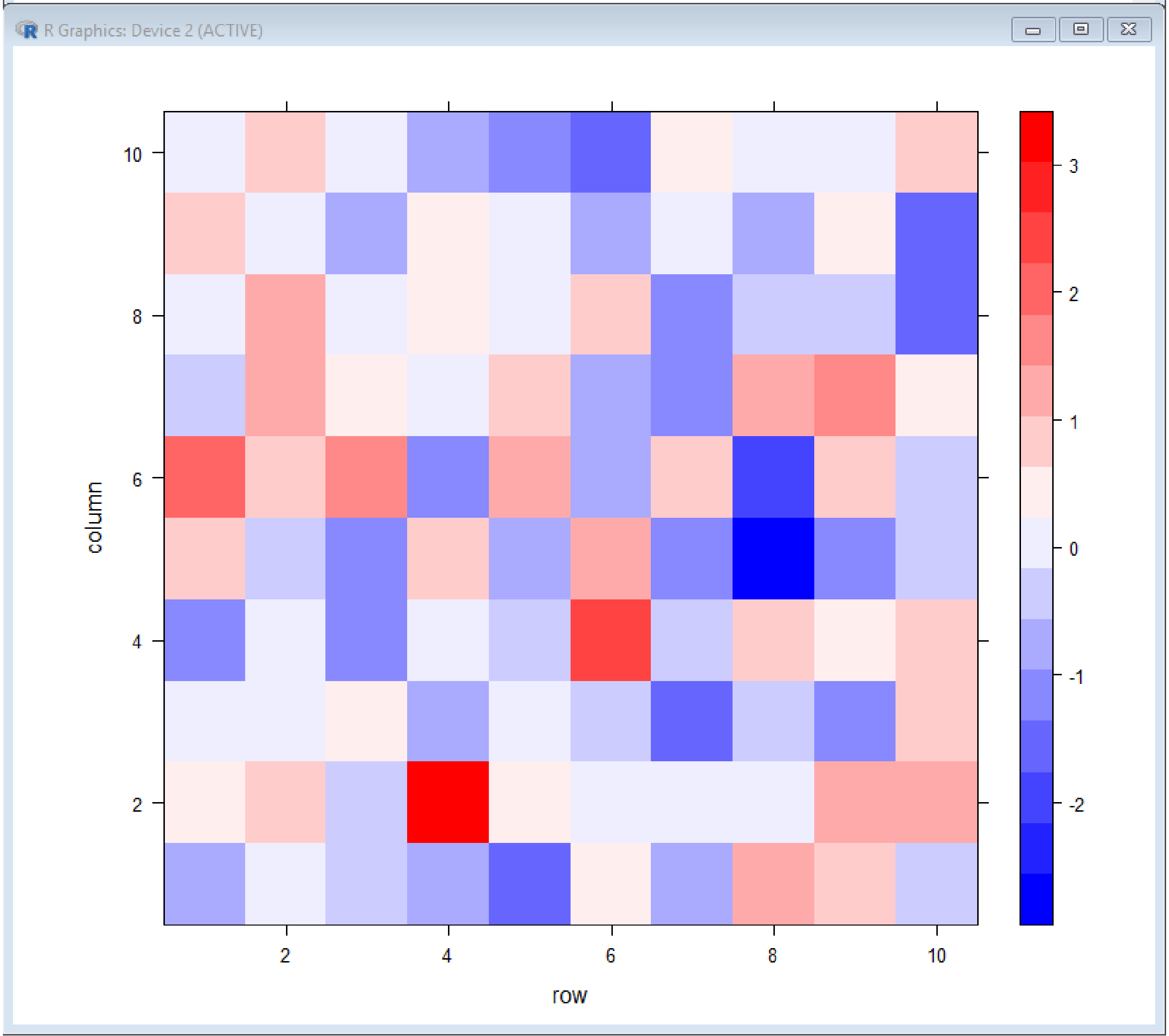
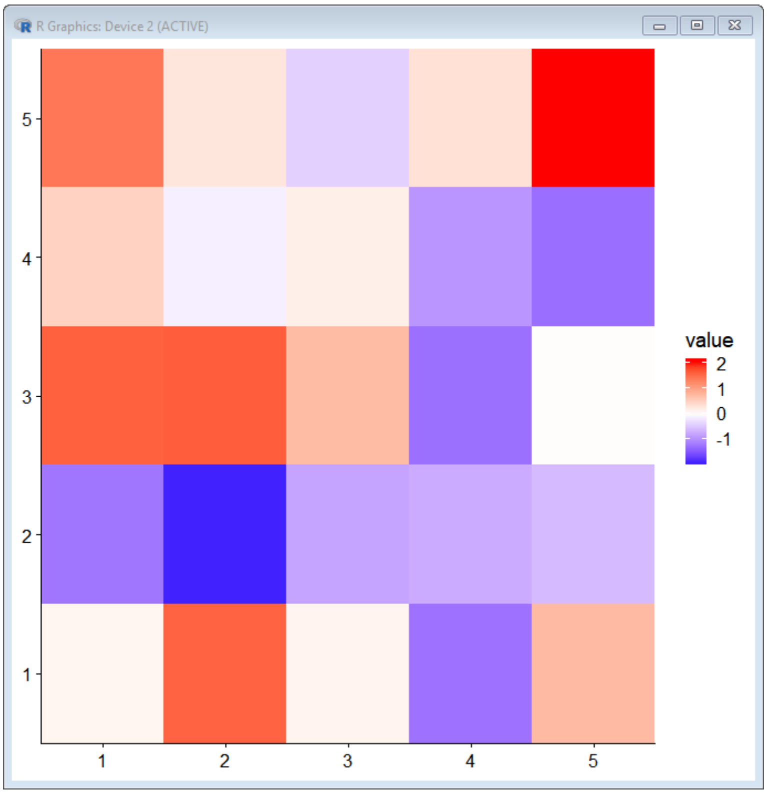


Figure 53 Graph de matrice (amplitudes de composantes)

On peut le faire aussi en ggplot2 (nous avons exprès réduit la dimension de la matrice car cela permet de mieux comprendre ce que fait le package reshape) à l'aide du paramètre **geom_tile( )**:

```
> library("ggplot2")
> library("reshape2")
> my_matrix<-replicate(5, rnorm(5))
> (my_matrix <- melt(my_matrix))
   Var1 Var2       value
1     1    1  0.11396292
2     2    1  1.58515916
3     3    1  0.11555344
4     4    1 -1.25399904
5     5    1  0.74186478
6     1    2 -1.22290161
7     2    2 -1.91889413
8     3    2 -0.80993639
9     4    2 -0.74467723
10    5    2 -0.60750592
11    1    3  1.60675957
12    2    3  1.63559945
13    3    3  0.72609110
14    4    3 -1.27383493
15    5    3  0.02314656
16    1    4  0.48712123
17    2    4 -0.13702750
18    3    4  0.17174548
19    4    4 -0.94479391
20    5    4 -1.28762031
21    1    5  1.40776574
22    2    5  0.26737269
23    3    5 -0.40839764
24    4    5  0.32527122
25    5    5  2.06248097
> ggplot(my_matrix, aes(x = Var1, y = Var2, fill = value)) + geom_tile()+
+ scale_fill_gradient2(low = "blue", mid = "white", high = "red")+
+ scale_y_continuous(expand = c(0,0)) + scale_x_continuous(expand = c(0,0))+
+ theme(axis.title.x=element_blank(),axis.title.y=element_blank())
> 
```


Ce qui donne:



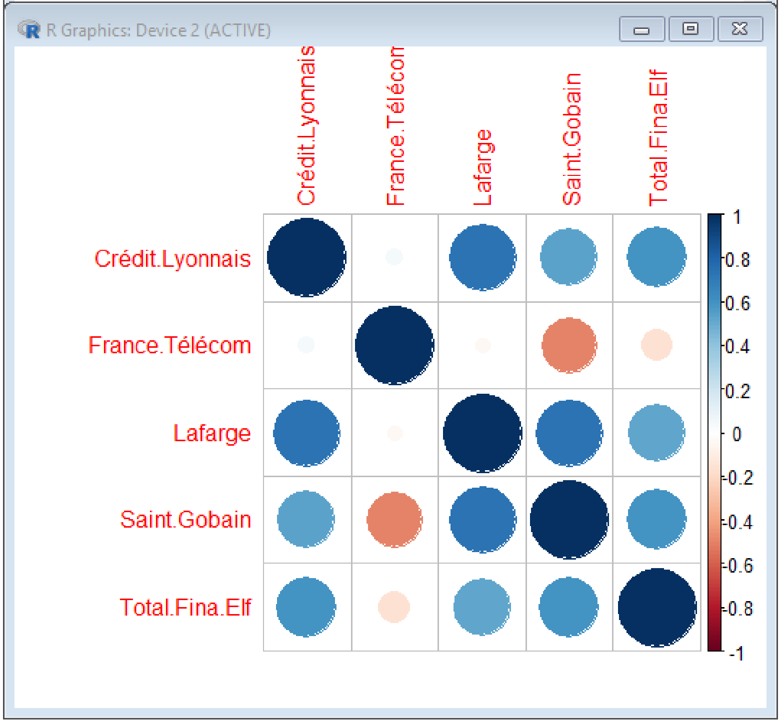
Sinon pour les matrices de corrélation, nous avons aussi la fonction **corrplot( )** du package **corrplot**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

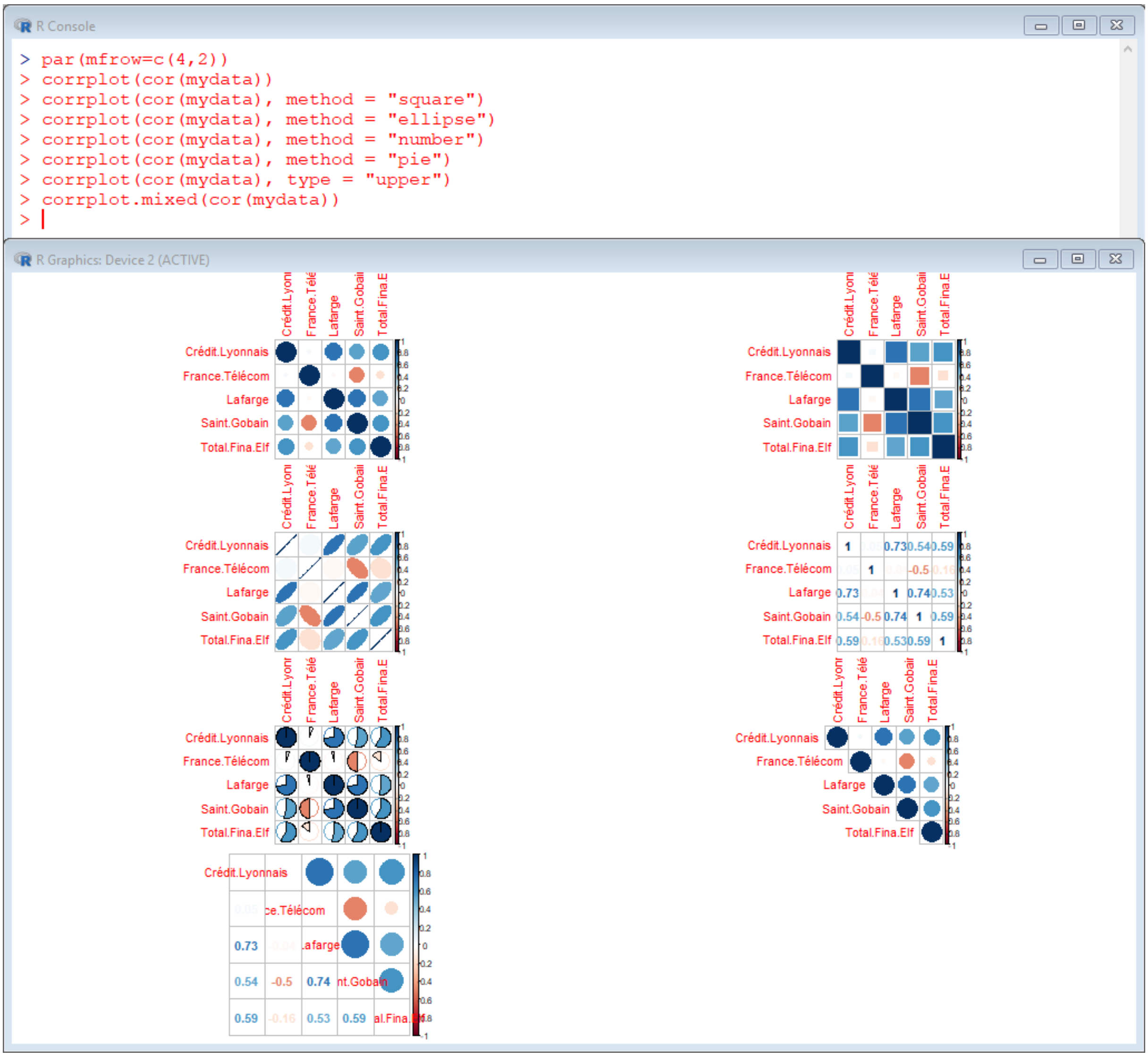
> mydata<-read.csv("c:/tmp/VariationsTitres.csv",header=T,sep=";")
> mydata
   Crédit.Lyonnais France.Télécom   Lafarge Saint.Gobain Total.Fina.Elf
1        -0.017516      -0.328775  0.023053     0.003086       0.000615
2        -0.198426      -0.020374 -0.104156    -0.106509      -0.093423
3         0.122761       0.197863  0.103410     0.023046       0.057627
4        -0.034988       0.063419  0.038501     0.054376      -0.086538
5        -0.000267       0.018141  0.023415     0.040521       0.125614
6        -0.002667      -0.172160 -0.060057    -0.023602       0.015586
7         0.021390      -0.180791  0.009128     0.084602       0.042971
8         0.142932       0.153366  0.030151     0.042345       0.041789
9         0.072148      -0.232346  0.027317     0.015608      -0.049718
10       -0.034822      -0.229599  0.037037     0.000000      -0.007729
11       -0.039398      -0.545980 -0.075092    -0.043158      -0.014979
12       -0.082719       0.558855 -0.110891    -0.329373      -0.102798
> cor(mydata)
                Crédit.Lyonnais France.Télécom     Lafarge Saint.Gobain Total.Fina.Elf
Crédit.Lyonnais      1.00000000     0.04780981  0.73338852    0.5370196      0.5929914
France.Télécom       0.04780981     1.00000000 -0.03863644   -0.4997799     -0.1576209
Lafarge              0.73338852    -0.03863644  1.00000000    0.7391915      0.5288331
Saint.Gobain         0.53701959    -0.49977992  0.73919150    1.0000000      0.5917240
Total.Fina.Elf       0.59299138    -0.15762088  0.52883313    0.5917240      1.0000000
> |
```

Et finalement:

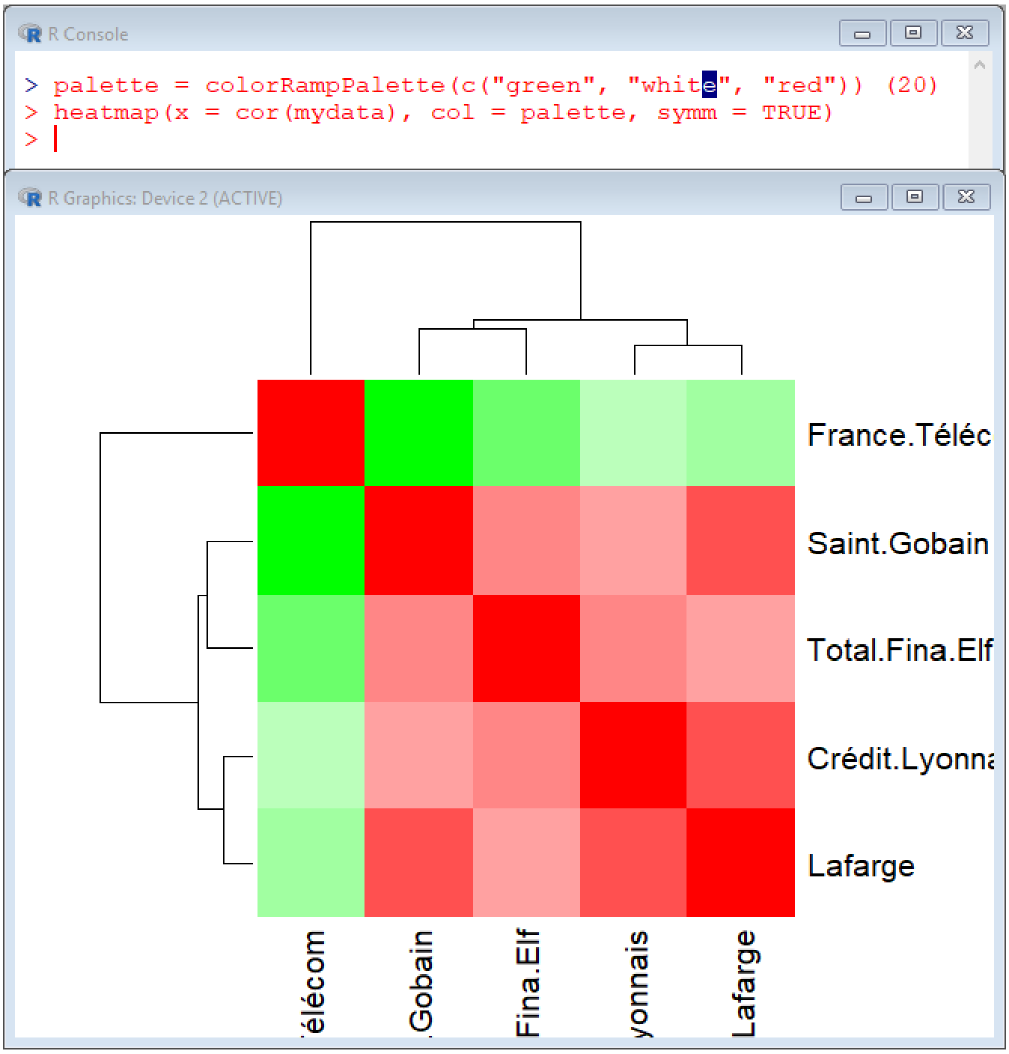

```
R Console
> library("corrplot")
> corrplot(cor(mydata))
> |
```



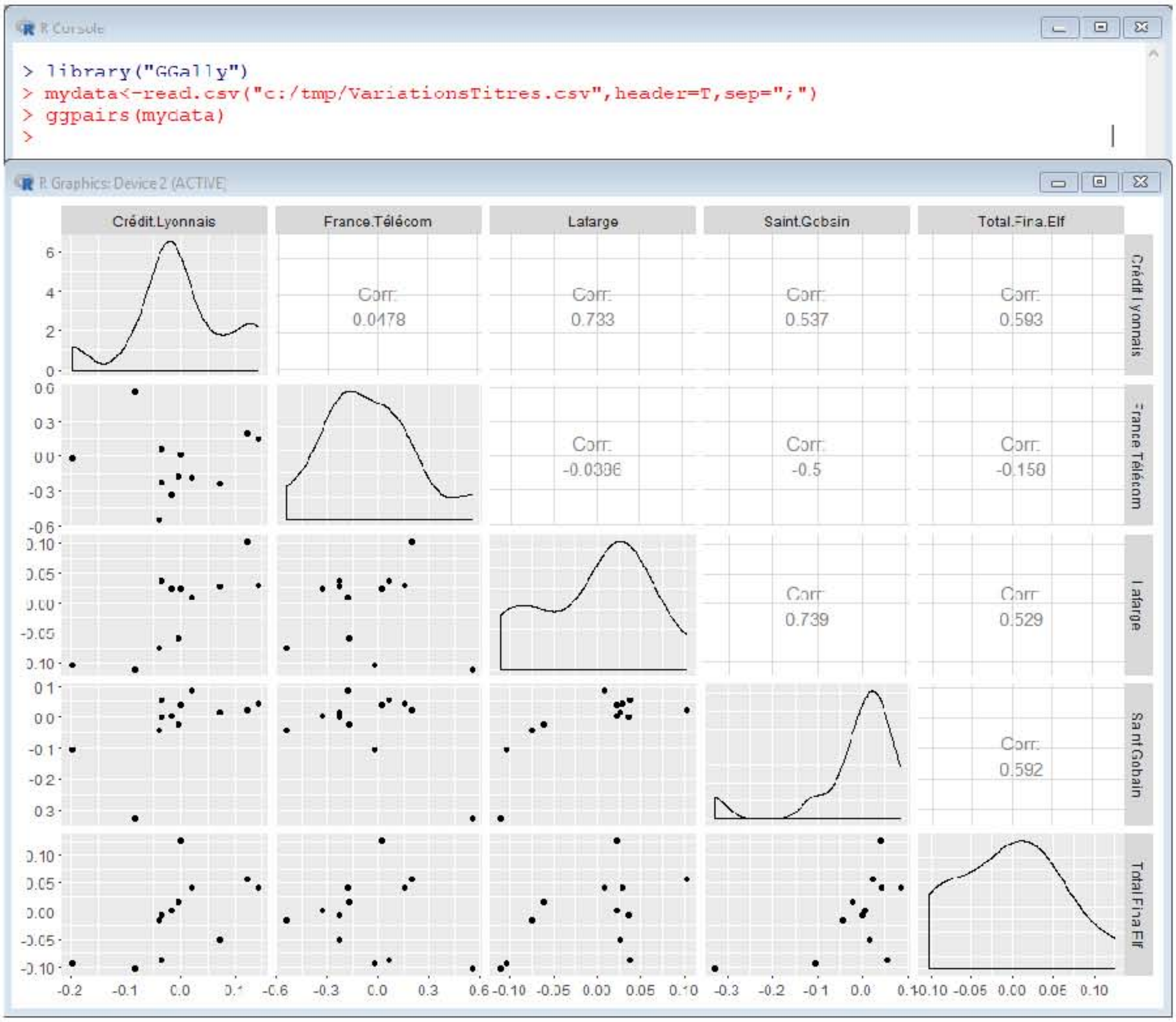
On peut faire plein d'autres choses avec le package **corrplot** dont voici un échantillon non exhaustif:



On peut faire aussi une analyse descendante hiérarchique avec en même temps une carte de chaleur:



Ou aussi la fonction **ggpairs( )** package **GGally**:

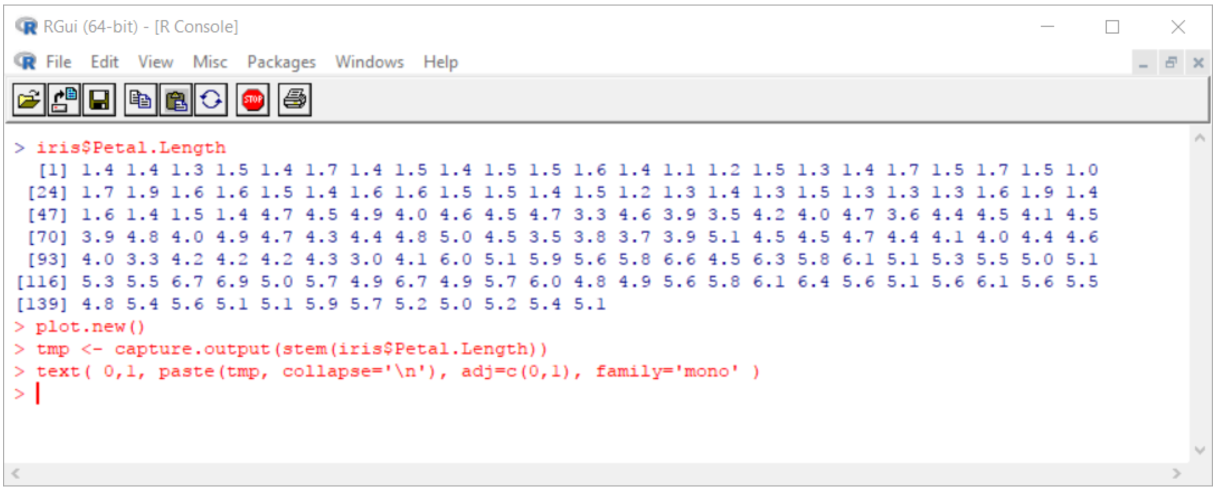

R Console
> library("GGally")
> mydata<-read.csv("c:/tmp/VariationsTitres.csv",header=T,sep=";")
> ggpairs(mydata)
>
R Graphics: Device 2 (ACTIVE)
Crédit.Lyonnais
France.Télécom
Lafarge
Saint.Gobain
Total.Fina.Elf
Corr:
0.0478
0.733
0.537
0.593
-0.0336
-0.5
-0.158
0.739
0.529
0.592


R Console
> library("GGally")
> ggcoef(fit,vline_color = "red",vline_linetype = "solid",errorbar_color = "blue",
+ errorbar_height = .25,exclude_intercept = TRUE)
> ggpairs(mydata)
>
Mois
Couts
CoutA
CoutB
CoutC
Corr:
0.156
0.204
-0.0376
0.49
0.601
0.611
0.52
0.543
0.268
0.0582

Exemple 114.: Diagrammes "stem-and-leaf"

R 3.4.1

Nous partons du jeu de données suivant et utilisons la fonction native **stem( )** :



pour obtenir:

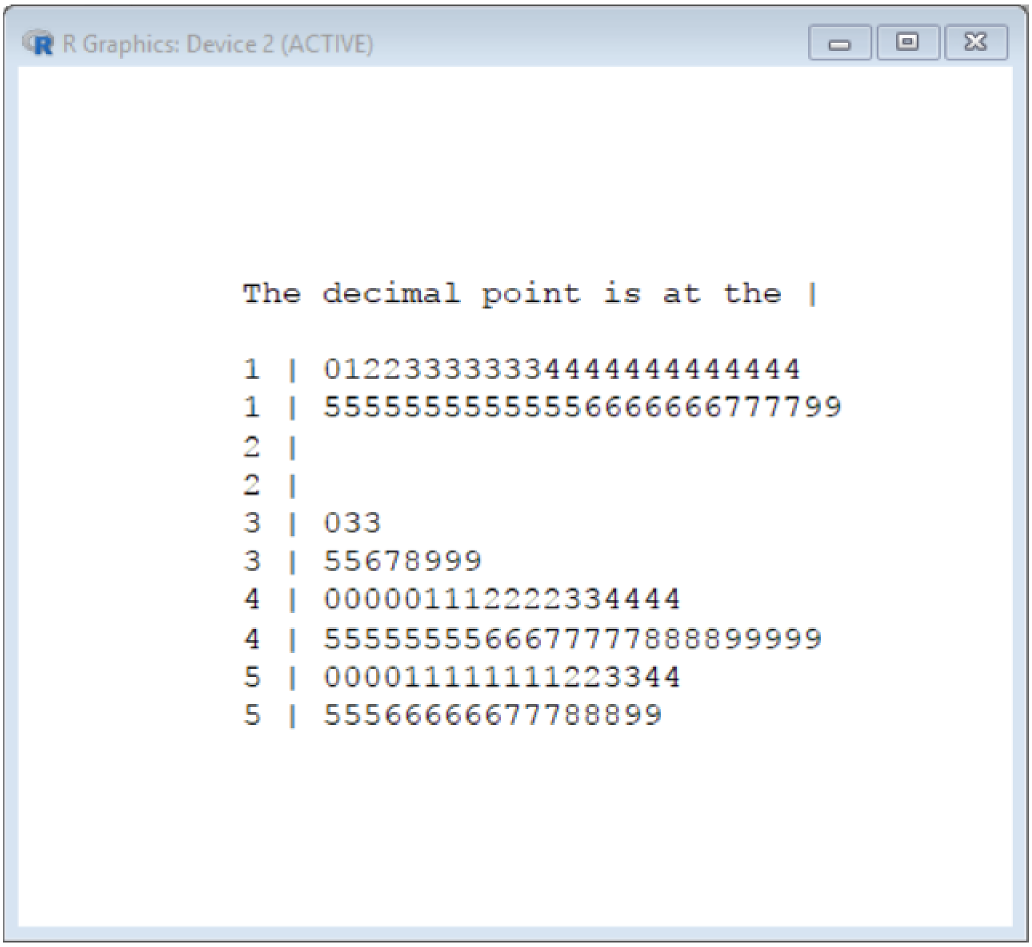
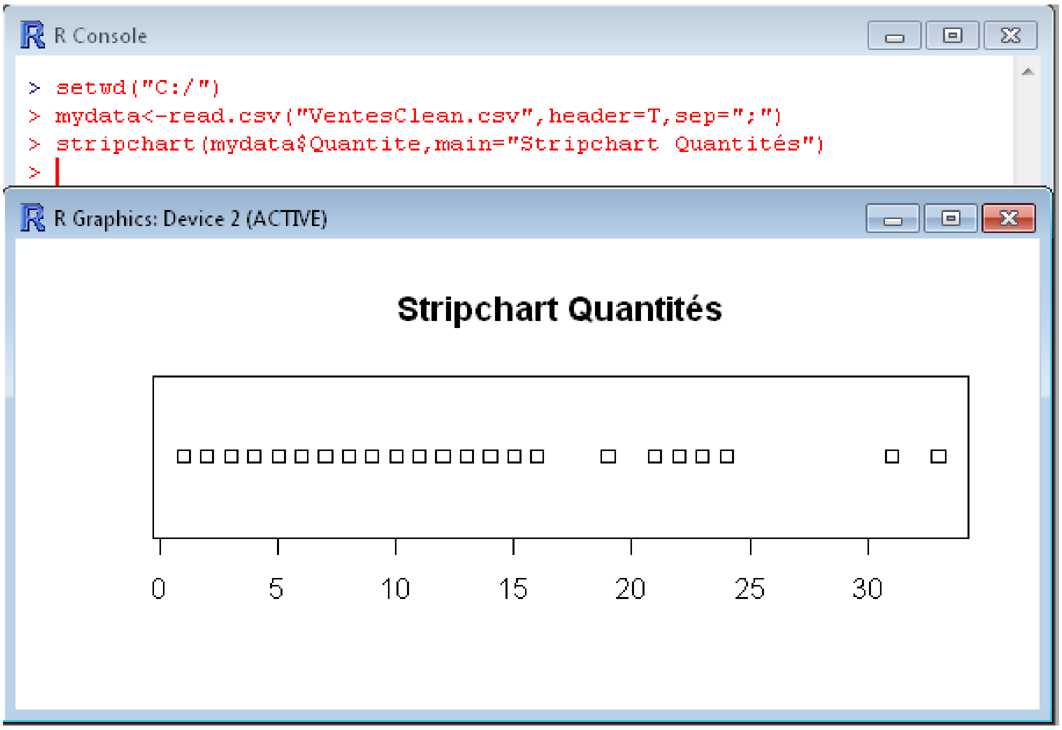


Figure 54 Graph de type stem-and-leafs

Exemple 115.: Diagrammes "strips"

R 3.0.2

Une famille de diagramme parfois by sympathique et facilement lisible par le plus grand nombre en utilisant la commande **stripchart( )** :



On peut ajouter un bruit pour voir la grossièrement la densité avec l'attribut *jitter* du paramètre *method*:

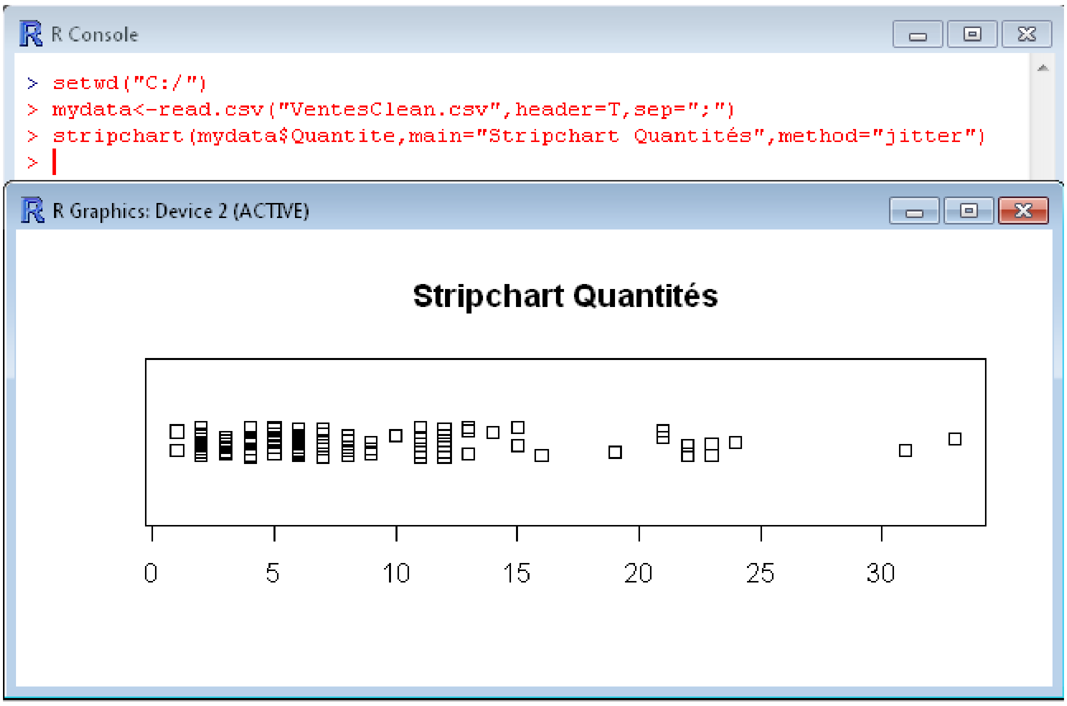
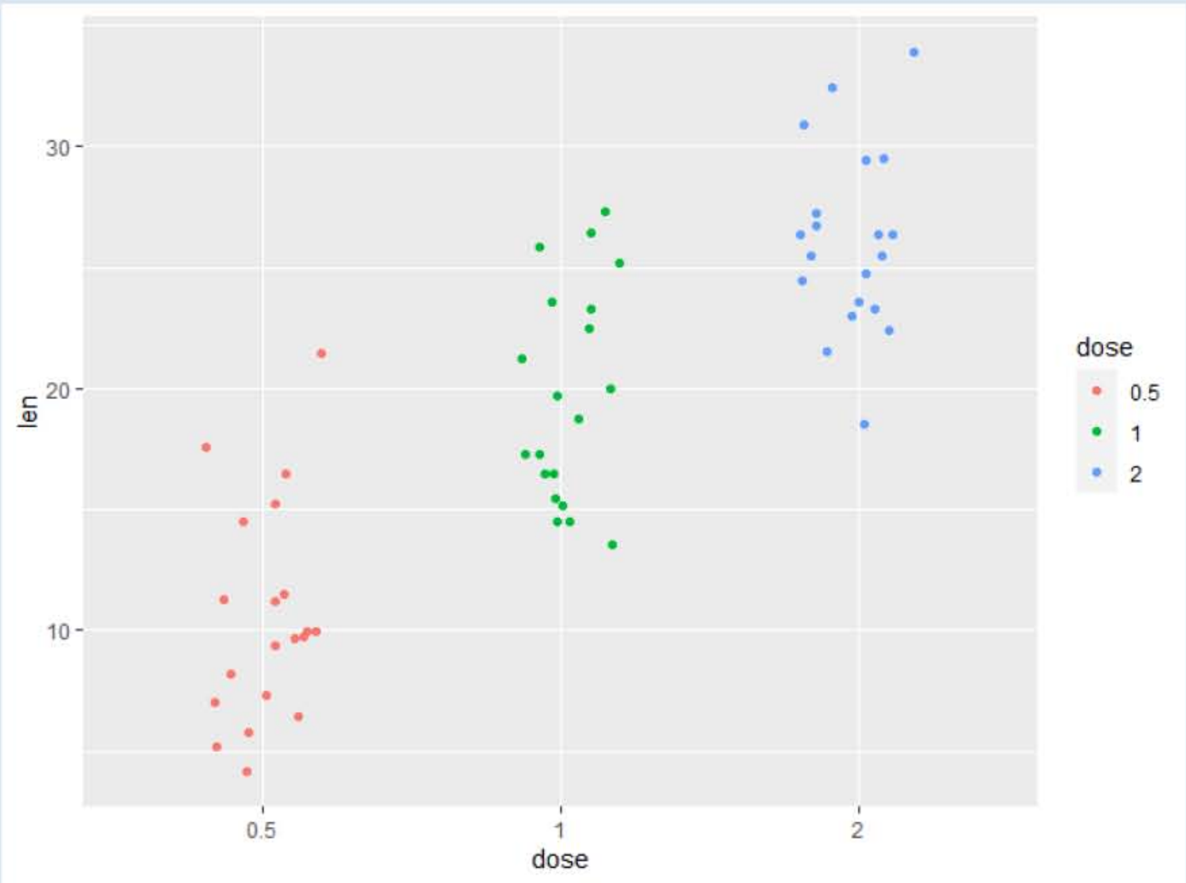


Figure 55 Graph de type stripchart

On peut faire des graphiques jitter plus perfectionnés bien évidemment, typiquement:

```
> data(ToothGrowth)
> head(ToothGrowth)
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
> ToothGrowth$dose <- as.factor(ToothGrowth$dose)
> library("ggplot2")
> ggplot(ToothGrowth, aes(x=dose, y=len)) +
+     geom_jitter(position=position_jitter(0.2), color="red")
> # Change stripchart colors by groups
> p<-ggplot(ToothGrowth, aes(x=dose, y=len, color=dose)) +
+   geom_jitter(position=position_jitter(0.2))
> p
>
```



Exemples 116.: Diagrammes X-Y et à lignes

R 3.0.2

Commençons donc par le plus simple des graphiques à points **plot( )** imaginable avec une petite couleur de fond différente du blanc (comme cela on apprend deux petites choses d'un coup):

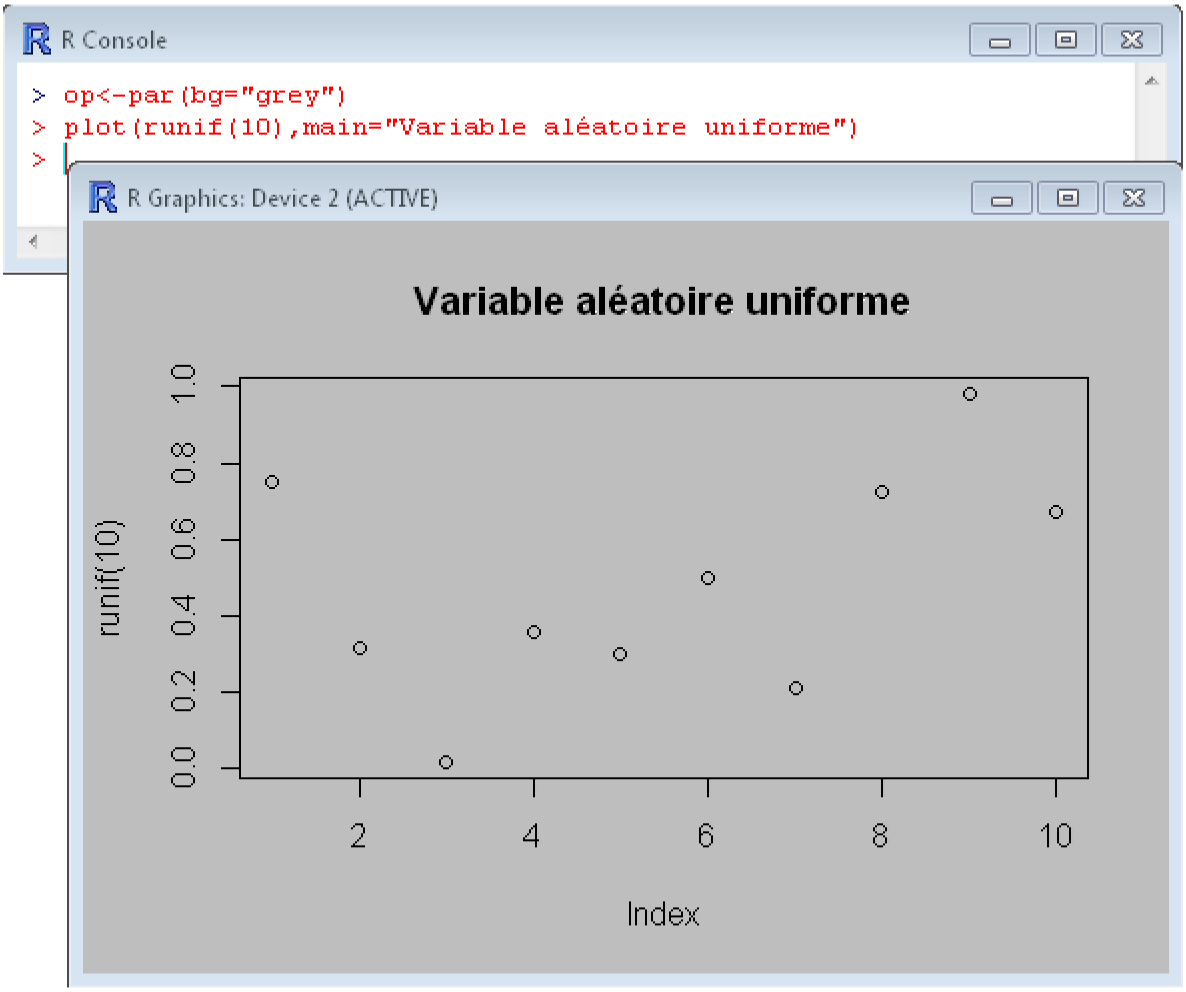
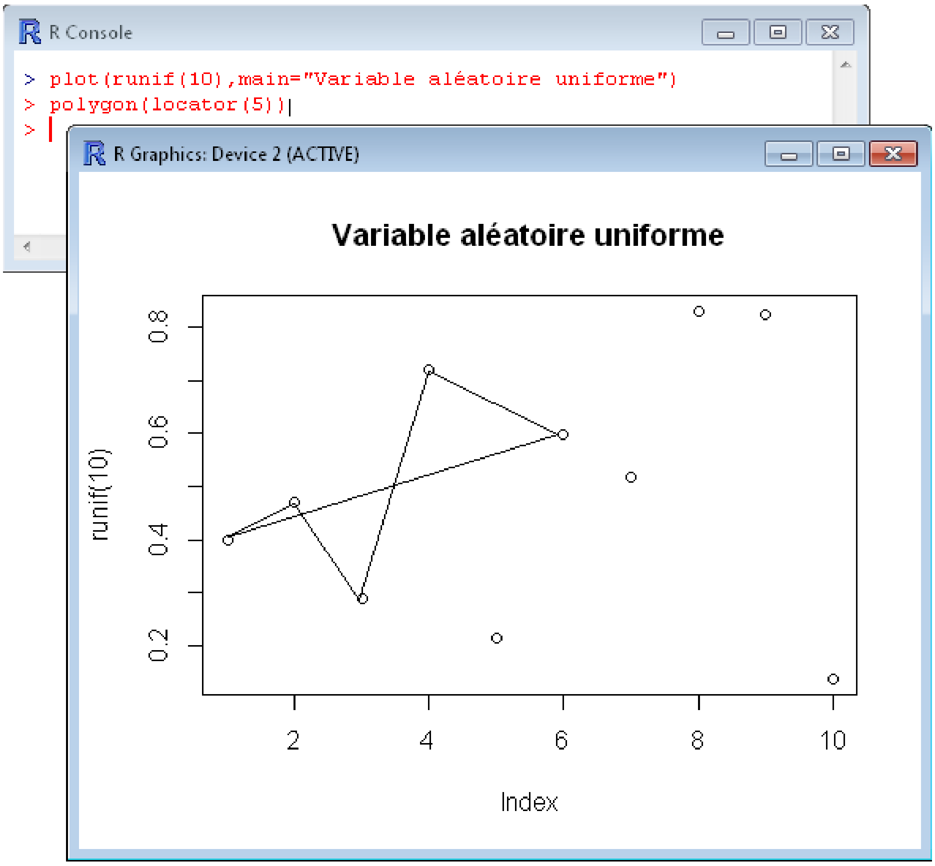
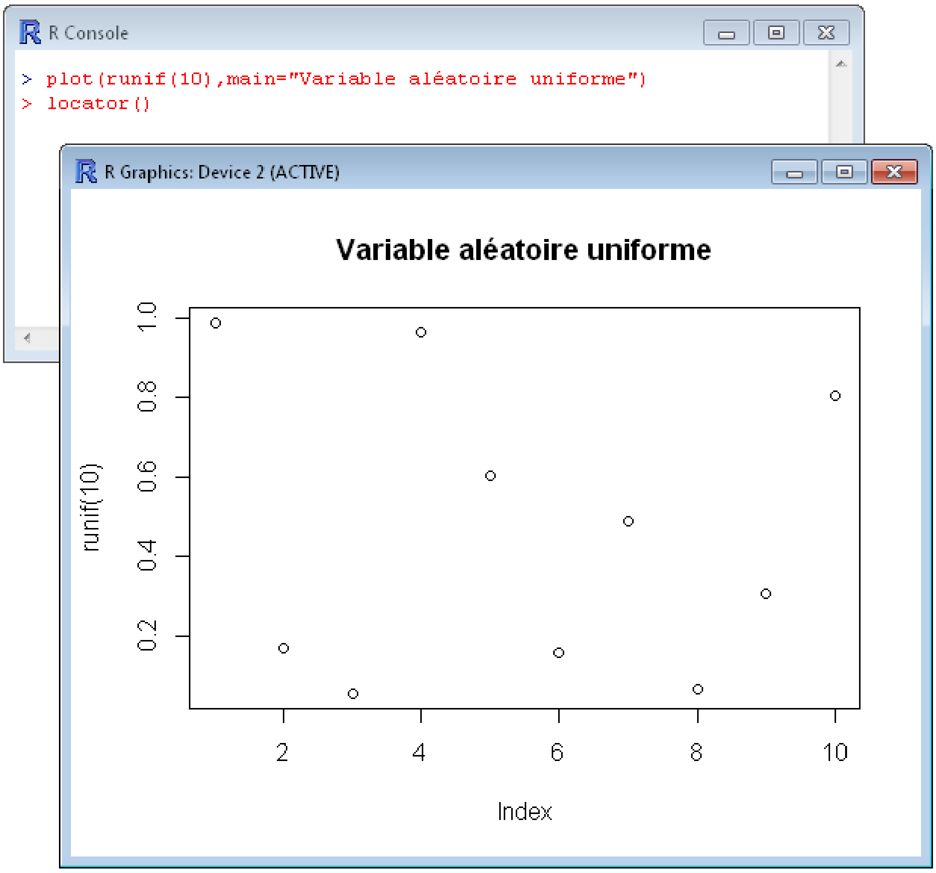


Figure 56 Graph X-Y à points

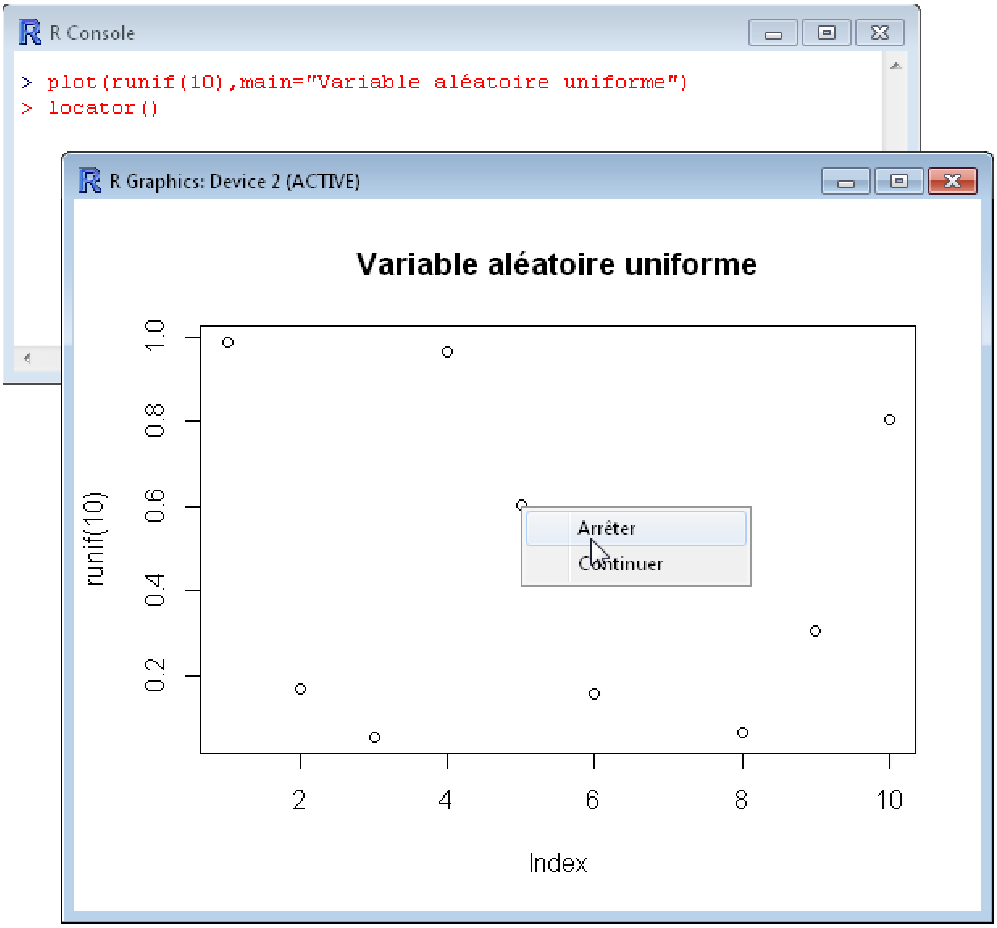
Avec la commande **polygon( )** associée à **locator( )** nous pouvons laisser l'utilisateur cliquer de façon interactive dans le graphique pour créer un polygone:



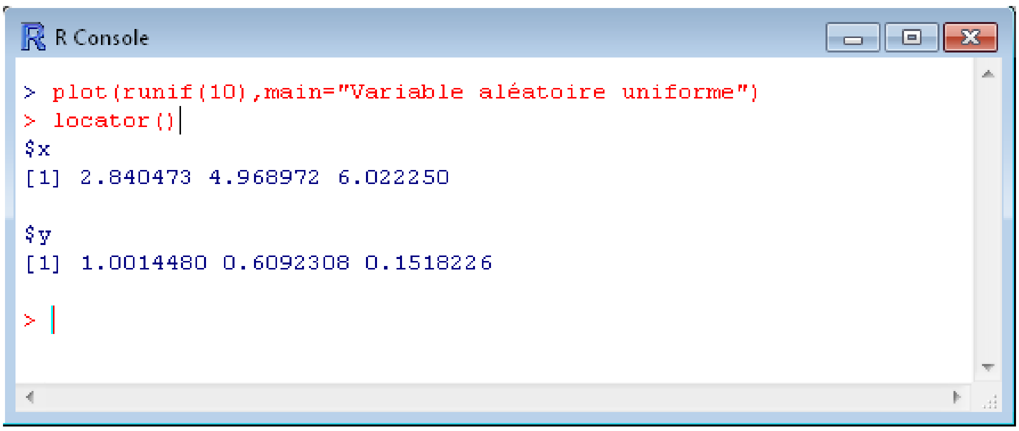
Ou encore avec la commande native **locator( )** seule en cliquant ensuite à différents endroits du graphe:



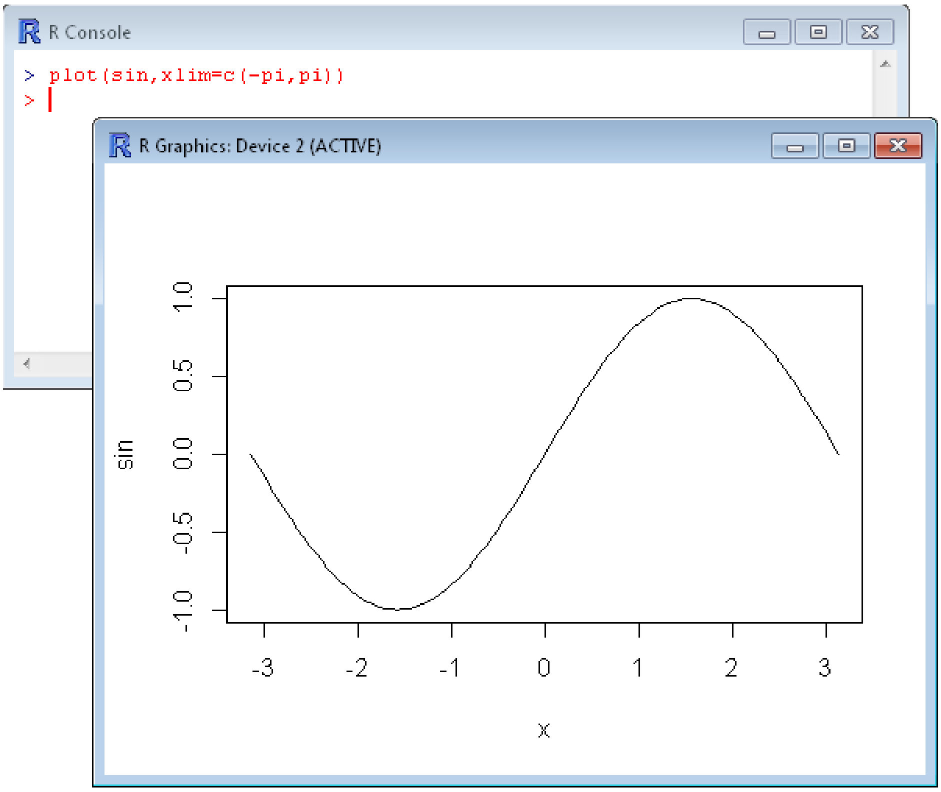
Et ensuite si nous arrêtons le "cliquage" par un clic droit:



Nous obtenons alors les coordonnées des points sur lesquels nous avons cliqués:



Ou encore dans les classiques scolaires de la petite école (attention! pour ploter toute fonction algébrique dans le plan reportez vous à l'exemple de la page 309):



ou encore un graphique de type "stem":

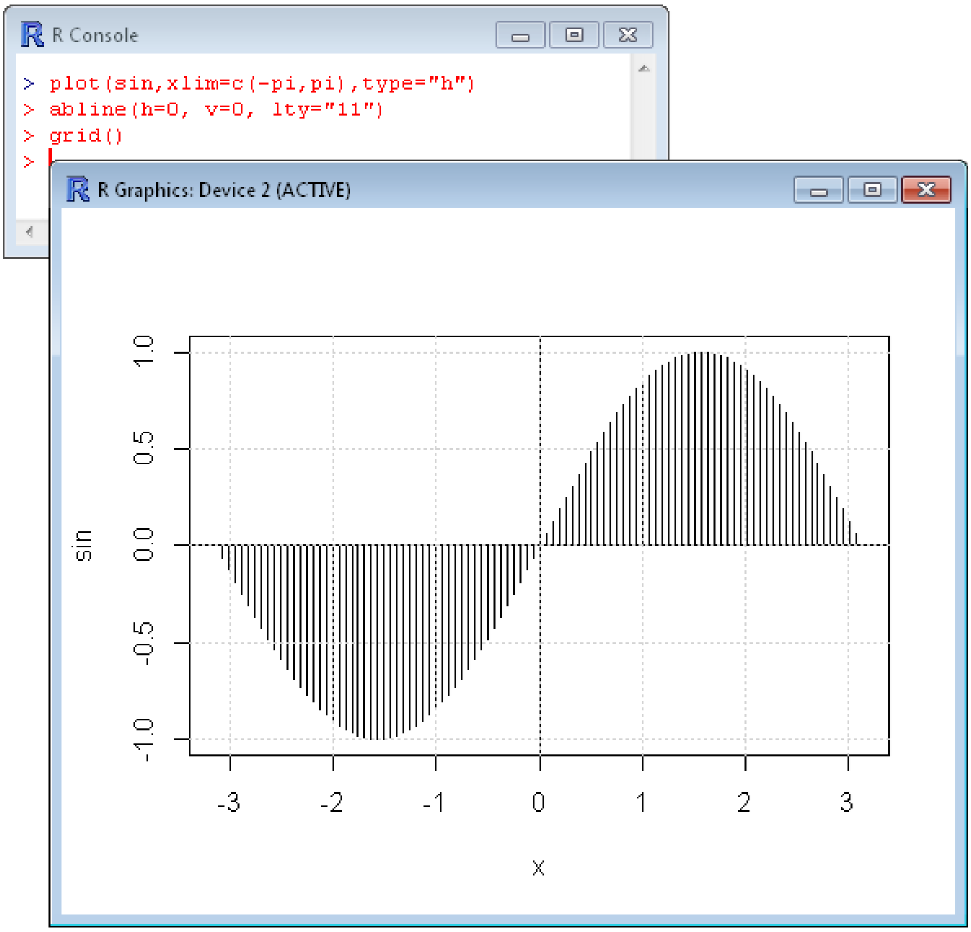


Figure 57 Graph de type stem (à "tiges")

Sinon dans le même genre on a le cas classique en traitement du signal (ici il s'agit d'une convolution discrète):

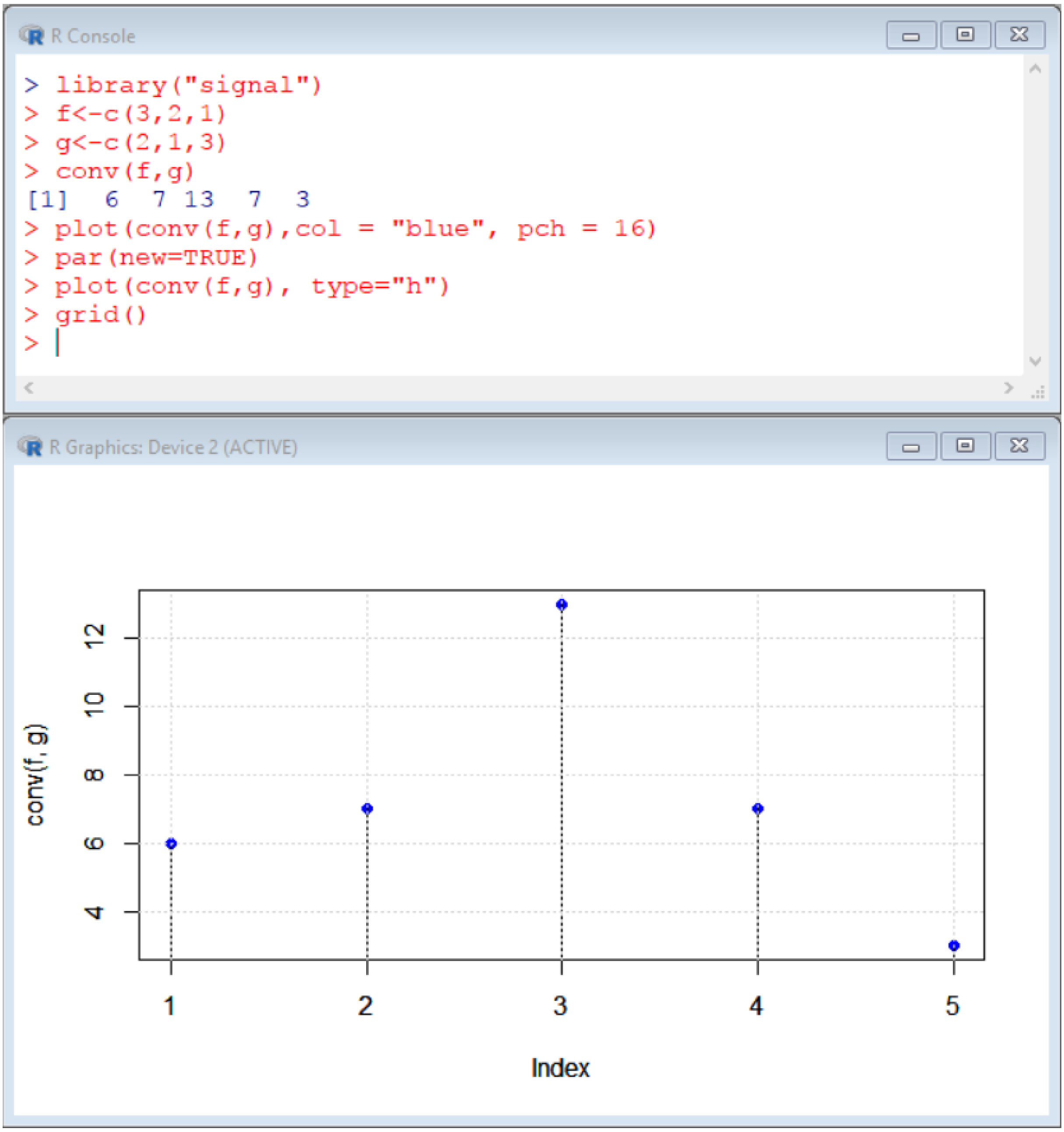
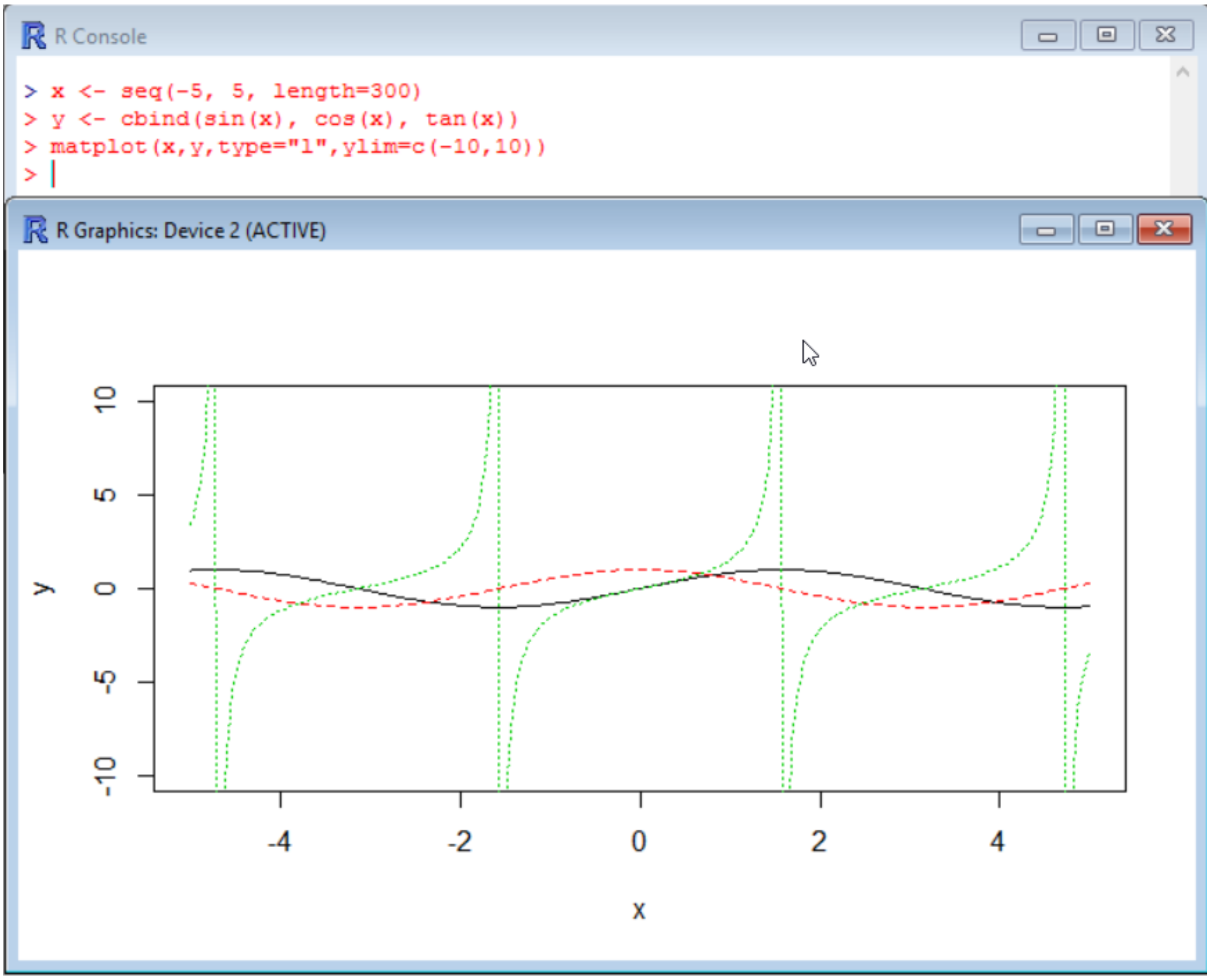
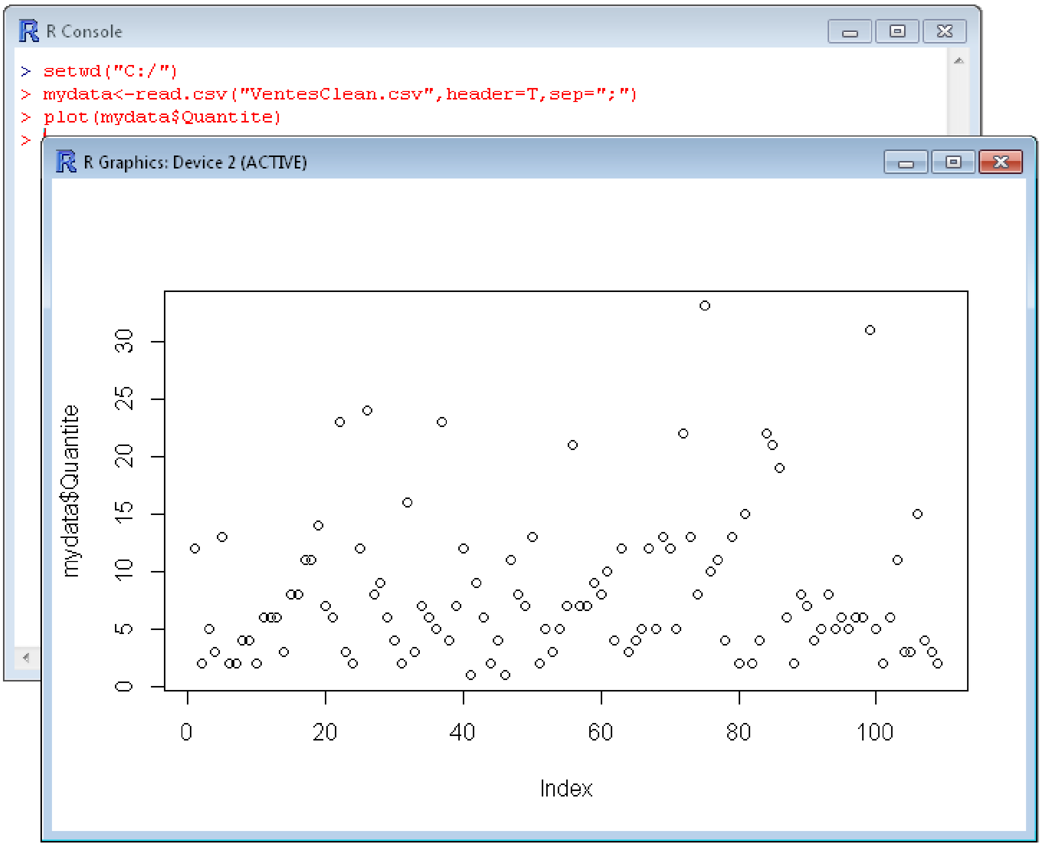


Figure 58 Graph de type stem (avec disques aux sommets)

et avec la fonction native **matplot( ) :**



Revenons à des points:



On peut s'amuser à les relier par des droites (puisque l'index correspond dans le cas présent aussi à l'ordre des commandes) en utilisant une technique combinant **plot( )** et **lines( )** :

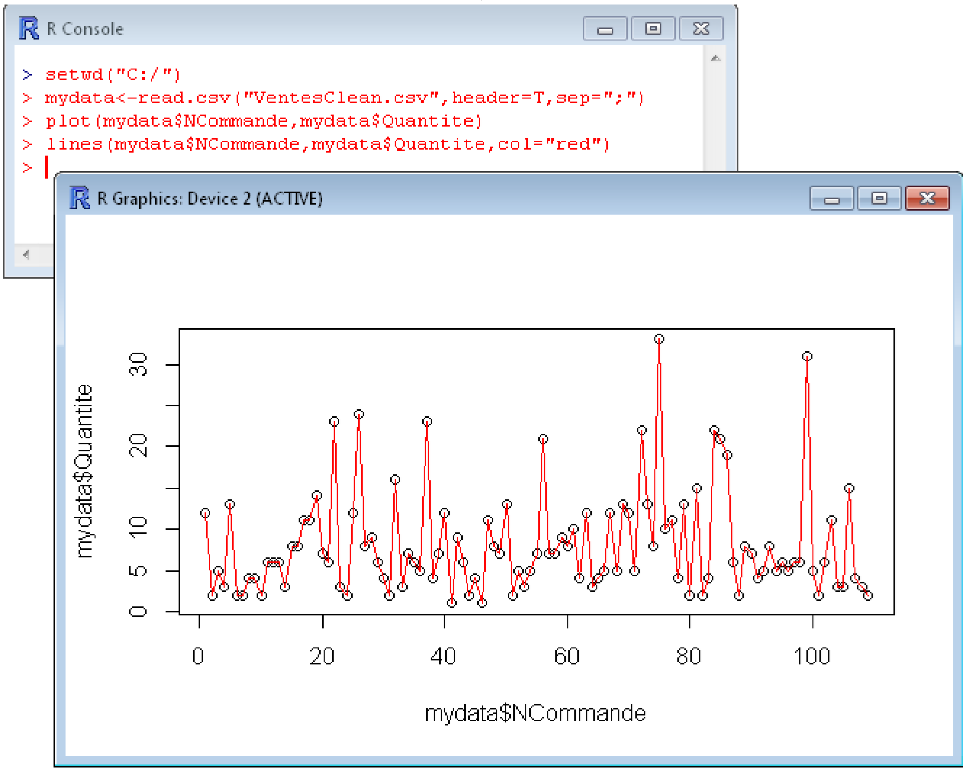
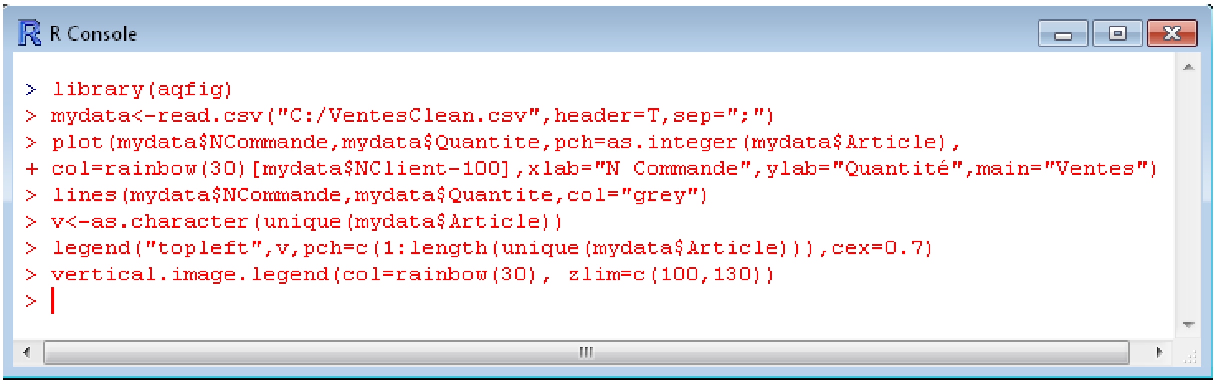
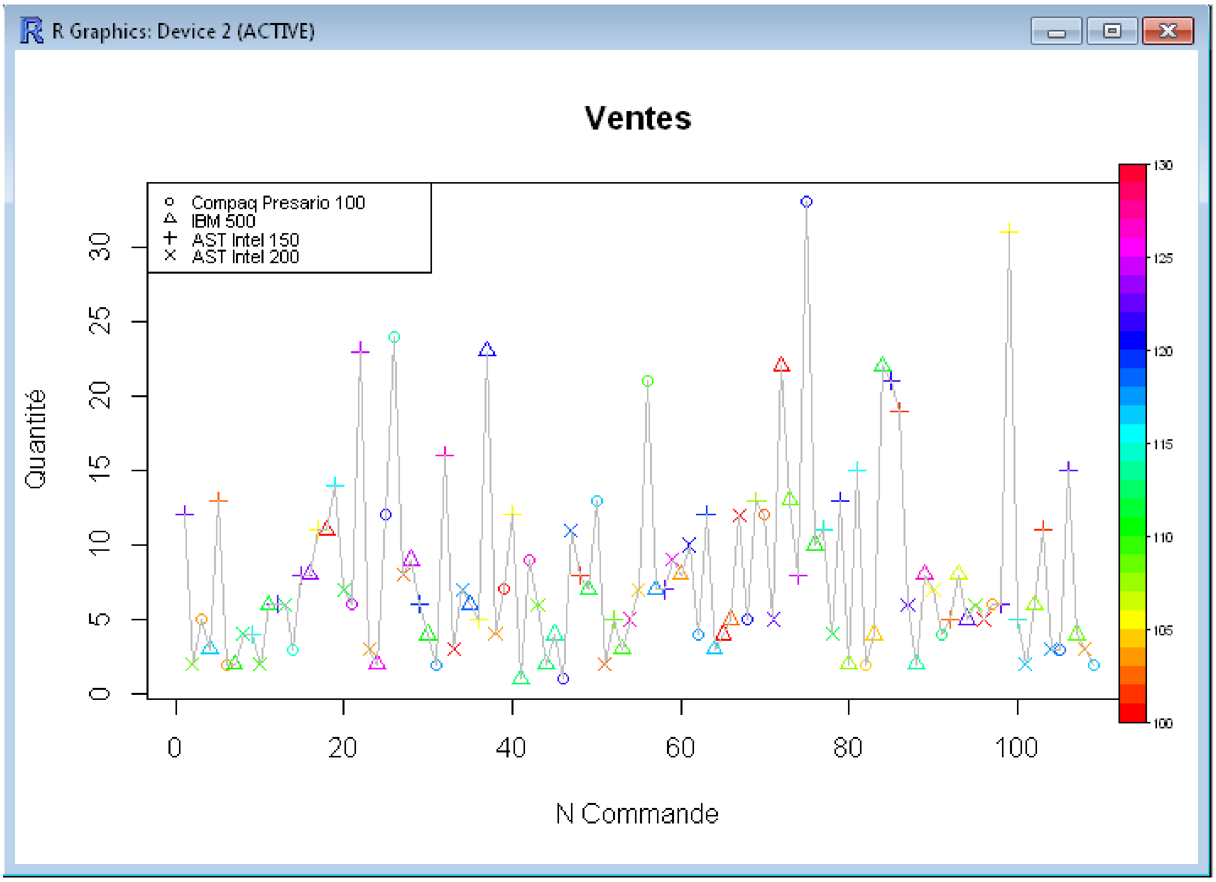


Figure 59 Graph X-Y à points et lignes

Nous avons aussi (on ajoute un titre, des points différentes pour chaque type d'articles, des couleurs en fonction du numéro de client et des légendes) en utilisant la commande **vertical.image.legend( )** du package **aqfig**:



Ce qui donne:



Ou pour revenir à un cas plus simple en ajoutant une ligne de rég linéaire avec la commande **abline( )** :

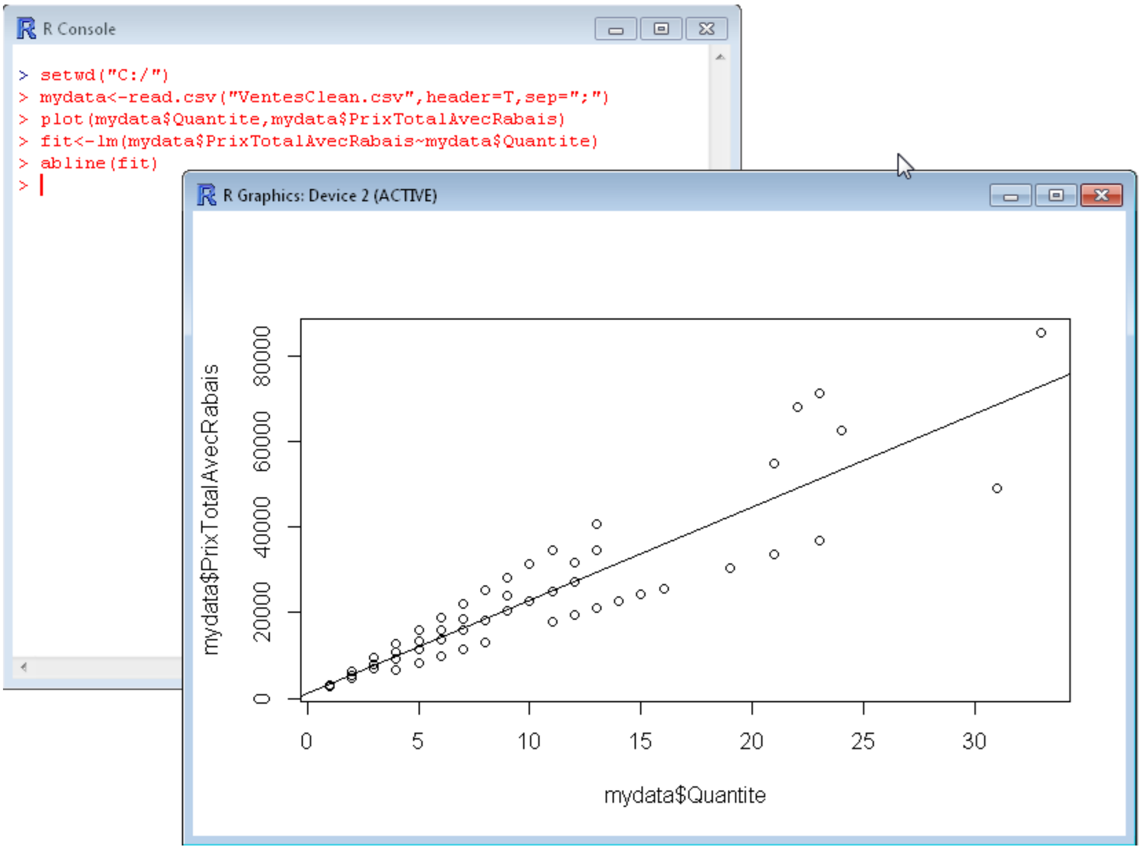
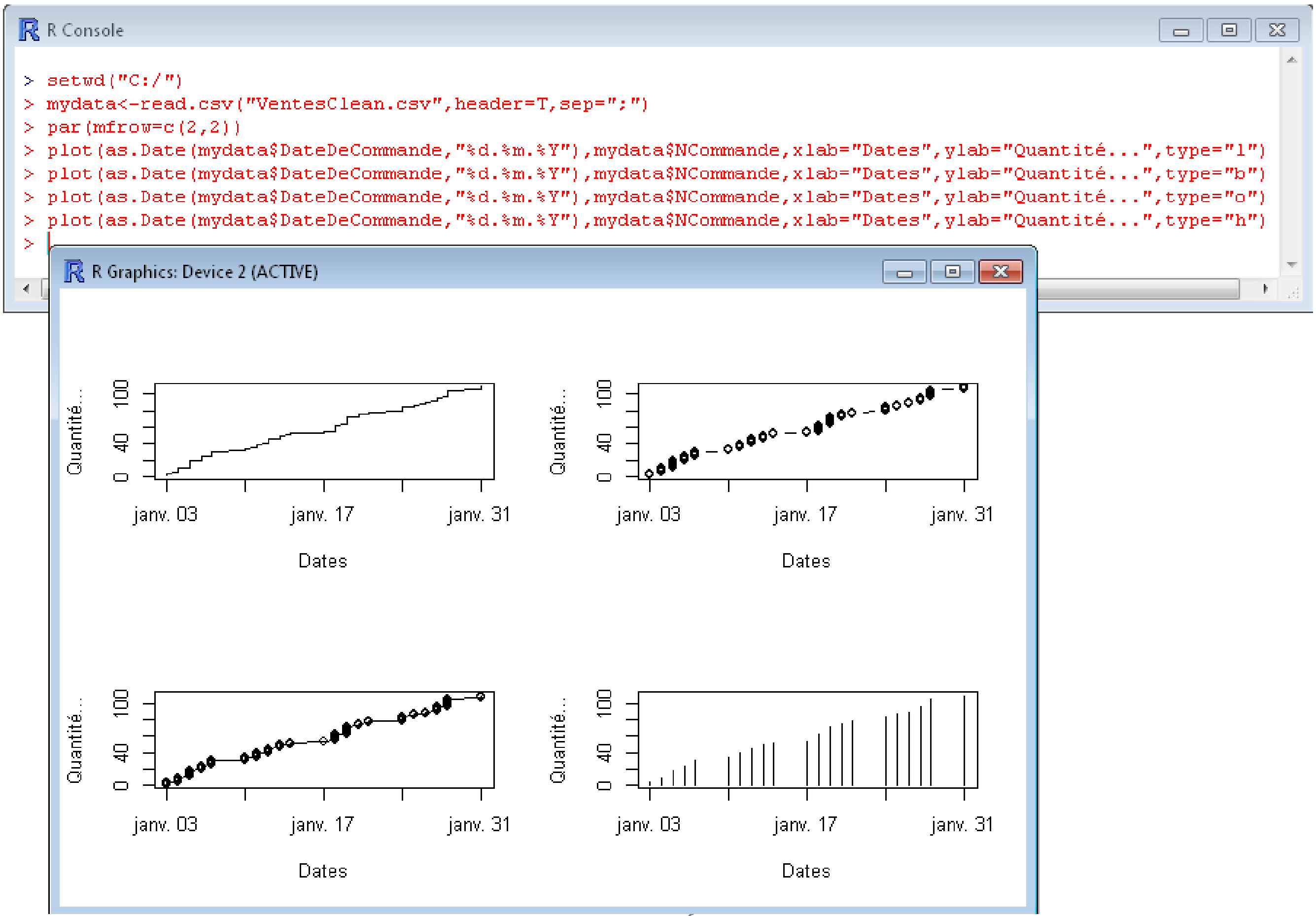
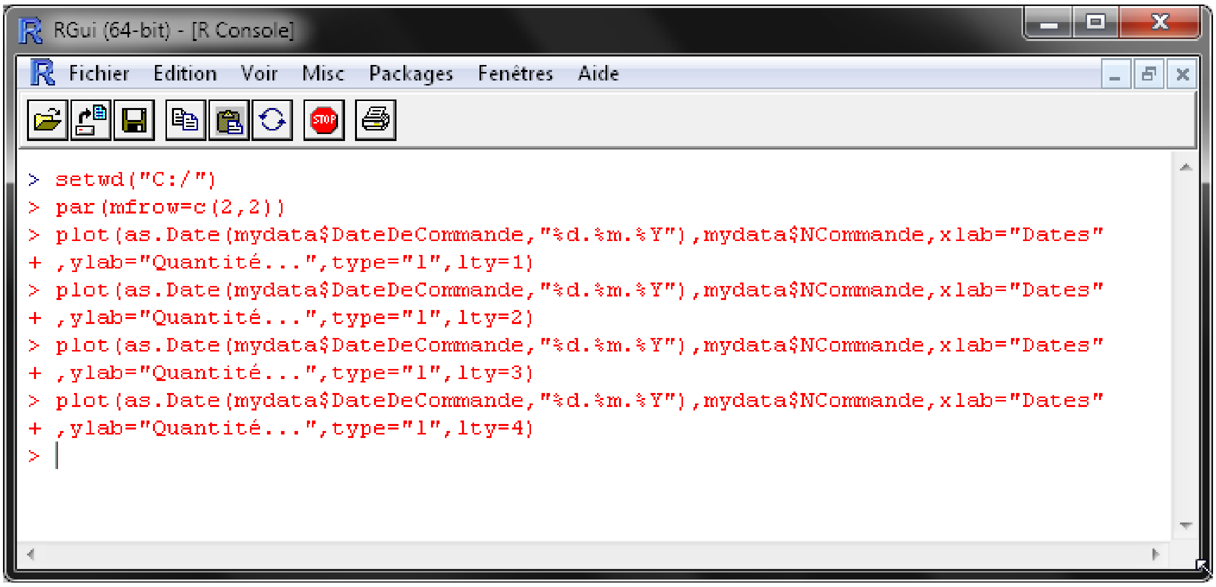


Figure 60 Graph X-Y à points avec simple fit

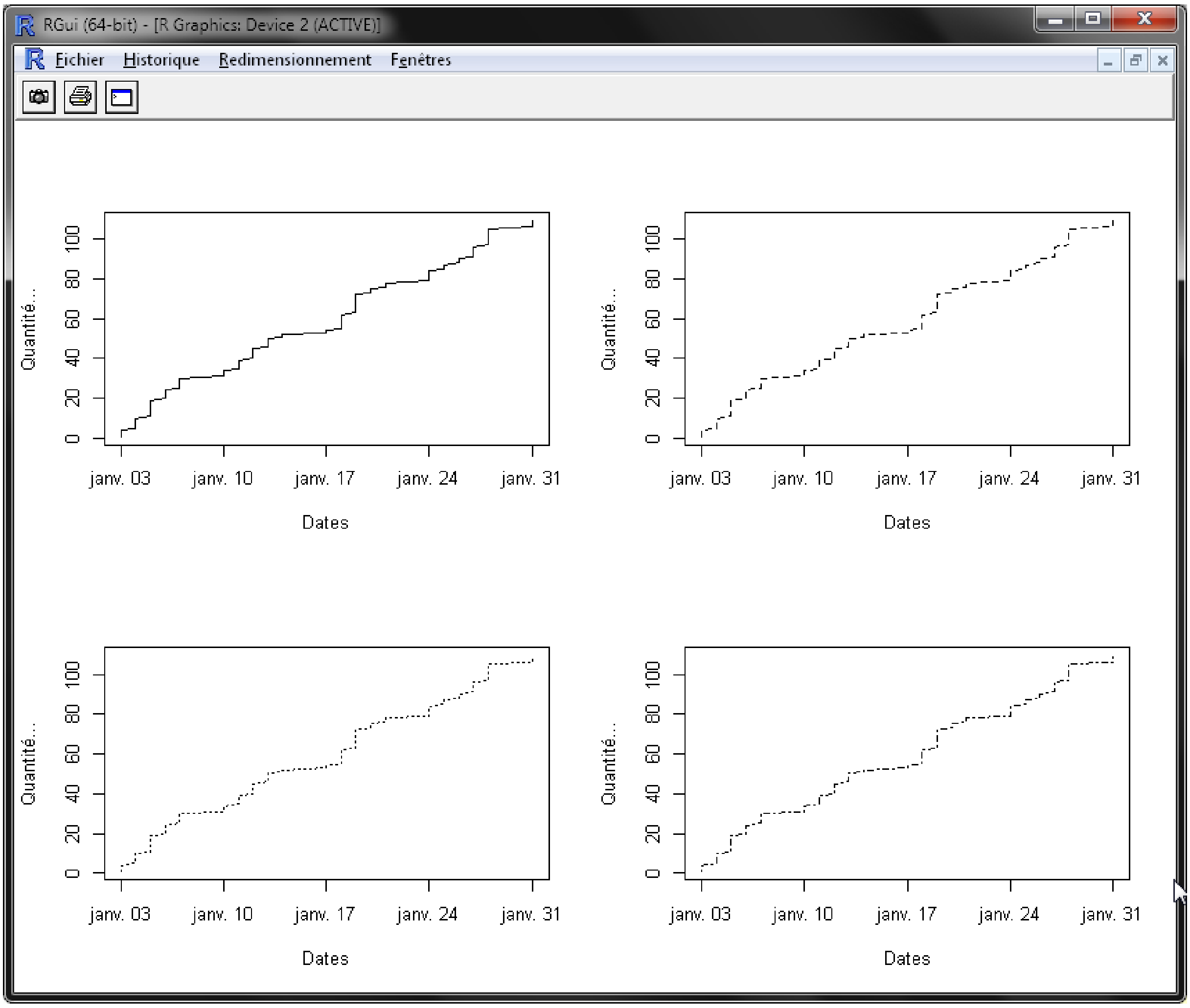
ou encore:



ou considérons encore l'exemple suivant présentant le paramètre *lty*:

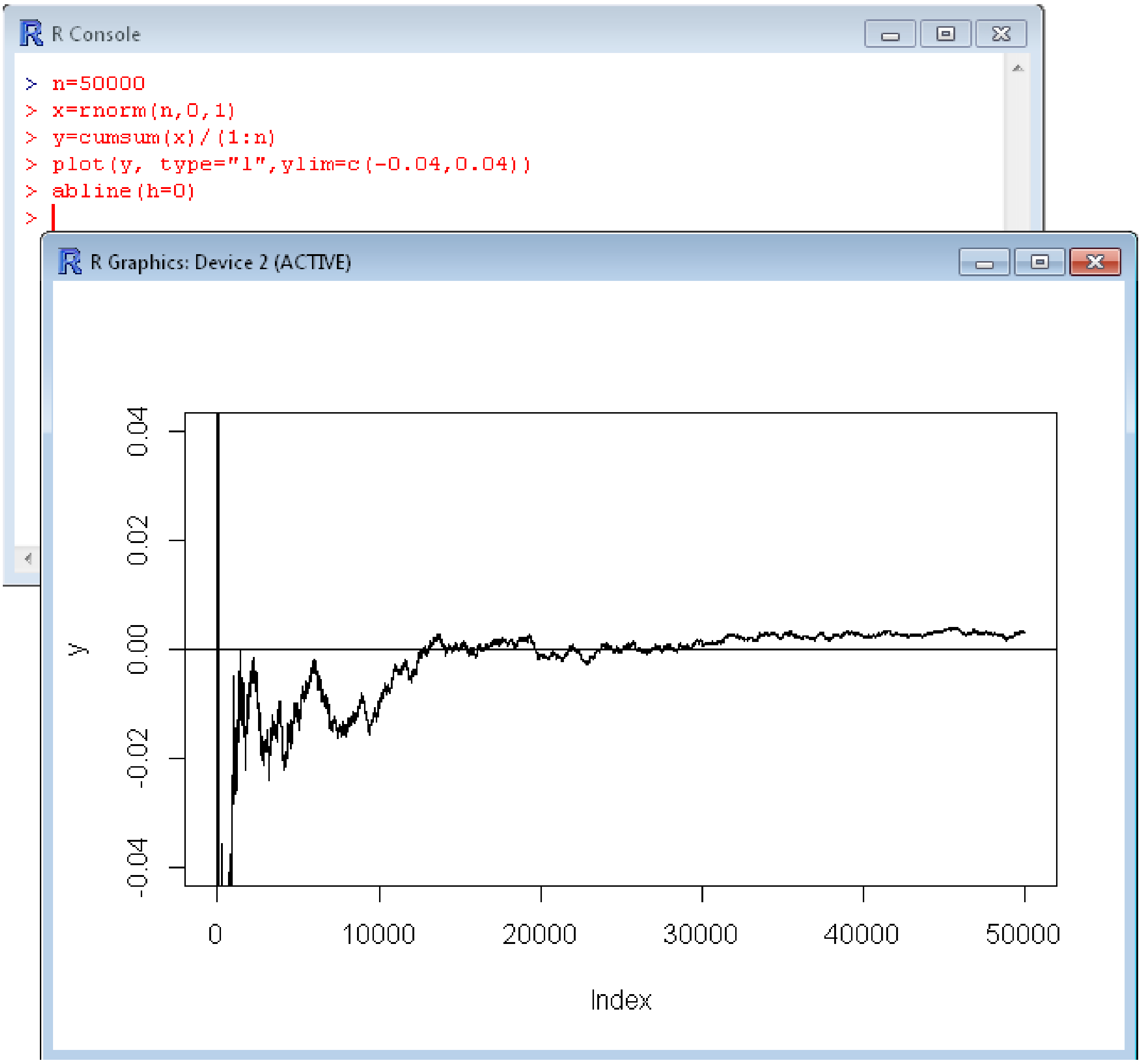


qui donne:

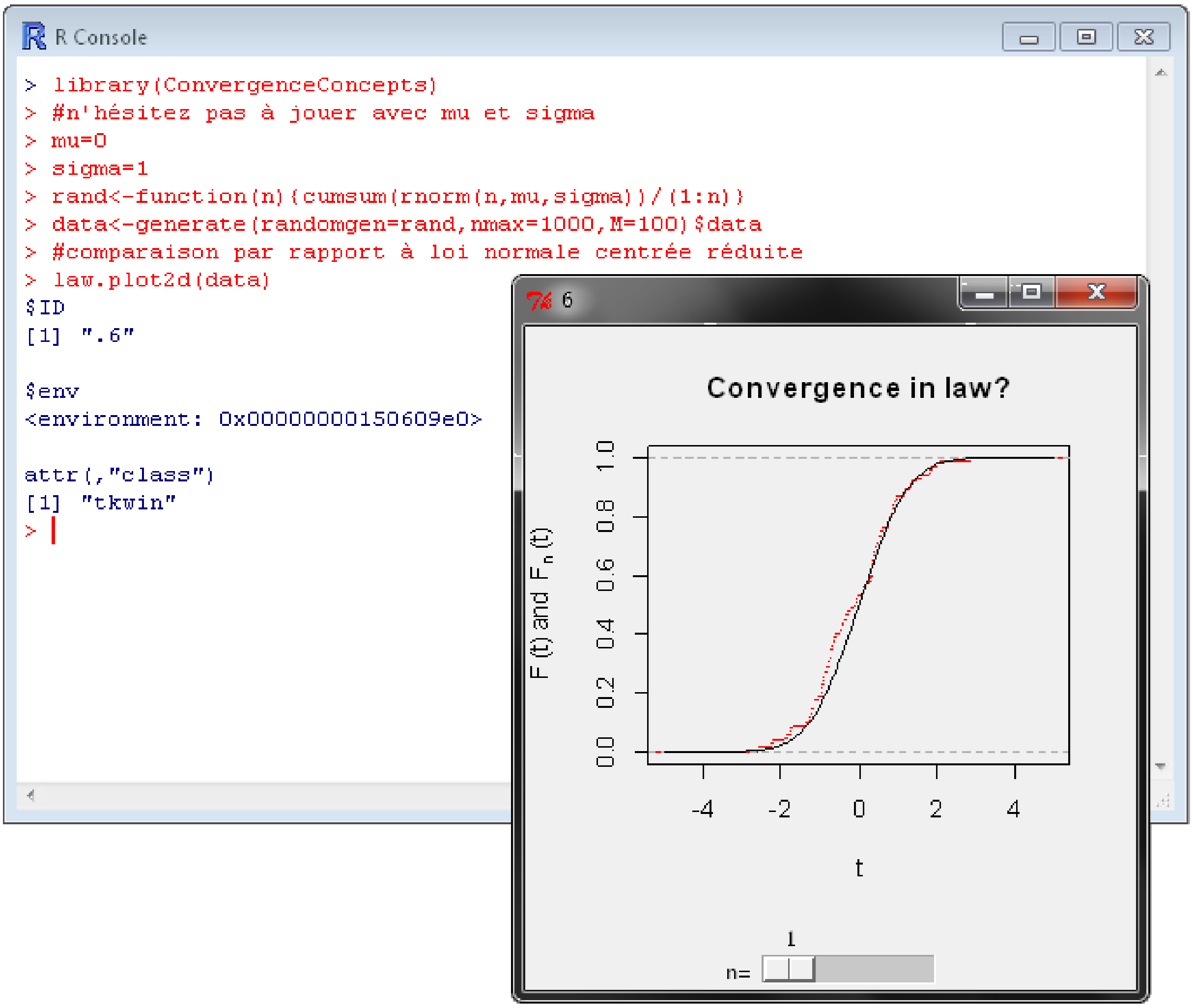


et ainsi de suite (il y a plus d'une quarantaine de types de traits).

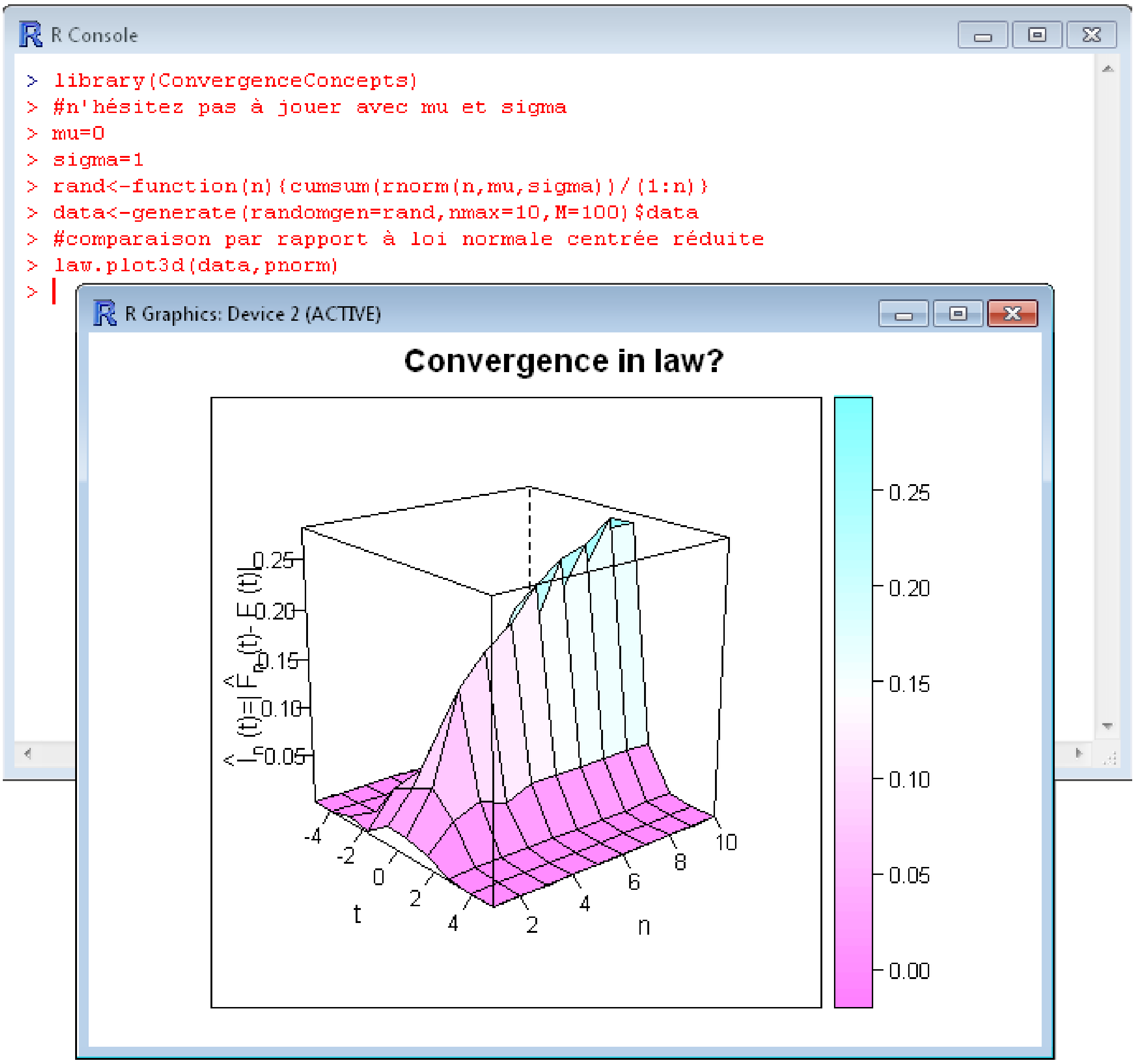
Un cas classique pour montrer la convergence des estimateurs (loi des grands nombres) lors de simulations de Monte-Carlo et une application intéressant de **abline( )** pour créer de simples lignes:



ou pour jouer avec la convergence à l'aide du package **ConvergenceConcepts** et les commandes **generate( )** et **law.plot2d( )** :



ou toujours avec ce même package voir l'évolution de la convergence en 3D mais avec la commande **law.plot3d( )** :



Avec la commande **plot( )** , nous pouvons aussi avoir une représentation sympathique dont données *Iris* que nous avons utilisé dans le cours théorique pour décortiquer les développements mathématiques de l'analyse en composantes principales et en associant en plus des couleurs à chaque catégorie!:

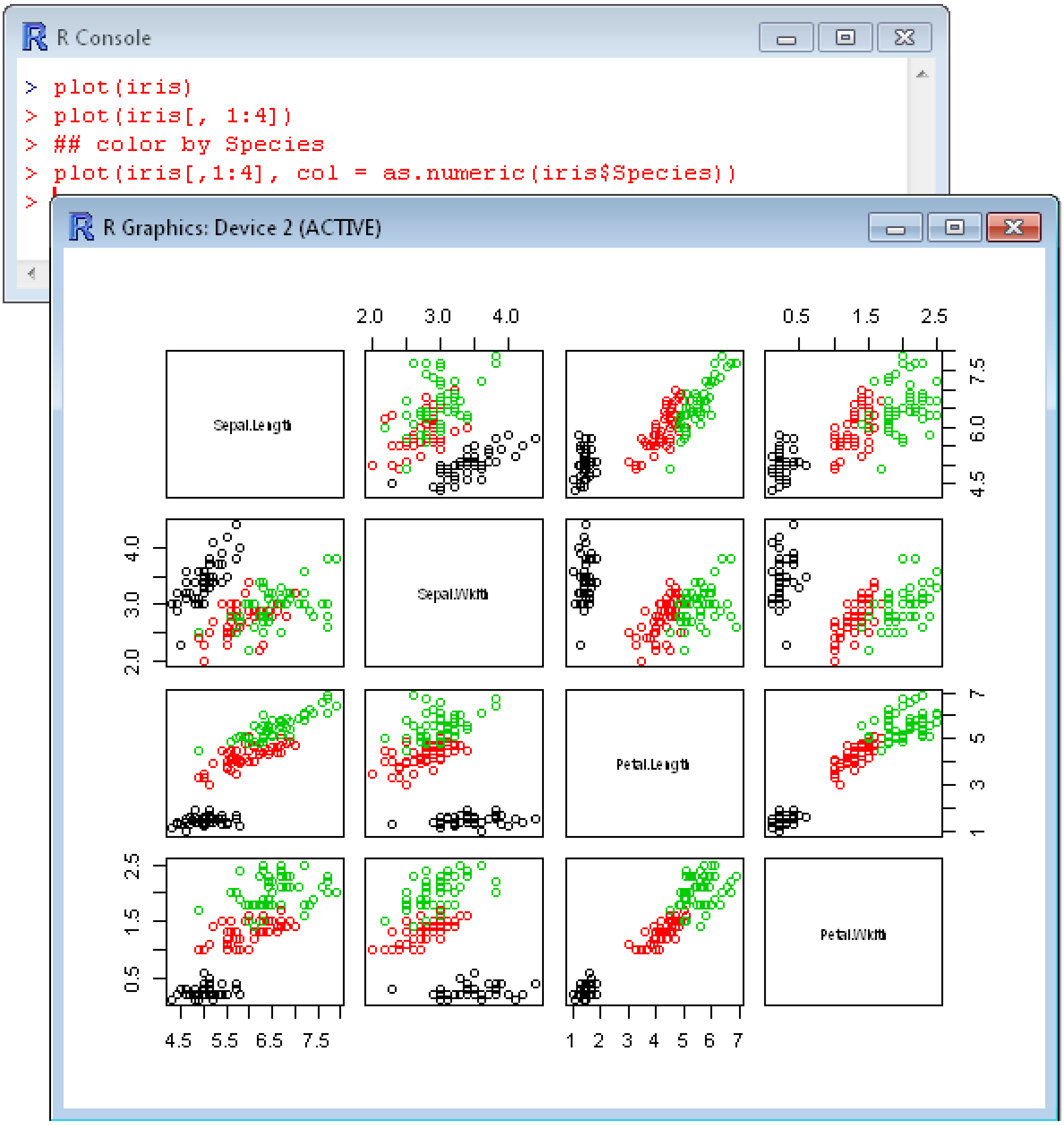


Figure 61 Graph de type lattices avec sous-graphs X-Y

Et dans une version nettement plus compliquée (très utile en finance):

R Console

```
> panel.hist <- function(x, ...) {
+ usr <- par("usr");
+ on.exit(par(usr))
+ par(usr = c(usr[1:2], 0, 1.5) )
+ h <- hist(x,plot=FALSE)
+ breaks <- h$breaks
+ nB <- length(breaks)
+ y <- h$counts; y <- y/max(y)
+ rect(breaks[-nB], 0, breaks[-1], y, ...)
+ }
> # Correlation coefficient
> my.panel.smooth <-function (x, y,
+ col = par("col"),bg = NA,pch = par("pch"),cex = 1,
+ col.smooth = "red",span = 2/3,iter = 3, ...) {
+ points(x, y,pch = pch, col = col,bg = bg, cex = cex)
+ ok <- is.finite(x) & is.finite(y)
+ if (any(ok))
+ lines(lowess(x[ok], y[ok],
+ f = span,iter = iter),col = col.smooth,...)
+ usr <- par('usr')
+ text( (usr[1]+usr[2])/2, (usr[3]+9*usr[4])/10,floor(100*cor(x,y))/100,
+ col="blue", cex=3, adj=c(.5,1) )
+ }
> plot(iris[,1:4],diag.panel = panel.hist,upper.panel = panel.smooth,
+ lower.panel = my.panel.smooth,gap = 0,col=as.numeric(iris$Species))
> 
```


Ce qui donne:

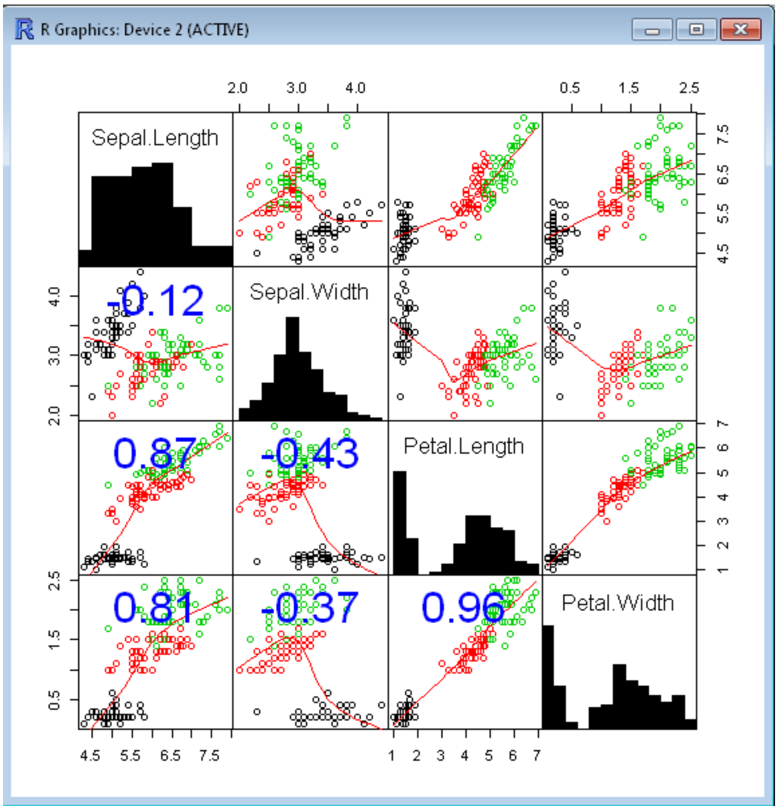


Figure 62 Graph à lattrices avec histogrammes, corrélation et régressions

Ou, toujours avec les couleurs et en jouant avec notre source *Ventes* habituelle:

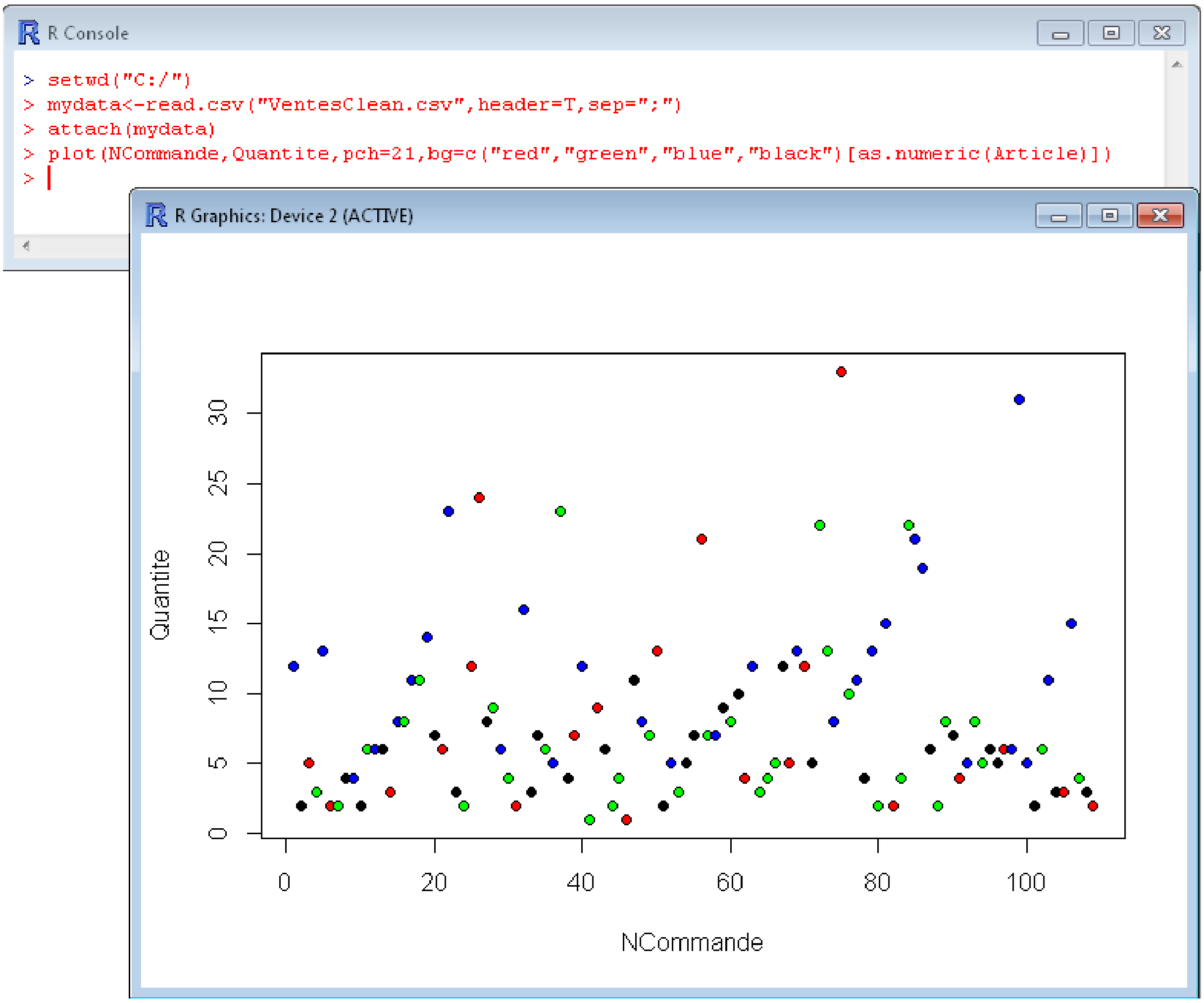
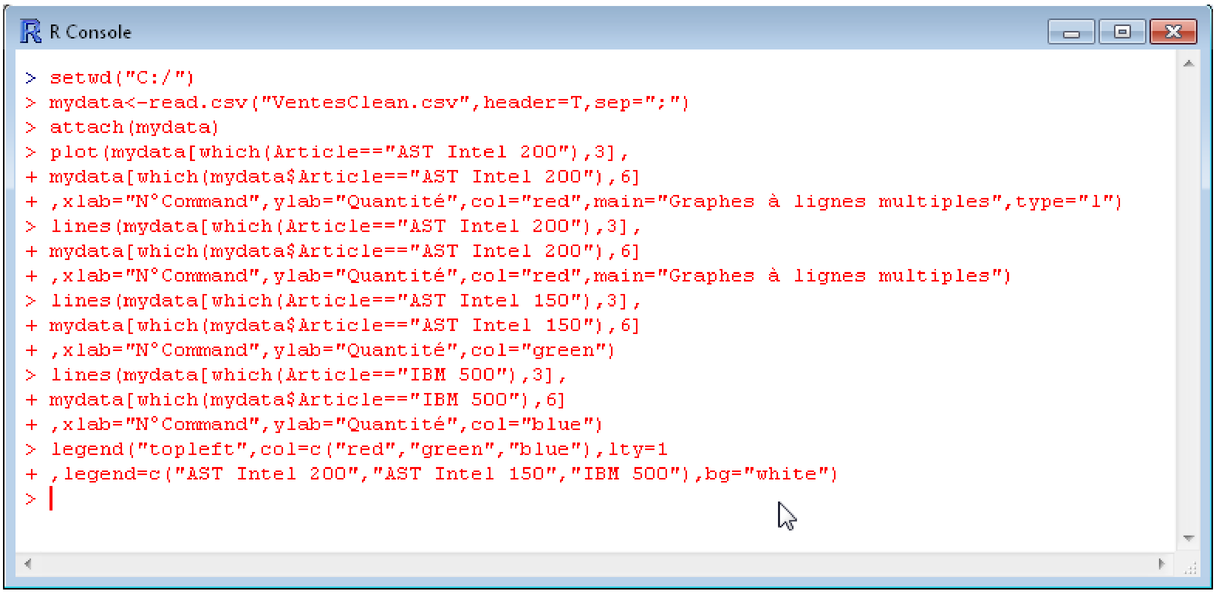


Figure 63 Graph X-Y à points coloriés par catégorie

ou toujours en s'amusant avec les couleurs:



Ce qui donne:

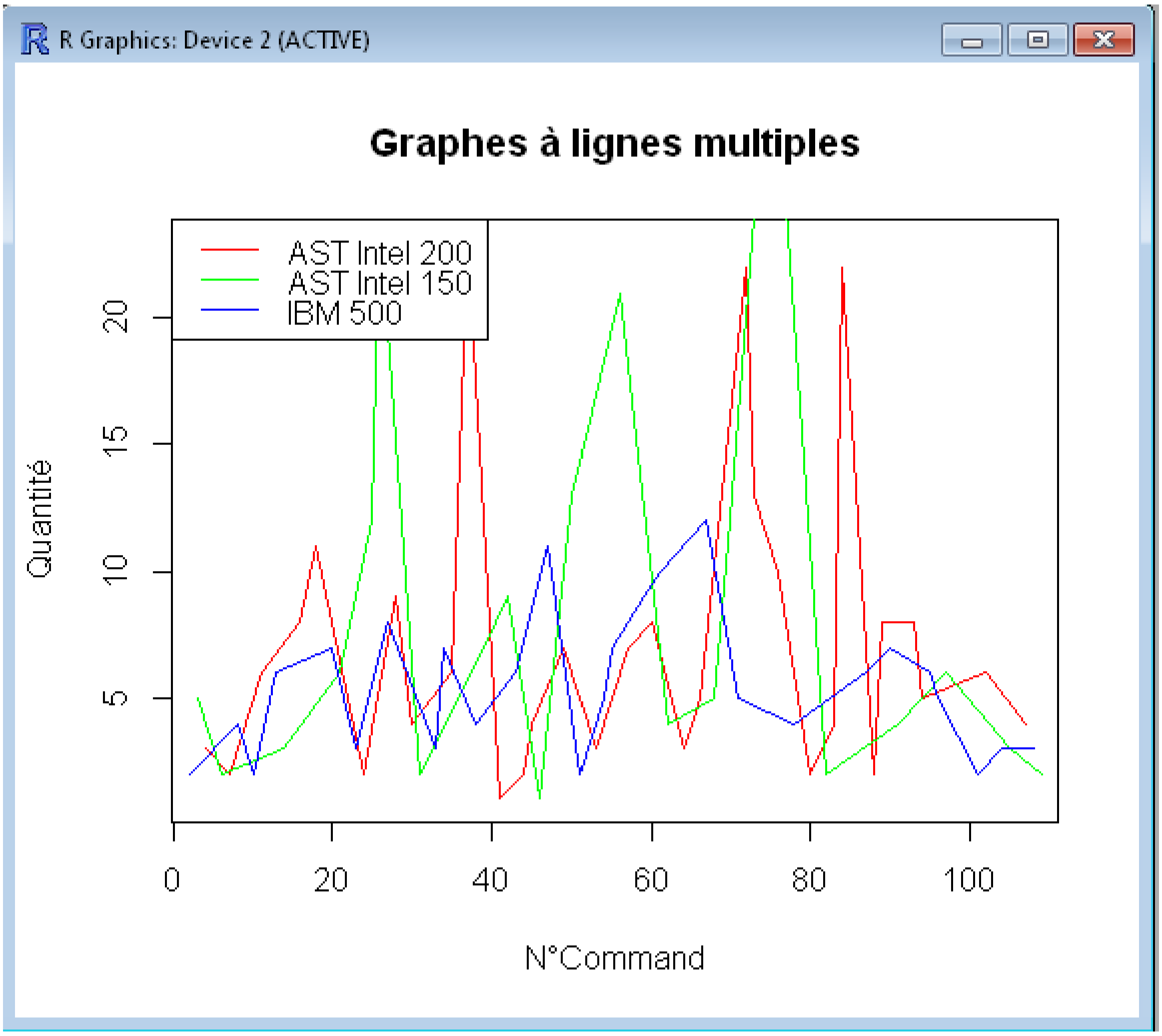
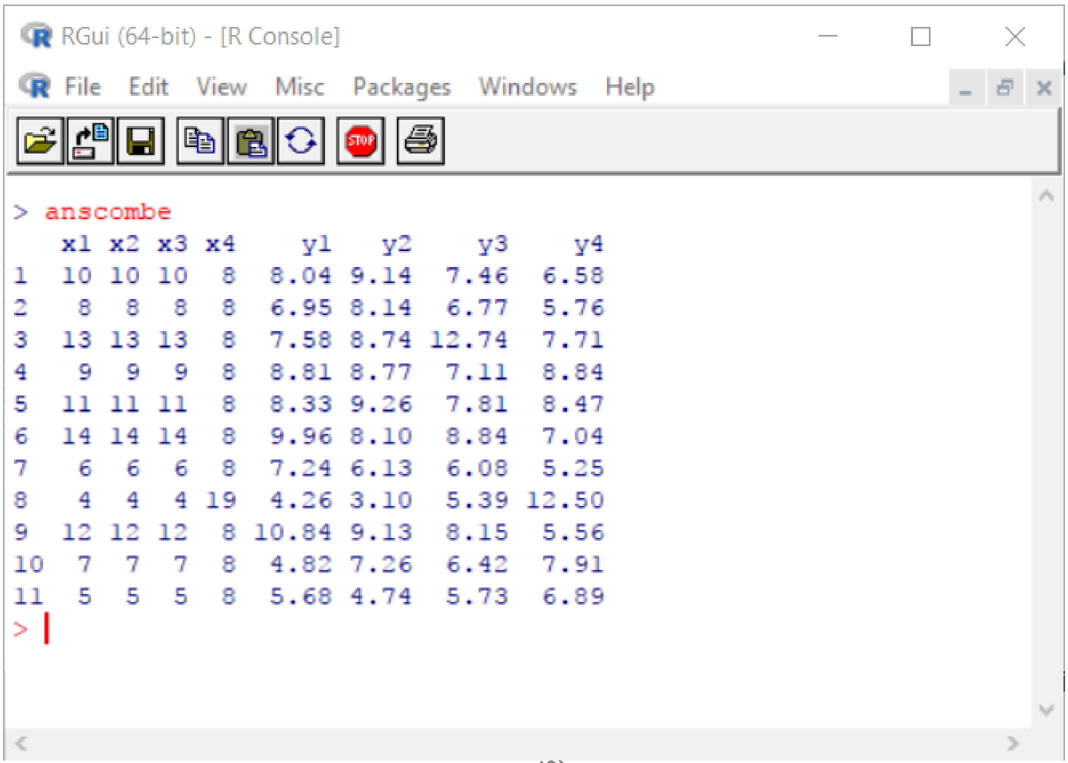


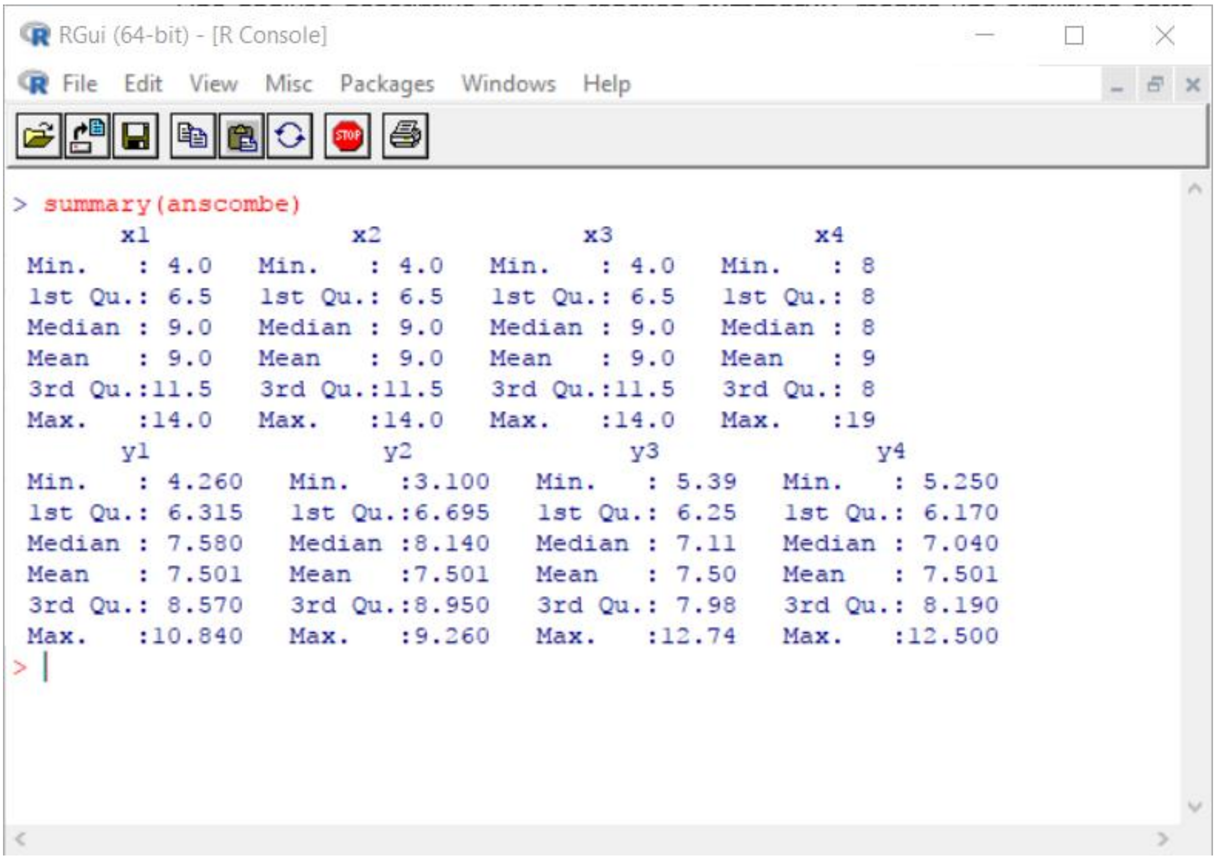
Figure 64 Graph à lignes coloriées

Quartet d'Anscombe

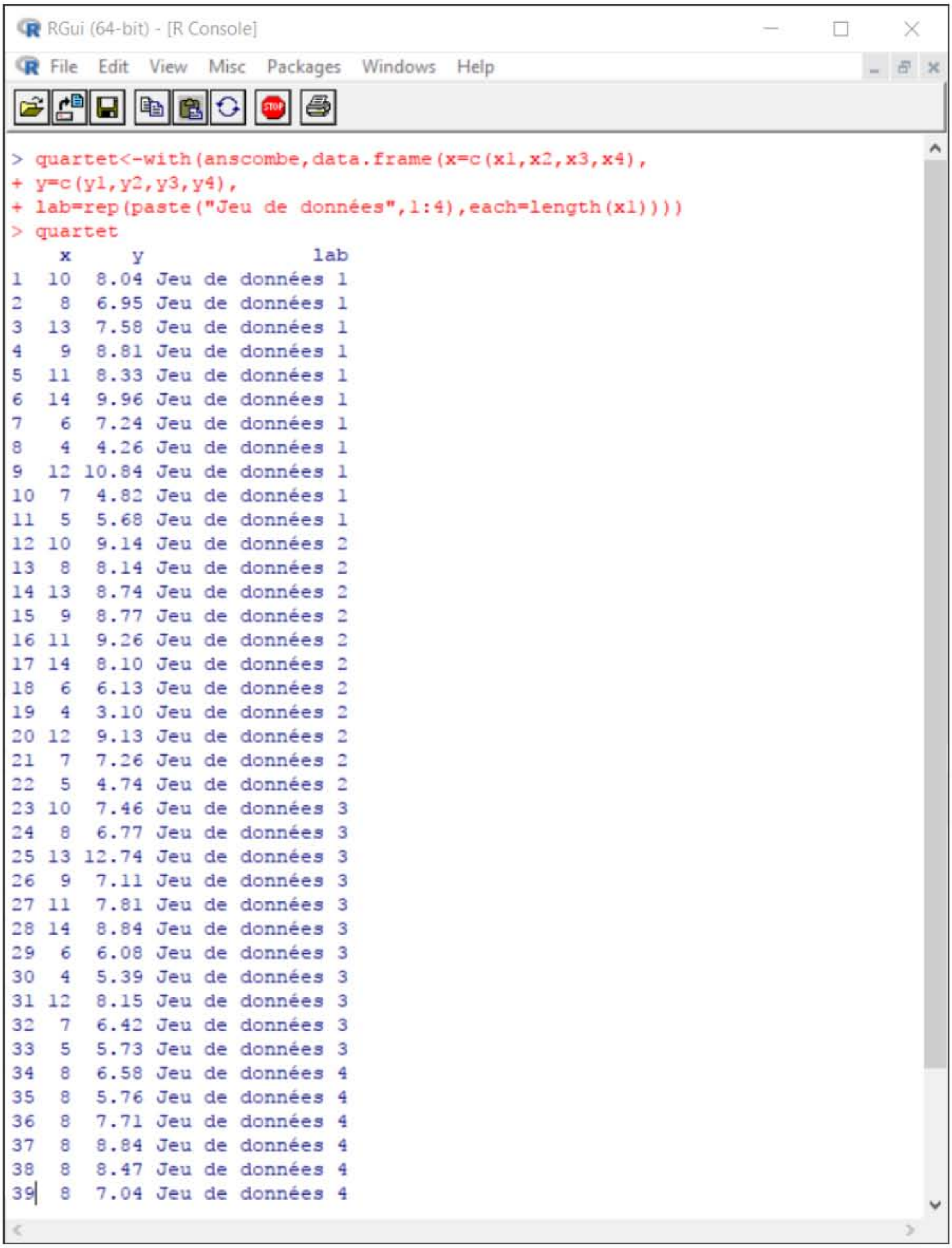
Nous ne pouvons pas, ne pas traiter du cas du quarter d'Anscombe! Voyons voir comment le recréer avec ggplot2:



Nous voyons bien que les statistiques sont similaires:



On adapte le jeu de données pour en faire un graphique (ce qui reste un bon exercice… chiant):



et finalement le graphique:

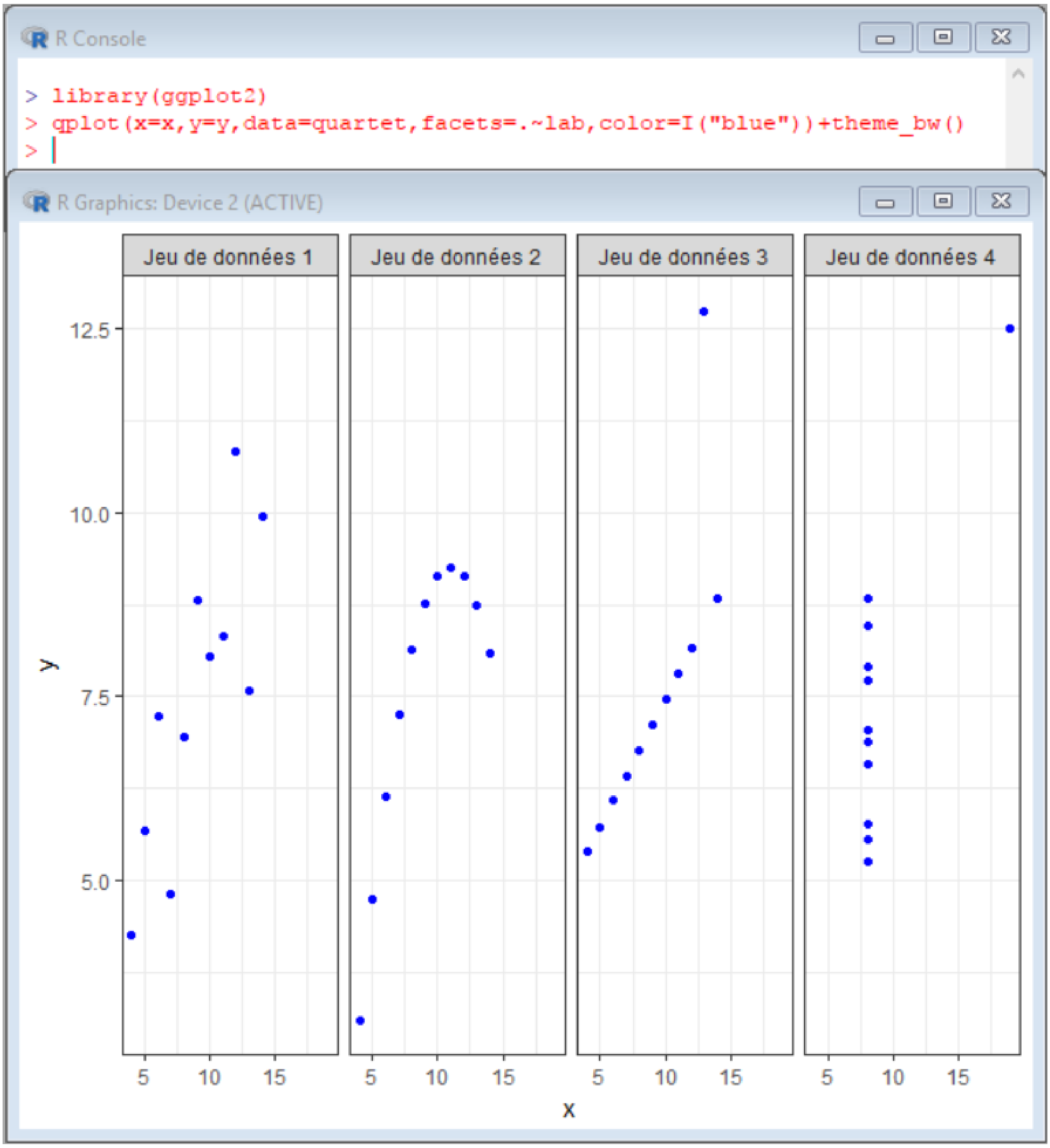


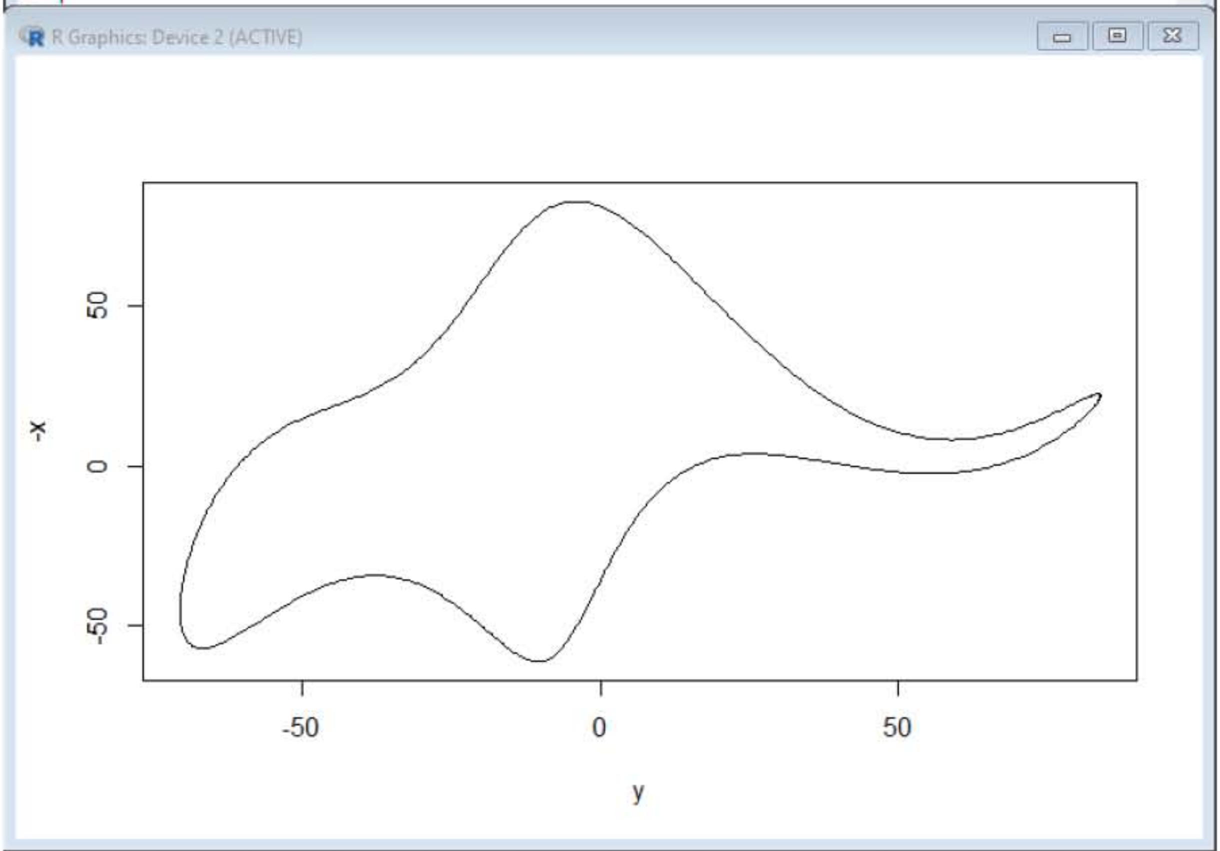
Figure 65 Graph lattice du quartet d'Anscombe

Éléphant de Fermi

Comme metionné dans le cours théorique (chose promie, chose due!):

R Console

```
> x_coordinate <- function(t) {
+   x <- 50*sin(t) + 18*sin(2*t) + 12 * cos(3*t) - 14*cos(5*t)
+   x
+ }
>
> y_coordinate <- function(t) {
+   y <- -30*sin(t) + 8*sin(2*t) - 10*sin(3*t) -60*cos(t)
+   y
+ }
>
> x <- numeric()
> y <- numeric()
> i <- 0.
> for (tt in seq(0,6.28,0.01))#10
+ { i<-i+1
+   x[i] <- x_coordinate(tt)
+   y[i] <- y_coordinate(tt)
+ }
>
> plot(y,-x,type='l')
>
```



Ajouter des marques mineures aux axes

Il n'existe pas de commande par défaut dans la commande **plot( )** pour ajouter des marques mineures. Il faudra installer (du moins dans le cas le plus simple), le package **Hmisc**.

Voici une démonstration utilisant donc la commande **minor.tick( )** :

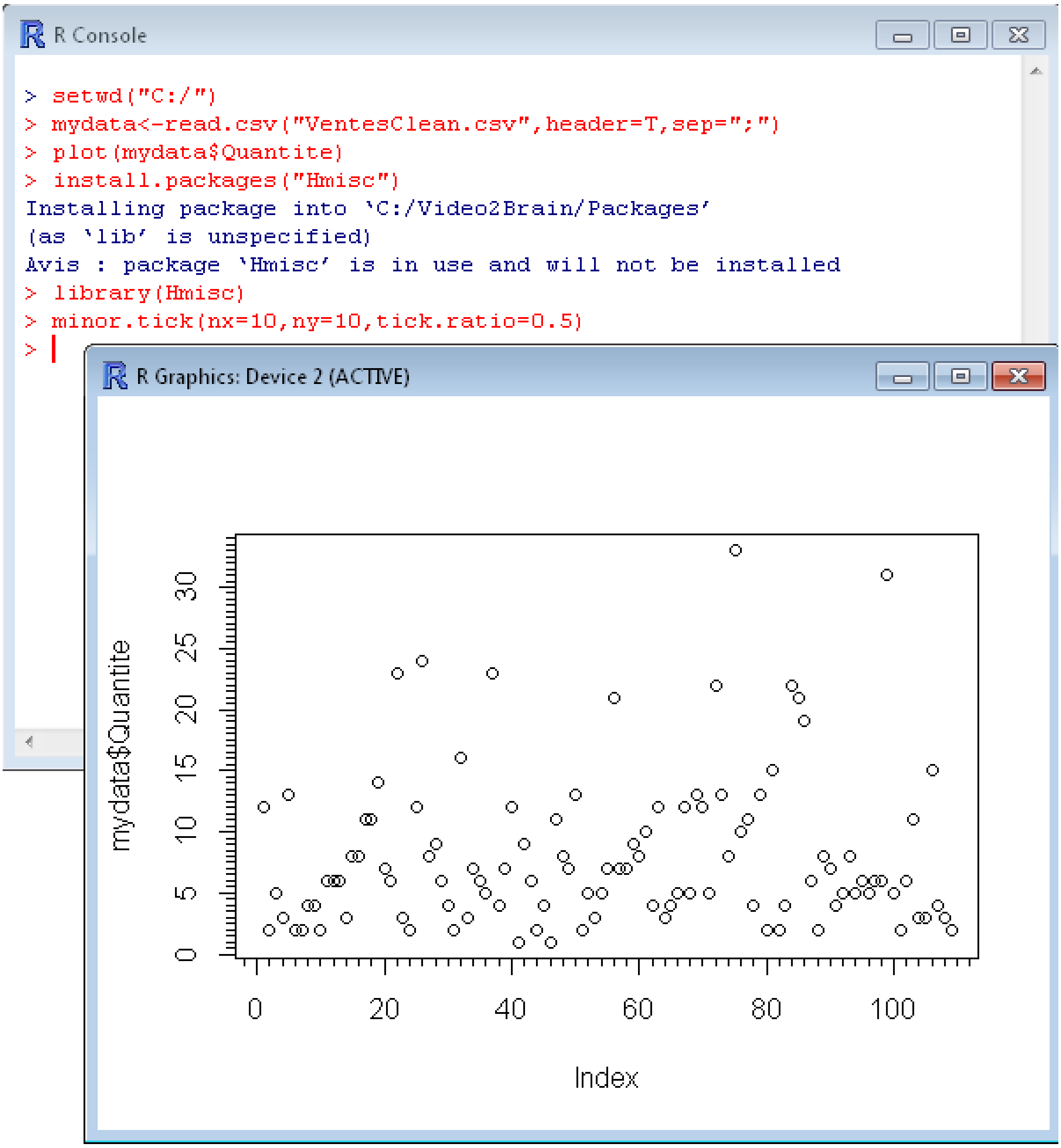
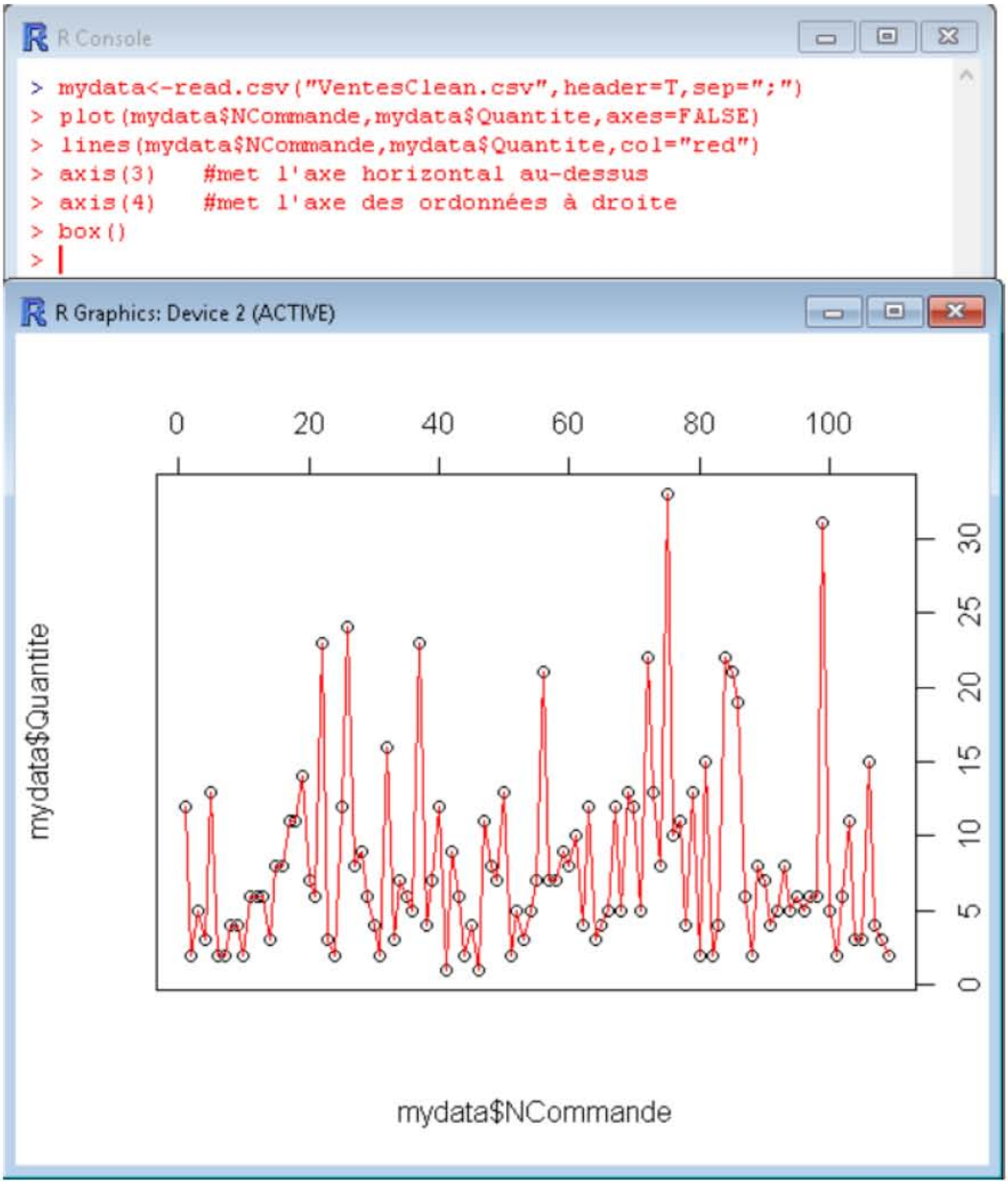


Figure 66 Graph X-Y avec marques mineures aux axes

Positions des axes/Inversion des axes et Dates sur les Axes

Pour déplacer les axes c'est relativement simple:



Maintenant voyons plus grand…

R Console

```
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> x<-as.Date(mydata$DateDeCommande, format="%d.%m.%Y")
> y1 <- mydata$Quantite
> yi1<-y1[rev(order(y1))]
> y2 <- mydata$PrixTotalAvecRabais
> yi2<-y2[rev(order(y2))]
> par(mar=c(5,4,4,5)+.1)
> plot(x,y1,type="l",col="red",ylim=c(max(mydata$Quantite)+70,0),ylab="Quantité",axes=F)
> #astuce pour hacker l'axe des X avec un format lisible
> axis(3,at=NULL, labels=F)
> text(x, par("usr")[3]*(-0.1), labels = paste(x,' '), srt = 90, pos = 3, xpd = TRUE,cex=.7)
> #Sinon remplacer les deux lignes du dessus par... (résultant décevent)
> #axis(3,x,format(x,"%b %d"),cex.axis=.7)
> axis(2)
> par(new=TRUE)
> plot(x, y2,,type="l",col="blue",xaxt="n",yaxt="n",xlab="",ylab="",ylim=rev(range(yi2)))
> axis(4)
> mtext("Cash",side=4,line=3)
> legend("bottomleft",col=c("red","blue"),lty=1,legend=c("Quantités","Cash"))
>
```

Ce qui donne:

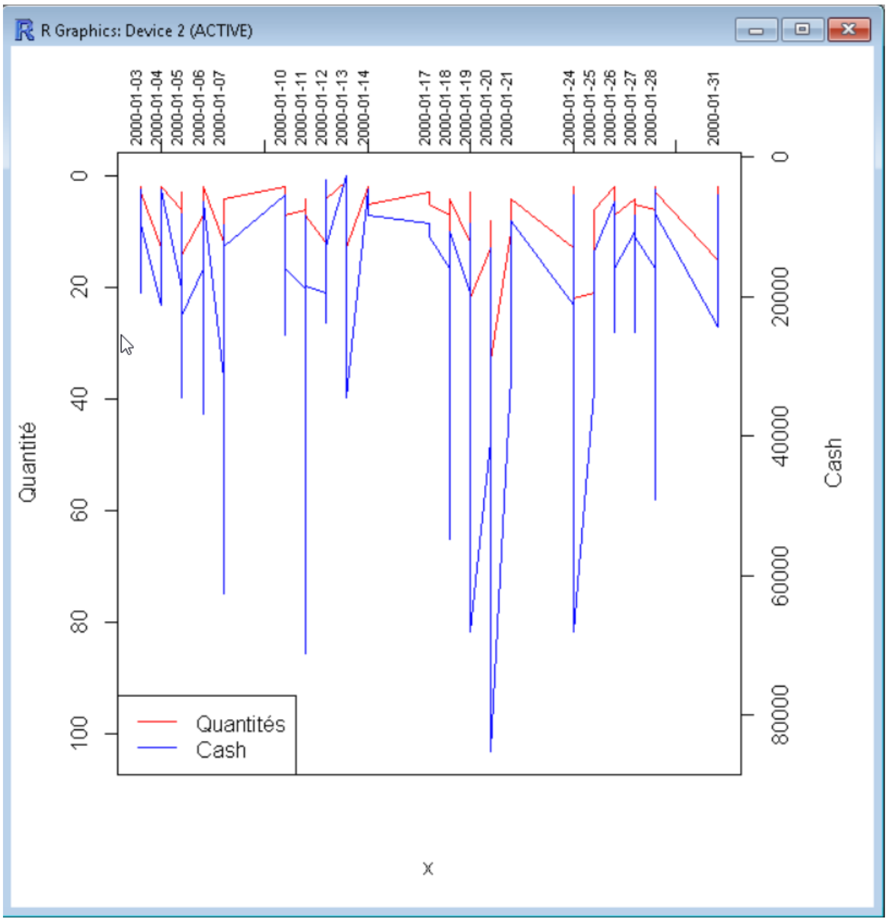


Figure 67 Graph à lignes avec triple axes

Ajouter des légendes aux points

Nous pouvons également ajouter des légendes à chaque point. Par exemple:

```
> setwd("C:/")
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> mydata<-aggregate(mydata$Prix.total.avec.rabais,by=list(mydata$Article,mydata$Quantité),FUN=sum)
> colnames(mydata)<-c("Articles","Quantité","PTAR")
> plot(mydata$Quantité,mydata$PTAR,main="PTAR f(Quantité) par article",
+ xlab="Quantités",ylab="PTAR",pch=18,col="blue")
> text(mydata$Quantité,mydata$PTAR,mydata$Articles,cex=0.6,pos=4,col="red")
```

Donnera:

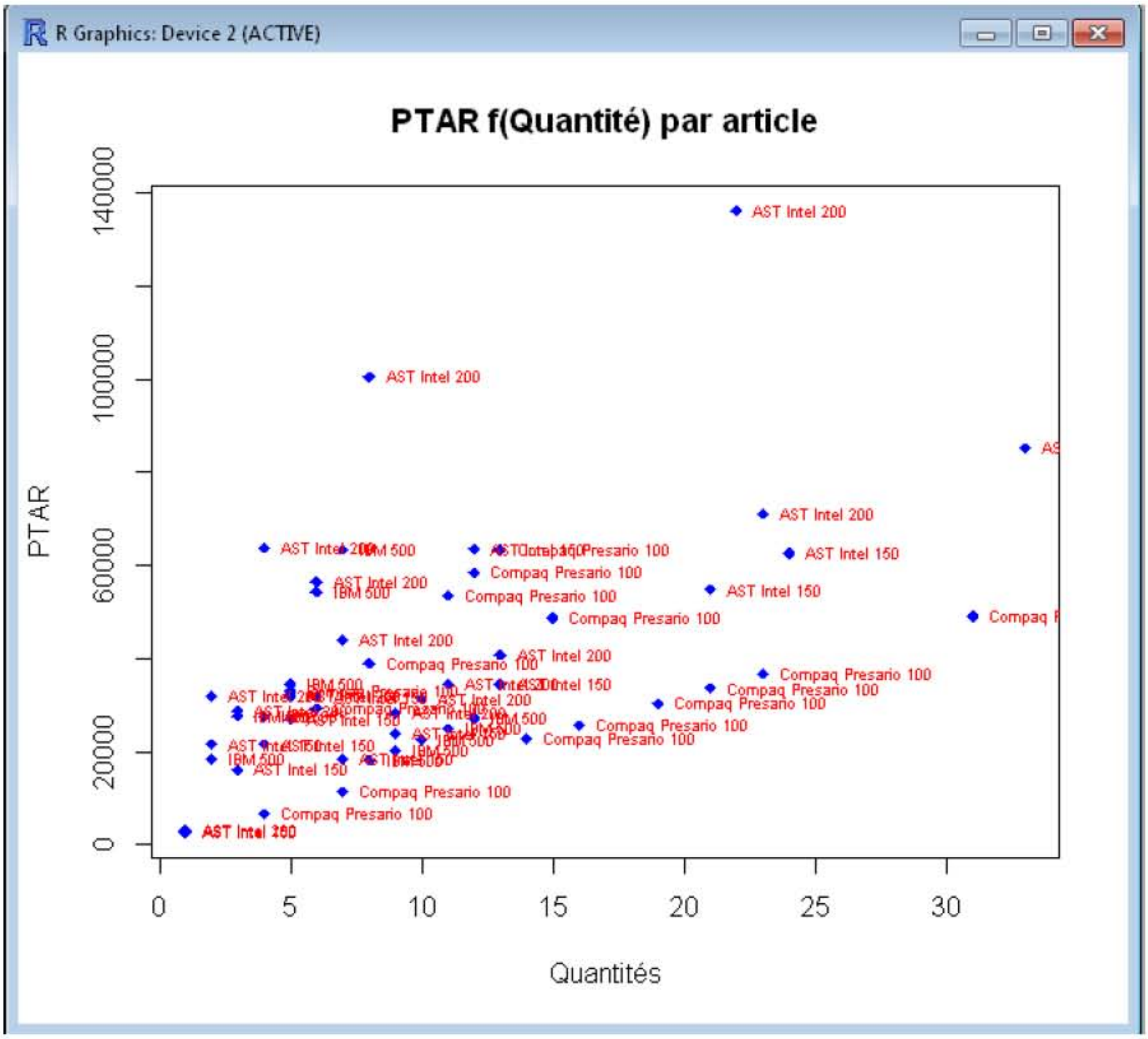


Figure 68 Graphy X-Y avec légendes sur chaque point

Zoomer sur un sous-ensemble

En installant le package **plotrix** (que nous retrouverons de nombreuses fois) nous pouvons dans certains cas faire une manipulation utile.

```
> library(plotrix)
> setwd("C:/")
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> mydata<-aggregate(mydata$Prix.total.avec.rabais,by=list(mydata$Article,mydata$Quantité),FUN=sum)
> colnames(mydata)<-c("Articles","Quantité","PTAR")
> zoomInPlot(mydata$Quantité,mydata$PTAR,rxlim=c(2.5,7.5),rylim=c(17000,30000),
+ zoomtitle="Zoom In Plot",titlepos=-1.5,col="blue",pch=18)
>
```

Ce qui donne:

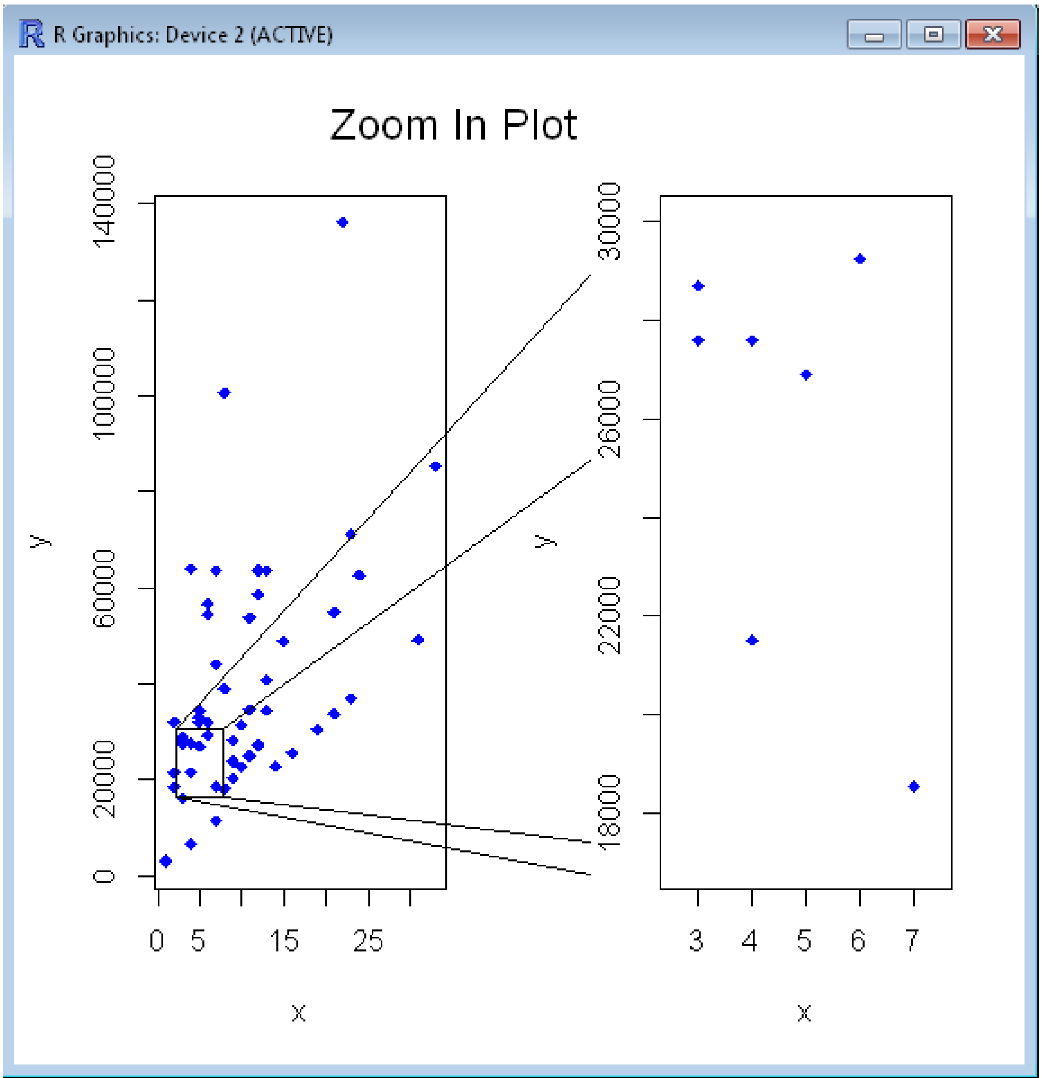


Figure 69 Graph X-Y avec zoom sur zone Graph X-Y sans ggplot2

On peut faire la même chose avec ggplot:

```
> library("ggplot2")
>
> set.seed(42)
> n <- 1000
> x <- runif(n) * 3
> y <- x * sin(1/x) + rnorm(n) / 25
> df <- data.frame(x = x, y = y)
>
> #le graphique dans son ensemble
> p1 <- ggplot(df, aes(x, y)) + geom_point(alpha = 0.3) +
+     geom_smooth(se = FALSE) + theme_bw()
> #le sous-ensemble qui nous intéresse
>
> p2 <- ggplot(df, aes(x, y)) + geom_point(alpha = 0.3) +
+     geom_smooth(se = FALSE) + scale_x_continuous(limits = c(0, 0.5)) +
+     scale_y_continuous(limits = c(-0.3, 0.6)) + theme_bw()
>
> #et au final
> p1 + annotation_custom(ggplotGrob(p2), xmin = 1, xmax = 3, ymin = -0.3, ymax = 0.6)
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
Warning messages:
1: Removed 812 rows containing non-finite values (stat_smooth).
2: Removed 812 rows containing missing values (geom_point).
>
```

Ce qui donne:

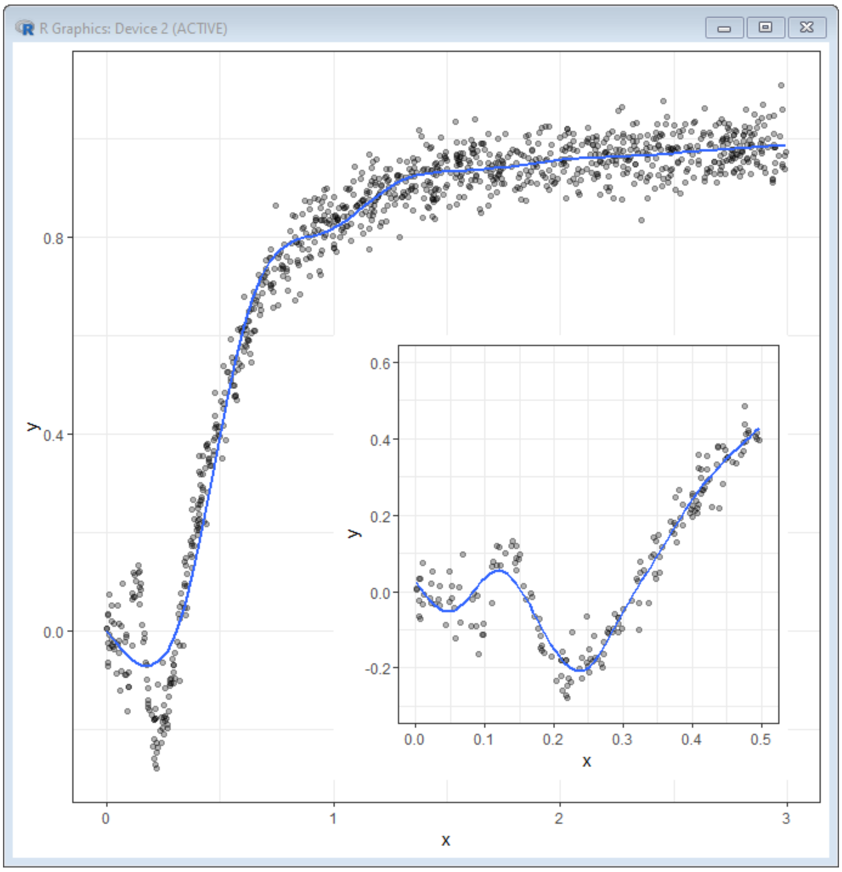
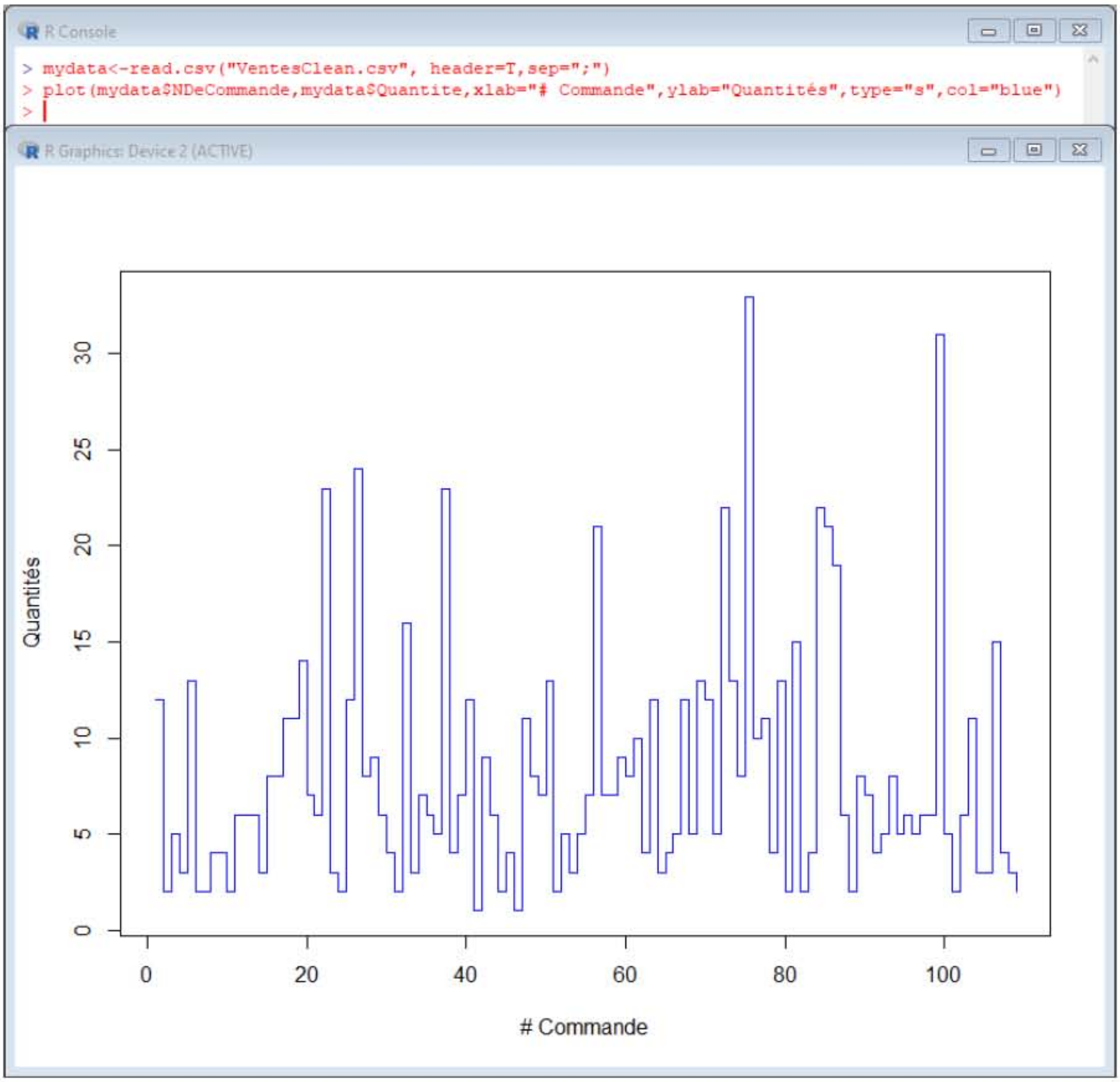


Figure 70 Graph X-Y avec zoom sur zone avec ggplot2

Diagramme à ligne en escalier

Rien de spécial ici à part que c'est un graph souvent apprécié pour sa lisibilité:

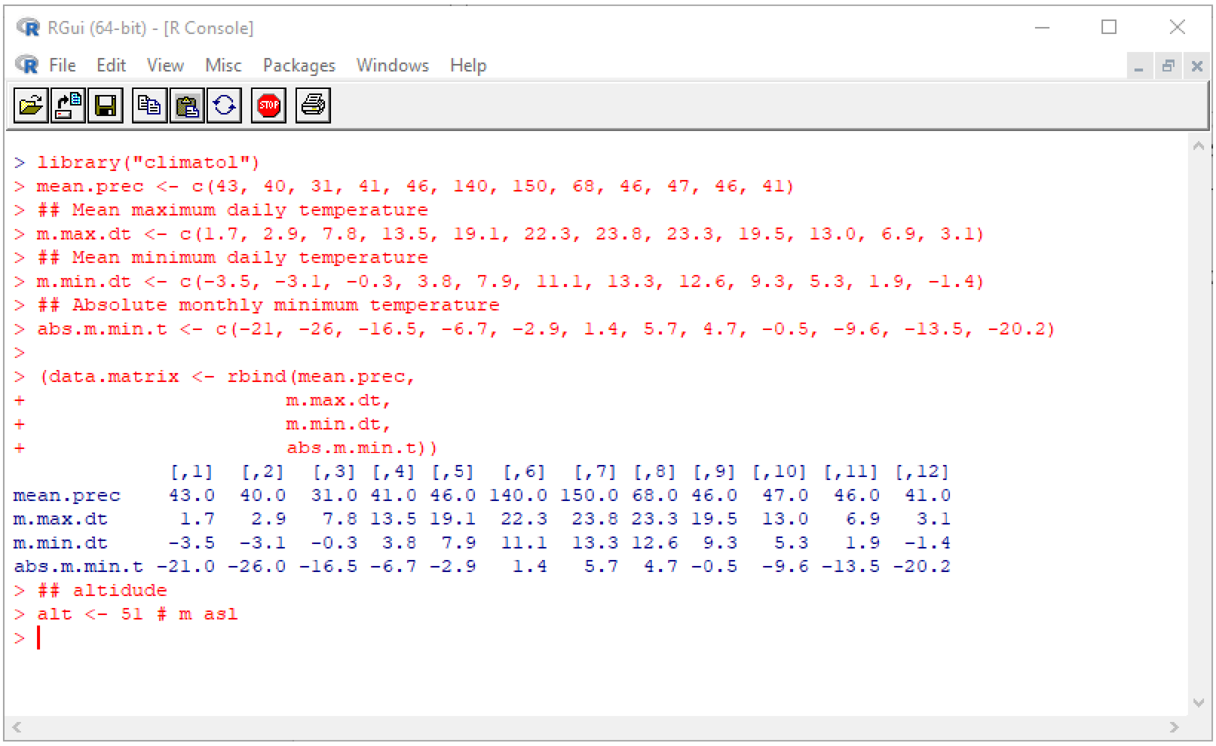


Graphique de Walter-Lieth

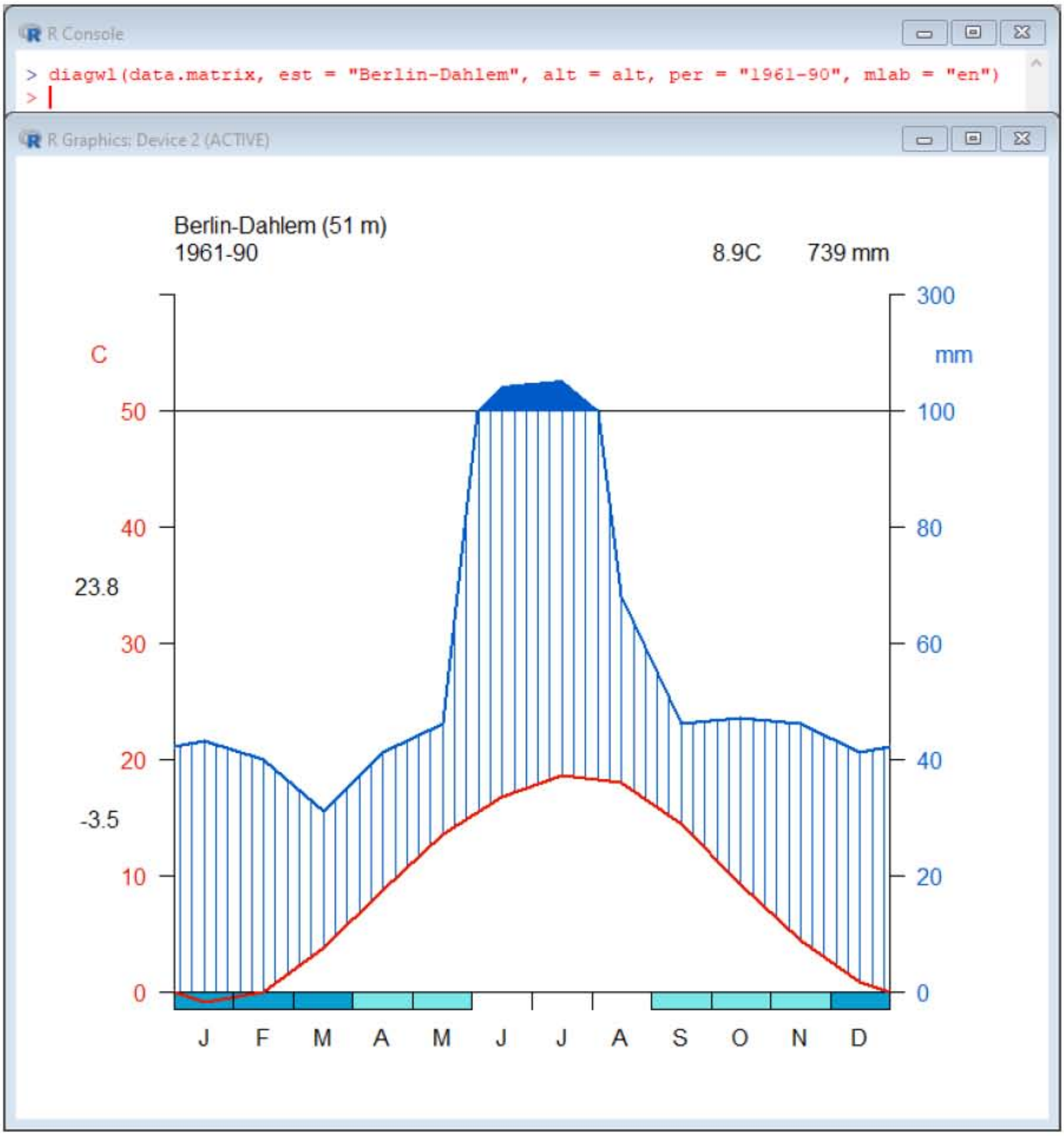
Une forme très populaire de visualisation des données météorologiques est le graphique de Walter-Lieth. Un tel graphique multi-axes à lignes illustre les précipitations et les changements de température tout au long de l'année. Il est particulièrement utile de déterminer les mois arides et humides tout au long du cycle annuel.

Voyons comment en produire un avec le package climatlor et sa fonction diagwl( )!

D'abord nous créons le jeu de données à la main:



Une fois les variables et données créées (ou importées…!), nous pouvons faire la graphique correspondant très simplement:


R Console
> diagwl(data.matrix, est = "Berlin-Dahlem", alt = alt, per = "1961-90", mlab = "en")
R Graphics: Device 2 (ACTIVE)
Berlin-Dahlem (51 m)
1961-90
8.9C 739 mm
C
mm
23.8
-3.5
J F M A M J J A S O N D

Points avec courbes d'iso-densité

Pour cet exemple, il vous faudra avoir téléchargé d'abord le package **ggplot2**. Une fois ceci fait, nous pouvons utiliser la commande **ggplot( )** avec **geom_density2d( )** pour obtenir:

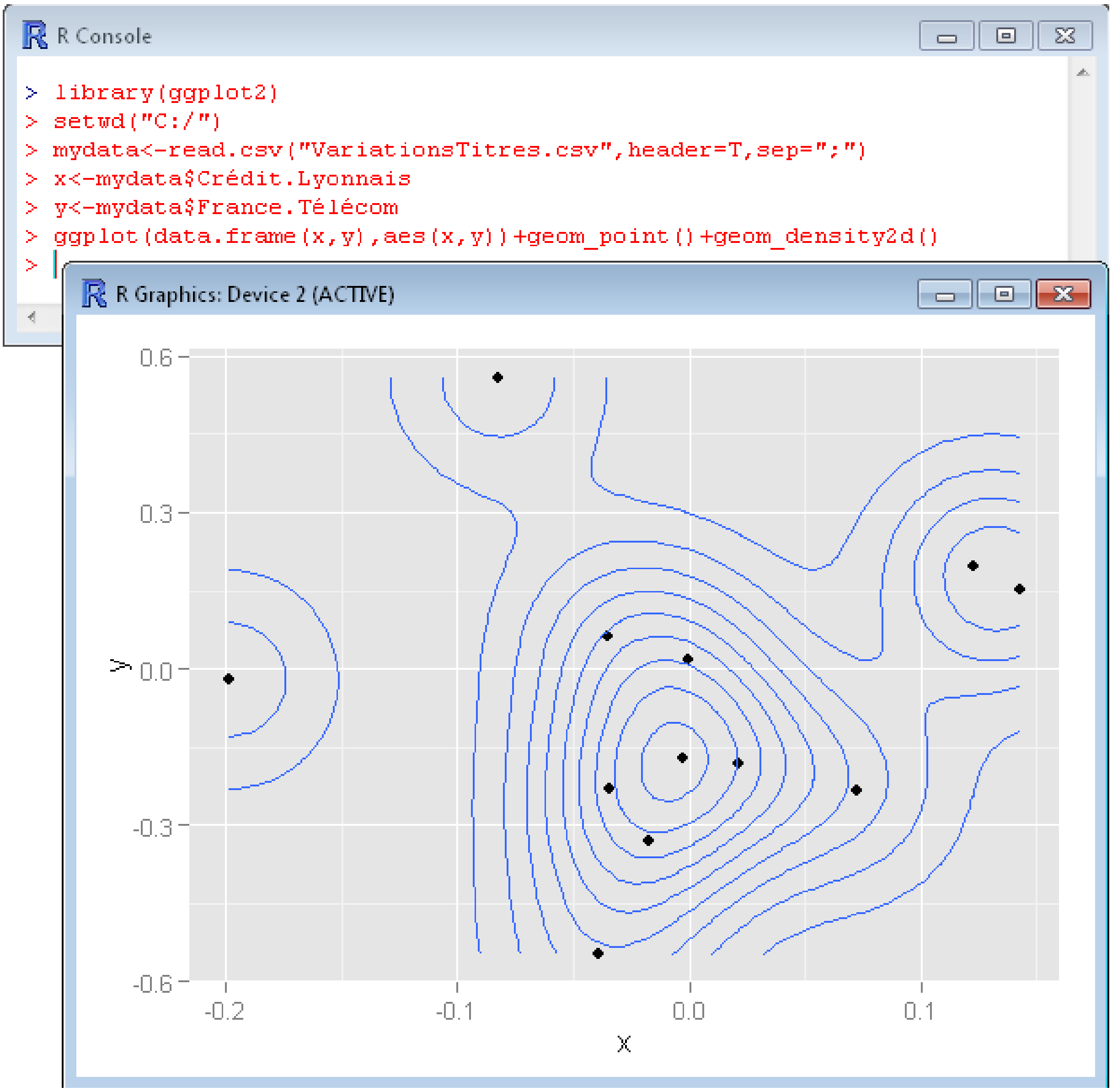


Figure 71 Graph X-Y avec courbes de densité (isoclines)

ou bien en version améliorée:

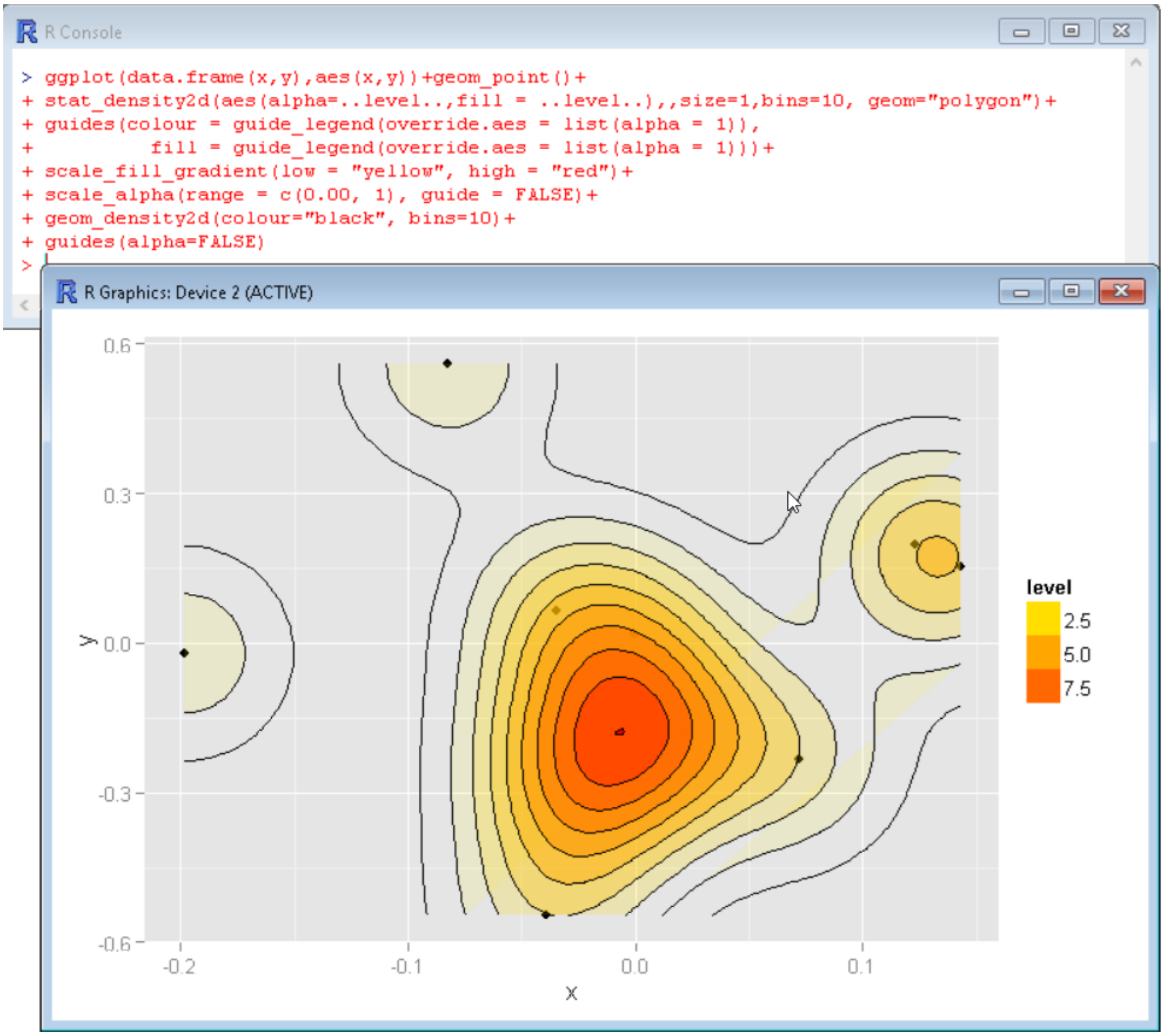


Figure 72 Graph X-Y avec courbes de densité (isoclines) et gradients de couleurs (ggplot)

Nous pouvons faire mieux avec les packages **MASS** et **RColorBrewer** en associant les noyaux de densité locale avec la commande **kde2d( )** et la commande **contours( )** utilisant le dégradé de couleurs du package **RColorBrewer**:

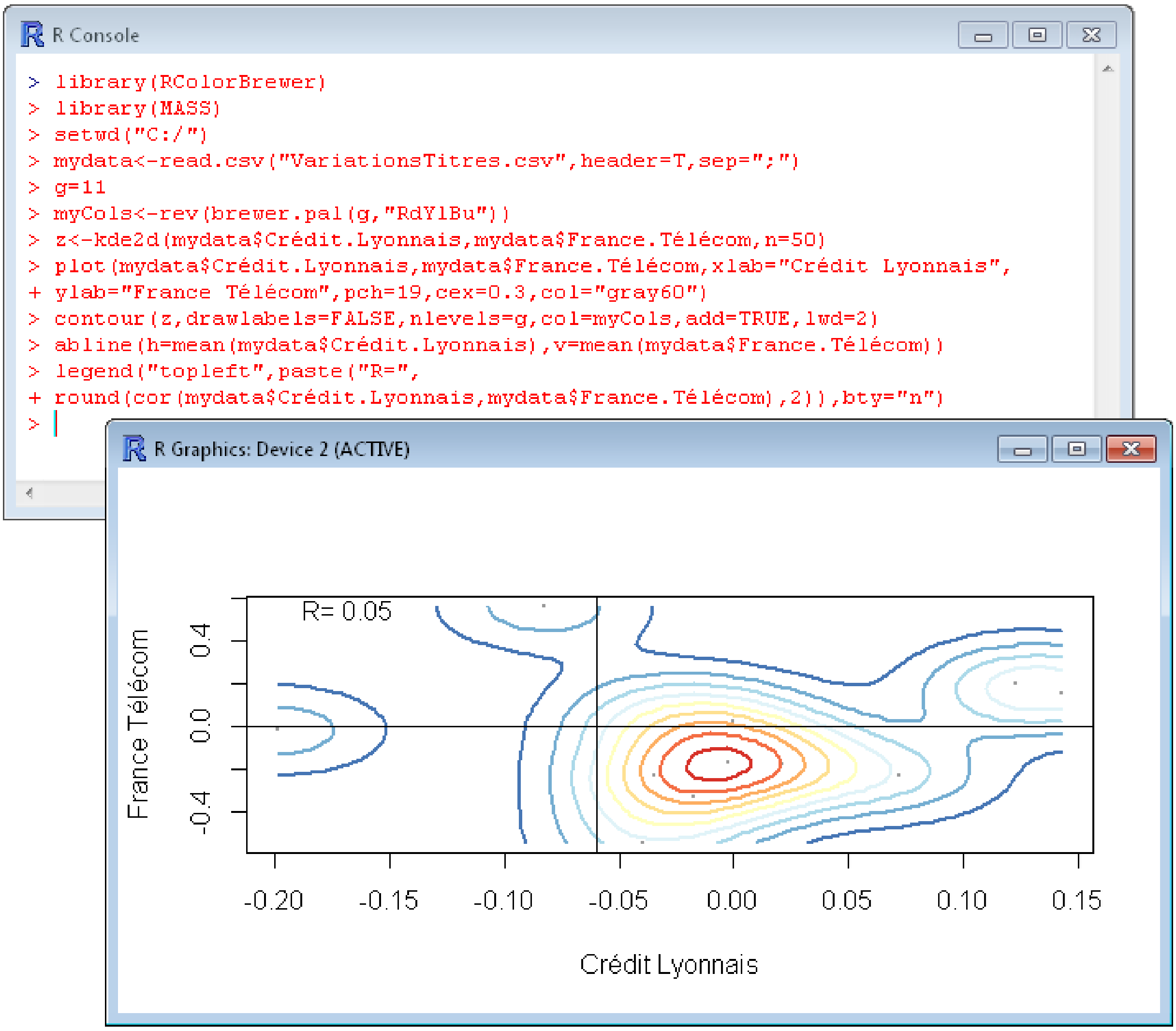
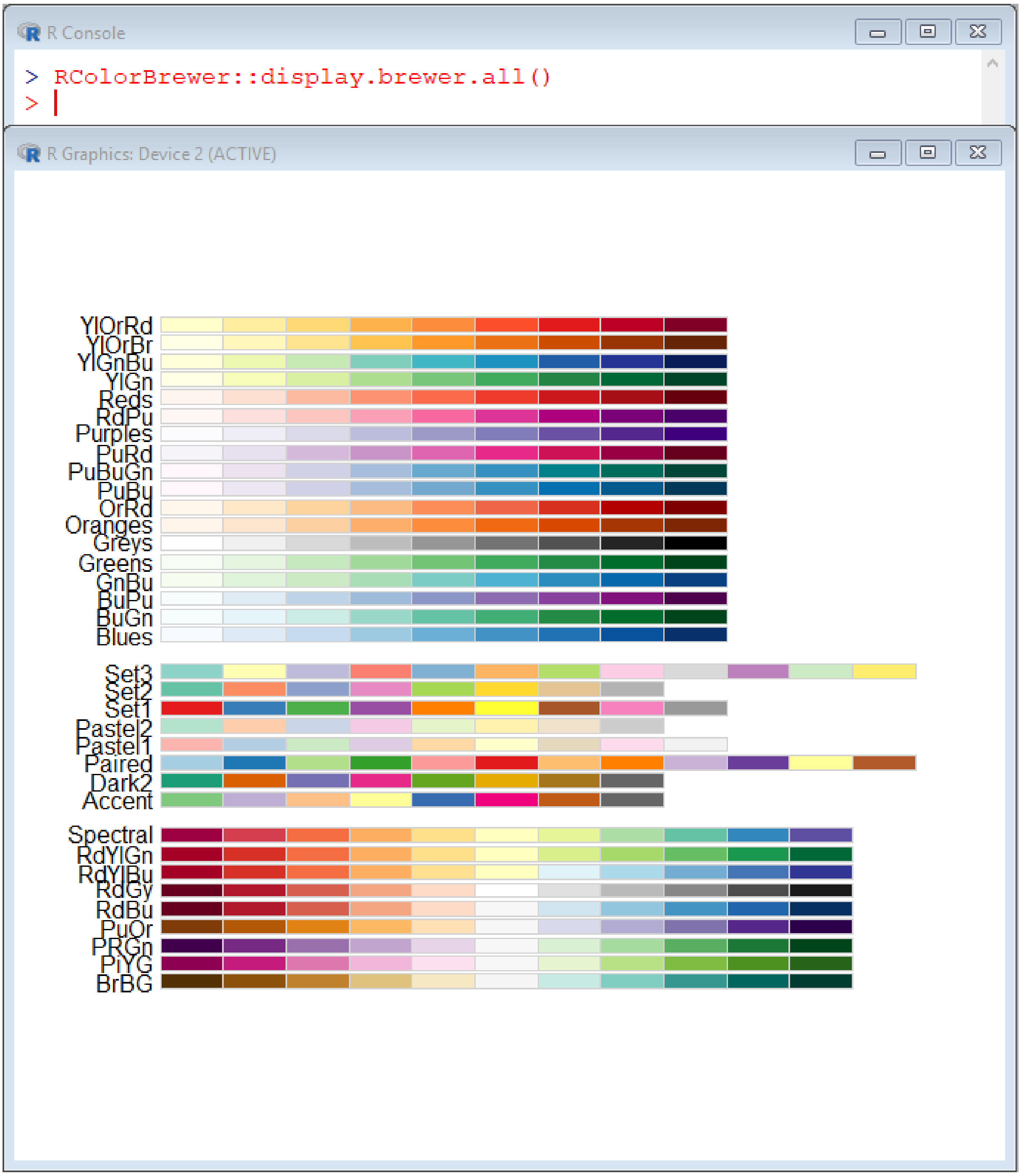


Figure 73 Graph X-Y avec courbes de densité (isoclines) en couleurs

Au besoin pour le package RColorBrewer n'hésitez pas à utiliser la commande **RColorBrewer::display.brewer.all()** qui vous donner un rappel des types de dégradés:


R Console
> RColorBrewer::display.brewer.all()
>
R Graphics: Device 2 (ACTIVE)
YlOrRd
YlOrBr
YlGnBu
YlGn
Reds
RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG

Points avec variation de couleurs et zone géographique

Nous allons ici utiliser la fonction **spplot( )** du package **sp**.

D'abord nous chargeons et préparons les données brutes:

```
> library(sp)
>
> meuse<-read.csv("c:/tmp/Meuse.csv",header=T,sep=";")
> #on convertit les colonnes x,y en coordonnées
> coordinates(meuse)=~x+y
> head(meuse)
       coordinates cadmium copper lead zinc  elev       dist   om ffreq soil lime landuse dist.m optional
1 (181072, 333611)    11.7     85  299 1022 7.909 0.00135803 13.6     1    1    1      Ah     50     TRUE
2 (181025, 333558)     8.6     81  277 1141 6.983 0.01222430 14.0     1    1    1      Ah     30     TRUE
3 (181165, 333537)     6.5     68  199  640 7.800 0.10302900 13.0     1    1    1      Ah    150     TRUE
4 (181298, 333484)     2.6     81  116  257 7.655 0.19009400  8.0     1    2    0      Ga    270     TRUE
5 (181307, 333330)     2.8     48  117  269 7.480 0.27709000  8.7     1    2    0      Ah    380     TRUE
6 (181390, 333260)     3.0     61  137  281 7.791 0.36406700  7.8     1    2    0      Ga    470     TRUE
> #on charge les coordonnées de la riviere et on construit son polygone
> meuse.riv<-read.csv("c:/tmp/MeuseRiviere.csv",header=T,sep=";")
> head(meuse.riv)
        V1       V2
1 182003.7 337678.6
2 182136.6 337569.6
3 182252.1 337413.6
4 182314.5 337284.7
5 182331.5 337122.3
6 182323.9 336986.2
> meuse.sr = SpatialPolygons(list(Polygons(list(Polygon(meuse.riv)),"meuse.riv")))
> river = list("sp.polygons", meuse.sr, fill = "lightblue")
> 
```

Ce qui donne (voir page suivante):

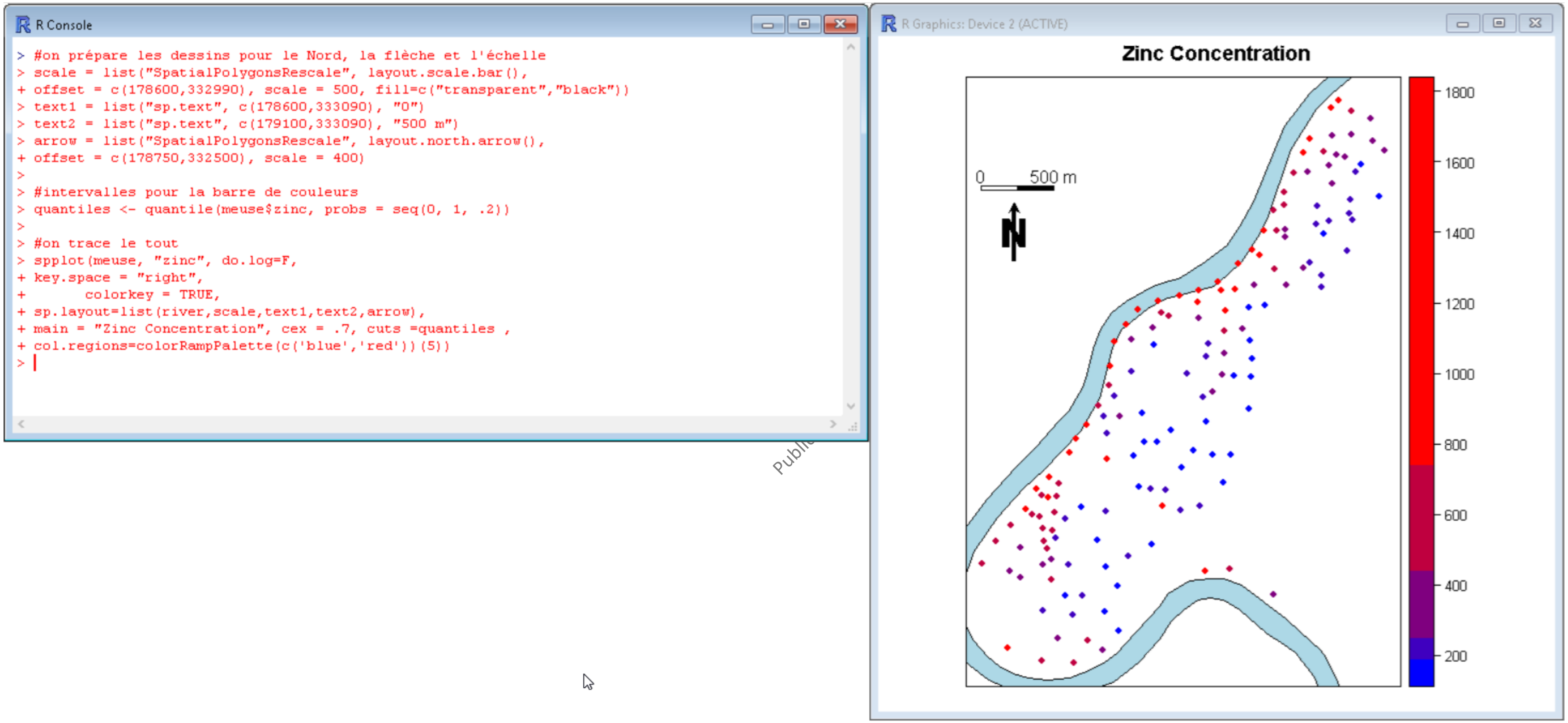


Figure 74 Graphe X-Y avec rivière et légende de couleurs d'amplitudes

Points avec histogrammes et lissage

En utilisant la package **Hmisc** et la commande **histSpike( )** nous pouvons associer dans un plot de points des histogrammes avec des fonctions de densité au plus proche sur chacun des axes de projection:

```
R Console
> library(Hmisc)
Il y a eu 11 avis (utilisez warnings() pour les visionner)
> setwd("C:/")
> mydata<-read.csv("VariationsTitres.csv",header=T,sep=";")
> x<-mydata$Crédit.Lyonnais;y<-mydata$France.Télécom
> plot(x,y,pch=19,xlab="Crédit Lyonnais",ylab="France Télécom")
> grid()
> histSpike(x,add=TRUE,col="green4",lwd=2)
> histSpike(y,4,add=TRUE,col="blue",lwd=2)
> histSpike(x,type="density",col="red",add=TRUE)
> histSpike(y,4,type="density",col="red",add=TRUE)
> 
```

Ce qui donne:

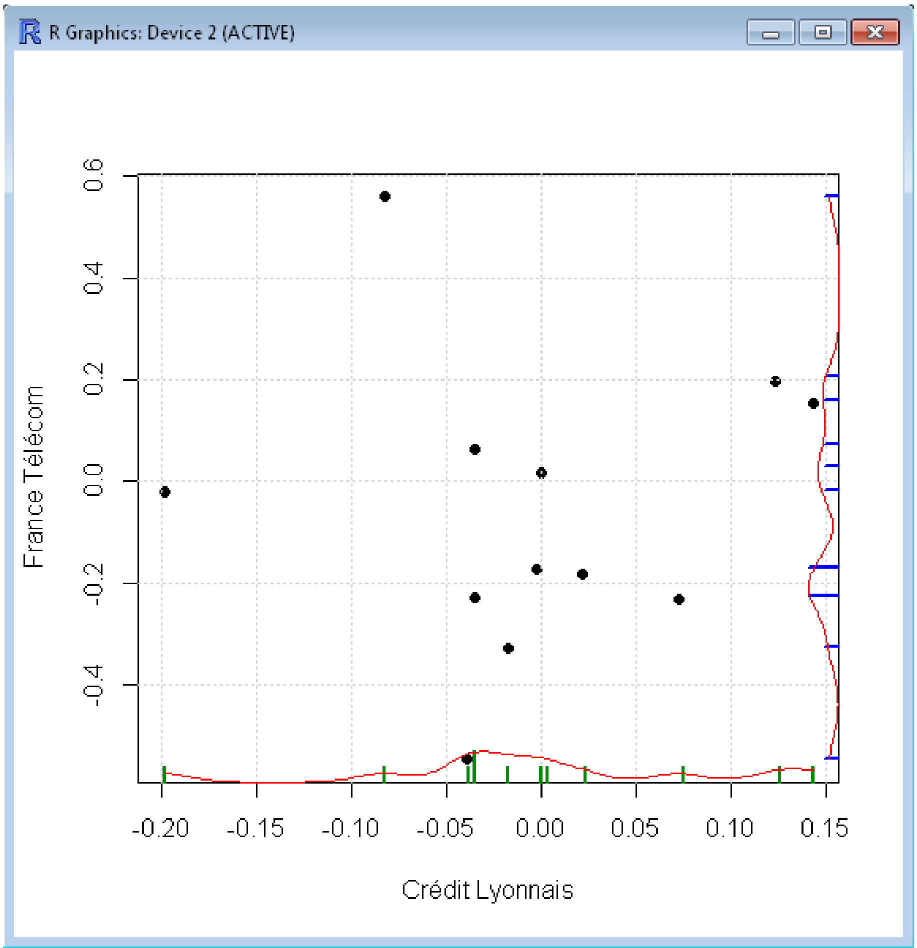
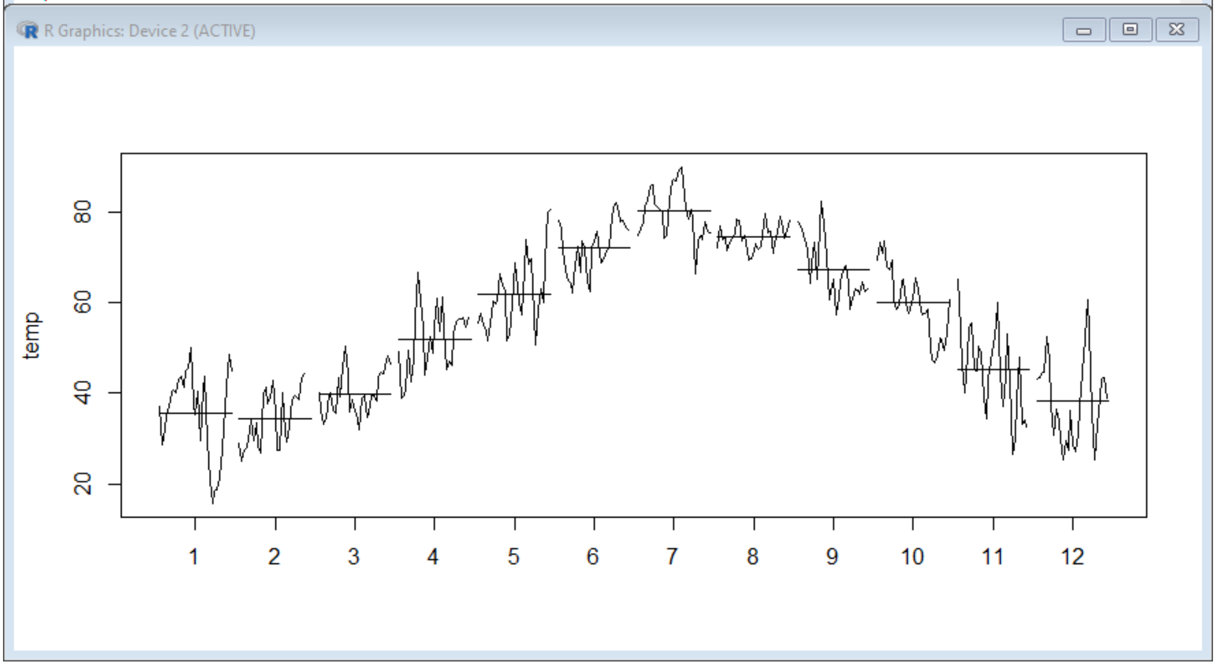


Figure 75 Graph X-Y avec stem (tiges) et courbes de densité en marge

Graphiques à cycles

Ici nous utilisons simplement la fonction **monthplot( )** intégrée par défaut dans R:

```
> library("nycflights13")
> str(weather[,(3:6)])
tibble [26,115 x 4] (S3: tbl_df/tbl/data.frame)
 $ month: int [1:26115] 1 1 1 1 1 1 1 1 1 1 ...
 $ day  : int [1:26115] 1 1 1 1 1 1 1 1 1 1 ...
 $ hour : int [1:26115] 1 2 3 4 5 6 7 8 9 10 ...
 $ temp : num [1:26115] 39 39 39 39.9 39 ...
> # Trouver la valeur moyenne
> tempdata <- aggregate(temp ~ month + day, data=weather, mean)
>
> #monthplot fait partie du package graphics de base de R
> with(tempdata, monthplot(temp, times=day , phase=month, ylab = "temp"))
>
```



Ajout de barres d'erreurs

Il manque un grand classique cependant que sont les barres d'erreurs (elles sont disponibles dans les tableurs par défaut) et il nous les faut absolument dans un logiciel comme R. Pour cela, il va nous falloir associer un script à notre source de données.

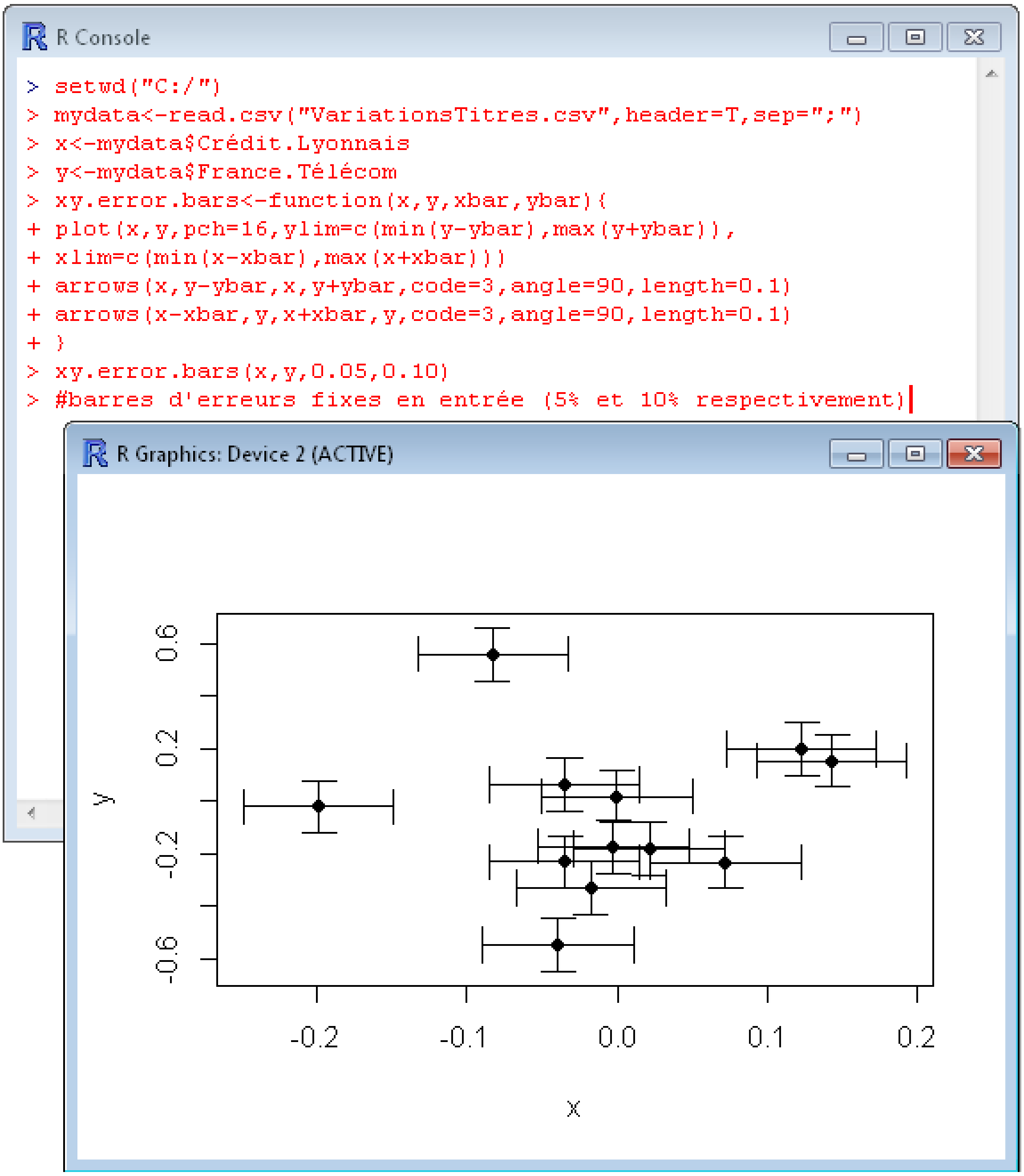


Figure 76 Graph X-Y avec marges d'erreurs

Libre à vous ensuite de relier les points par des lignes comme nous l'avons déjà vu plus haut.

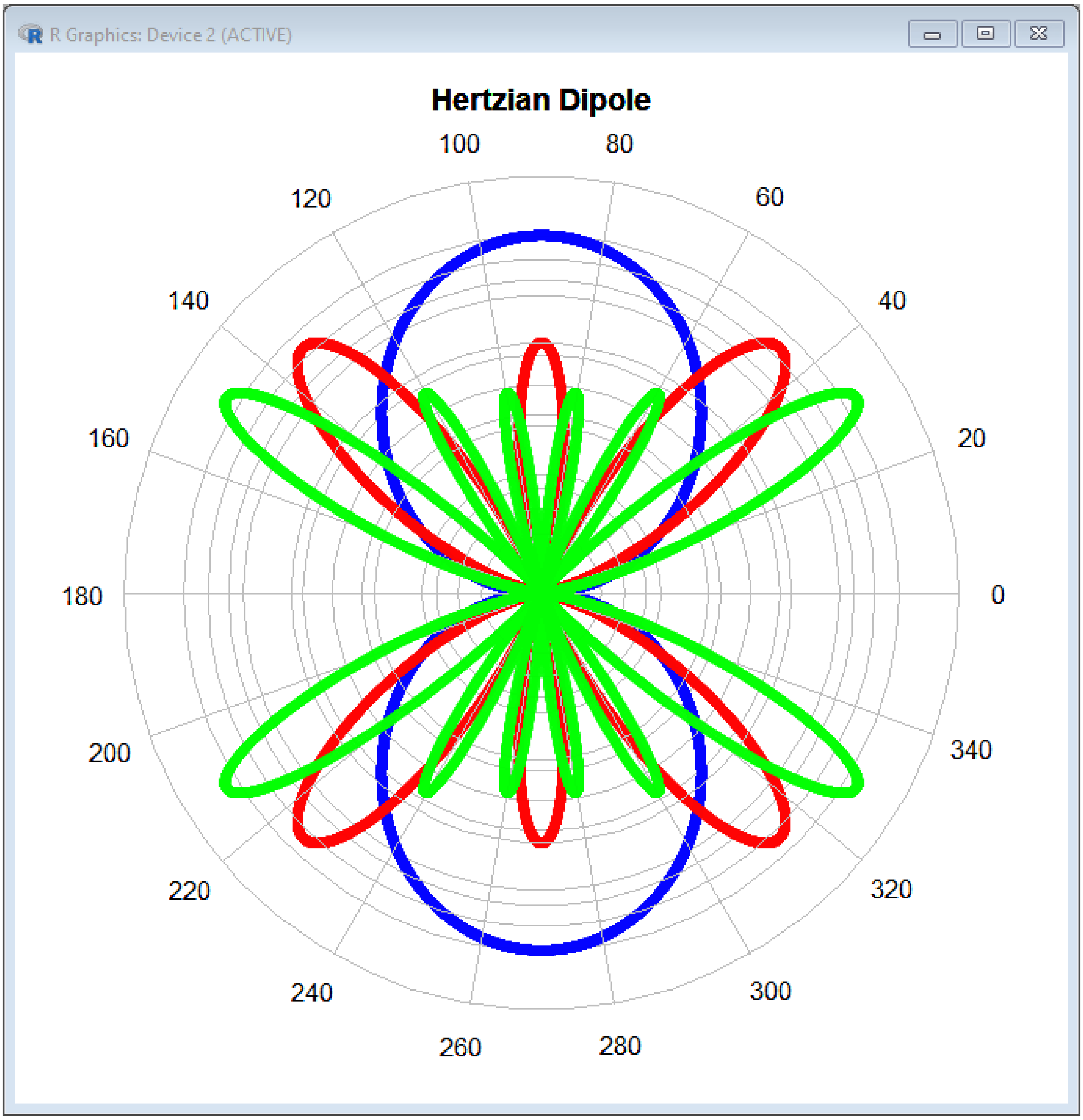
Graphiques en coordonnées polaires

Avec le package **plotrix** et sa fonction **polar.plot( )**, nous pouvons sans problèmes, reproduire le diagramme de radiation d'un dipôle de Hertz en régime harmonique pour l'approximation en champs lointains. Ce qui donne:

Nous avons alors en reproduisant à l'identique l'exemple fait dans le cours MATLAB:

```
> library("plotrix")
> n<-377
> I0<-1
> r<-10
> lambda<-0.3
> k<-(2*pi)/lambda
> L<-lambda/2
> theta<-seq(0.01,360,by=0.01)
> thetaRad<-(theta/360)*2*pi
> E<- -complex(0,imaginary=1)*n*I0*exp(-complex(0,imaginary=1)*k*r)*(1/(2*pi*r))*
+ ((cos(k*L*cos(thetaRad)/2)-cos(k*L/2))/sin(thetaRad))/r
> polar.plot(abs(E),theta,main="Hertzian Dipole",show.grid.labels=0,
+ lwd=3,rp.type="s",point.col=rgb(0,0,1,0.1),point.symbols=19)
>
> par(new=TRUE)
>
> L<-lambda*3
> E<- -complex(0,imaginary=1)*n*I0*exp(-complex(0,imaginary=1)*k*r)*(1/(2*pi*r))*
+ ((cos(k*L*cos(thetaRad)/2)-cos(k*L/2))/sin(thetaRad))/r
> polar.plot(abs(E),theta,main="Hertzian Dipole",show.grid.labels=0,
+ lwd=3,rp.type="s",point.col=rgb(1,0,0,0.1),point.symbols=19)
>
> par(new=TRUE)
>
> L<-lambda*6
> E<- -complex(0,imaginary=1)*n*I0*exp(-complex(0,imaginary=1)*k*r)*(1/(2*pi*r))*
+ ((cos(k*L*cos(thetaRad)/2)-cos(k*L/2))/sin(thetaRad))/r
> polar.plot(abs(E),theta,main="Hertzian Dipole",show.grid.labels=0,
+ lwd=3,rp.type="s",point.col=rgb(0,1,0,0.1),point.symbols=19)
>
```

Ce qui donne:



Un autre graphique polaire bien connu en science (optique) est celui utilisé comme diagnostique avec les opérations LASER (LASIK) pour l'œil: les topographies cornéennes!

Voyons comment reproduire un tel graphique!

Le code est relativement long donc nous n'allons pas faire de capture d'écran mais juste le fournir ici en texte brut avec le résultat final qui lui sera une capture d'écran par contre:

```
#Code crée par Tim Riffe
#Source: https://gist.github.com/timriffe/2893780

PolarImageInterpolate <- function(x, y, z, outer.radius = 1,
            breaks, contours = TRUE, legend = TRUE,
            axes = TRUE){

        #Quelques paramètres globaux
        NumberPoints <- 1000
        circle.rads = pretty(c(0,outer.radius))
```

```
        couleur.cercles.concentriques = "#66666650"

        minitics <- seq(-outer.radius, outer.radius, length.out =
NumberPoints )
        # interpolation des données (c'est subtil)
        Interp <- akima::interp(x = x, y = y, z = z,
                extrap = TRUE,
                xo = minitics,
                yo = minitics,
                linear = FALSE)
        Mat <- Interp[[3]]

        # marquer les cellules à l'extérieur du cercle comme NA
        markNA <- matrix(minitics, ncol = NumberPoints , nrow =
NumberPoints )
        Mat[!sqrt(markNA ^ 2 + t(markNA) ^ 2) < outer.radius] <- NA

        if (!missing(breaks)){
            col <- rev(heat.colors(length(breaks) - 1))
            nlevels <- length(breaks) - 1
        }

        # si une légende est souhaitée, elle se place à droite et nécessite
un peu d'espace
        if (legend) {
            par(mai = c(1,1,1.5,1.5))
        }

        # on commence le plot
        image(x = minitics, y = minitics, t(Mat), useRaster = TRUE, asp =
1,
            axes = FALSE, xlab = "", ylab = "", col = col, breaks = breaks)

        # ajout des courbes de niveau si souhaités
        if (contours){
            CL <- contourLines(x = minitics, y = minitics, t(Mat), levels =
breaks)
            A <- lapply(CL, function(xy){
                        lines(xy$x, xy$y, col = gray(.2), lwd = .5)
                    })
        }

        # ajouter des axes radiaux si vous le souhaitez
        if (axes){
            # internals for axis markup
            RMat <- function(radians){
                matrix(c(cos(radians), sin(radians), -sin(radians),
cos(radians)), ncol = 2)
            }

            circle <- function(x, y, rad = 1, nvert = 500){
                rads <- seq(0,2*pi,length.out = nvert)
                xcoords <- cos(rads) * rad + x
                ycoords <- sin(rads) * rad + y
                cbind(xcoords, ycoords)
            }

            for (i in pretty(c(0,outer.radius))){
                lines(circle(0, 0, i), col = couleur.cercles.concentriques)
            }
```

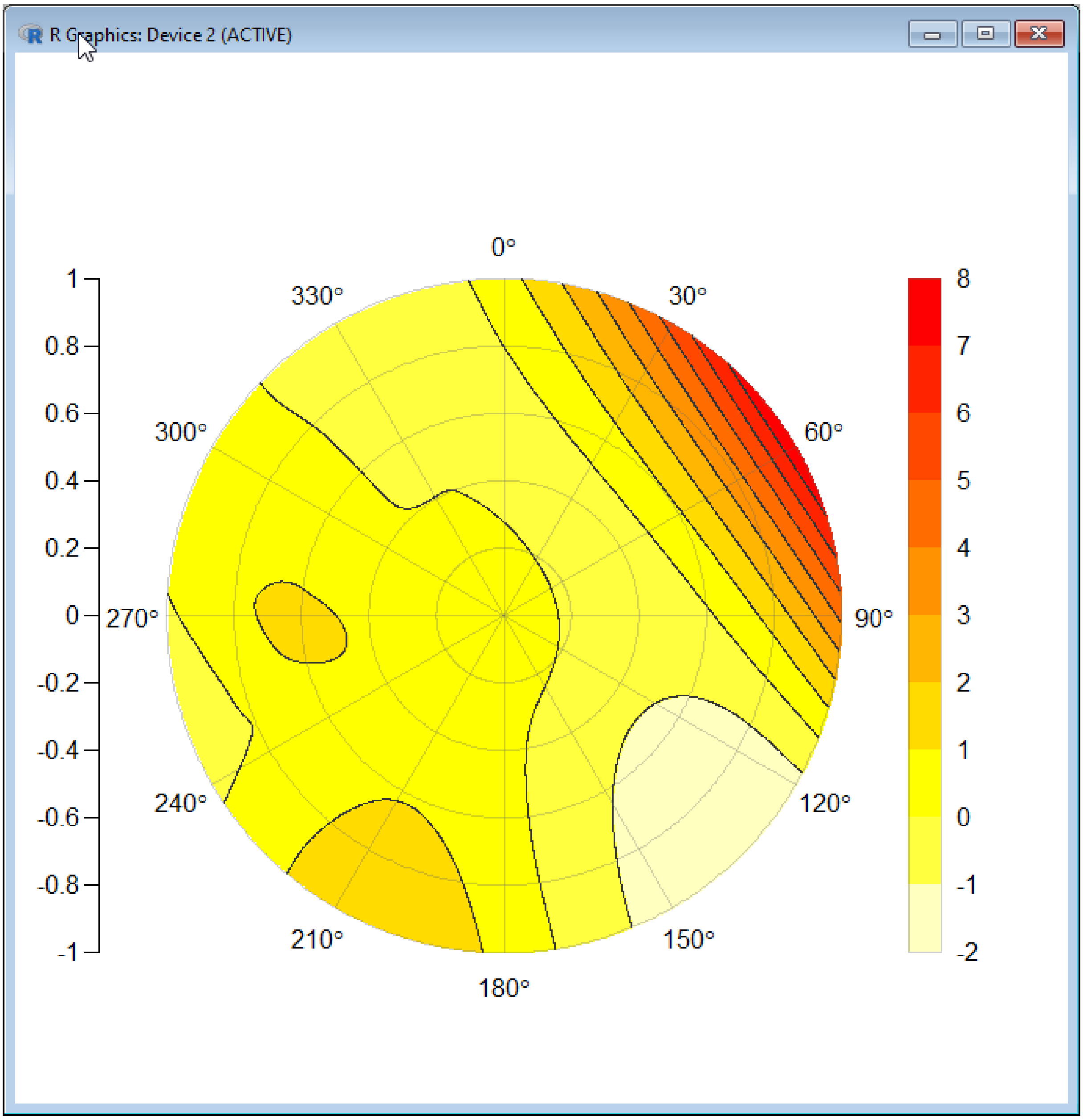
```
            # préparation des vecteurs pour les dessin des axes à rayons
radiaux
            axis.rads <- c(0, pi / 6, pi / 3, pi / 2, 2 * pi / 3, 5 * pi /
6)
            r.labs <- c(90, 60, 30, 0, 330, 300)
            l.labs <- c(270, 240, 210, 180, 150, 120)

            # maintenant on dessine les axies radiaux et on pose leurs
étiquettes
            for (i in 1:length(axis.rads)){
                endpoints <- zapsmall(c(RMat(axis.rads[i]) %*% matrix(c(1,
0, -1, 0) * outer.radius,ncol = 2)))
                segments(endpoints[1], endpoints[2], endpoints[3],
endpoints[4], col = couleur.cercles.concentriques)
                endpoints <- c(RMat(axis.rads[i]) %*% matrix(c(1.1, 0, -
1.1, 0) * outer.radius, ncol = 2))
                lab1 <- bquote(.(r.labs[i]) * degree)
                lab2 <- bquote(.(l.labs[i]) * degree)
                text(endpoints[1], endpoints[2], lab1, xpd = TRUE)
                text(endpoints[3], endpoints[4], lab2, xpd = TRUE)
            }
            #dessin de l'axe vertical gauche
            axis(2, pos = -1.2 * outer.radius, at =
sort(union(circle.rads,-circle.rads)), labels = NA)
            text( -1.21 * outer.radius, sort(union(circle.rads, -
circle.rads)),sort(union(circle.rads, -circle.rads)), xpd = TRUE, pos = 2)
        }

        # dessin de la barre de légende de droite avec le dégradé de
couleurs
        if (legend){
            ylevs <- seq(-outer.radius, outer.radius, length = nlevels + 1)
            rect(1.2 * outer.radius, ylevs[1:(length(ylevs) - 1)], 1.3 *
outer.radius, ylevs[2:length(ylevs)], col = col, border = NA, xpd = TRUE)
            rect(1.2 * outer.radius, min(ylevs), 1.3 * outer.radius,
max(ylevs), border = couleur.cercles.concentriques, xpd = TRUE)
            text(1.3 * outer.radius, ylevs,round(breaks, 1), pos = 4, xpd =
TRUE)
        }
    }

    # Example
    set.seed(10)
    x <- rnorm(20)
    y <- rnorm(20)
    z <- rnorm(20)
    PolarImageInterpolate(x,y,z, breaks = seq(-2,8,by = 1))
```


Ce qui donne:


R Graphics: Device 2 (ACTIVE)
0°
30°
60°
90°
120°
150°
180°
210°
240°
270°
300°
330°
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
8
7
6
5
4
3
2
1
0
-1
-2

Graphiques à pentes (interactions)

C'est actuellement assez laborieux à faire comme graphique mais voici un exemple:

```
> library(ggplot2)
> library(scales)
> theme_set(theme_classic())
>
> # prep data
> df <- read.csv("https://raw.githubusercontent.com/selva86/datasets/master/gdppercap.csv")
> colnames(df) <- c("continent", "1952", "1957")
> left_label <- paste(df$continent, round(df$`1952`),sep=", ")
> right_label <- paste(df$continent, round(df$`1957`),sep=", ")
> df$class <- ifelse((df$`1957` - df$`1952`) < 0, "red", "green")
>
> # Plot
> p <- ggplot(df) + geom_segment(aes(x=1, xend=2, y=`1952`, yend=`1957`, col=class), size=.75, show.legend=F) +
+                   geom_vline(xintercept=1, linetype="dashed", size=.1) +
+                   geom_vline(xintercept=2, linetype="dashed", size=.1) +
+                   scale_color_manual(labels = c("Up", "Down"),
+                                      values = c("green"="#00ba38", "red"="#f8766d")) +  # color of lines
+                   labs(x="", y="Mean GdpPerCap") +  # Axis labels
+                   xlim(.5, 2.5) + ylim(0,(1.1*(max(df$`1952`, df$`1957`))))  # X and Y axis limits
>
> # Add texts
> p <- p + geom_text(label=left_label, y=df$`1952`, x=rep(1, NROW(df)), hjust=1.1, size=3.5)
> p <- p + geom_text(label=right_label, y=df$`1957`, x=rep(2, NROW(df)), hjust=-0.1, size=3.5)
> p <- p + geom_text(label="Time 1", x=1, y=1.1*(max(df$`1952`, df$`1957`)), hjust=1.2, size=5)  # title
> p <- p + geom_text(label="Time 2", x=2, y=1.1*(max(df$`1952`, df$`1957`)), hjust=-0.1, size=5)  # title
>
> # Minify theme
> p + theme(panel.background = element_blank(),
+           panel.grid = element_blank(),
+           axis.ticks = element_blank(),
+           axis.text.x = element_blank(),
+           panel.border = element_blank(),
+           plot.margin = unit(c(1,2,1,2), "cm"))
>
```

Ce qui donne:

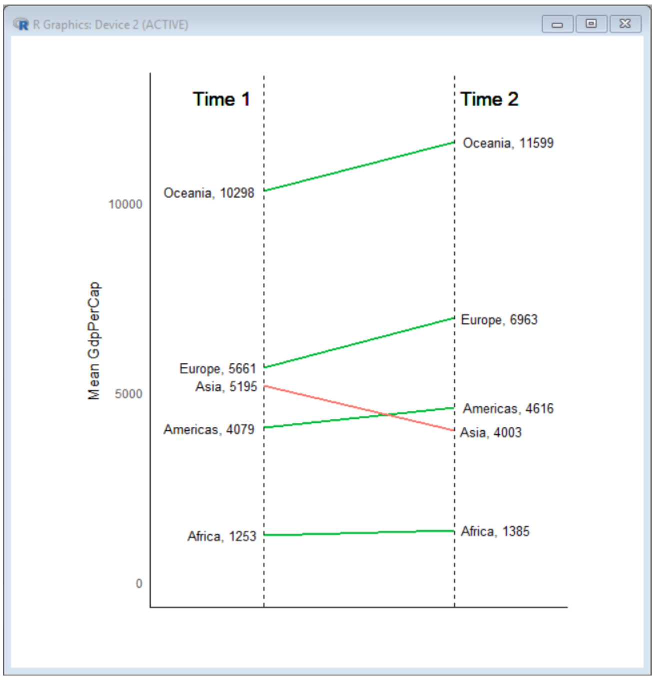


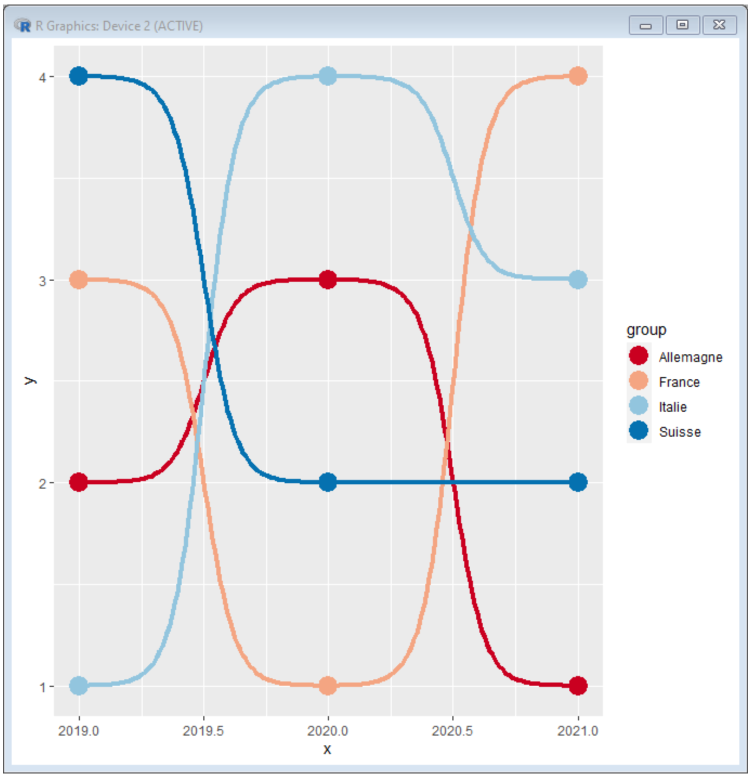
Figure 77 Graph à pentes et interactions temporelles et catégorielles (ggplot)

Graphiques Bump

Voyons donc comment à l'aide du package **ggbump** faire un tel graphique!

```
> library("tidyverse")
> library("ggbump")
>
> year <- rep(2019:2021, 4)
> position <- c(4, 2, 2, 3, 1, 4, 2, 3, 1, 1, 4, 3)
> player <- c("Suisse", "Suisse", "Suisse",
+             "France", "France", "France",
+             "Allemagne", "Allemagne", "Allemagne",
+             "Italie", "Italie", "Italie")
>
> (df <- data.frame(x = year,
+                   y = position,
+                   group = player))
      x y     group
1  2019 4    Suisse
2  2020 2    Suisse
3  2021 2    Suisse
4  2019 3    France
5  2020 1    France
6  2021 4    France
7  2019 2 Allemagne
8  2020 3 Allemagne
9  2021 1 Allemagne
10 2019 1    Italie
11 2020 4    Italie
12 2021 3    Italie
>
> ggplot(df, aes(x = x, y = y, color = group)) +
+   geom_bump(size = 1.5) +
+   geom_point(size = 6)+
+   scale_color_brewer(palette = "RdBu")
>
```

Ce qui donne:

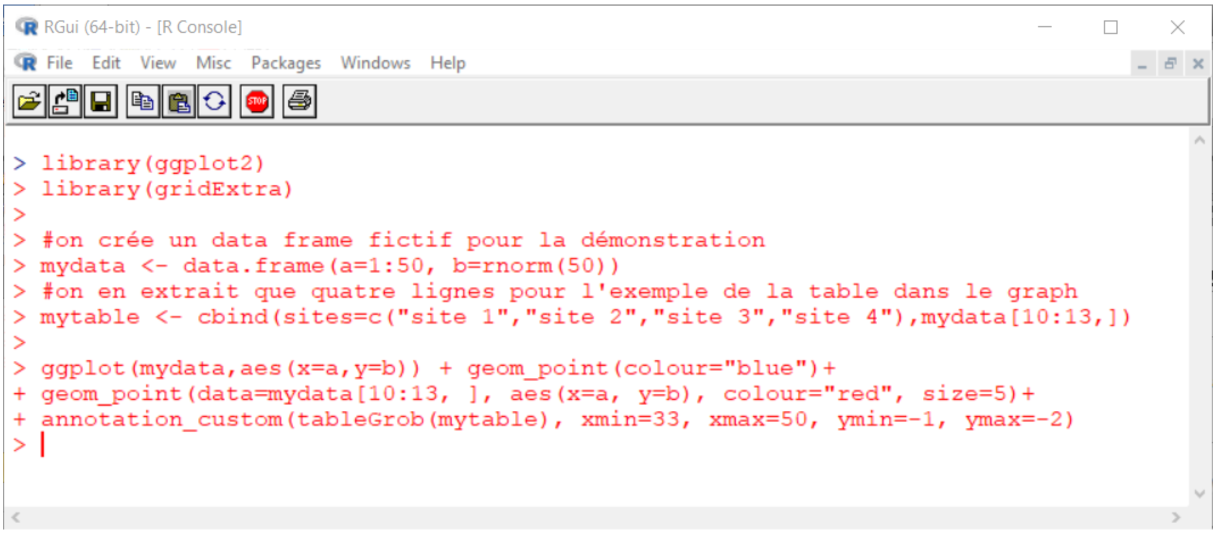


Exemple 117.: Diagrammes à points

Diagramme à points (ou autre) avec table

R 3.5

Voici un traitement rare mais pas inexistant qui consiste à mettre une table dans un graphique en utilisant le package **ggplot2** et **gridExtra**:



```
> library(ggplot2)
> library(gridExtra)
>
> #on crée un data frame fictif pour la démonstration
> mydata <- data.frame(a=1:50, b=rnorm(50))
> #on en extrait que quatre lignes pour l'exemple de la table dans le graph
> mytable <- cbind(sites=c("site 1","site 2","site 3","site 4"),mydata[10:13,])
>
> ggplot(mydata,aes(x=a,y=b)) + geom_point(colour="blue")+
+ geom_point(data=mydata[10:13, ], aes(x=a, y=b), colour="red", size=5)+
+ annotation_custom(tableGrob(mytable), xmin=33, xmax=50, ymin=-1, ymax=-2)
>
```

Comme on peut le deviner à la lecture du script ci-dessus c'est **tableGrob( )** avec **annotation_custom( )** qui vont faire le travail dont voici le résultat (**attention!!! vu que les valeurs sont générées aléatoirement, le résultat sur votre écran peut changer significativement de celui visible ci-dessous!**):

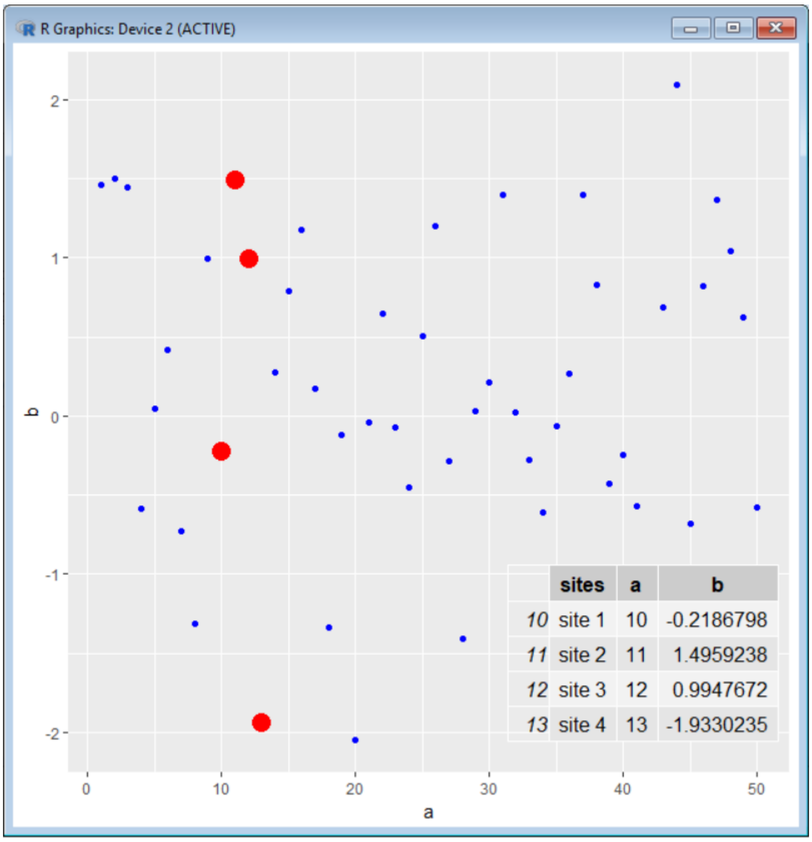
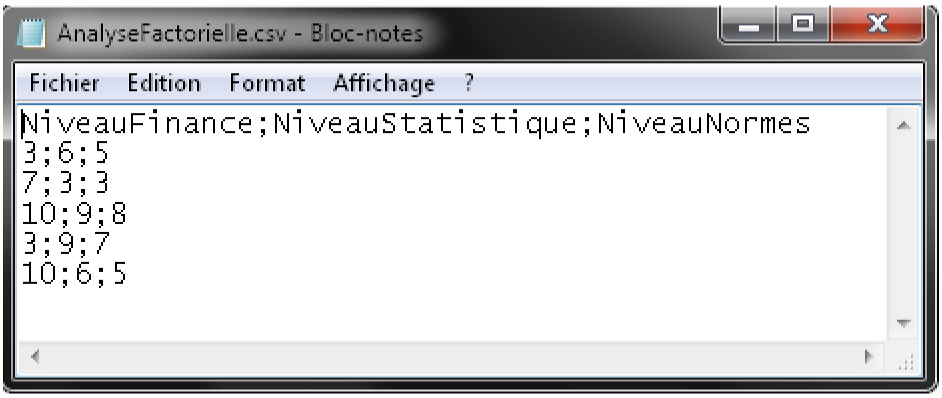


Figure 78 Graph X-Y avec table en coin

Diagrammes à points catégoriels (diagramme de Cleveland)

R 3.0.2

Nous allons voir ce type de diagramme juste histoire de le faire car dans la pratique il est très peu utilisé. Pour cela, nous allons utiliser les données utilisées lors de notre étude théorique de l'analyse factorielles (mêmes données que celles utilisées dans le support de cours Minitab), soit:



et nous obtenons en utilisant la commande **dotchart( )** :

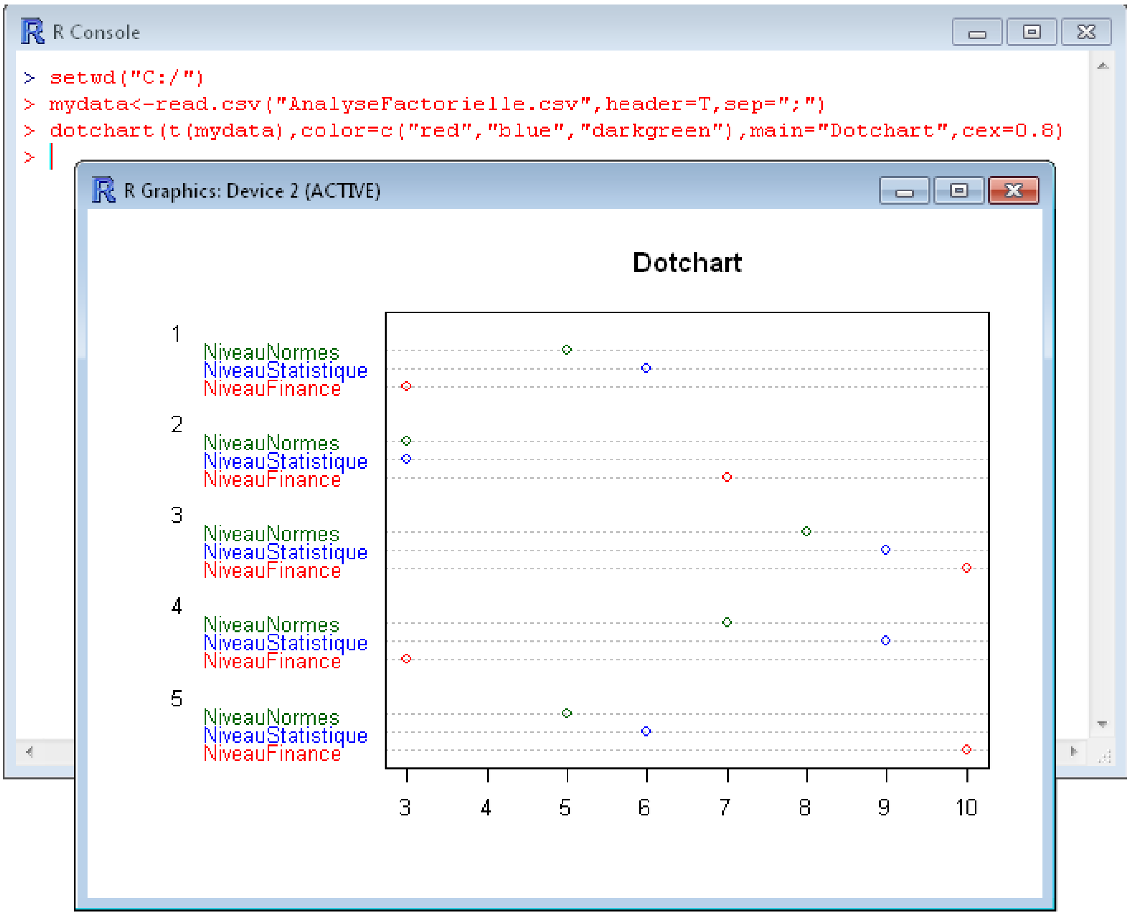


Figure 79 Graph de Cleveland (comptage horizontal variables catégorielles)

Diagrammes à points (dot plots)

R 3.0.2

Bon ici il s'agit simplement de voir que l'on peut obtenir le même type de diagrammes à points que celui que nous avions fait dans le cours Minitab (mtb) en utilisant le package **plotrix** et la commande **dotplot.mtb( )** :

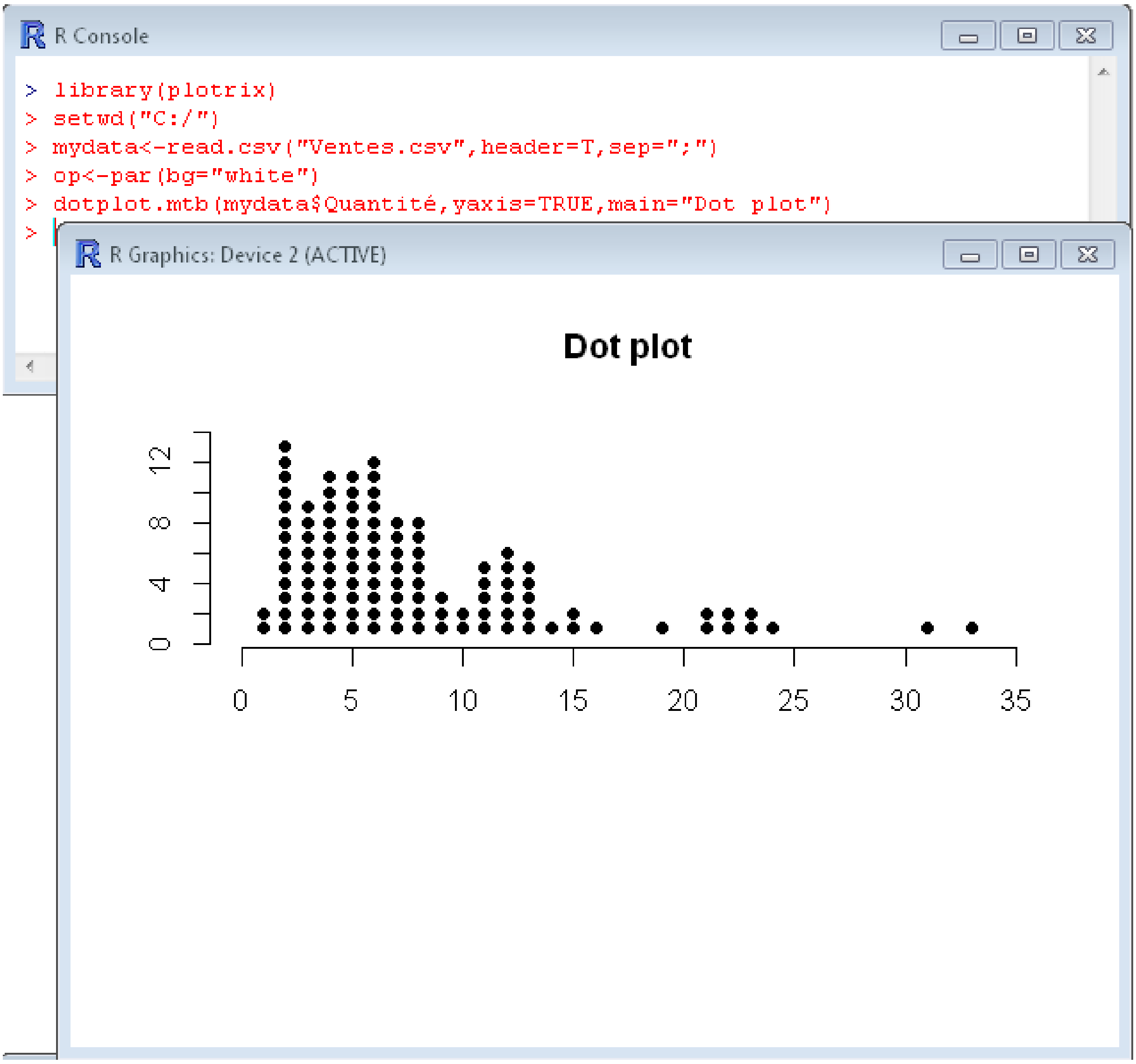


Figure 80 Graph dot-plot

Donc nous voyons que comme dans Minitab... l'esthétique n'est pas des meilleures...

Sinon un autre grand classique dans les dot plot en se basant sur le fichier suivant:



Pièces;Mesure;Opérateurs
Pièce 1;9;Opérateur 1
Pièce 2;6;Opérateur 1
Pièce 3;8;Opérateur 1
Pièce 4;7;Opérateur 1
Pièce 5;10;Opérateur 1
Pièce 6;6;Opérateur 1
Pièce 1;2;Opérateur 2
Pièce 2;1;Opérateur 2
Pièce 3;4;Opérateur 2
Pièce 4;1;Opérateur 2
Pièce 5;5;Opérateur 2
Pièce 6;2;Opérateur 2
Pièce 1;5;Opérateur 3
Pièce 2;3;Opérateur 3
Pièce 3;6;Opérateur 3
Pièce 4;2;Opérateur 3
Pièce 5;6;Opérateur 3
Pièce 6;4;Opérateur 3
Pièce 1;8;Opérateur 4
Pièce 2;2;Opérateur 4
Pièce 3;8;Opérateur 4
Pièce 4;6;Opérateur 4
Pièce 5;9;Opérateur 4
Pièce 6;7;Opérateur 4

avec la fonction **dotplot( )** du package **lattice**:

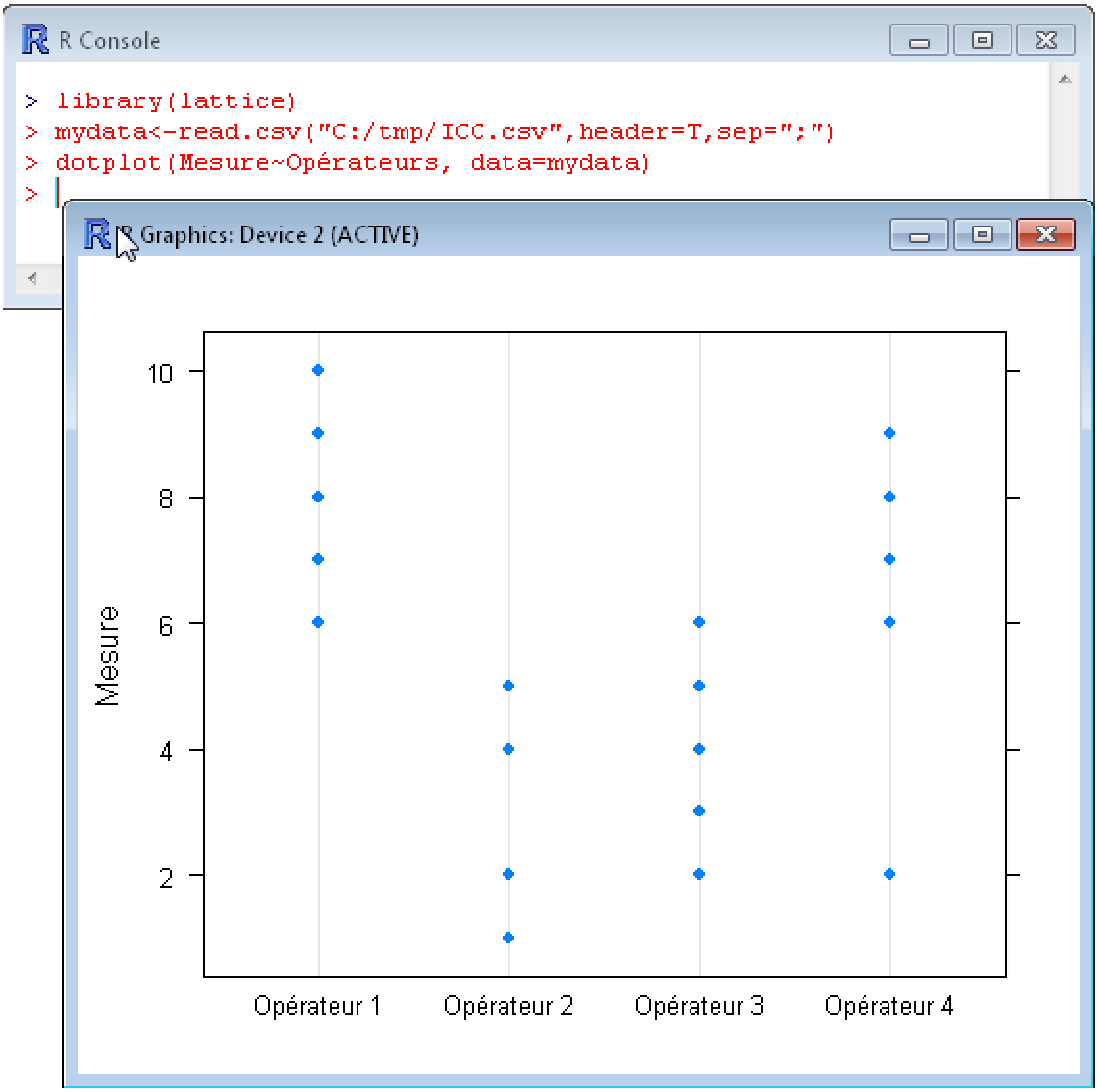


Figure 81 Graph de Cleveland vertical

Diagramme tournesols (analyses de densité 2D)

R 3.0.2

L'idée du diagramme tournesol est simple: à chaque fois qu'un point se superpose à un autre de coordonnées identiques, un pétale sous forme de trait est rajouté et le centre de ses "pétales" successives sont les points superposés eux-mêmes.

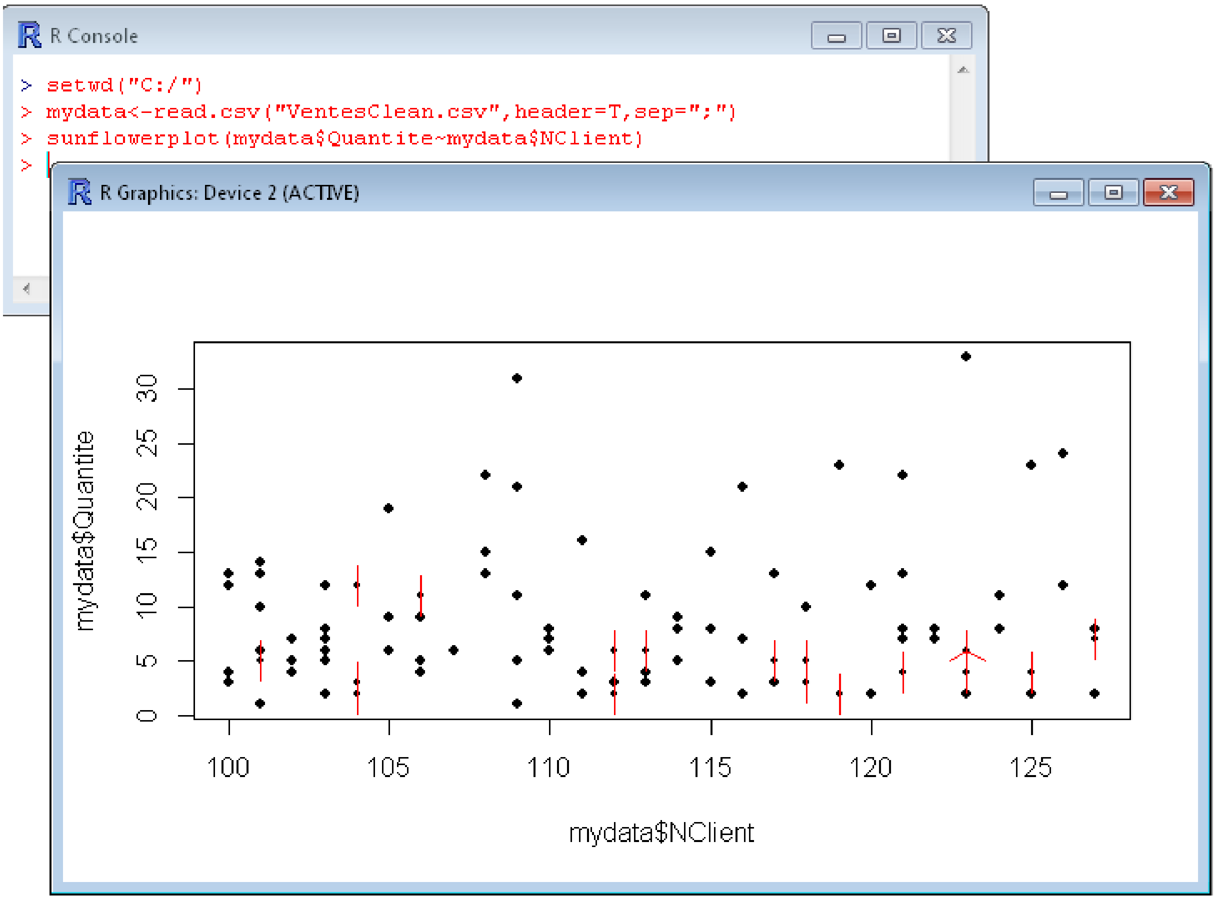


Figure 82 Graph X-Y Tournesols

Ou en bruitant les données avec la commande **jitter( )** :

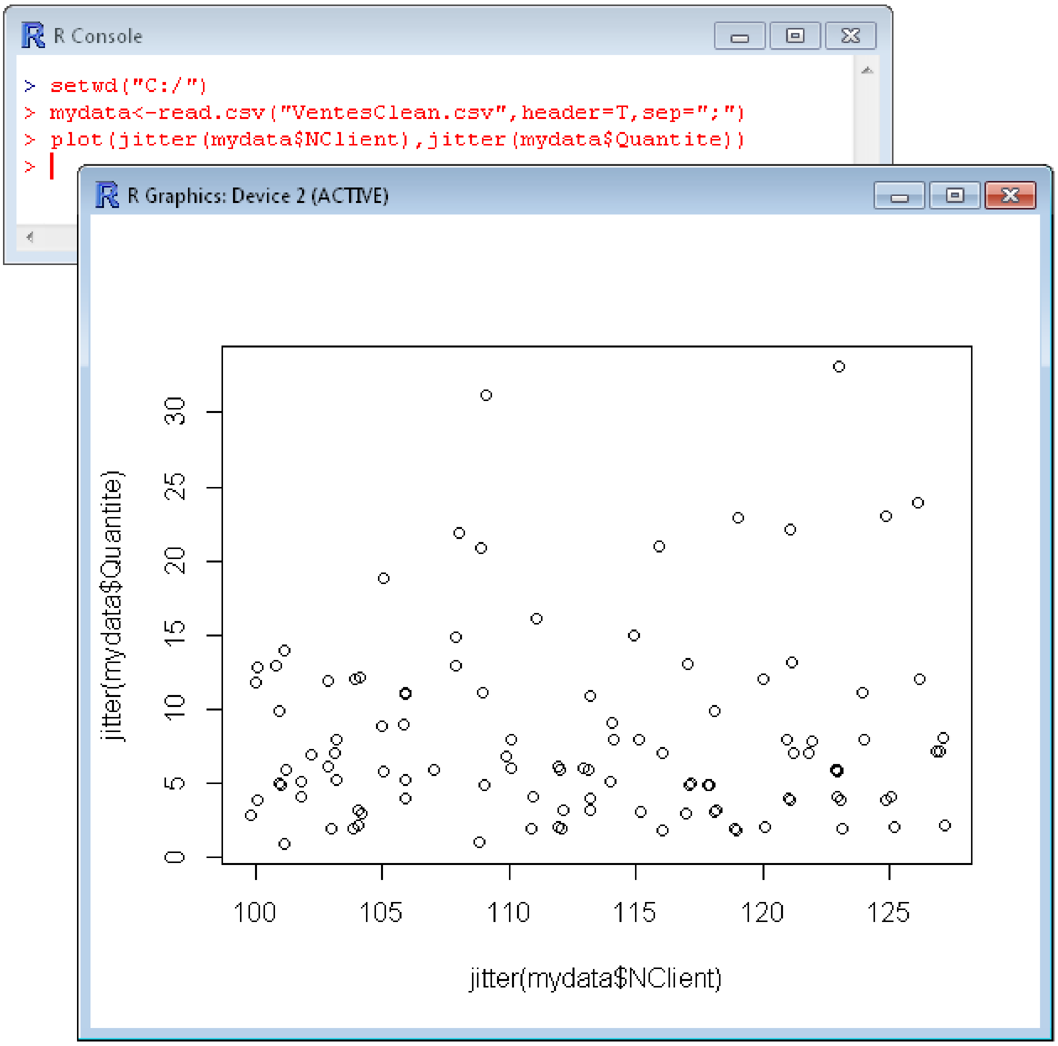


Figure 83 Graph X-Y jitter

Ou en utilisant un estimateur local de densité associant la commande **image( )** et la commande **kde2d( )** :

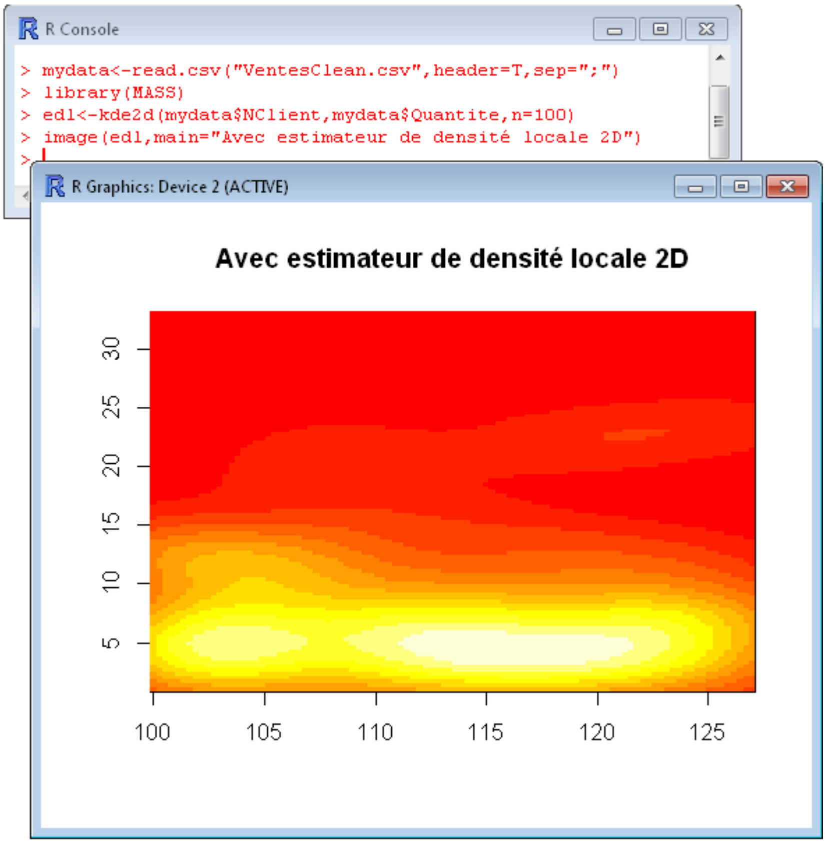
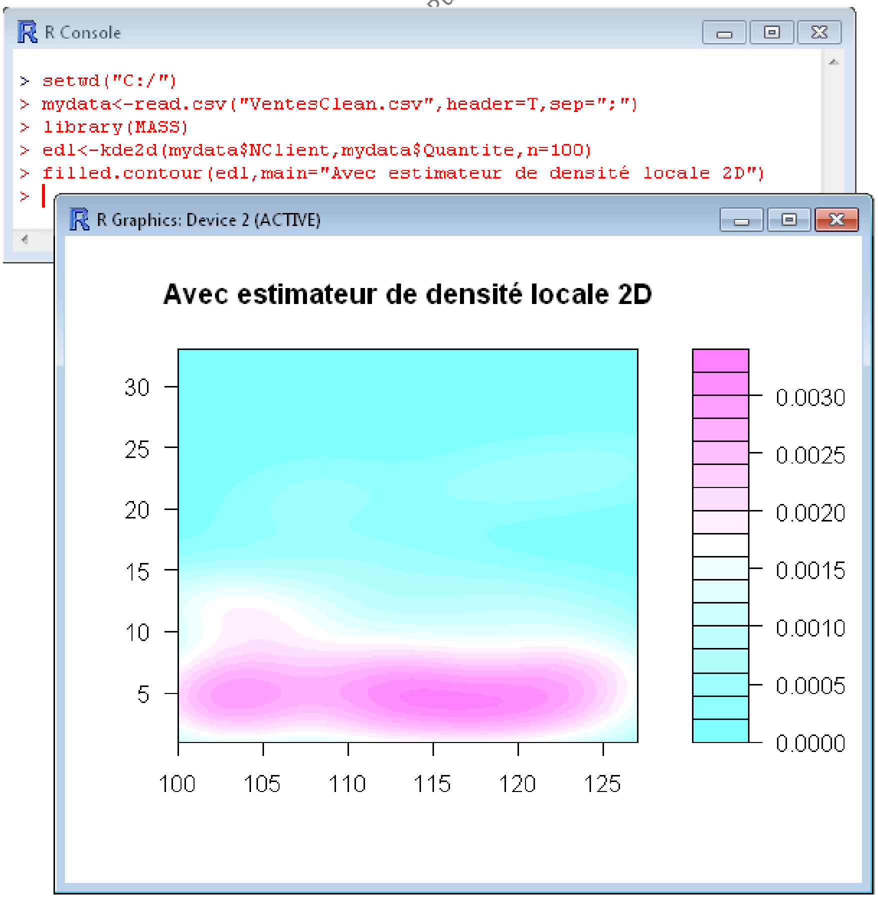
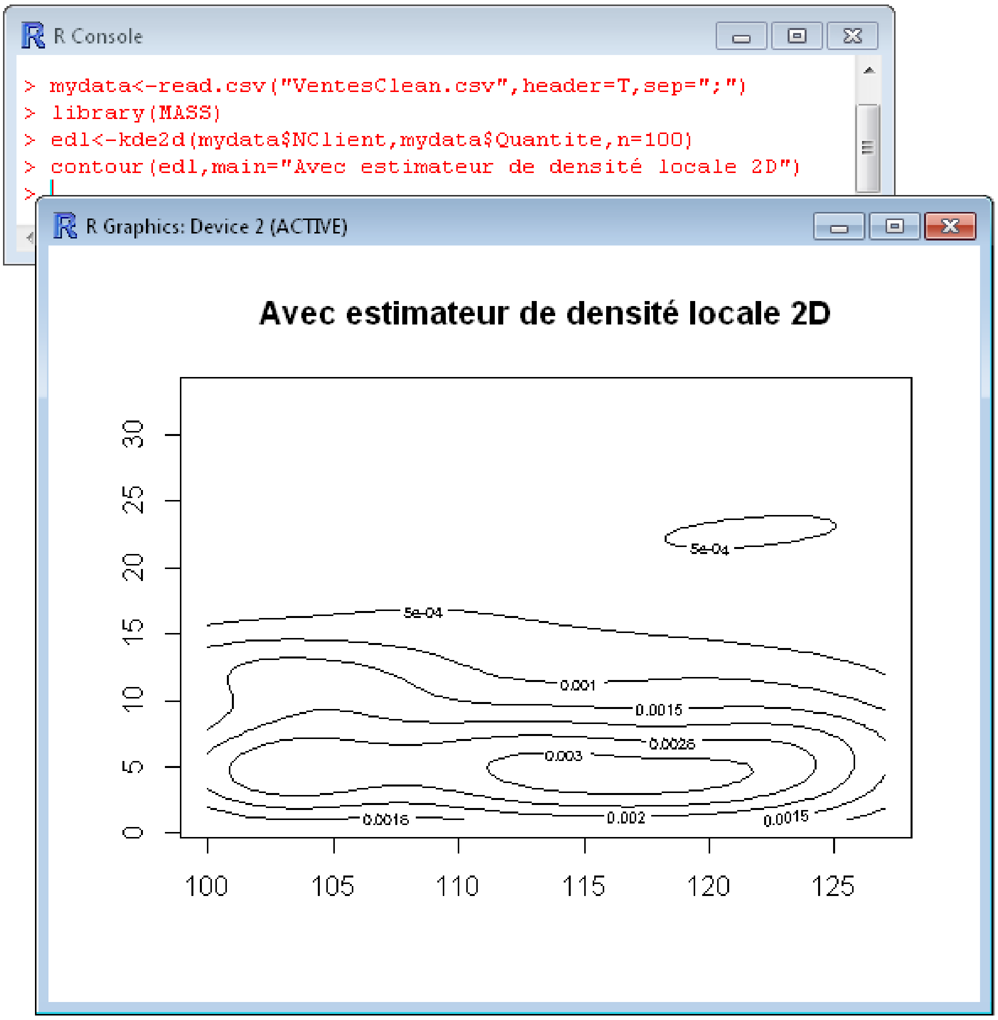


Figure 84 Graph X-Y avec carte de densité

Ou encore:



Et en associant avec la commande **contour( )** :



Ou nous pouvons passer à une vue 3D:

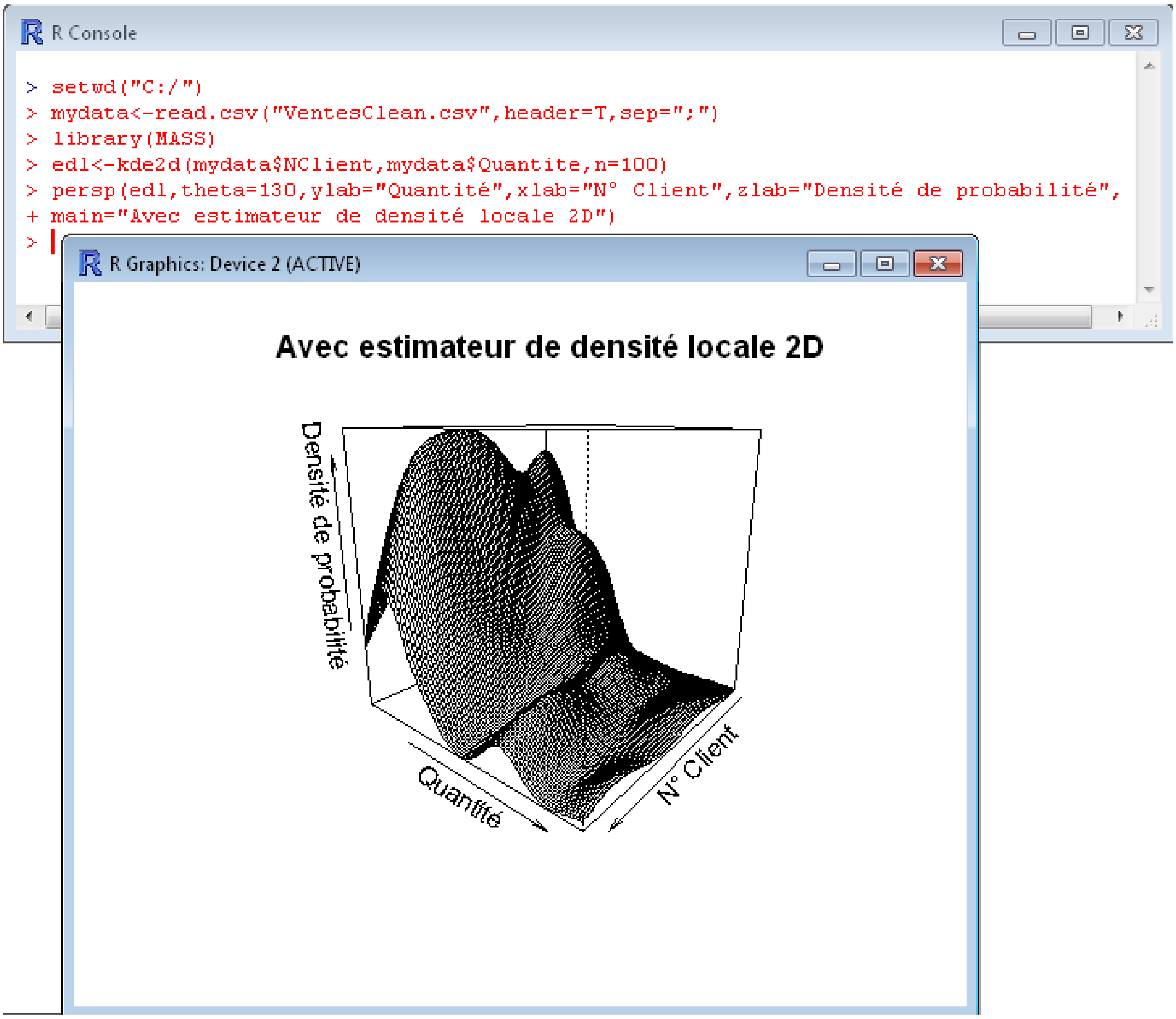


Figure 85 Graph X-Y avec carte de densité 3D filaire

Évidemment comme toujours avec tout diagramme... l'interprétation peut être délicate (raison pour laquelle je me rabats toujours si possible sur des indicateurs empiriques et acceptés par la communauté scientifique comme étant robustes).

Co-diagramme

R 3.0.2

La commande **coplot( )** , pour plot conditionnel, génère en quelque sorte des projections de plot à 3 dimensions sur de multiples plans de 2 dimensions.

Par exemple, supposons que nous souhaitons comparer les factures payées et non payées en fonction des articles et de la quantité commandée.

Nous avons alors:

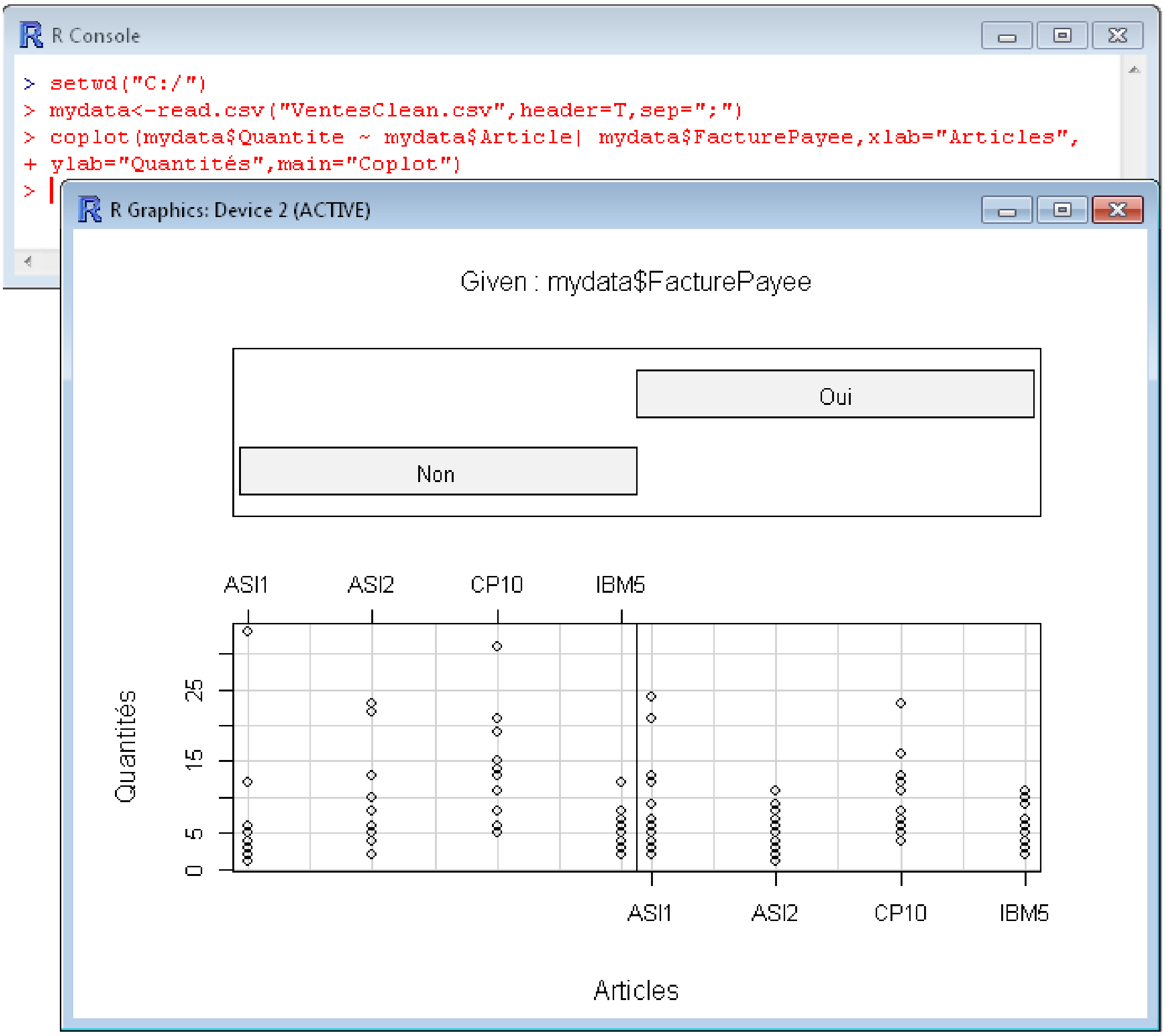
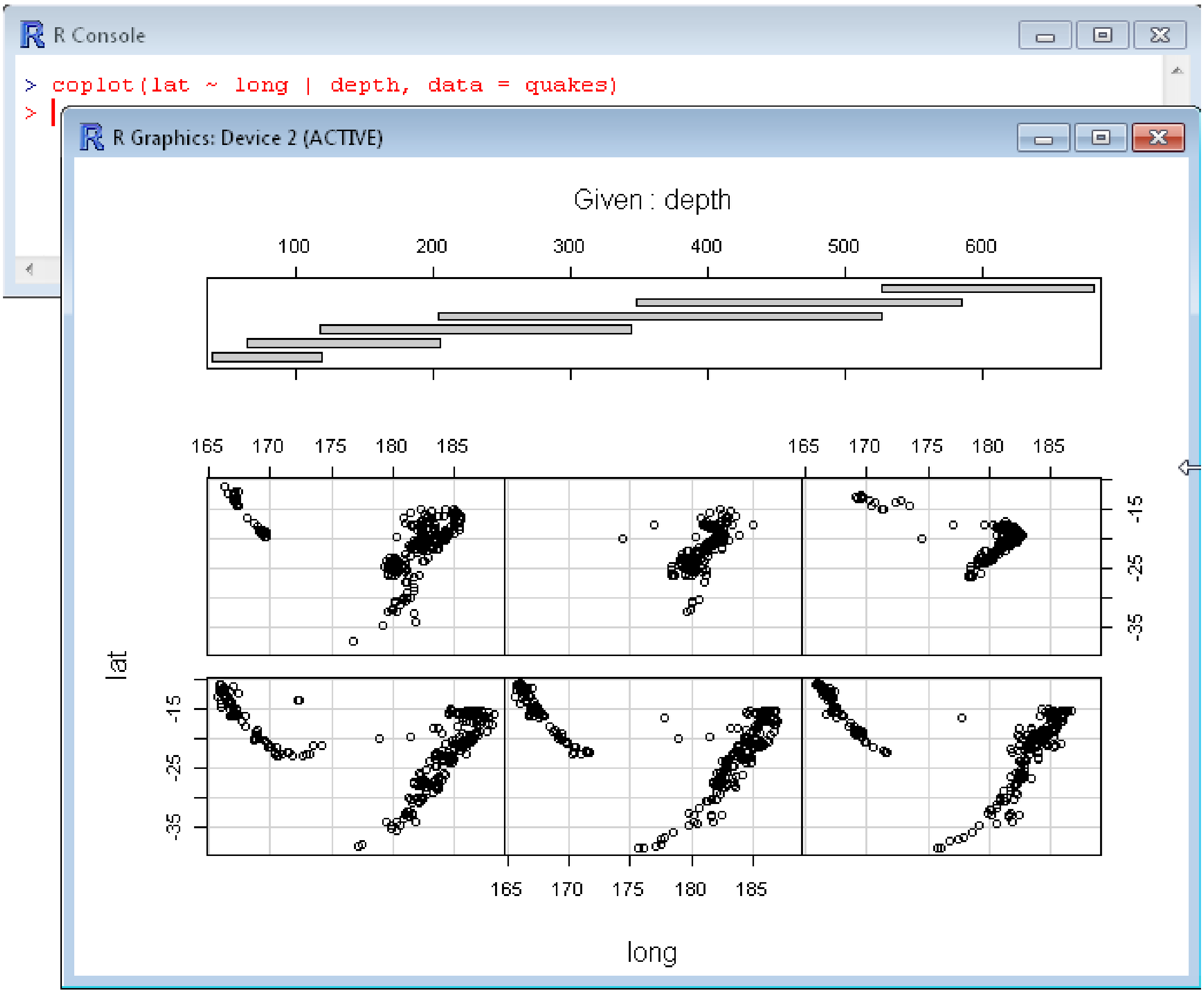


Figure 86 Graph de type co-diagramme

Nous pouvons donc conclure que les factures non payées ne sont pas distinctes pour un article donné ou une quantité donnée. Les bons ou mauvais payeurs se comportent donc vraiment de manière aléatoire et non systématique.

Vous pouvez jouer avec ce graphique en remplaçant les factures payées par une variable numérique ainsi que les noms des articles. C'est très efficace mais la lecture peut vite devenir compliquée.

Par exemple en s'amusant avec un jeu de données intégré à R, nous obtenons uniquement avec des valeurs numériques:

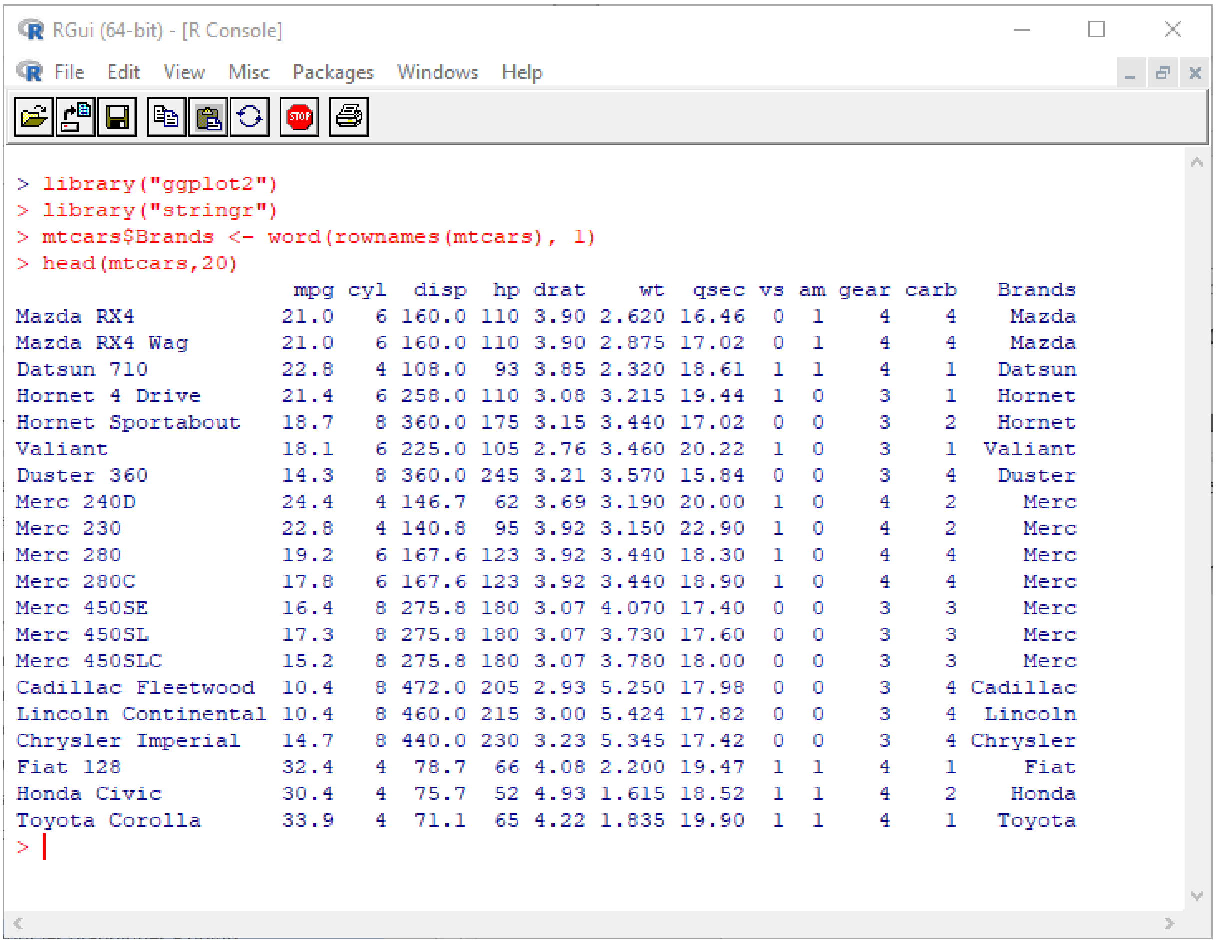


Symboles disponibles pour les graphiques à points

R 4.1.1

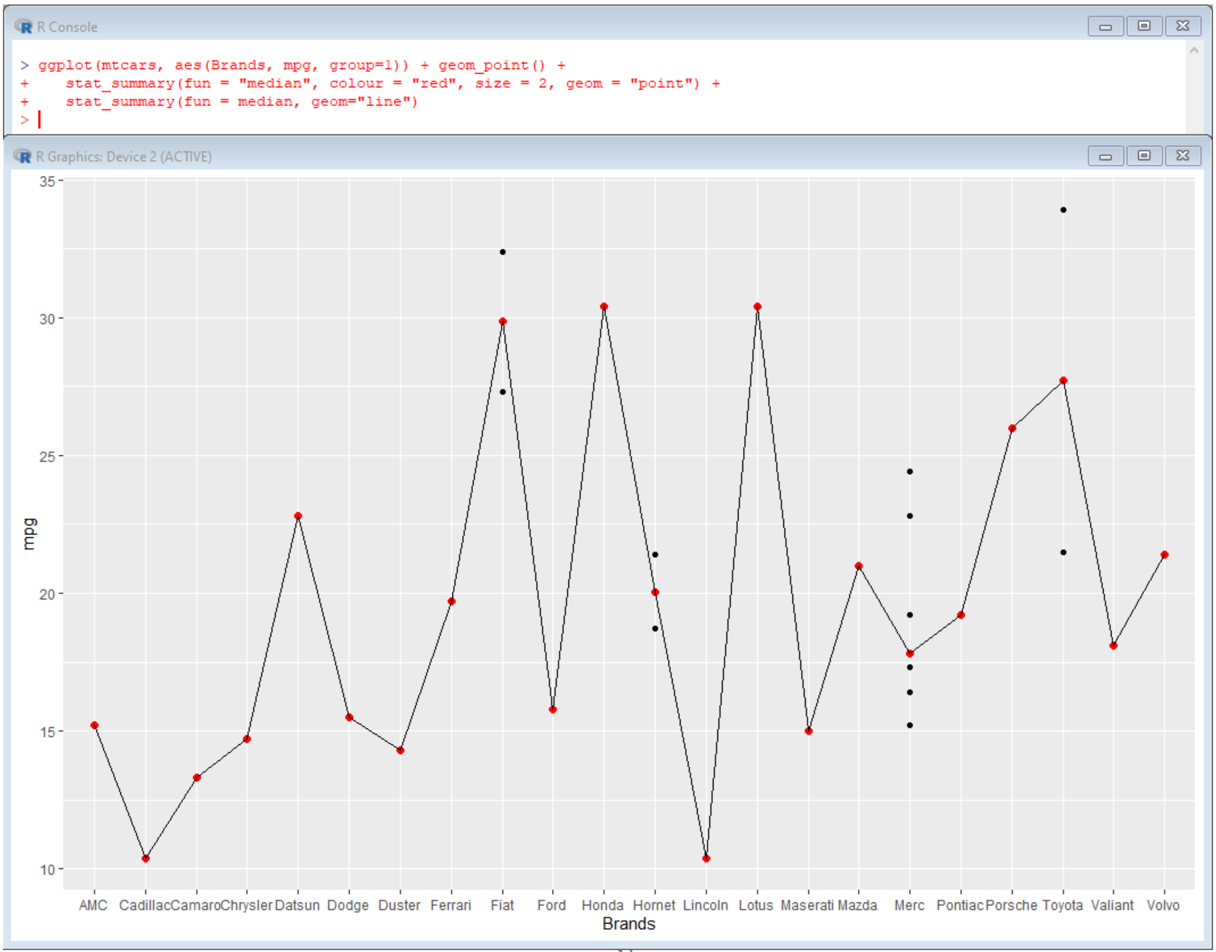
Voici un type de diagramme relativement souvent utilisé par les ingénieurs et pas évident à deviner au niveau du code (surtout à cause du group=1) qu'il faut mettre… pour avoir des catégories sur l'axe des abscisses.

D'abord on prépare un jeu de données pour l'exemple:



```
> library("ggplot2")
> library("stringr")
> mtcars$Brands <- word(rownames(mtcars), 1)
> head(mtcars,20)
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb   Brands
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4    Mazda
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4    Mazda
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1   Datsun
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1   Hornet
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2   Hornet
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1  Valiant
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4   Duster
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2     Merc
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2     Merc
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4     Merc
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4     Merc
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3     Merc
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3     Merc
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3     Merc
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 Cadillac
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4  Lincoln
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 Chrysler
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1     Fiat
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2    Honda
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1   Toyota
>
```

Et ensuite faisons le graphique:


R Console
> ggplot(mtcars, aes(Brands, mpg, group=1)) + geom_point() +
+   stat_summary(fun = "median", colour = "red", size = 2, geom = "point") +
+   stat_summary(fun = median, geom="line")
>
R Graphics: Device 2 (ACTIVE)
mpg
35
30
25
20
15
10
AMC Cadillac Camaro Chrysler Datsun Dodge Duster Ferrari Fiat Ford Honda Hornet Lincoln Lotus Maserati Mazda Merc Pontiac Porsche Toyota Valiant Volvo
Brands

Symboles disponibles pour les graphiques à points

R 3.0.2

Voici la liste des symboles disponibles pour les graphiques à points présentant le type de symbole *pch* et leur taille avec *cex*:

```
> # Make an empty chart
> plot(1,1, xlim=c(1,5.5), ylim=c(0,7), axes=FALSE, type="n", ann=FALSE)
>
> # Plot digits 0-4 with increasing size and color
> text(1:5, rep(6,5), labels=c(0:4), cex=1:5, col=1:5)
>
> # Plot symbols 0-4 with increasing size and color
> points(1:5, rep(5,5), cex=1:5, col=1:5, pch=0:4)
> text((1:5)+0.4, rep(5,5), cex=0.6, (0:4))
>
> # Plot symbols 5-9 with labels
> points(1:5, rep(4,5), cex=2, pch=(5:9))
> text((1:5)+0.4, rep(4,5), cex=0.6, (5:9))
>
> # Plot symbols 10-14 with labels
> points(1:5, rep(3,5), cex=2, pch=(10:14))
> text((1:5)+0.4, rep(3,5), cex=0.6, (10:14))
>
> # Plot symbols 15-19 with labels
> points(1:5, rep(2,5), cex=2, pch=(15:19))
> text((1:5)+0.4, rep(2,5), cex=0.6, (15:19))
>
> # Plot symbols 20-25 with labels
> points((1:6)*0.8+0.2, rep(1,6), cex=2, pch=(20:25))
> text((1:6)*0.8+0.5, rep(1,6), cex=0.6, (20:25))
```

ce qui donne:

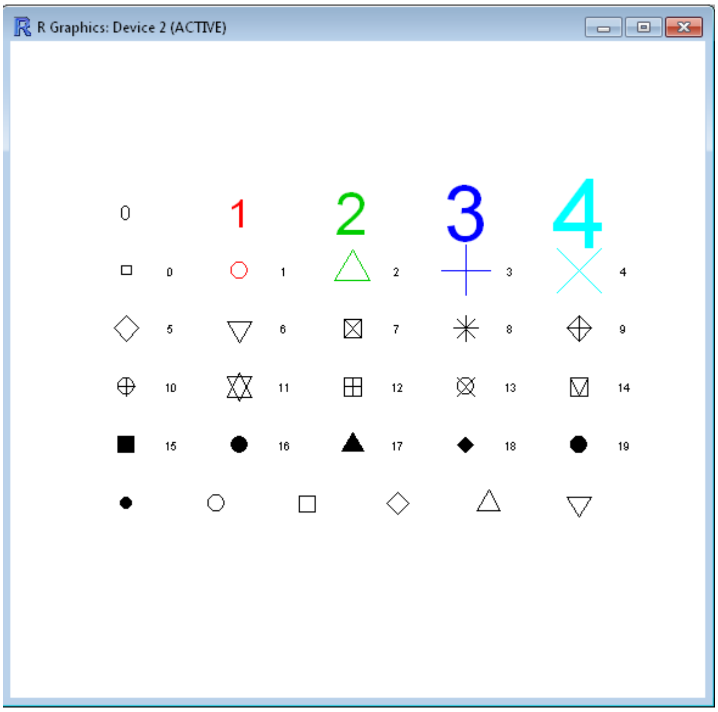


Figure 87 Types de symboles de points disponibles dans R

Exemple 118.: Diagramme d'associations

R 3.0.2

Il existe deux familles de diagrammes d'associations! Voyons donc chacune d'entre elle!

Diagramme d'association pour table de contingence

sont souvent utilisés pour les tables de contingence.

Voici un exemple:



Figure 88 Graph d'association

Il faut lire ce graphique dans l'optique d'un tableau de contingence avec la règle du "toute chose égale par ailleurs" qui nous sera donné par:

```
R Console

> mytable

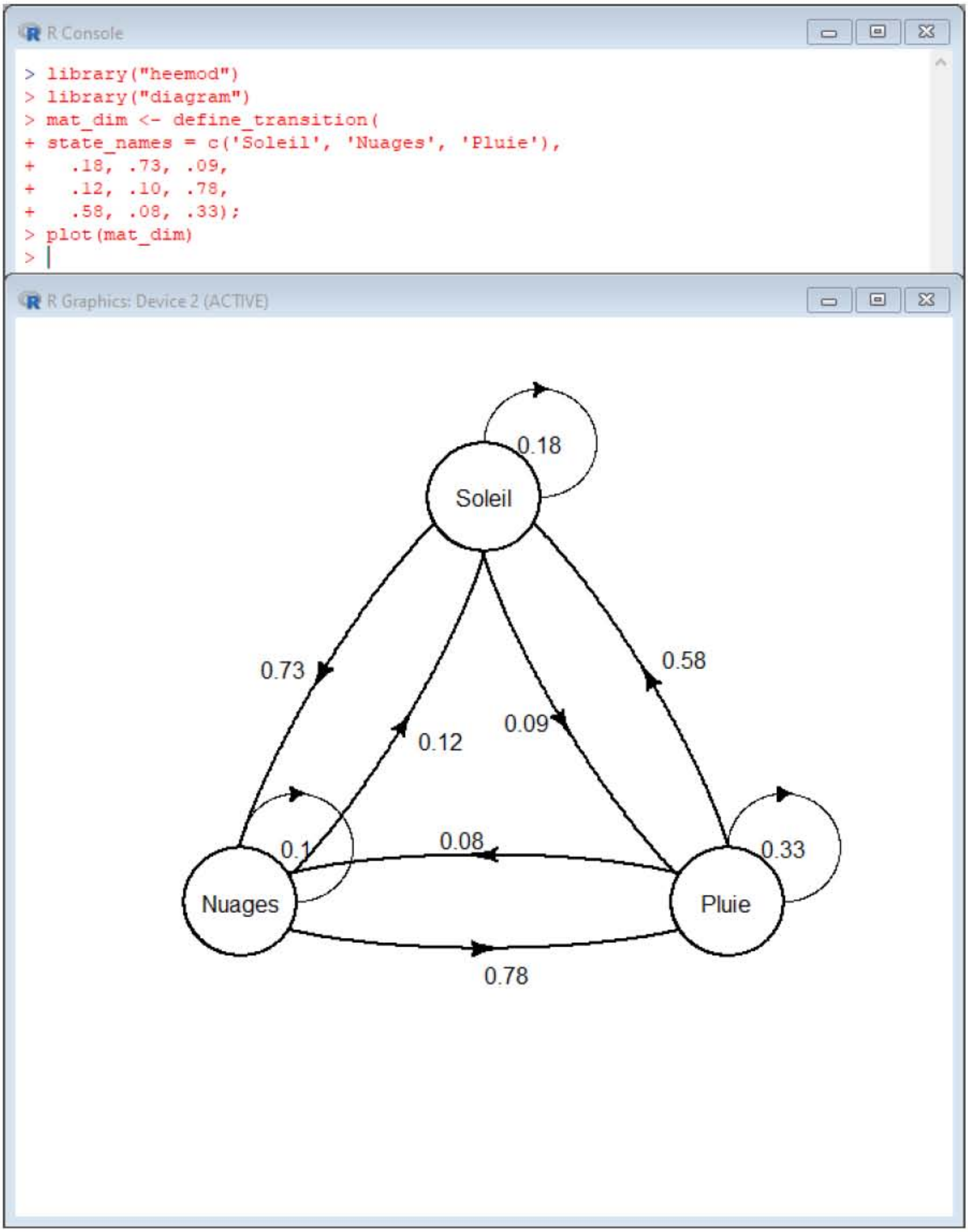
                     Non Oui
  AST Intel 150        9  12
  AST Intel 200       13  19
  Compaq Presario 100 14  15
  IBM 500             12  15
> as.array(margin.table(mytable,1))%*%t(as.array(margin.table(mytable,2)))/margin.table(mytable)

                          Non      Oui
  AST Intel 150        9.247706 11.75229
  AST Intel 200       14.091743 17.90826
  Compaq Presario 100 12.770642 16.22936
  IBM 500             11.889908 15.11009
> 
```

Ainsi, toute chose étant égale par ailleurs, les AST Intel 150 non payés sont un peu en-dessous de la valeur à laquelle on aurait pu s'attendre si toutes les choses avaient été égales par ailleurs, les AST Intel 200 très en-dessous, les Compaq très au-dessus et les IBM un tout petit peu au-dessus.

Diagramme d'association pour table de contingence

Nous allons ici refaire le même diagramme de Markov que dans le cous théorique avec les packages **diagram** et **heemod** et la fonction **define_transition( )**:



Exemple 119.: Diagramme à mosaïques

R 3.1.1 & 3.2.1

Considérons le fichier suivant de traitement contre l'Ulcère:

```
Traitement;Guerison;Nombre
0 mg C;Oui;69
0 mg C;Non;99
400 mg C;Oui;113
400 mg C;Non;69
800 mg C;Oui;120
800 mg C;Non;45
1600 mg C;Oui;145
1600 mg C;Non;43
```

Nous souhaiterions afficher ces résultats sous la forme d'un mosaïque de barre. Pour cela, nous utilisons d'abord la commande **xtabs( )** pour faire une synthèse simple et ensuite les fonctions **dotchart( )** et **mosaicplot( )** pour les graphiques qui nous intéressent (c'est surtout ce dernier qui est l'élément d'intérêt!):

```
> mydata<-read.csv("C:/tmp/Ulcere.csv",header=T,sep=";")
> tab.Ulcer=xtabs(Nombre~Traitement+Guerison,mydata)
> tab.Ulcer
           Guerison
Traitement  Non Oui
  0 mg C     99  69
  1600 mg C  43 145
  400 mg C   69 113
  800 mg C   45 120
> #nous combinons les deux types traditionnels de graphiques
> par(mfrow=c(1,2),mar=c(4,2,1,1))
> dotchart(tab.Ulcer)
> mosaicplot(tab.Ulcer,color=T,las=1,main=" ",xlab="Traitement",ylab="Status")
> 
```

Nous obtenons alors (voir page suivante):

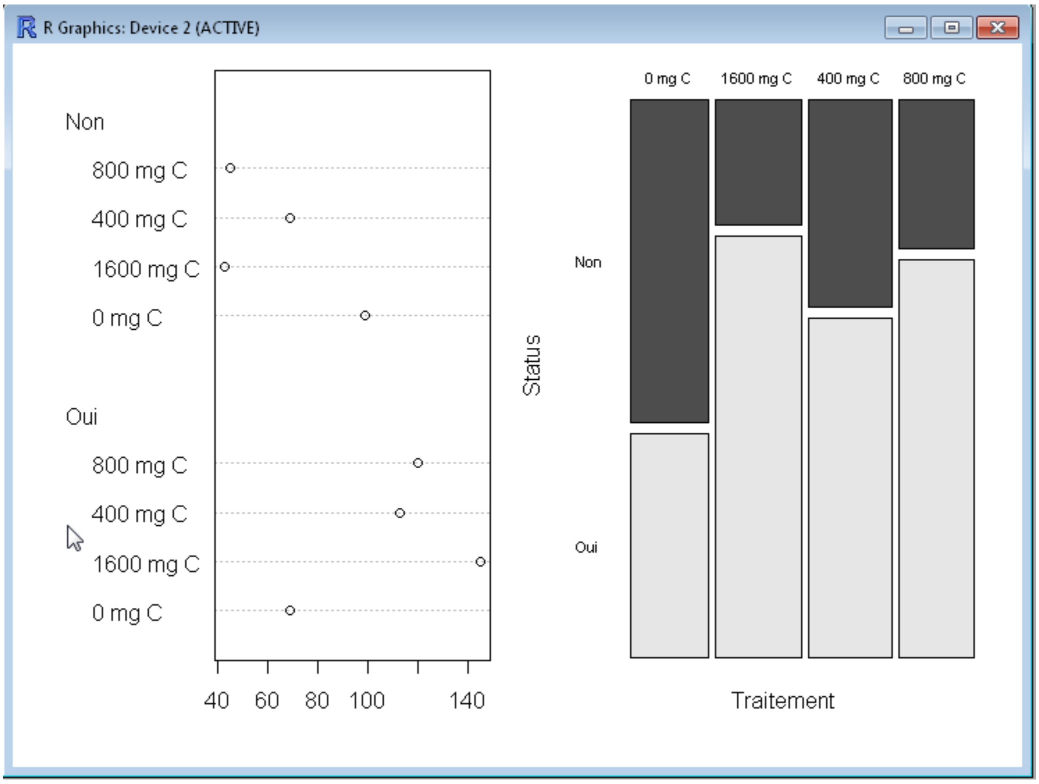


Figure 89 Graph à mosaïques

Toujours dans les diagrammes à mosaïques considérons le jeu de données suivant:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> mydata<-read.csv("c:/tmp/CountriesGNI.csv",header=T,sep=";");mydata
   iso3                  country     continent population   GNI
1   ABW                    Aruba North America        108     0
2   AFG              Afghanistan          Asia      34385   410
3   AGO                   Angola        Africa      19082  3960
4   ALB                  Albania        Europe       3205  3960
5   ARE     United Arab Emirates          Asia       7512     0
6   ARG                Argentina South America      40412  8620
7   ARM                  Armenia          Asia       3092  3200
8   ASM           American Samoa       Oceania         68     0
9   ATG      Antigua and Barbuda North America         88 13280
10  AUS                Australia       Oceania      22299 46200
11  AUT                  Austria        Europe       8390 47030
12  AZE               Azerbaijan          Asia       9054  5330
13  BDI                  Burundi        Africa       8382   170
14  BEL                  Belgium        Europe      10896 45840
15  BEN                    Benin        Africa       8850   780
16  BFA             Burkina Faso        Africa      16468   550
17  BGD               Bangladesh          Asia     148692   700
18  BGR                 Bulgaria        Europe       7534  6280
19  BHR                  Bahrain          Asia       1262     0
20  BHS                  Bahamas North America        343 22240
21  BIH   Bosnia and Herzegovina        Europe       3760  4770
22  BLR                  Belarus        Europe       9490  5950
23  BLZ                   Belize North America        345  3810
24  BMU                  Bermuda North America         65     0
25  BOL                  Bolivia South America       9929  1810
26  BRA                   Brazil South America     194946  9390
27  BRB                 Barbados North America        274     0
```

Pour faire un diagramme mosaïque nous utiliser la commande du package **treemap**:

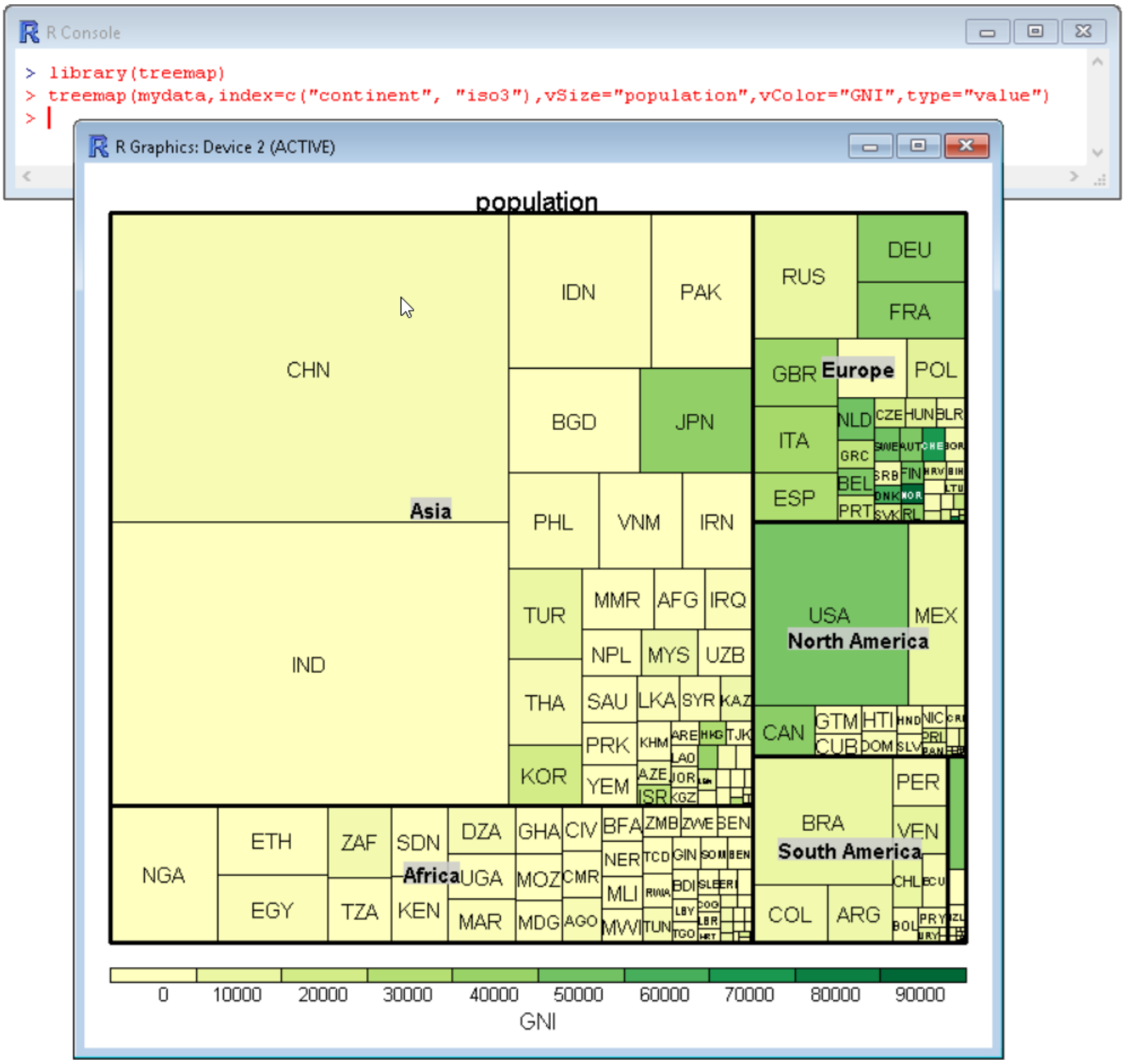


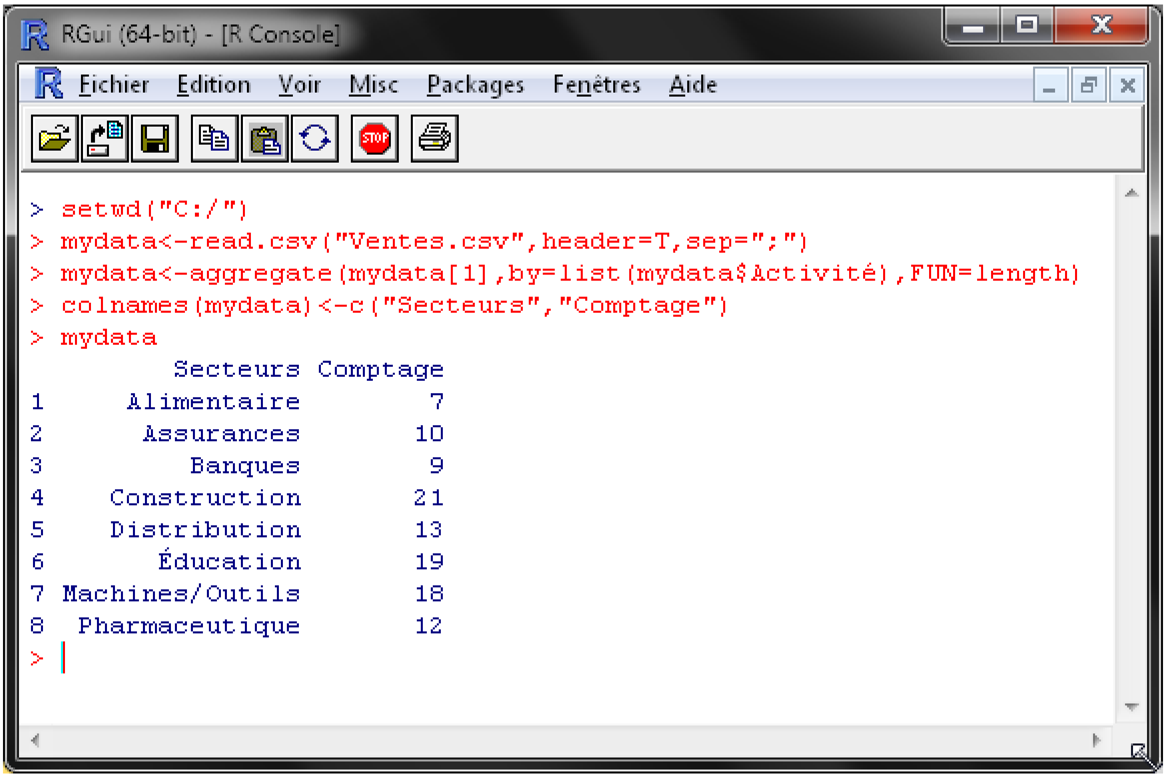
Figure 90 Graph de type TreeMap

Exemple 120.: Diagrammes à barres (fréquences et autres)

R 3.0.2

Diagramme à barres simples

Nous allons ici construire un diagramme de type carte à barre représentant le nombre d'occurrence des secteurs dans tout le tableau. Pour ceci, nous écrivons dans un premier temps:



et ensuite, nous utilisons la commande **barplot( )** avec quelques raffinements et la commande **text( )** pour ajouter les valeurs au-dessus des barres de l'histogramme:

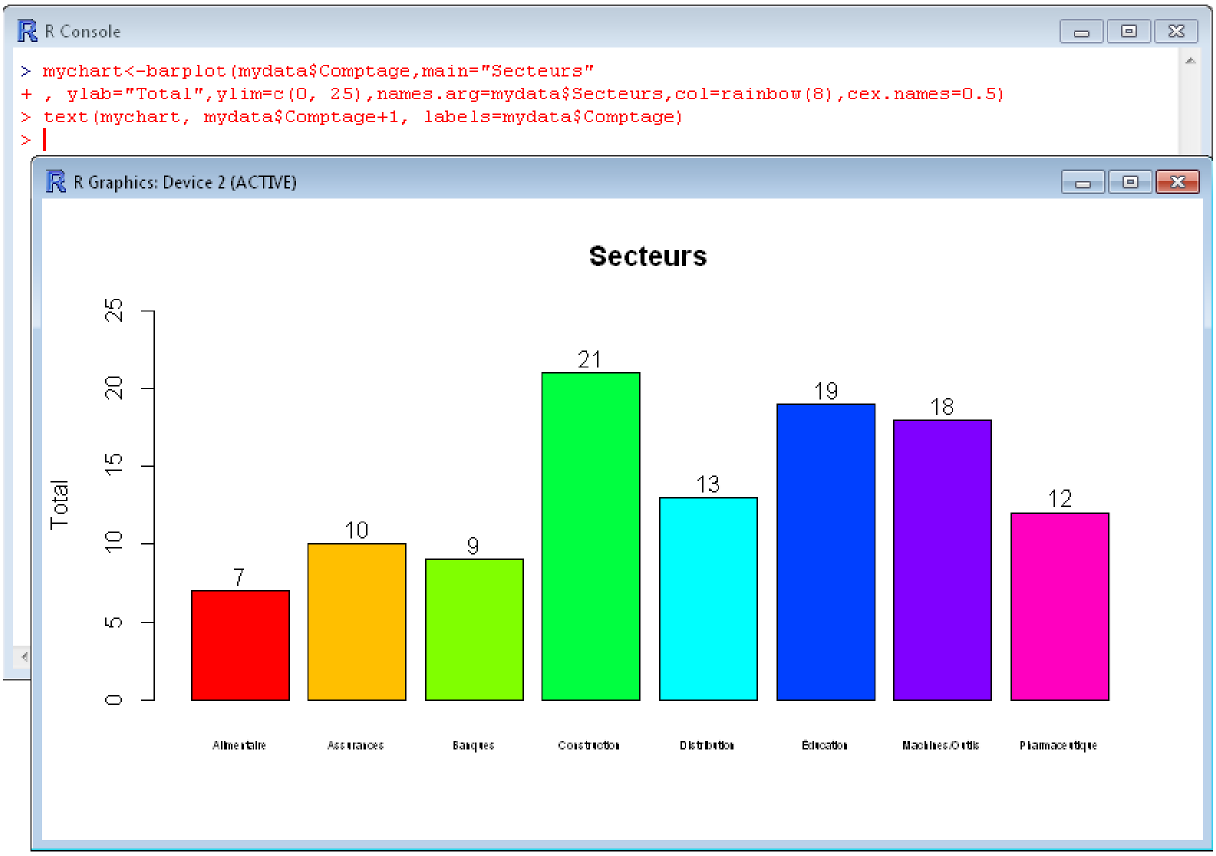


Figure 91 Graph à barres et couleurs

ou une petite variante toujours avec **barplot( )** :

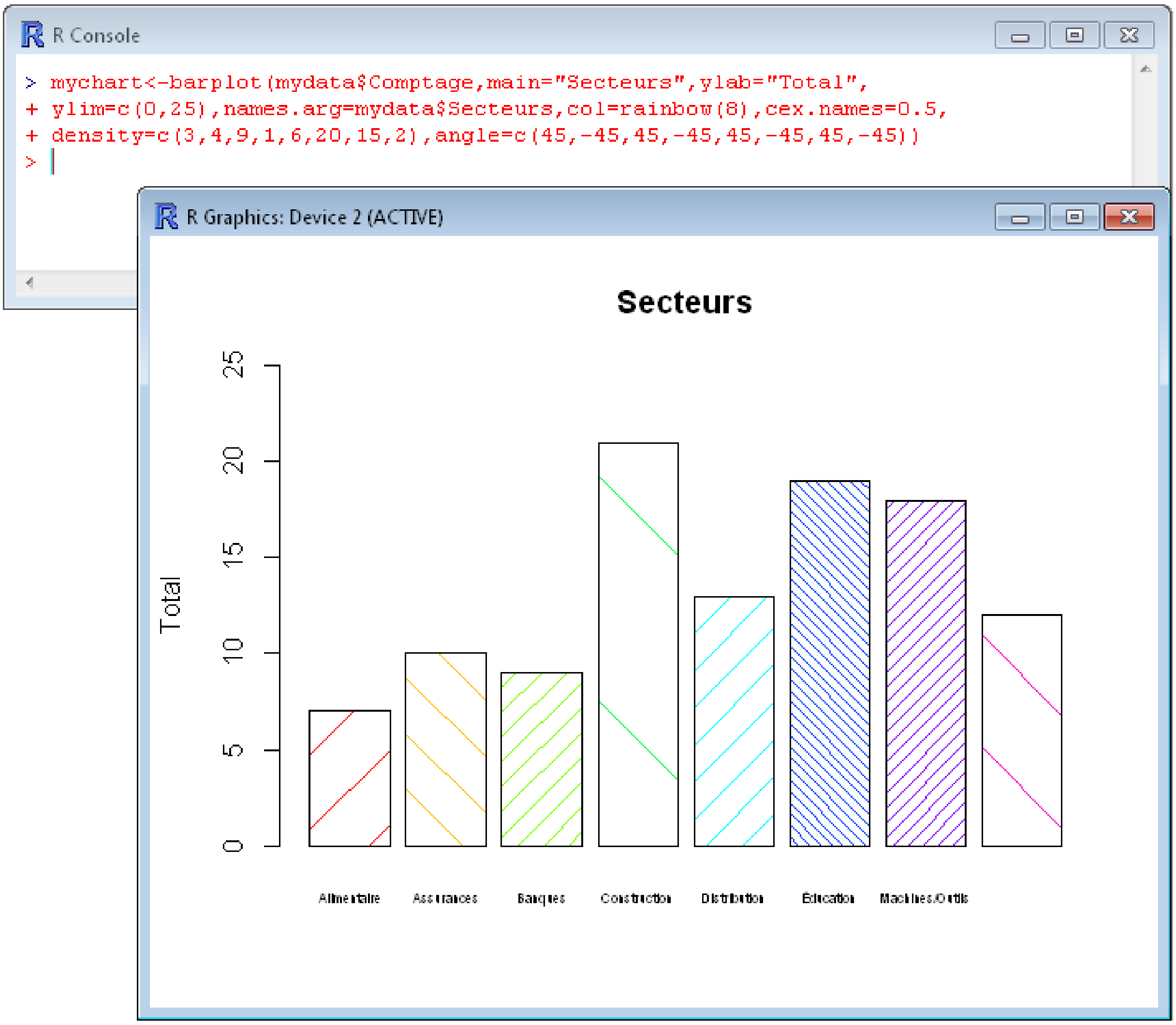


Figure 92 Graph à barres avec remplissages

Diagramme à barres empilées

Sinon voici comment faire un diagramme à barres simples empilées avec ggplot2 et selon la demande d'une de mes participants, à partir d'un jeu de données dont la structure de base n'est pas idéale à la base:

Graph_Barres_Cumul.csv - Notepad

File Edit Format View Help

```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

Pour en faire un diagramme à barres, nous écrirons:

```
> mydata<-read.csv("c:/tmp/Graph_Barres_Cumul.csv",header=T,sep=";")
> Graph_Barres_Cumul<-mydata[,-2]
> mydata
   Mois Couts CoutA CoutB CoutC
1     1 44439   515   541   928
2     2 43936   929   692   711
3     3 44464   800   710   824
4     4 41533   979   675   758
5     5 46343  1165  1147   635
6     6 44922   651   939   901
7     7 43203   847   755   580
8     8 43000   942   908   589
9     9 40967   630   738   682
10   10 48582  1113  1175  1050
11   11 45003  1086  1075   984
12   12 44303   843   640   828
13   13 42070   500   752   708
14   14 44353   813   989   804
15   15 45968  1190   823   904
16   16 47781  1200  1108  1120
17   17 43202   731   590  1065
18   18 44074  1089   607  1132
19   19 44610   786   513   839
> library("reshape")
> library("ggplot2")
> mydata<-(melt(Graph_Barres_Cumul,id="Mois"))
>
> ggplot(mydata, aes(fill=variable, y=value, x=as.factor(Mois))) +
+   geom_bar( stat="identity")+labs(x = "Mois",y="Ventes")
> 
```

Ce qui donnera:

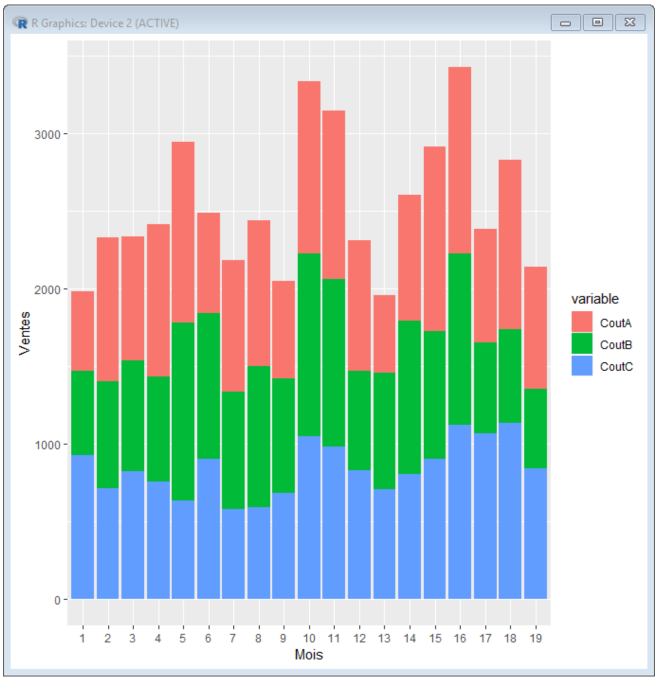


Diagramme à barres avec erreurs

Mais les barres d'erreurs manquent!!! Et comme elles ne sont pas natives dans R (mais elles le sont dans ggplot2 selon l'exemple présenté ci-après!), voici un petit script pour palier à ce problème (script tout à fait améliorable avec un choix des valeurs de barres d'erreurs empirique pour simplifier l'exemple):

```
> mydata<-read.csv("C:/tmp/Ventes.csv",header=T,sep=";")
> mydata<-aggregate(mydata,by=list(mydata$Activité),FUN=length)
> colnames(mydata)<-c("Secteurs","Comptage")
> error.bars<-function(yv,z,nn){
+     xv<-barplot(yv,ylim=c(0,(max(yv)+max(z))),names=nn,ylab=deparse(substitute(yv)))
+     g=(max(xv)-min(xv))/50
+     for(i in 1:length(xv)) {
+        lines(c(xv[i],xv[i]),c(yv[i]+z[i],yv[i]-z[i]))
+        lines(c(xv[i]-g,xv[i]+g),c(yv[i]+z[i], yv[i]+z[i]))
+         lines(c(xv[i]-g,xv[i]+g),c(yv[i]-z[i], yv[i]-z[i]))
+     }
+ }
> error.bars(mydata$Comptage,c(2,3,1,5,2,4,3,2),mydata$Secteurs)
>
```

Ce qui donne:

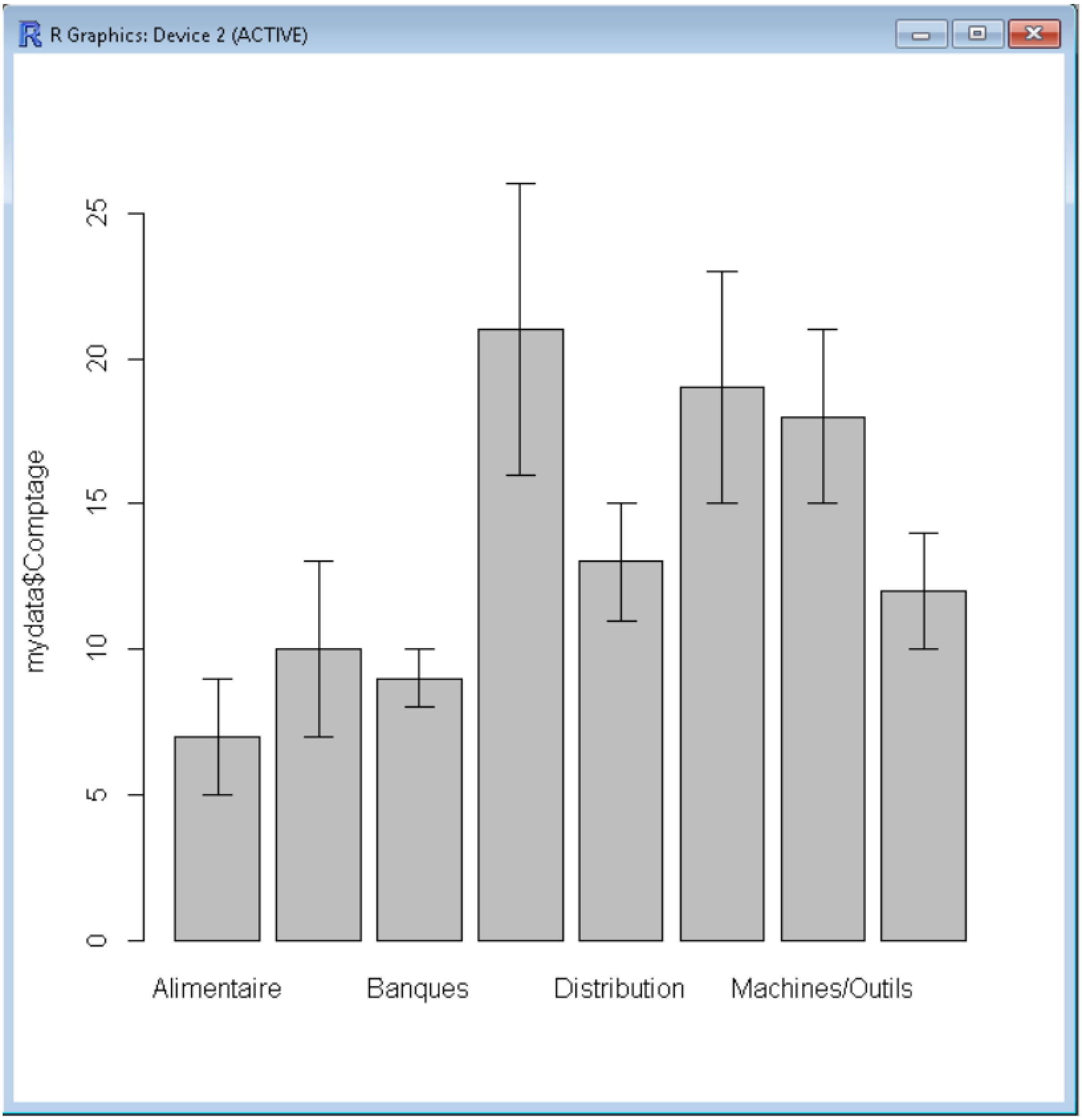
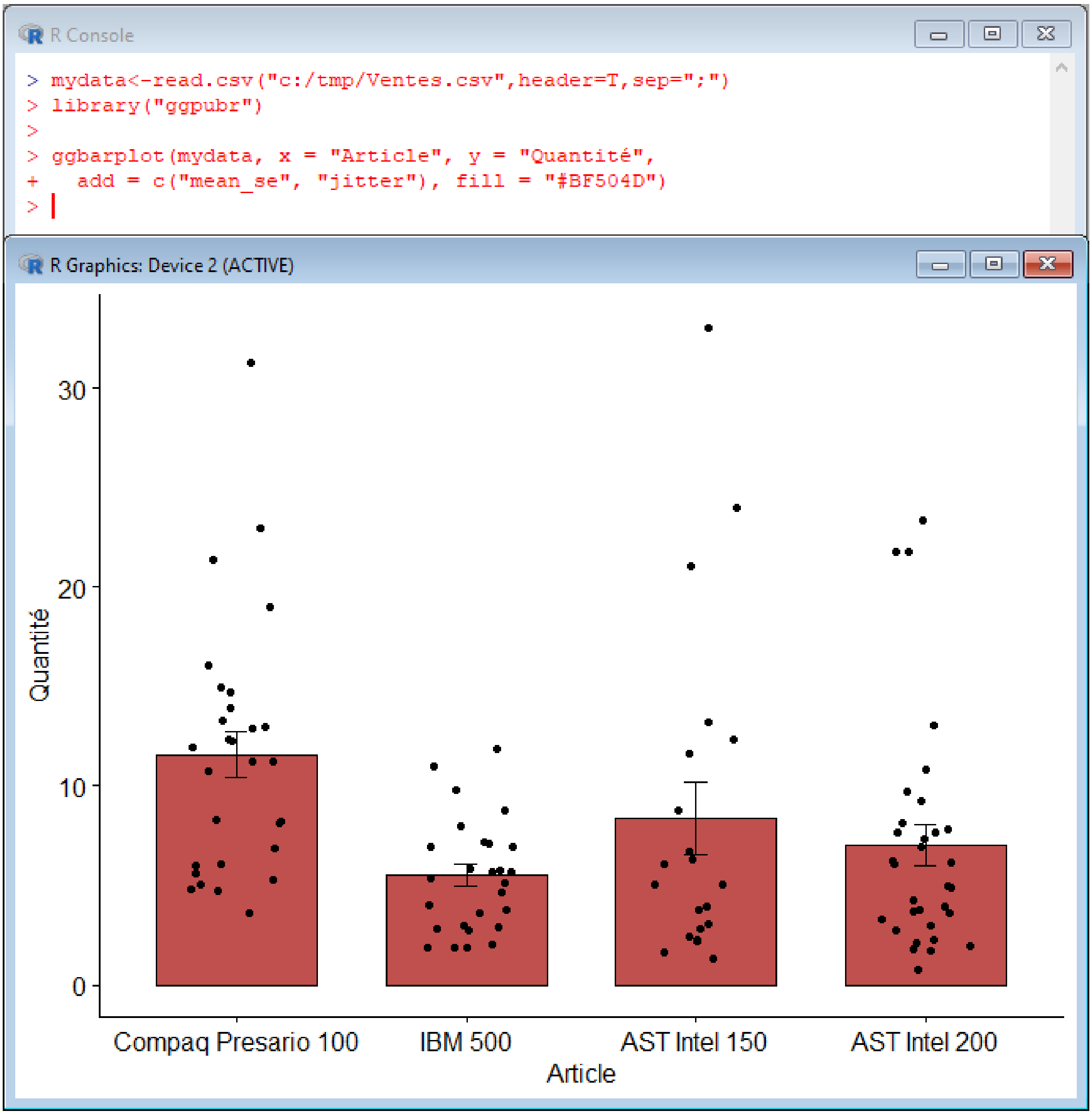


Figure 93 Graph à barres avec marges d'erreurs

En utilisant le package **ggpubr** et sa fonction **ggbarplot( )** c'est beaucoup plus facile:



On peut aussi faire des barres regroupées avec erreurs simplement avec ce package:

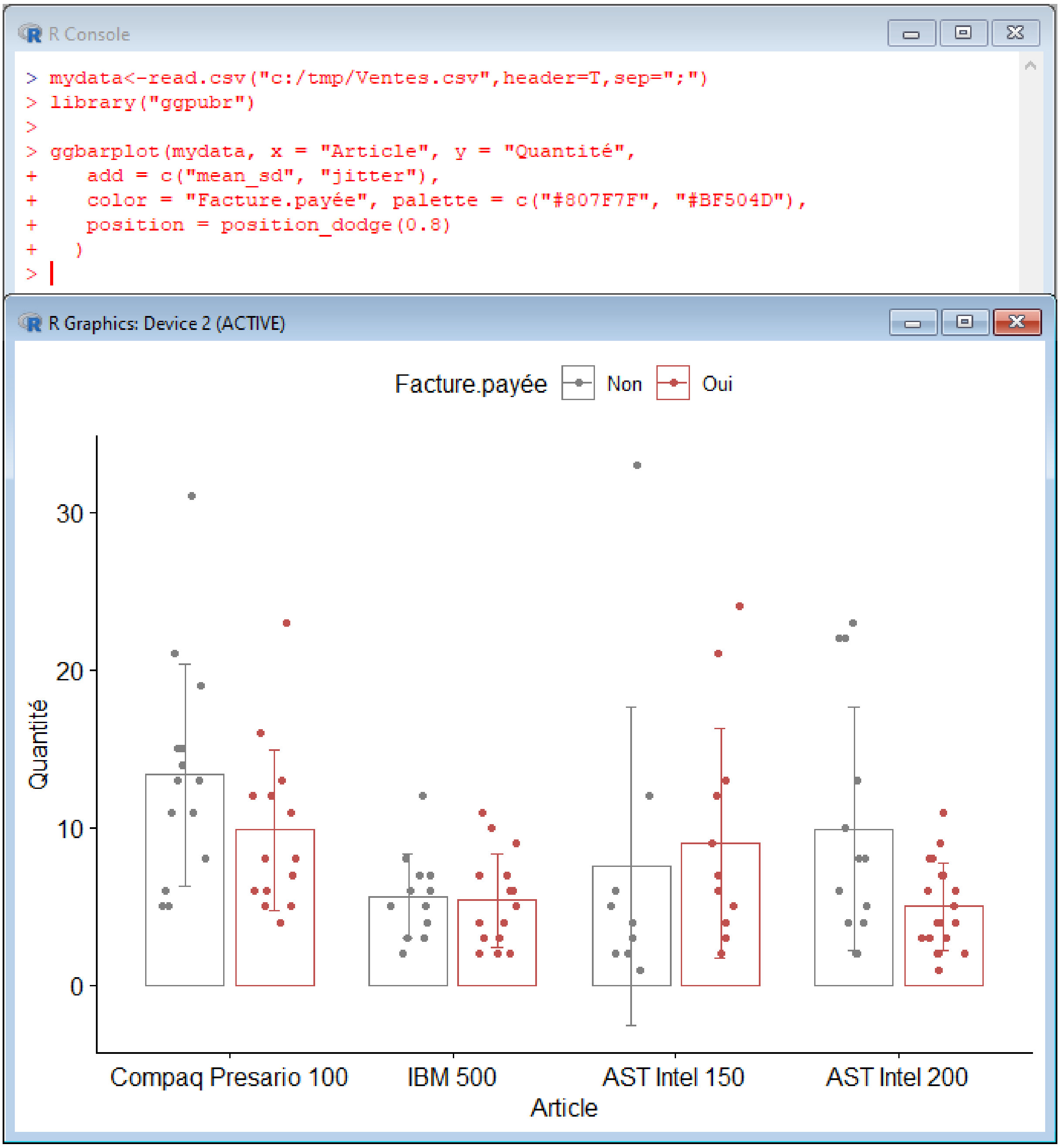


Diagramme à barres groupées

Nous pouvons également faire comme dans les tableurs des séries (pas facile à deviner) toujours avec **barplot( )** :

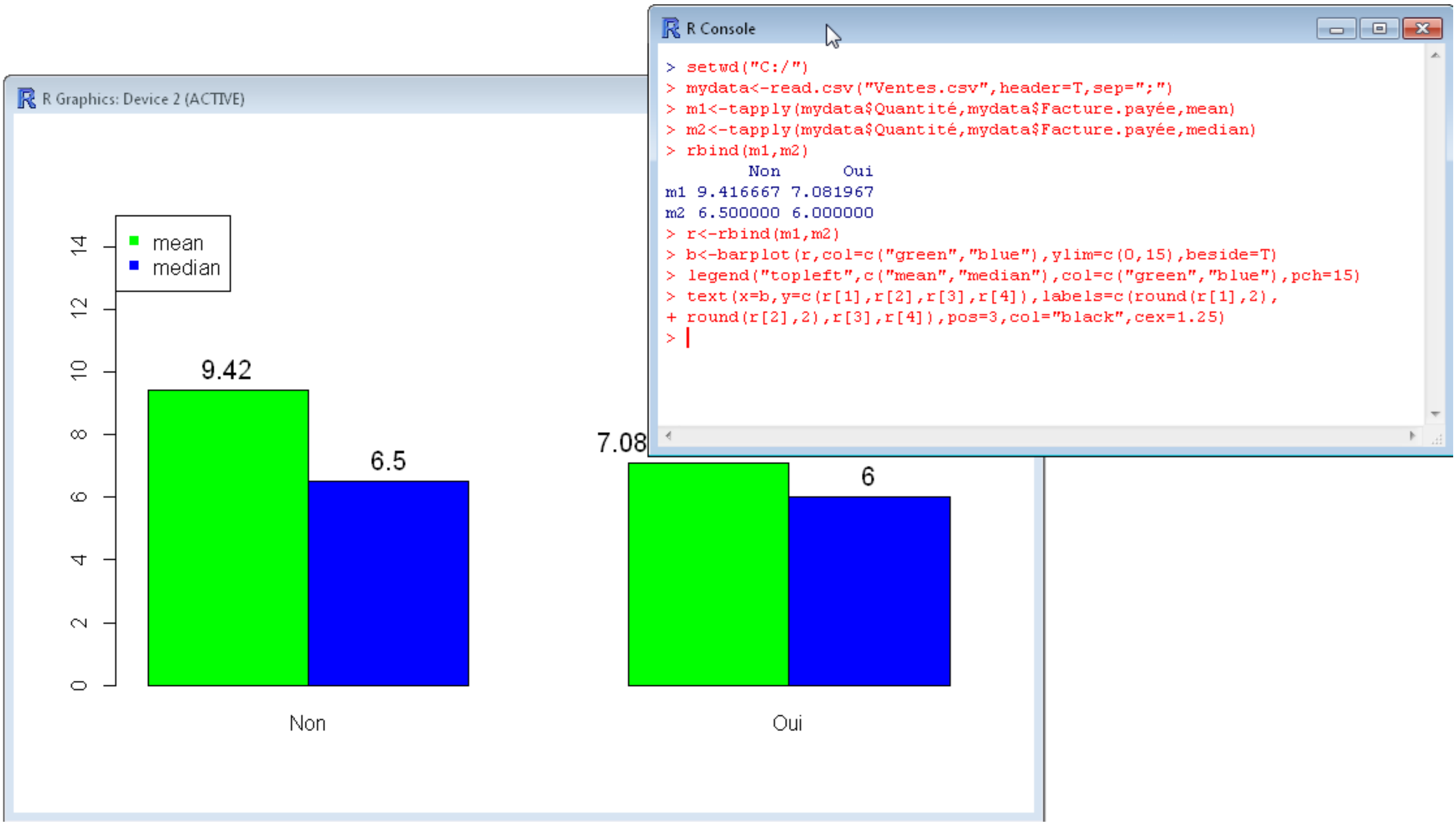
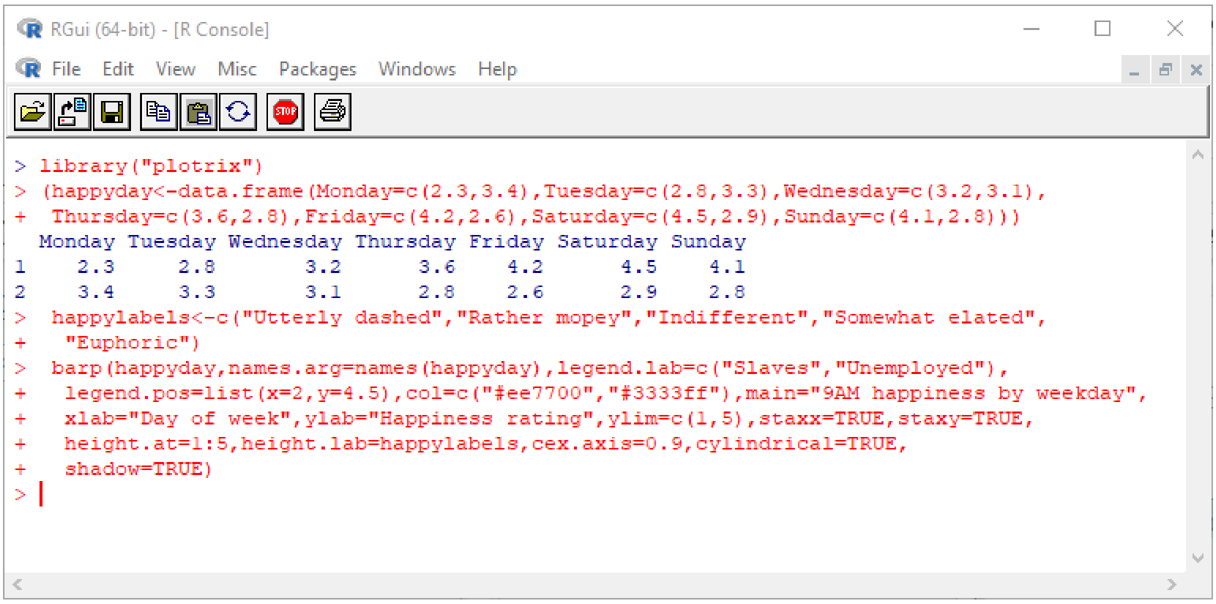


Figure 94 Graph à barres avec groupement par catégories

Sinon avec le package plotlix et sa fonction **barp( )** on peut fait des choses assez intéressantes:



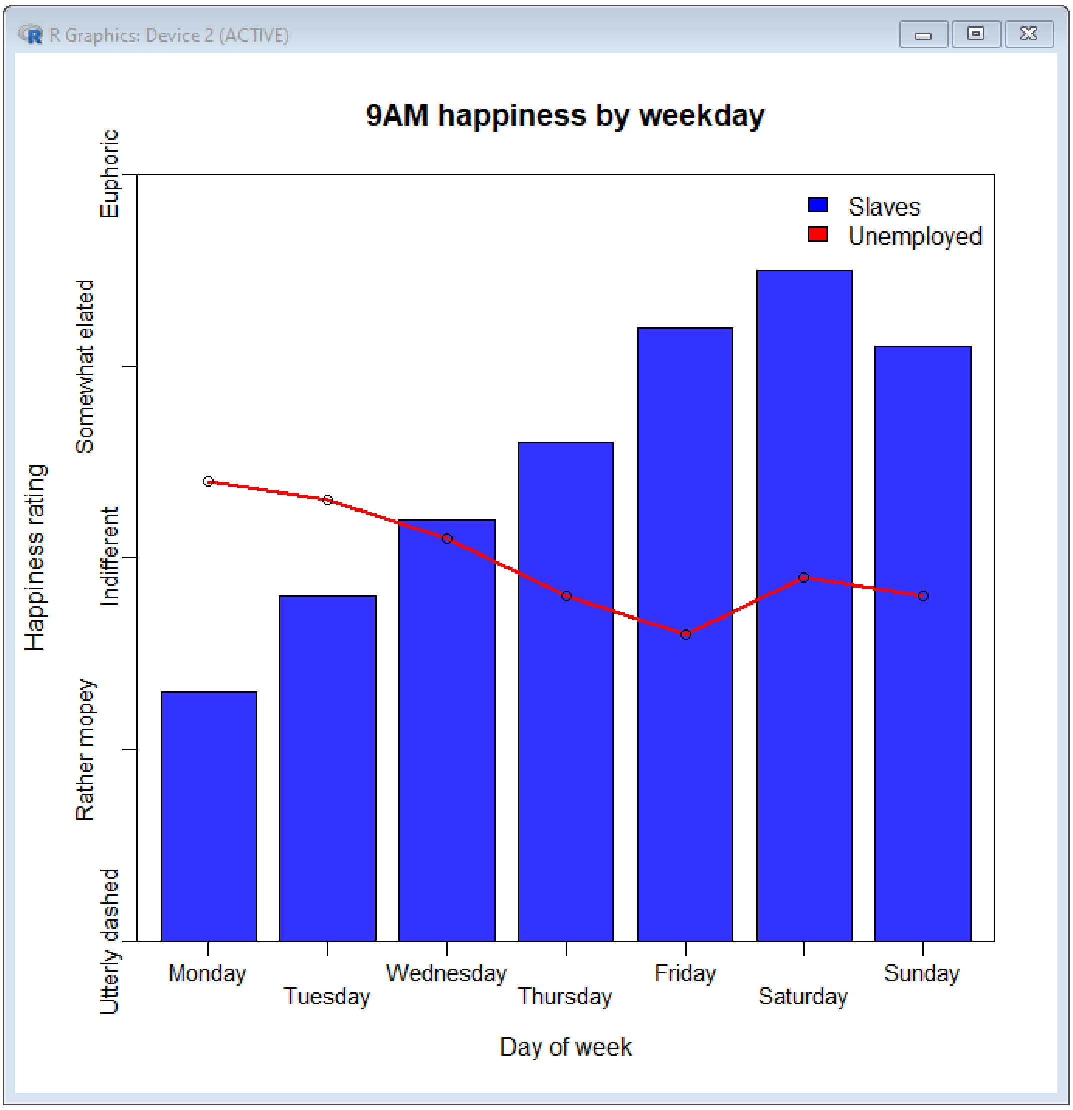
Ce qui donne:



Ou encore un fameux type de graphique à barres dit "graphique combiné":

```
> library("plotrix")
> (happyday<-data.frame(Monday=2.3,Tuesday=2.8,Wednesday=3.2,
+  Thursday=3.6,Friday=4.2,Saturday=4.5,Sunday=4.1))
  Monday Tuesday Wednesday Thursday Friday Saturday Sunday
1    2.3     2.8       3.2      3.6    4.2      4.5    4.1
>  happylabels<-c("Utterly dashed","Rather mopey","Indifferent","Somewhat elated",
+   "Euphoric")
>  barp(happyday,names.arg=names(happyday),
+   legend.pos=list(x=2,y=4.5),col="#3333ff",main="9AM happiness by weekday",
+   xlab="Day of week",ylab="Happiness rating",ylim=c(1,5),staxx=TRUE,staxy=TRUE,
+   height.at=1:5,height.lab=happylabels,cex.axis=0.9,cylindrical=FALSE,
+   shadow=FALSE)
> lines(c(3.4,3.3,3.1,2.8,2.6,2.9,2.8),col="red",lwd = 2)
> points(c(3.4,3.3,3.1,2.8,2.6,2.9,2.8))
> legend("topright", c("Slaves","Unemployed"),bty="n",fill=c("blue","red"))
>
```

Ce qui donne:



Pour revenir à **barplot( )** :

```
R Console
> setwd("C:/")
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> counts<-table(mydata$Activité,mydata$Facture.payée)
> counts

                   Non Oui
  Alimentaire        2   5
  Assurances         3   7
  Banques            5   4
  Construction       9  12
  Distribution       5   8
  Éducation          6  13
  Machines/Outils    9   9
  Pharmaceutique     9   3
> par(mar=c(10, 4, 4, 2))
> par(xpd=TRUE)
> barplot(counts,main="histogramme groupé",xlab="facture payée",
+ ylab="fréquences",col=rainbow(length(counts)))
> legend.text<-rownames(counts)
> legend(0,-3, legend.text, fill=rainbow(length(counts)),cex=0.5)
> 
```

ce qui donne:

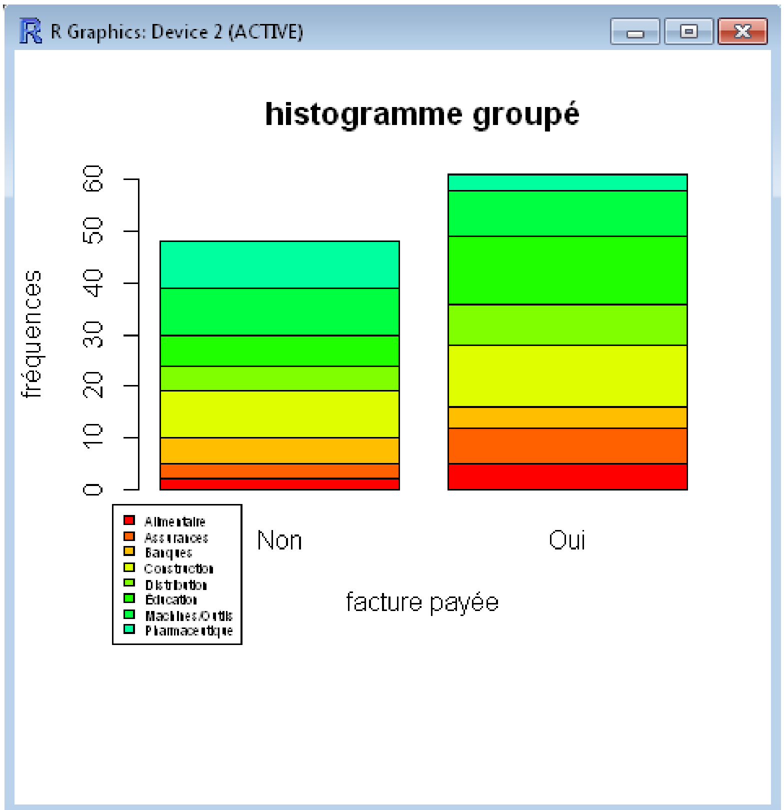
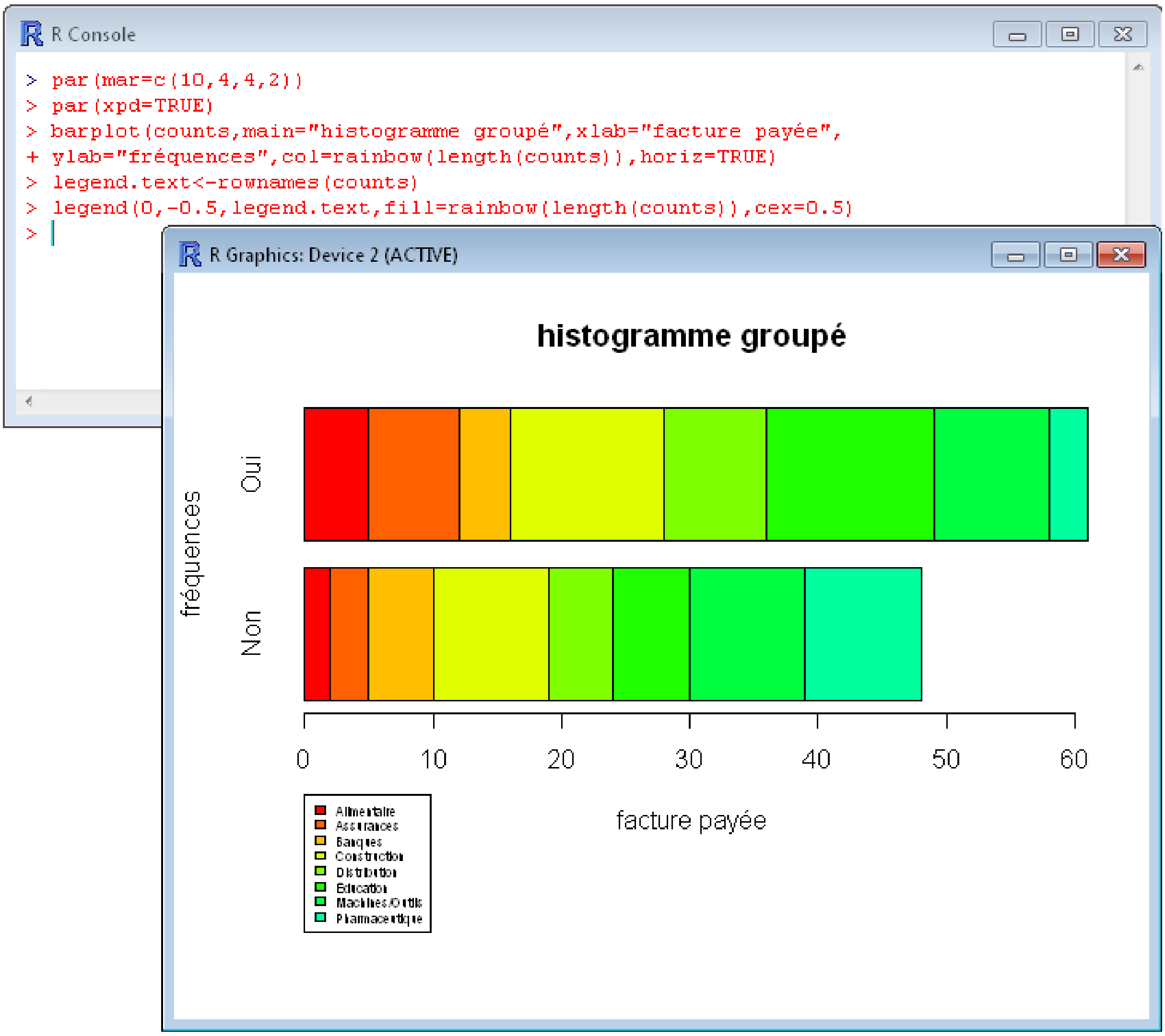


Figure 95 Graph à barres groupé (cumulé)

ou une version horizontale toujours avec **barplot( )** :



Compilation des diagrammes à barres classiques

Je l'ai fait pour le fun et pour des raisons pédagogiques! D'abord one prépare les jeux de données que nous allons utiliser:

```
> library("dplyr")
> library("ggplot2")
> library("ggpubr")
>
>
> df <- diamonds %>%
+   filter(color %in% c("J", "D")) %>%
+   group_by(cut, color) %>%
+   summarise(counts = n())
`summarise()` has grouped output by 'cut'. You can override using the `.groups`
argument.
> head(df, 4)
# A tibble: 4 x 3
# Groups:   cut [2]
  cut   color counts
  <ord> <ord>  <int>
1 Fair  D        163
2 Fair  J        119
3 Good  D        662
4 Good  J        307
>
> df_norm <- diamonds %>%
+   filter(color %in% c("J", "D"))
> head(df_norm)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
2  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
3  0.3  Good      J     SI1      64      55   339  4.25  4.28  2.73
4  0.23 Ideal     J     VS1      62.8    56   340  3.93  3.9   2.46
5  0.31 Ideal     J     SI2      62.2    54   344  4.35  4.37  2.71
6  0.3  Good      J     SI1      63.4    54   351  4.23  4.29  2.7
>
```

Ensuite on écrit le script pour les quatre graphiques qui nous intéressent:

```
> theme_set(theme_pubclean())
>
> StackedBars<-ggplot(df, aes(x = cut, y = counts)) +
+   ggtitle("Stacked Bars") +
+     geom_bar(aes(color = color, fill = color),
+     stat = "identity", position = position_stack() ) +
+   scale_color_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   scale_fill_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   theme(legend.position="bottom", legend.direction="horizontal",
+                  axis.text=element_text(size=8),legend.title = element_blank())
> #pour mettre à l'horizontale
> #+ coord_flip()
>
> GroupedBars<-ggplot(df, aes(x = cut, y = counts)) +
+   ggtitle("Grouped (absolute) Bars") +
+   geom_bar(aes(color = color, fill = color),
+     stat = "identity", position = position_dodge(0.8), width = 0.75 ) +
+   scale_color_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   scale_fill_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   theme(legend.position="bottom", legend.direction="horizontal",
+                  axis.text=element_text(size=8),legend.title = element_blank())
> #pour mettre à l'horizontale
> #+ coord_flip()
>
>
> EroorBars<-ggplot(df_norm , aes(cut, price, fill = color)) +
+   ggtitle("Grouped Bars with Errors") +
+   stat_summary(geom = "bar", fun = mean, position = "dodge") +
+   stat_summary(geom = "errorbar", fun.data = mean_se, position = "dodge")+
+   scale_color_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   scale_fill_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   theme(legend.position="bottom", legend.direction="horizontal",
+                  axis.text=element_text(size=8),legend.title = element_blank())
> #pour mettre à l'horizontale
> #+ coord_flip()
>
> NormalizedBars<-ggplot(df_norm , aes(x = cut,fill=color)) +
+   ggtitle("Normalized (relative) Bars") +
+   geom_bar(aes(color = color, fill = color),position = "fill")+
+   scale_color_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   scale_fill_manual(values = c("#0073C2FF", "#EFC000FF"))+
+   theme(legend.position="bottom", legend.direction="horizontal",
+                  axis.text=element_text(size=8),legend.title = element_blank())
> #pour mettre à l'horizontale
> #+ coord_flip()
>
> ggarrange(StackedBars, GroupedBars,NormalizedBars,EroorBars, ncol = 2, nrow = 2)
>
```

Ce qui donne:

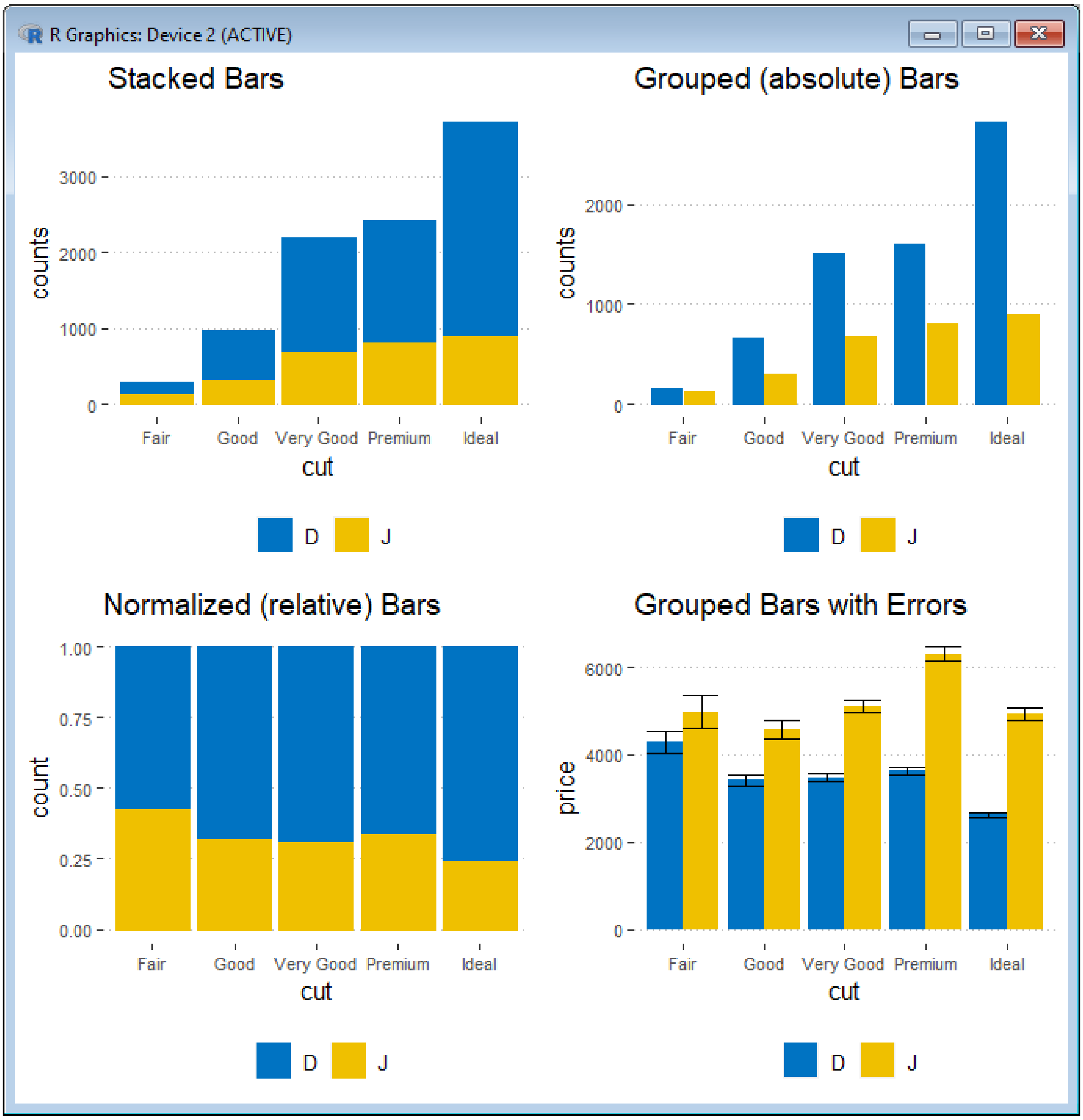
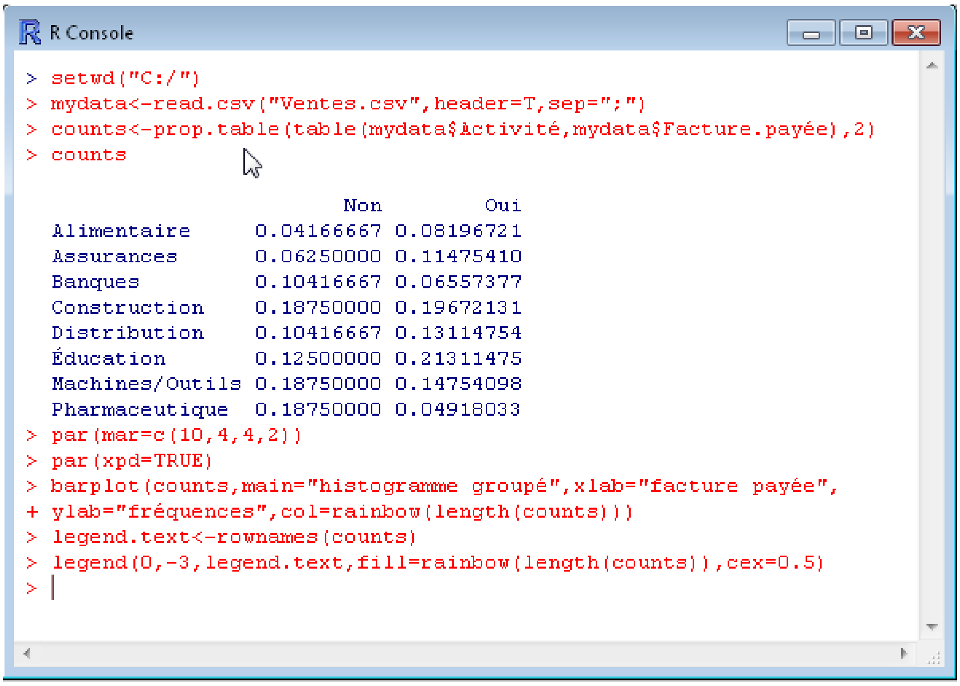


Diagramme à barres empilées à 100%

Nous pouvons mettre évidemment cela en normalisant à l'unité:



Ce qui donne (mais sans l'axe en pourcentages):

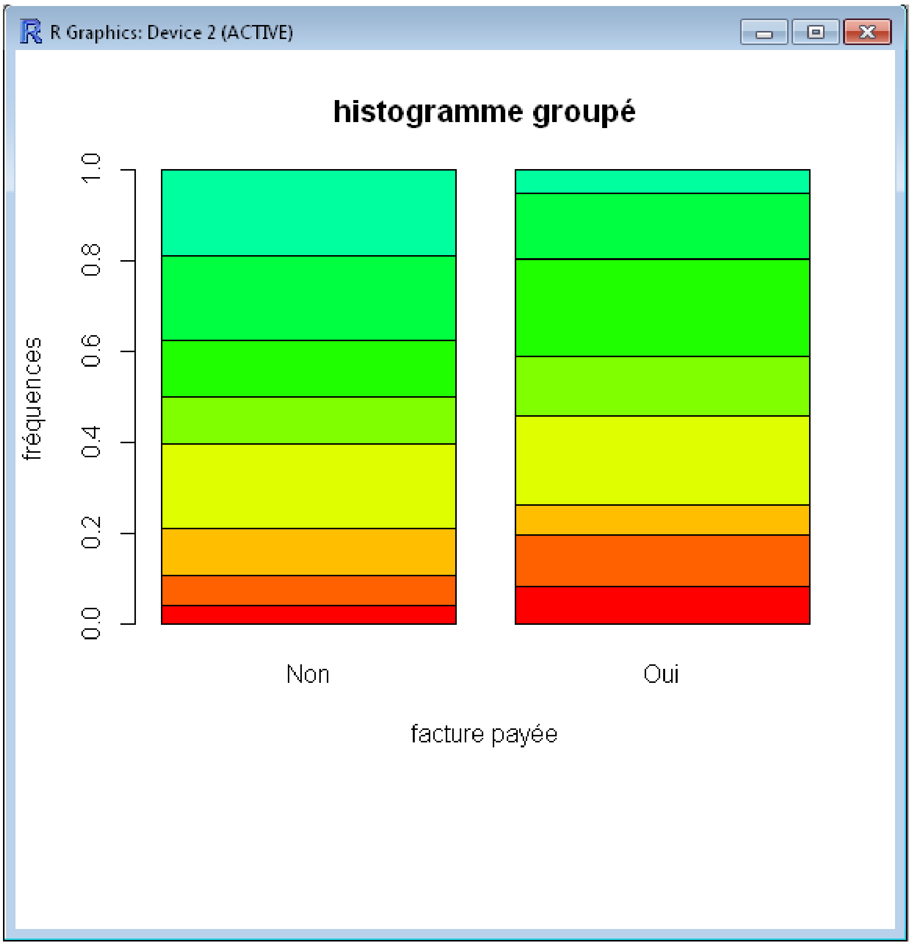


Figure 96 Graph à barres groupé et normalisé

Avec les packages **scales**, **reshape2** et **ggplot2** et en associant de façon très subtile les commandes **prop.table( )** , **as.data.frame.matrix( )** , **melt( )** et **ggplot( )** nous pouvons faire bien mieux quand il s'agit d'obtenir des diagrammes à barres groupés et normalisés:

```
> library(scales)
> library(reshape2)
> library(ggplot2)
> setwd("C:/")
> mydata<-read.csv("Ventes.csv",header=T,sep=";")
> mydata<-prop.table(table(mydata$Activité,mydata$Facture.payée),2)
> mydata<-as.data.frame.matrix(mydata)
> mydata<-melt(cbind(mydata,ind=rownames(mydata)),is.vars=c("ind"))
Using ind as id variables
> mydata
               ind variable      value
1      Alimentaire      Non 0.04166667
2       Assurances      Non 0.06250000
3          Banques      Non 0.10416667
4     Construction      Non 0.18750000
5     Distribution      Non 0.10416667
6        Éducation      Non 0.12500000
7  Machines/Outils      Non 0.18750000
8   Pharmaceutique      Non 0.18750000
9      Alimentaire      Oui 0.08196721
10      Assurances      Oui 0.11475410
11         Banques      Oui 0.06557377
12    Construction      Oui 0.19672131
13    Distribution      Oui 0.13114754
14       Éducation      Oui 0.21311475
15 Machines/Outils      Oui 0.14754098
16  Pharmaceutique      Oui 0.04918033
>
> ggplot(mydata,aes(x=factor(variable),y=value,label=paste(round(value*100),
+      "%"),fill=factor(ind))+ geom_bar(stat="identity") + geom_text(position="stack",
+      aes(ymax=1),vjust=1) + scale_y_continuous(labels = percent)
>
```

Ce qui donne:

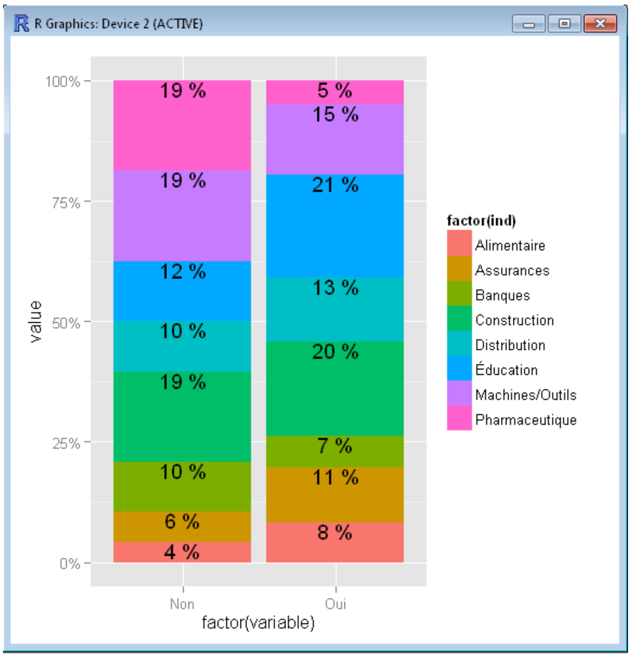
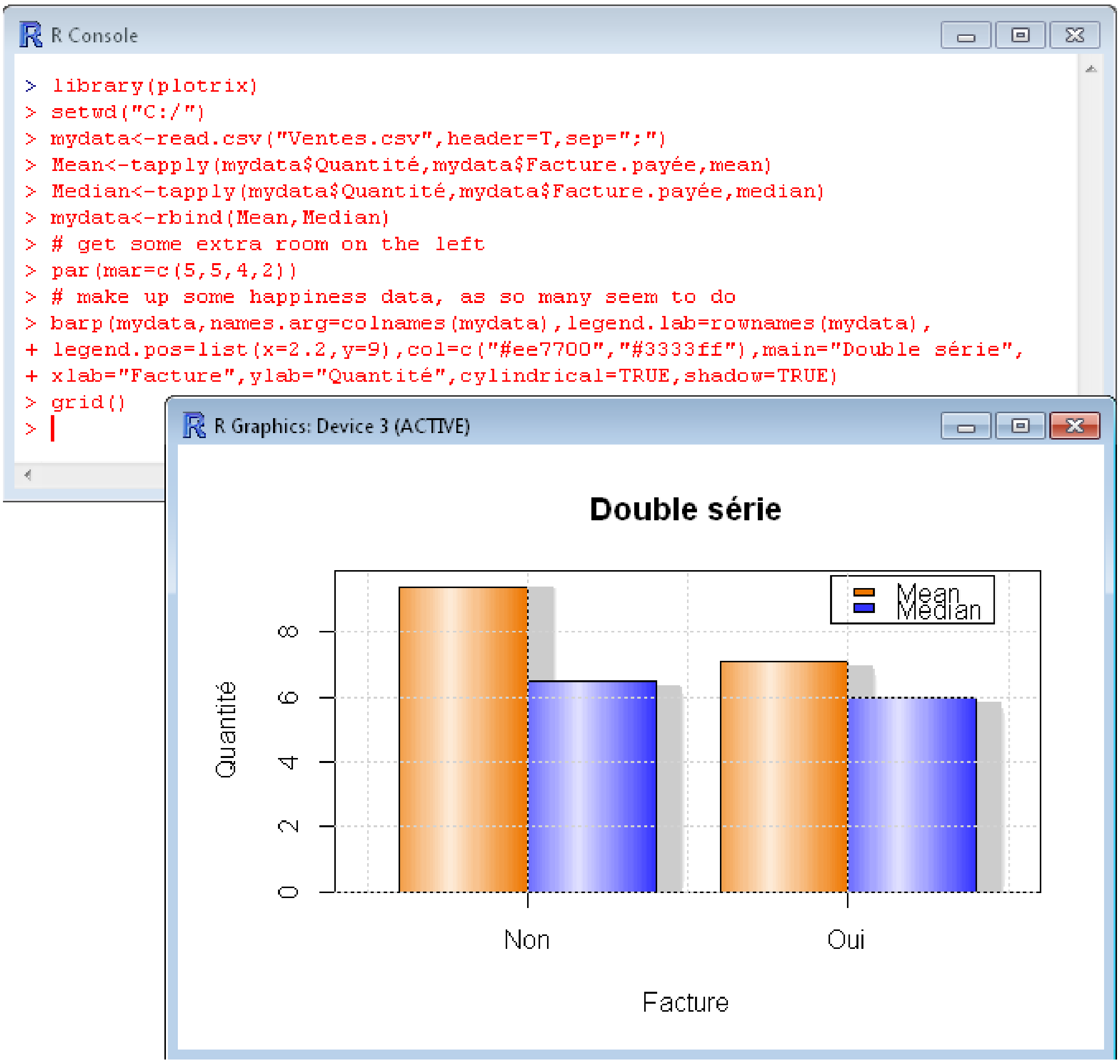


Figure 97 Graph à barres groupé et normalisé (ggplot)

Ou encore avec le package **plotrix** et en utilisant la commande **barp( )** :



Et considérons l'exercice de style suivant avec le package **ggplot2** et la commande **geom_bar( )** qui n'est pas dénuée d'intérêt:

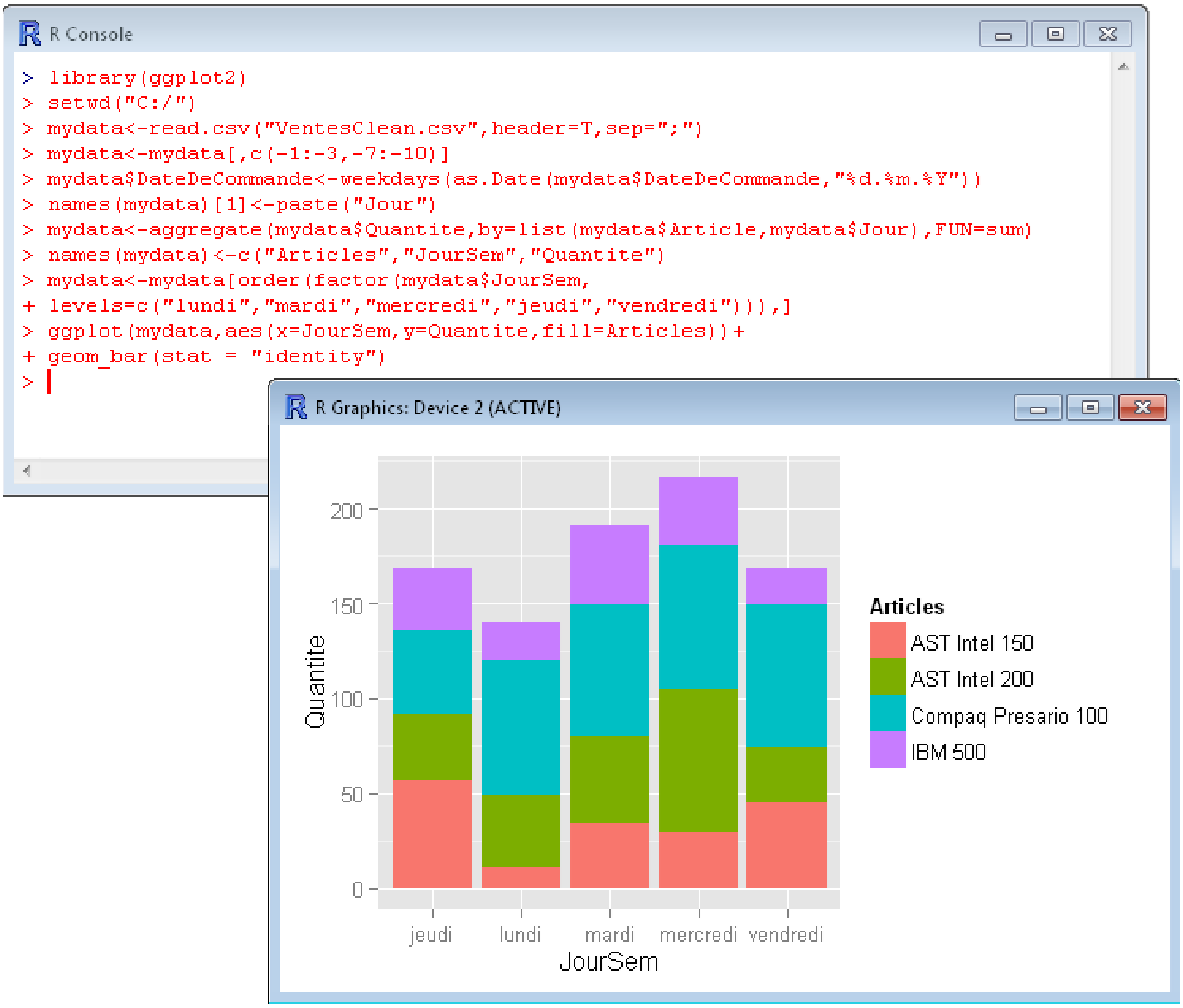


Figure 98 Graph à barres groupé cumulé (ggplot)

Le lecteur aura remarqué que trié le data frame par jour de la semaine n'influence pas l'ordre des jours dans le plot (sic!).

Enfin voyons comment faire un diagramme à barre groupé car c'est loin d'être trivial (contrairement à Microsoft Excel) en se basant sur le jeu de données suivant:

DiagrammeBarre_Groupe.csv - Notepad

```
Category;Recovery;Acids;Value
Milk;1st extraction;Formic acid;0.53
Milk;1st extraction;Chlorhydric acid;0.49
Milk;1st extraction;acetic acid;0.64
Milk;Recovery 3rd exractions;Formic acid;0.14
Milk;Recovery 3rd exractions;Chlorhydric acid;0.14
Milk;Recovery 3rd exractions;acetic acid;0.19
Water;total Extraction;Formic acid;0.76
Water;total Extraction;Chlorhydric acid;0.76
Water;total Extraction;acetic acid;0.73
```

En utilisant les packages **ggplot2**, **grid** et **gtable** et en écrivant (...):

RGui (64-bit) - [R Console]

```
> library(ggplot2)
> library(grid)
> library(gtable)
>
> mydata<-read.csv("DiagrammeBarre_Groupe.csv",header=T,sep=";")
>
> p <- ggplot(mydata, aes(x=interaction(Category, Acids),y=Value))+
+   geom_bar(aes(fill=Recovery),stat = "identity")+
+   scale_y_continuous(labels = scales::percent) +
+   scale_x_discrete(limits = levels(interaction(mydata$Category,
+                                               mydata$Acids)),
+                    labels = rep(c("Milk", "Water"), 3))+
+   coord_cartesian(ylim=c(0,0.9))+
+   annotate("text", c(1.5, 3.5, 5.5), y = - 0.15,
+            label = c("Acetic Acid",
+                      "Chlorhydric Acid",
+                      "Formic Acid"))+
+   theme_classic() +
+   theme(plot.margin = unit(c(3,1,3,1), "lines"),
+         axis.title.x = element_blank())
>
> g <- ggplot_gtable(ggplot_build(p))
> g$layout$clip[g$layout$name == "panel"] <- "off"
> grid.draw(g)
>
```

Nous arrivons à:

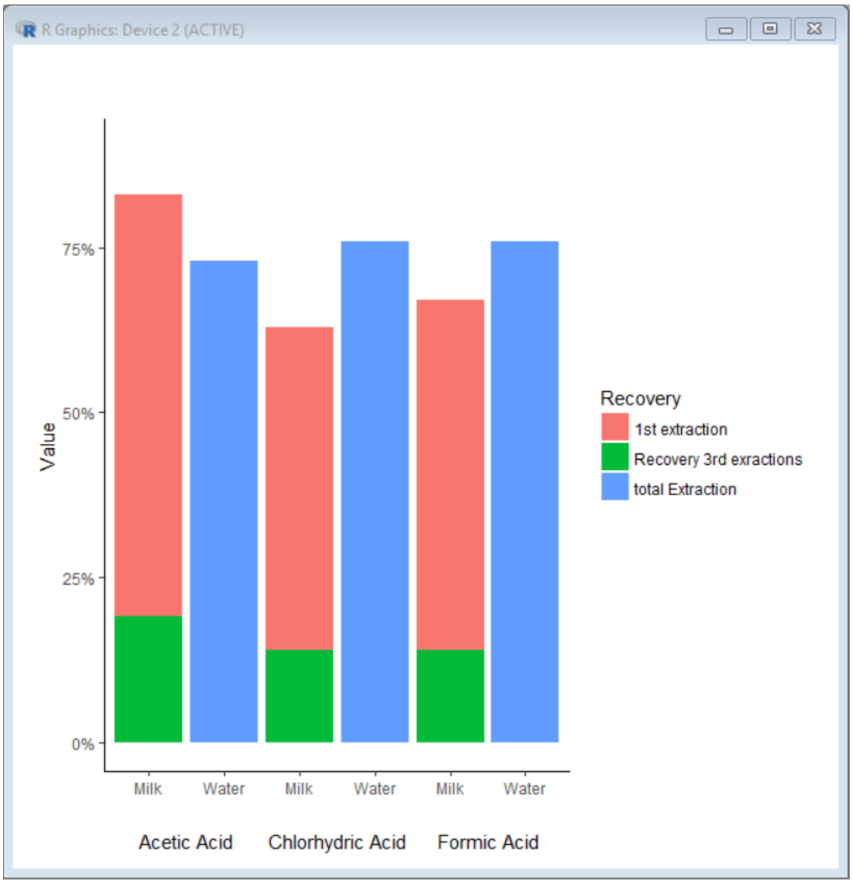


Figure 99 Graph à barres groupé cumulé avec sous-groupes

Diagramme à barres groupées avec double axe Y

Malheureusement je n'ai jamais trouvé une manière simple et élégante de faire très rapidement (en un minimum de lignes de codes), un graphique à barres groupées avec double axe Y.

Si un lecteur a une méthode plus brève pour le même résultat et aussi plus élégante nous sommes preneur!

Considérons que nous voulons de notre fichier *VentesCleans.csv* faire un graphique à barres des *Quantités* et *Prix total avec rabais*. Évidemment vu les différences de magnitude entre les deux il va nous falloir un double axe!

La solution que nous proposons est la suivante:

D'abord nous chargeons les données, supprimons le superflu, faisons un calcul de contrôle et préparons un data frame adapté au graphique à barre (le lecteur notera que cette préparation est subtile et très "bricolage"):

```
> library("ggplot2")
> library("tidyr")
> library("dplyr")
> library("scales")
>
> #on charge les données
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=";")
> #on ne garde que les colonnes qui nous intéressent
> mydata<-mydata[,c(5,6,9)]
> head(mydata)
             Article Quantite PrixTotalAvecRabais
1 Compaq Presario 100       12            19503.00
2             IBM 500        2             4598.00
3       AST Intel 150        5            13450.00
4       AST Intel 200        3             9570.00
5 Compaq Presario 100       13            21128.25
6       AST Intel 150        2             5380.00
>
> #un tableau résumé pour comparer avec les valeurs du graphique
> mydata %>% group_by(Article) %>% summarize_at(c("Quantite","PrixTotalAvecRabais"),sum,na.rm=TRUE)
# A tibble: 4 x 3
  Article             Quantite PrixTotalAvecRabais
  <chr>                  <int>               <dbl>
1 AST Intel 150            176             463635.
2 AST Intel 200            224             703028.
3 Compaq Presario 100      335             541819.
4 IBM 500                  149             339034.
>
> #on prépare un jeu de données particulier pour ce type de graph
> DFlong <- mydata |>
+   pivot_longer(cols = -Article, names_to = "Type") |>
+   group_by(Article, Type) |>
+   summarise(value = sum(value), .groups = "drop") |>
+   #ici il y a l'astuce d'un facteur 1000 qu'il faut adapter selon les situations!
+   mutate(scaled_value = ifelse(Type == "PrixTotalAvecRabais", value, value * 1000))
> DFlong
# A tibble: 8 x 4
  Article             Type                 value scaled_value
  <chr>               <chr>                <dbl>        <dbl>
1 AST Intel 150       PrixTotalAvecRabais 463635.      463635.
2 AST Intel 150       Quantite               176       176000
3 AST Intel 200       PrixTotalAvecRabais 703028.      703028.
4 AST Intel 200       Quantite               224       224000
5 Compaq Presario 100 PrixTotalAvecRabais 541819.      541819.
6 Compaq Presario 100 Quantite               335       335000
7 IBM 500             PrixTotalAvecRabais 339034.      339034.
8 IBM 500             Quantite               149       149000
>
```

Maintenant faisons le graphique:

```
> #le graphique
> options(scipen = 999) #pour éliminer la notation scientifique dans les axes Y
> ggplot(DFlong, aes(x = Article, y = scaled_value, fill = Type)) +
+   geom_col(position = "dodge", na.rm = TRUE, color = "black") +
+   #ici il y a l'astuce d'un facteur 1000 qu'il faut adapter selon les situations!
+   scale_y_continuous(sec.axis = sec_axis(~ . / 1000, name = "Quantite")) +
+   labs(y = "PrixTotalAvecRabais") +
+   theme(legend.position = "bottom") +
+   geom_text(aes(label = format(value, big.mark = "'", scientific = FALSE)),
+ vjust = -.3, position = position_dodge(width = .9))
>
```

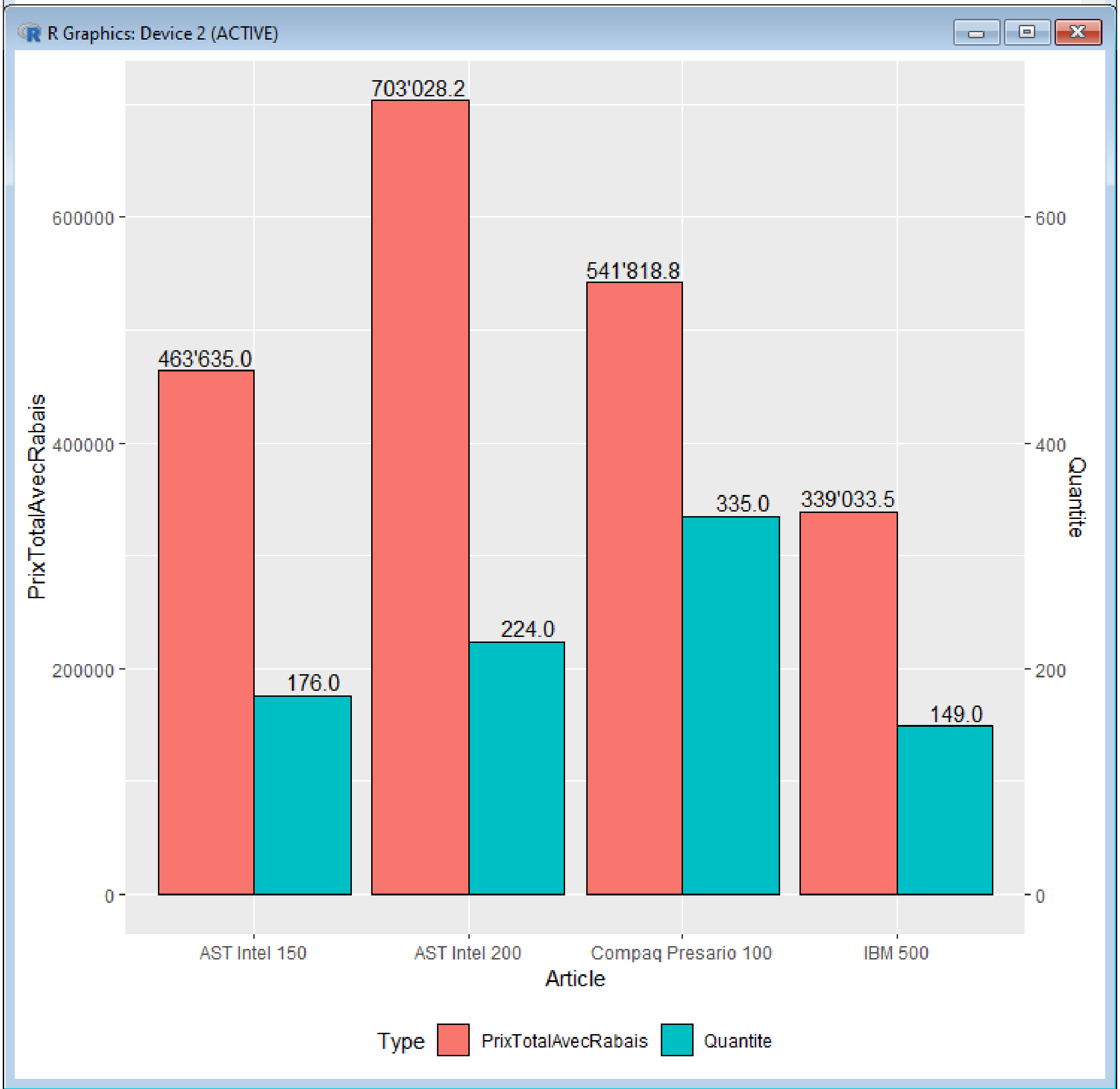


Diagramme à barres de type histogramme

Nous pouvons aussi avec la commande **hist( )** que nous retrouverons plus loin obtenir un diagramme en fréquences avec un nombre d'intervalles choisi dont nous colorions certains d'entre eux:

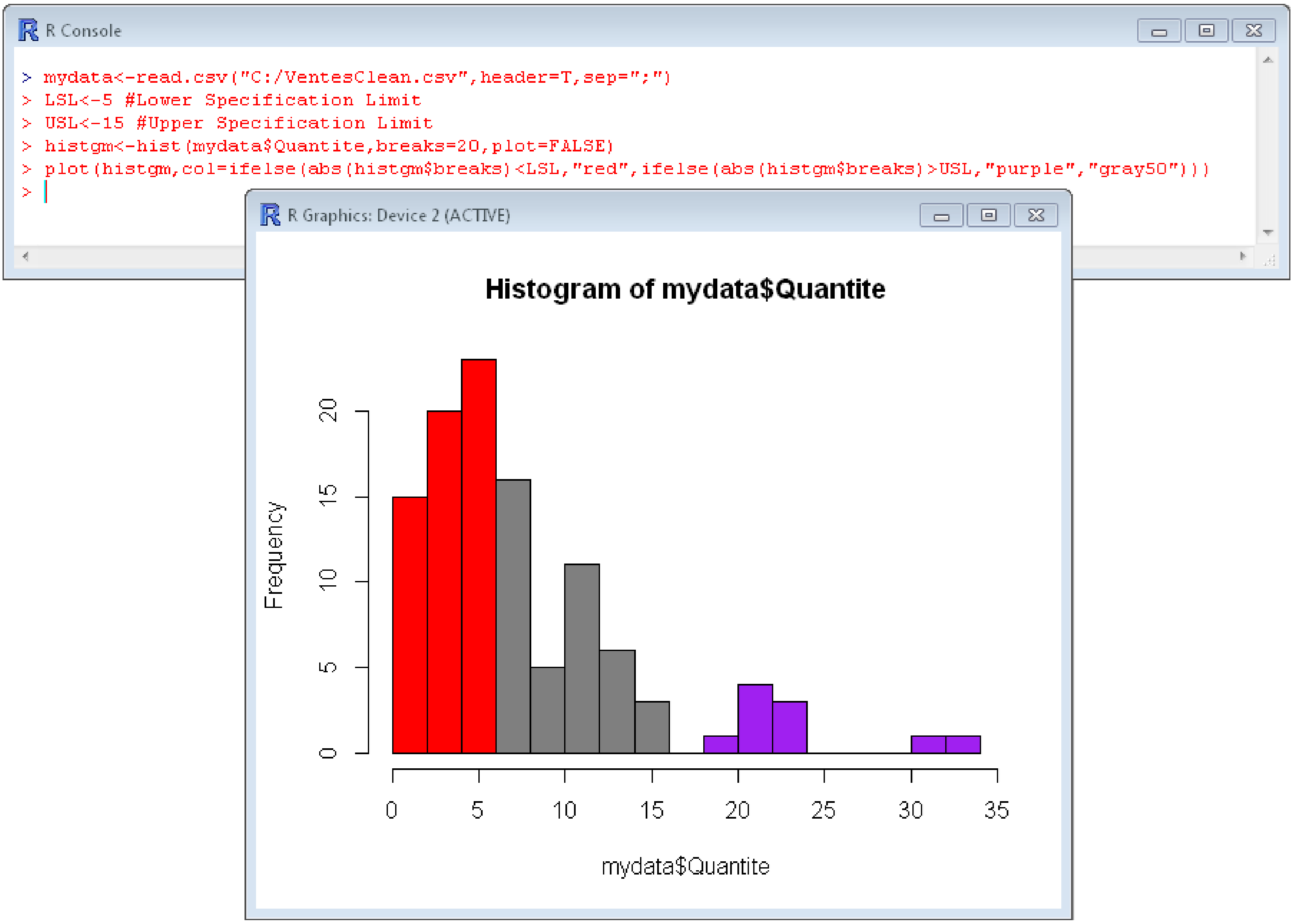
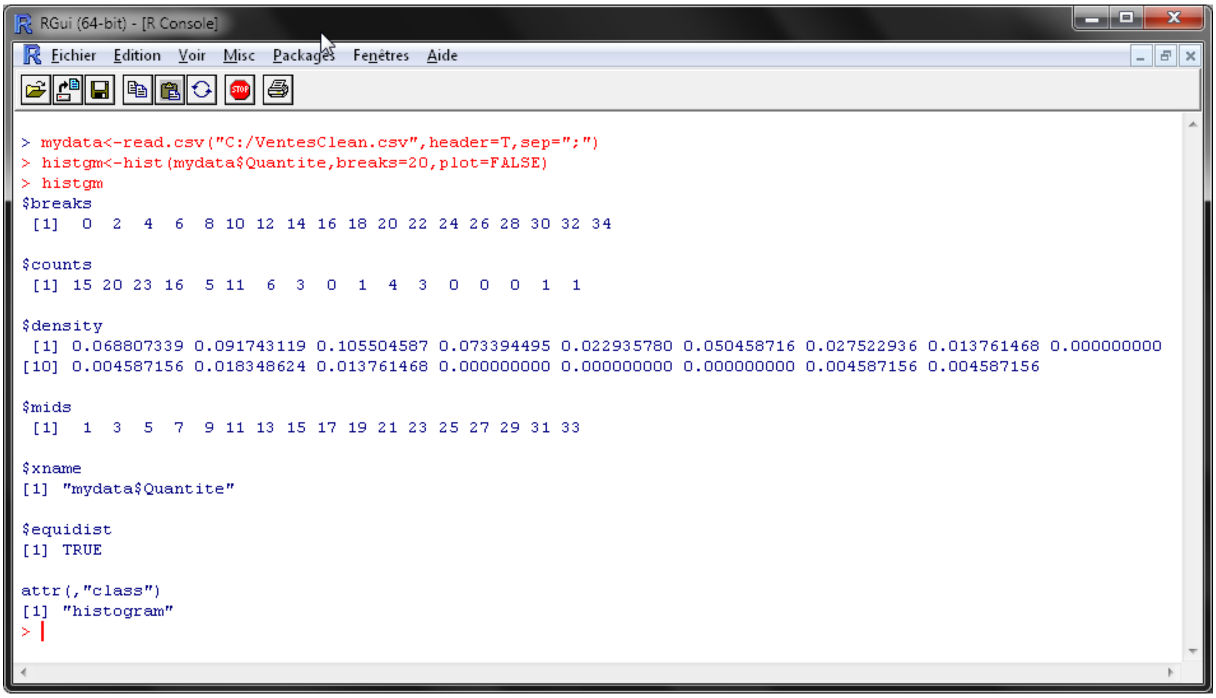


Figure 100 Graph de type histogramme avec couleurs différentes pour intervalles

Remarquez que nous pouvons obtenir les valeurs détaillées de l'histogramme qui sont implicitement contenues dans la commande **hist( )** :



Nous pouvons afficher aussi les quantiles en tant que lignes verticales en ajoutant le script suivant:

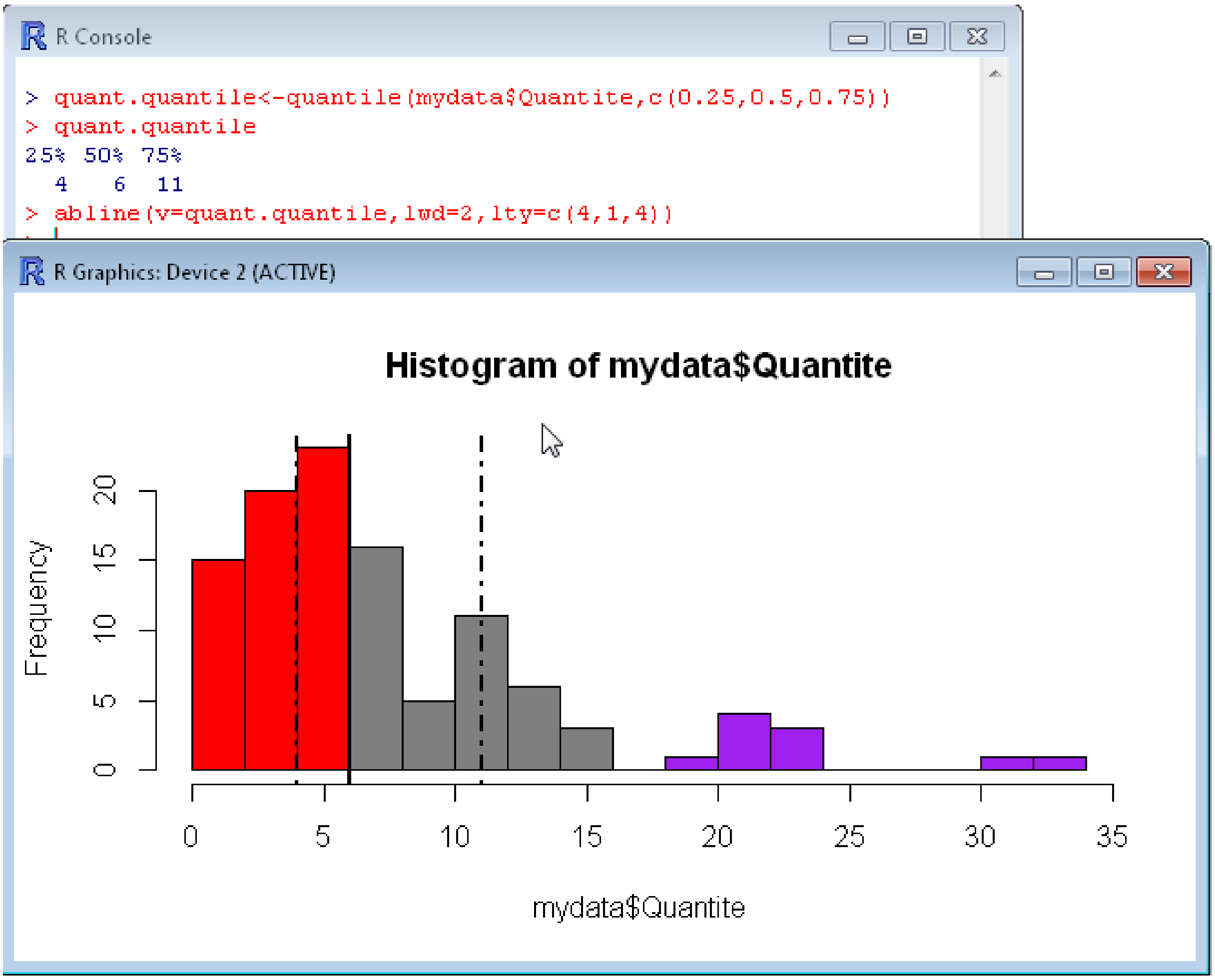


Diagramme à barres de type cascade

Au niveau des graphes de types barres en "cascade" ("waterfall chart" en anglais) je n'ai rien trouvé de mieux à ce jour que le package **waterfall** qui propose une fonction **waterfallchart( )** ayant cependant deux limitations majeures:

1) Elle ne différencie pas les variations positives et négatives par des couleurs

2) Le graphique ne conserve par l'ordre d'origine des données dans les vecteurs il faut donc tricher en précédent les noms des légendes par des numéros pour avoir l'ordre désiré

3) La fonction ajoute systématique une ligne *Total* qu'il n'est pas possible de supprimer ni de renommer

En reprenant le même exemple que dans la formation Microsoft Excel cela donne:



Figure 101 Graph Waterfall

A noter que la fonction **waterfallplot( )** à un paramètre *col* qui permet de changer la couleur de chaque barre que le souhaite (typiquement via un colonne codant les couleurs)!

On peut faire évidemment beaucoup mieux avec le package **waterfalls** et la fonction du même nom waterfalls( ):

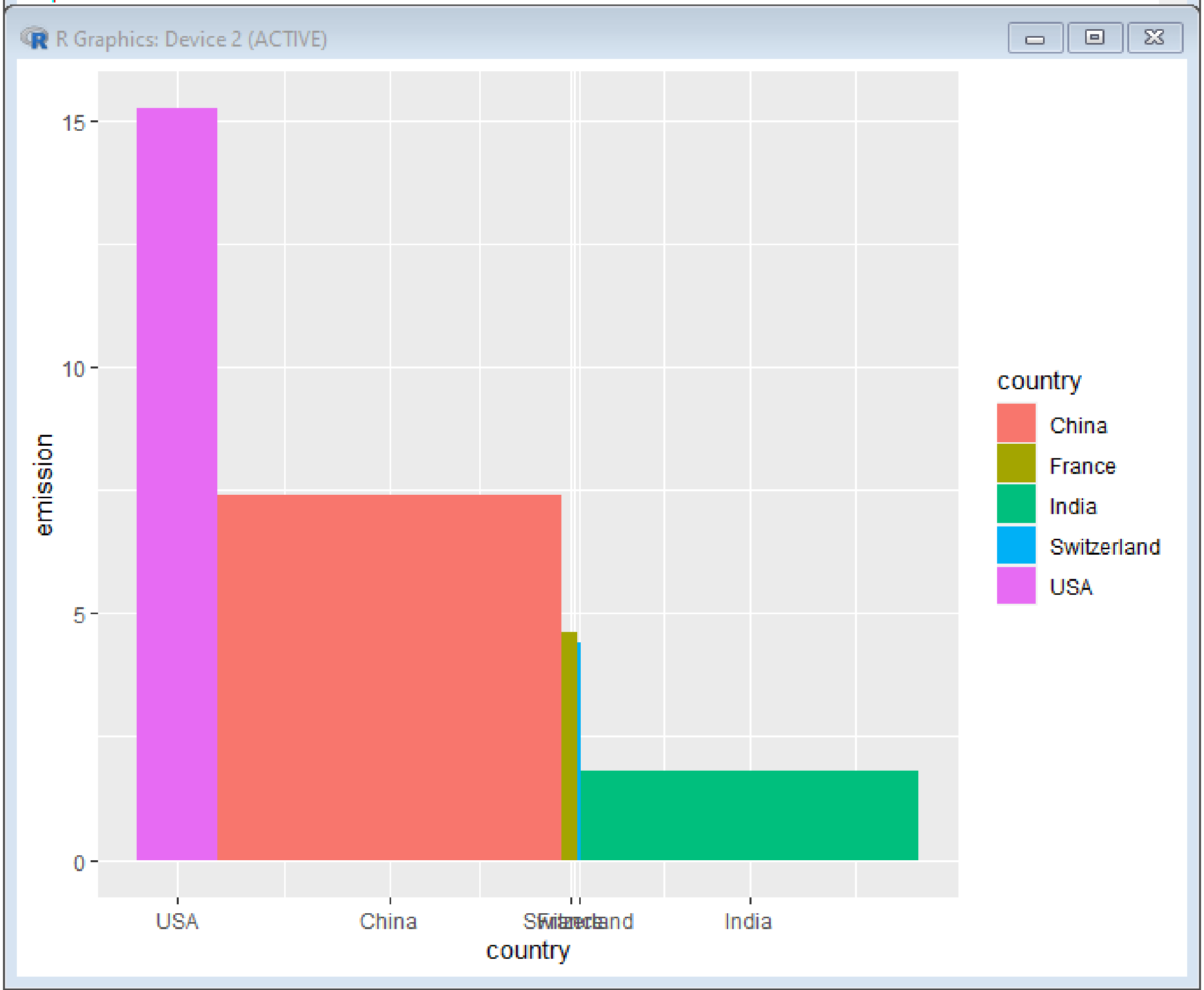


Diagrammes à barres mekko

R 4.1.1

Le diagramme à barre mekko à l'avantage de pouvoir représenter une variable supplémentaire dans un diagrammes à barres. Voyons le cas fameux de l'émission de CO_2 par habitant pour quelques pays où si on ne ne représentait pas la population des pays alors le diagramme à barres serait forcément biaisé grâce au package **mekko** et à sa fonction **barmekko( )**:

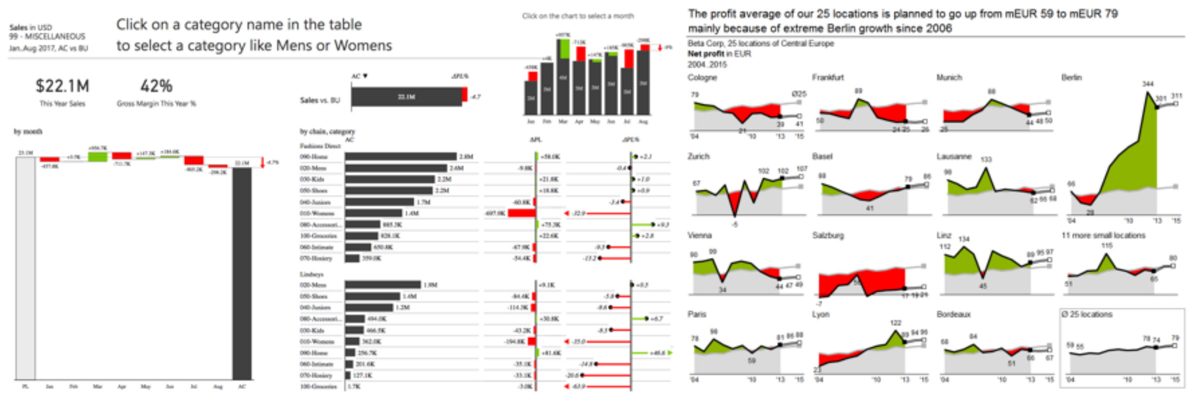
```
> library("ggplot2")
> library("mekko")
> CO2emission <- data.frame(
+   country = c('USA', 'China', 'France', 'Switzerland', 'India'),
+   emission = c(15.24, 7.41, 4.62, 4.4, 1.8),
+   population = c(329.5, 1400, 67.39, 8.6, 1380)
+   )
>
> barmekko(CO2emission , country, emission , population )
>
```



Diagrammes IBCS (International Business Communication System)

R 4.1.1

L'IBCS est un standard mondial de représentation visuelle de comptabilité et gestion financière. Voici un exemple (pris de Microsoft Power BI/Zebra) de ce quoi à quoi peut ressemble ce standard:



Nous allons voir que le package **tidycharts** permet dans un moindre mesure de créer quelques graphiques de l'IBCS (un peu est mieux que rien…).

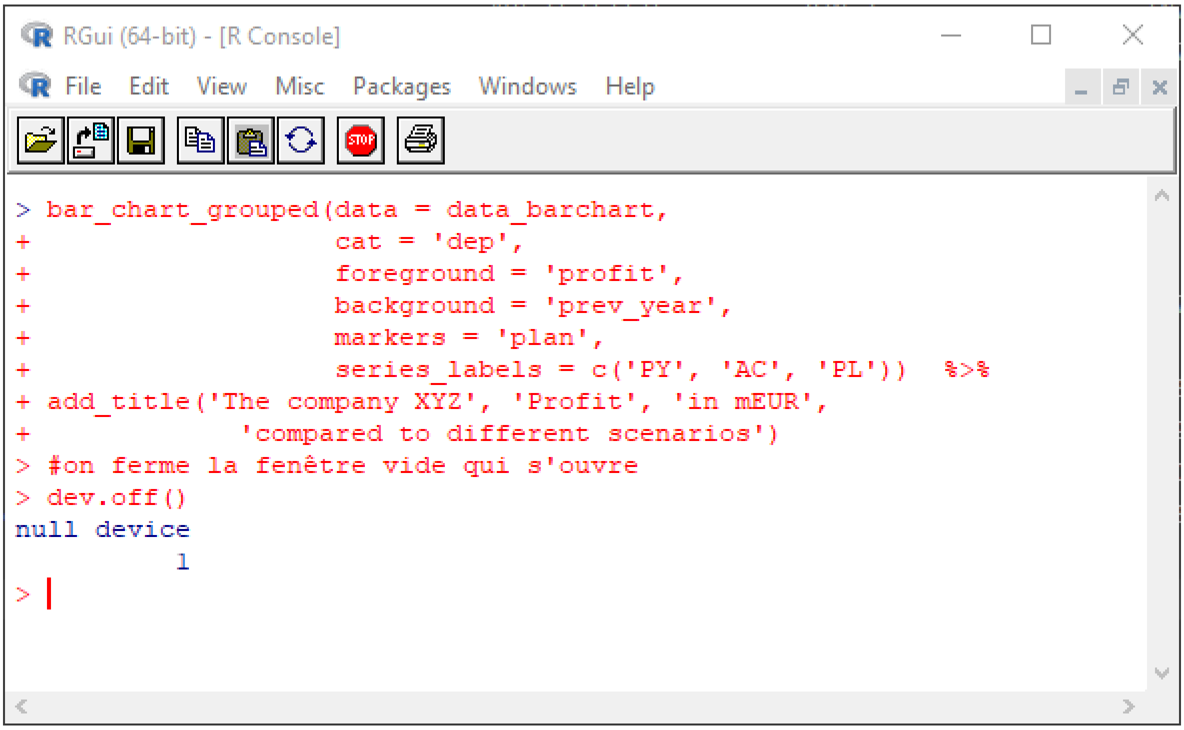
Les exemples donnés ci-dessous sont simplement pris de la vignette disponible ici:

https://cran.r-project.org/web/packages/tidycharts/vignettes/Getting_Started.html

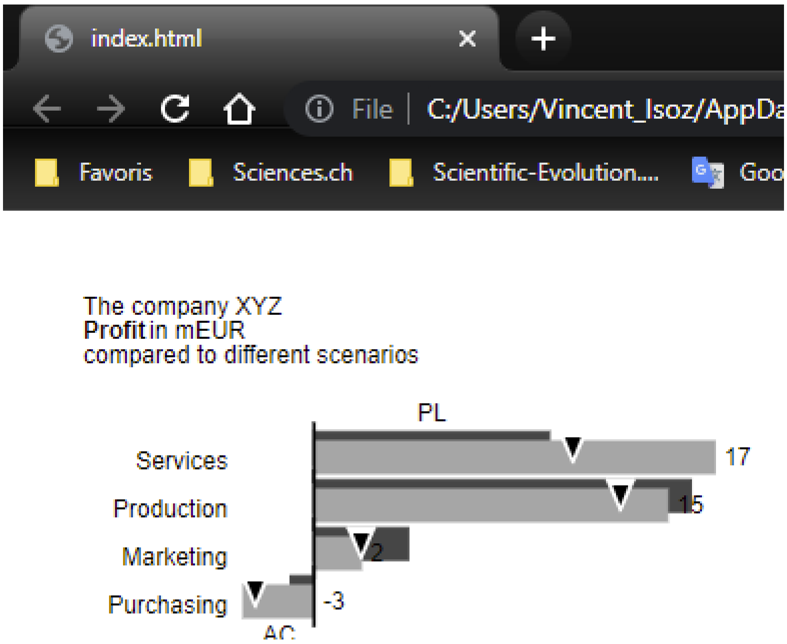
Nous créons donc un jeu de données à la volée:

```
> library("tidycharts")
> (data_barchart <- data.frame(
+   dep = c('Services', 'Production', 'Marketing', 'Purchasing'),
+   profit = c(17, 15, 2, -3),
+   operational = c(9, 7, 1.5, -0.4),
+   property = c(4, 4, 0.5, -0.6),
+   bonus = c(4, 4, 0, -2),
+   prev_year = c(10, 16, 4, -1),
+   plan = c(11, 13, 2, -2.5)
+ ))
         dep profit operational property bonus prev_year plan
1   Services     17         9.0      4.0     4        10 11.0
2 Production     15         7.0      4.0     4        16 13.0
3  Marketing      2         1.5      0.5     0         4  2.0
4 Purchasing     -3        -0.4     -0.6    -2        -1 -2.5
>
```

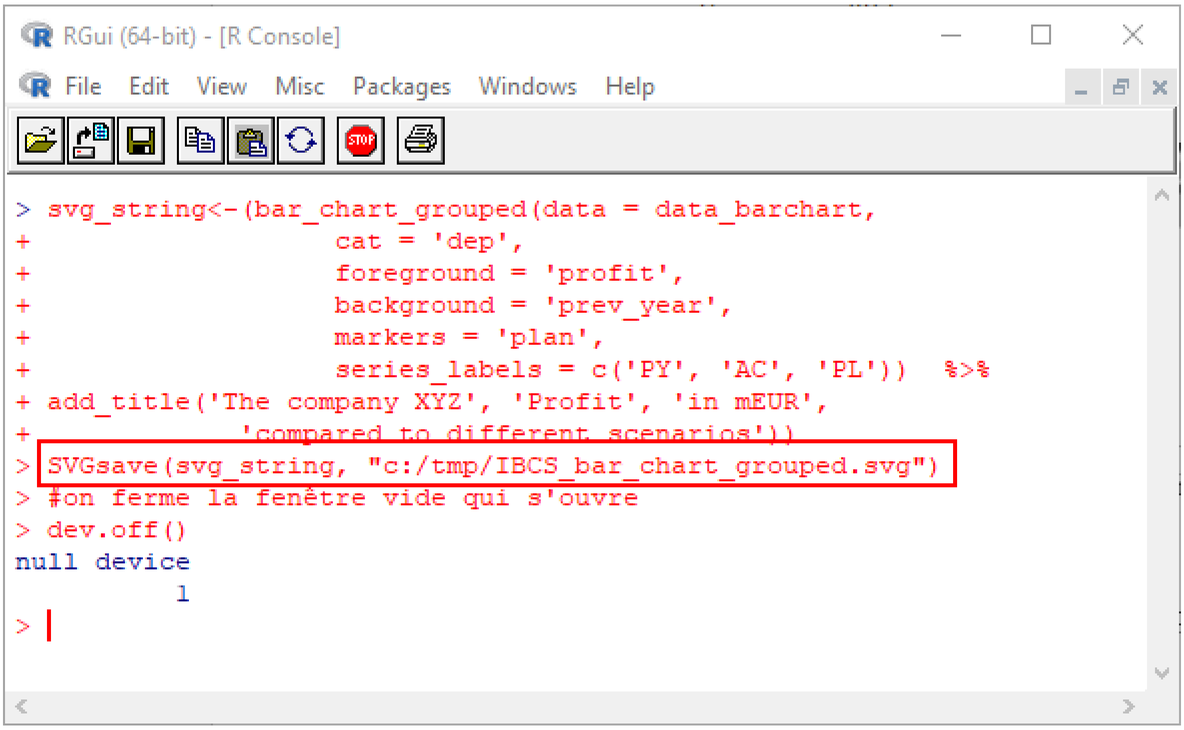
Création un graphique IBCS à barres groupées:



Ce qui donne dans le navigateur:



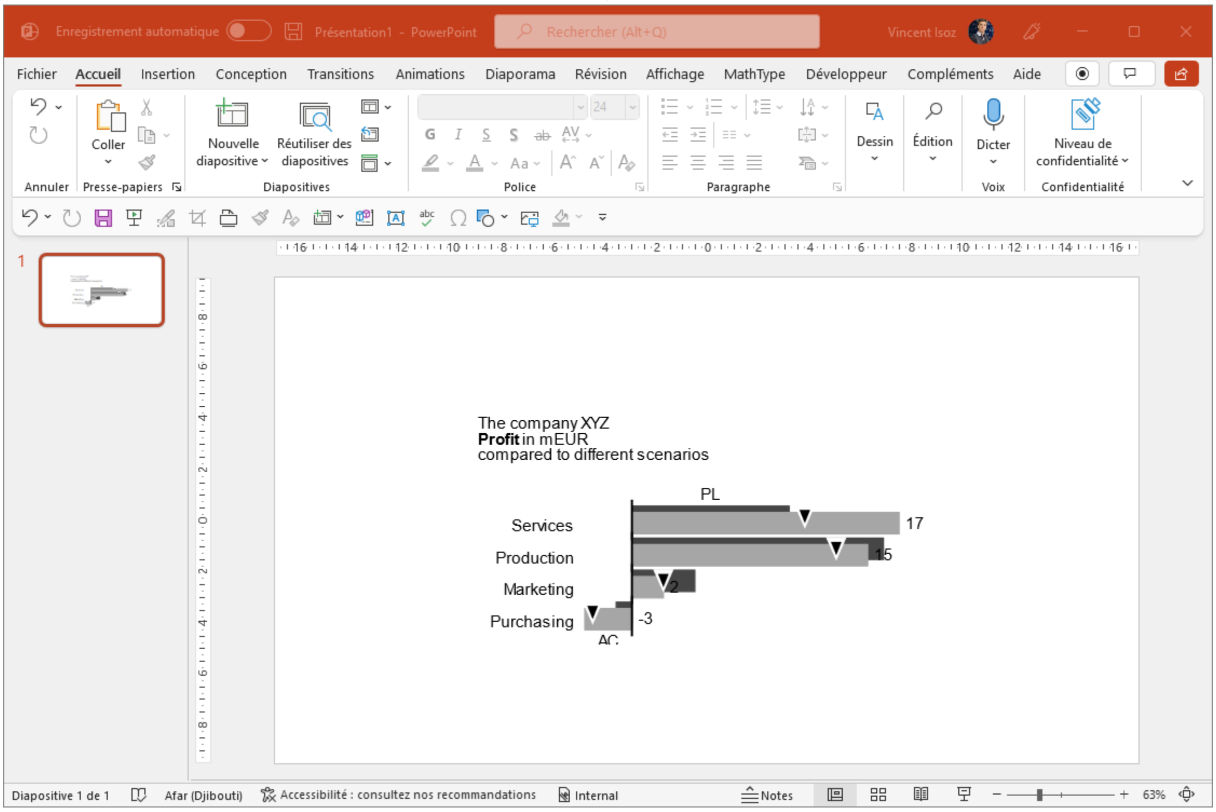
Le fait que ce soit dans le navigateur n'est pas idéal. Voyons donc comment créer un export avec une commande adaptée se trouvant dans le package **tydicharts**:



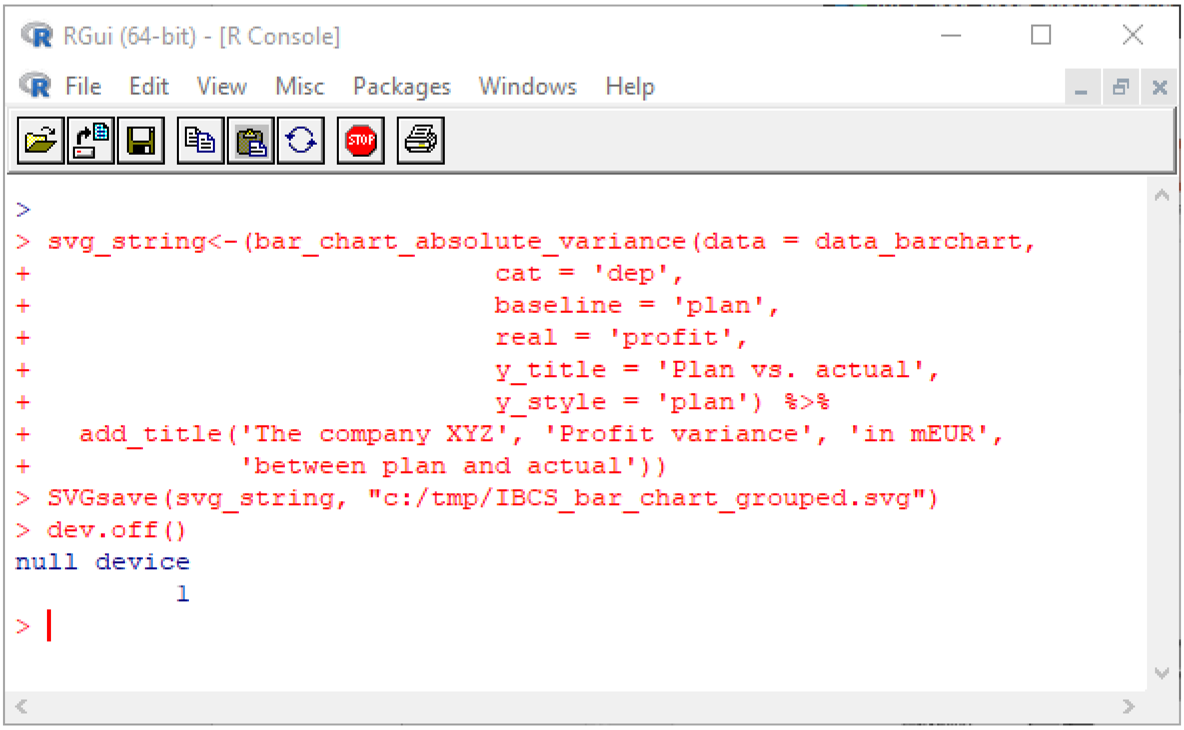
Et un fichier svg qu'on peut par exemple insérer dans Microsoft PowerPoint:



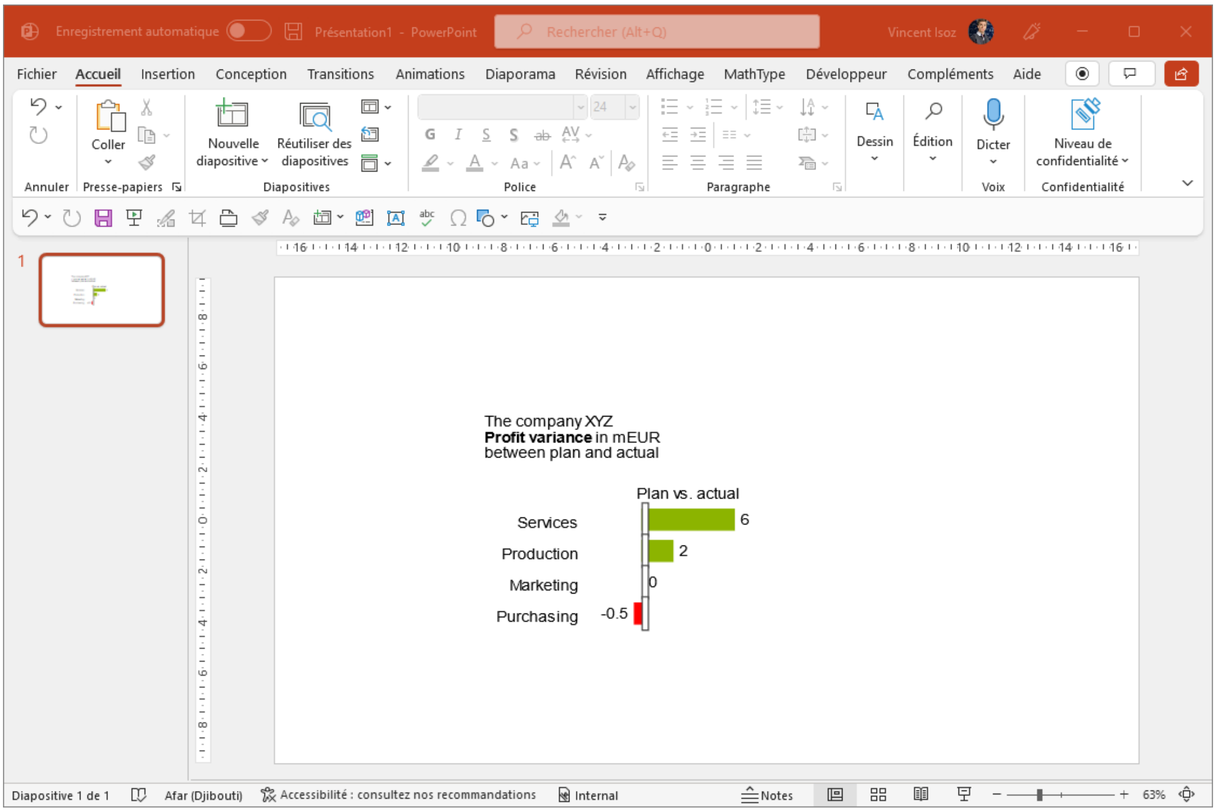
Ce qui donnera dans PowerPoint:



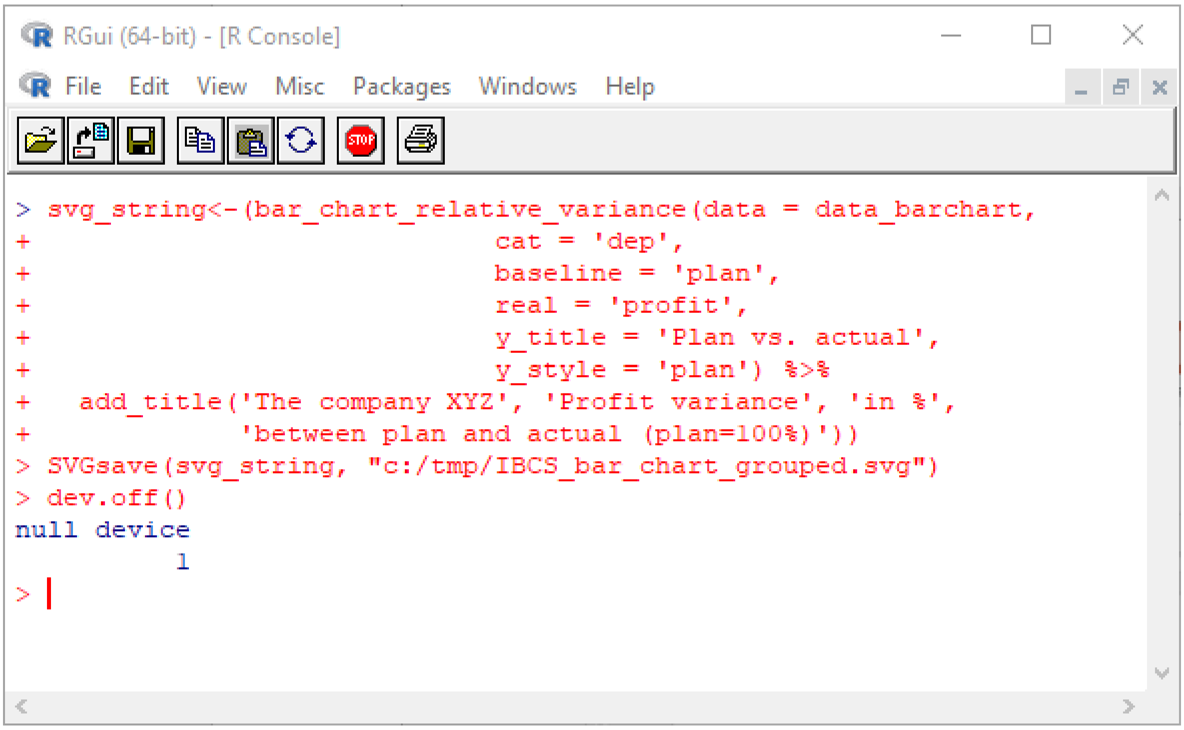
Ou:



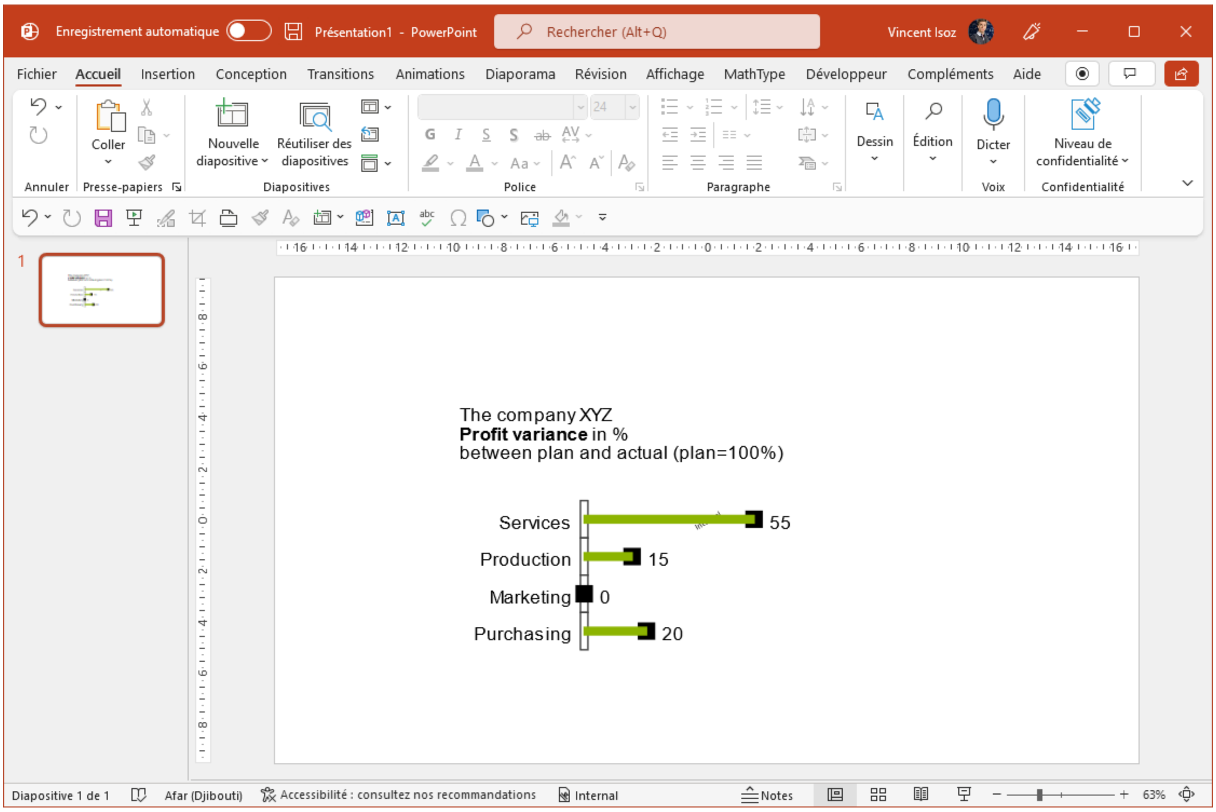
Ce qui donnera:



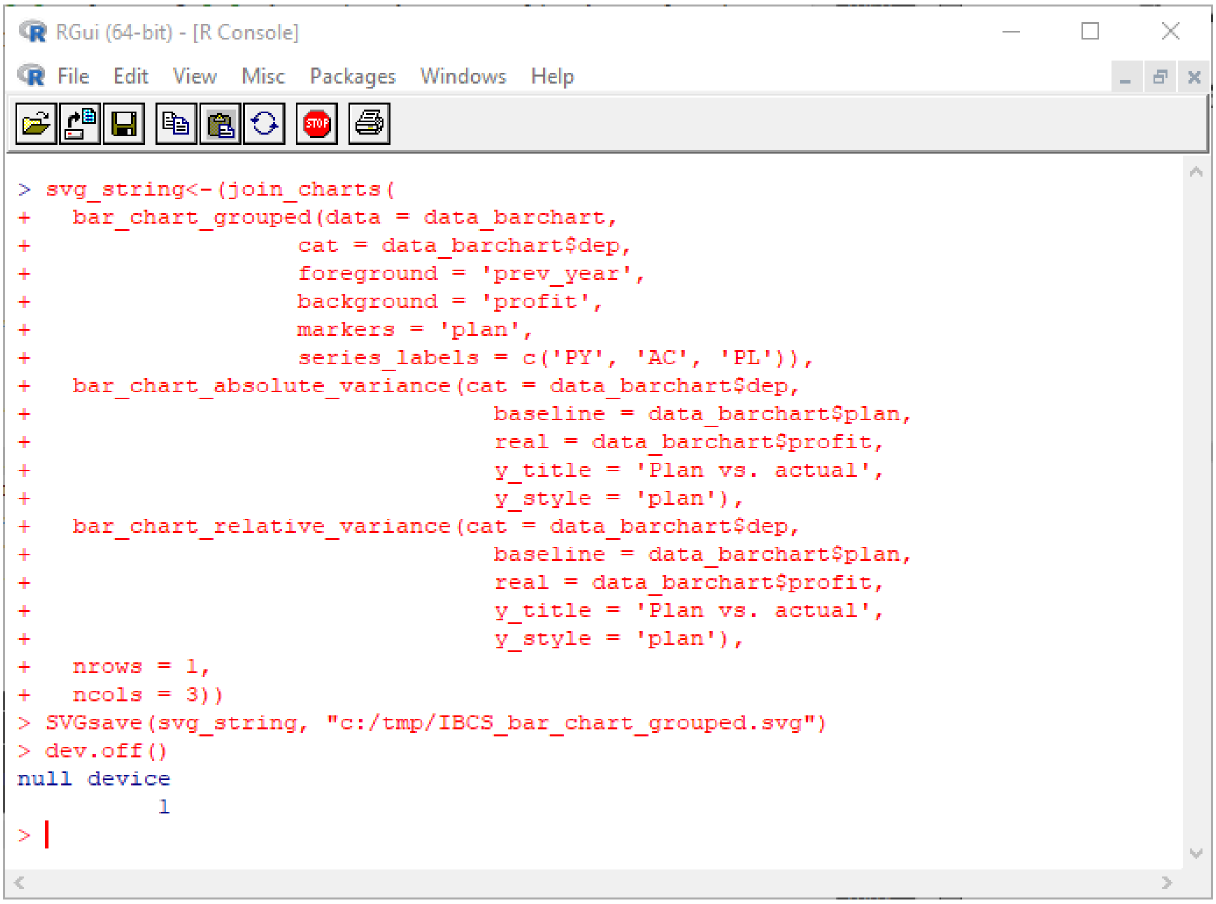
Ou encore:



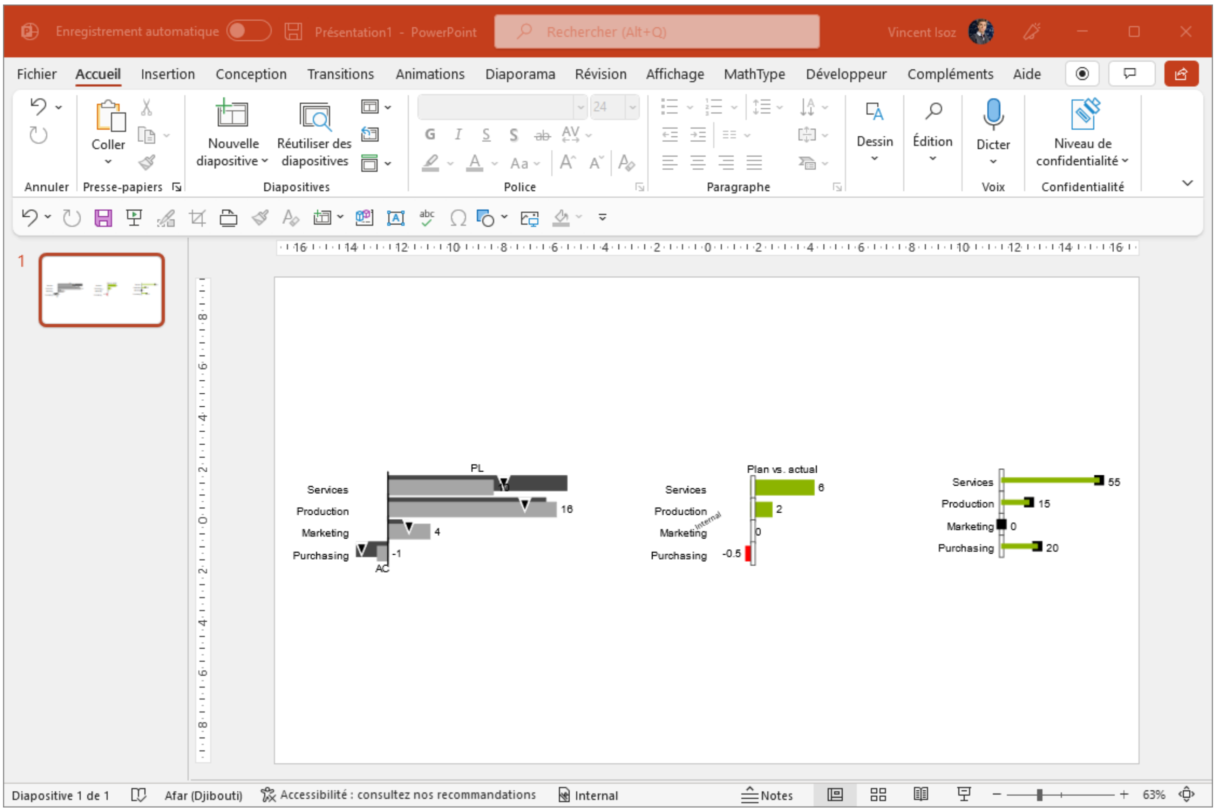
Ce qui donnera:



On peut compiler plusieurs graphs ensembles:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> svg_string<-(join_charts(
+   bar_chart_grouped(data = data_barchart,
+   cat = data_barchart$dep,
+   foreground = 'prev_year',
+   background = 'profit',
+   markers = 'plan',
+   series_labels = c('PY', 'AC', 'PL')),
+   bar_chart_absolute_variance(cat = data_barchart$dep,
+   baseline = data_barchart$plan,
+   real = data_barchart$profit,
+   y_title = 'Plan vs. actual',
+   y_style = 'plan'),
+   bar_chart_relative_variance(cat = data_barchart$dep,
+   baseline = data_barchart$plan,
+   real = data_barchart$profit,
+   y_title = 'Plan vs. actual',
+   y_style = 'plan'),
+   nrows = 1,
+   ncols = 3))
> SVGsave(svg_string, "c:/tmp/IBCS_bar_chart_grouped.svg")
> dev.off()
null device
          1
>

Ce qui donne:

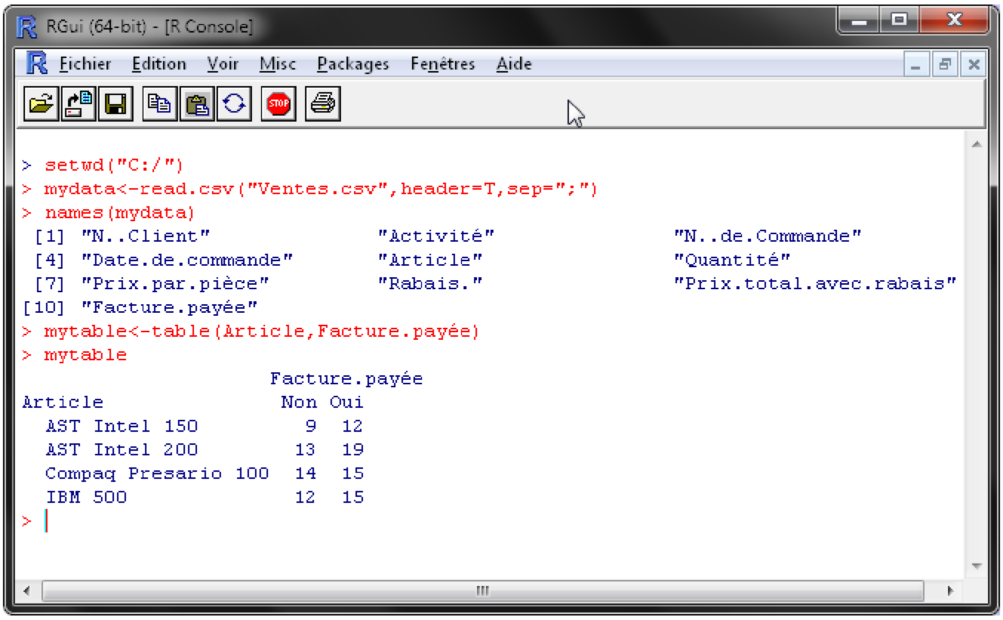

Présentation1 - PowerPoint
Fichier Accueil Insertion Conception Transitions Animations Diaporama Révision Affichage MathType Développeur Compléments Aide
PL
Services
Production
Marketing
Purchasing
AC
16
4
-1
Plan vs. actual
6
2
0
-0.5
55
15
0
20
Diapositive 1 de 1

Diagrammes à barres 3D

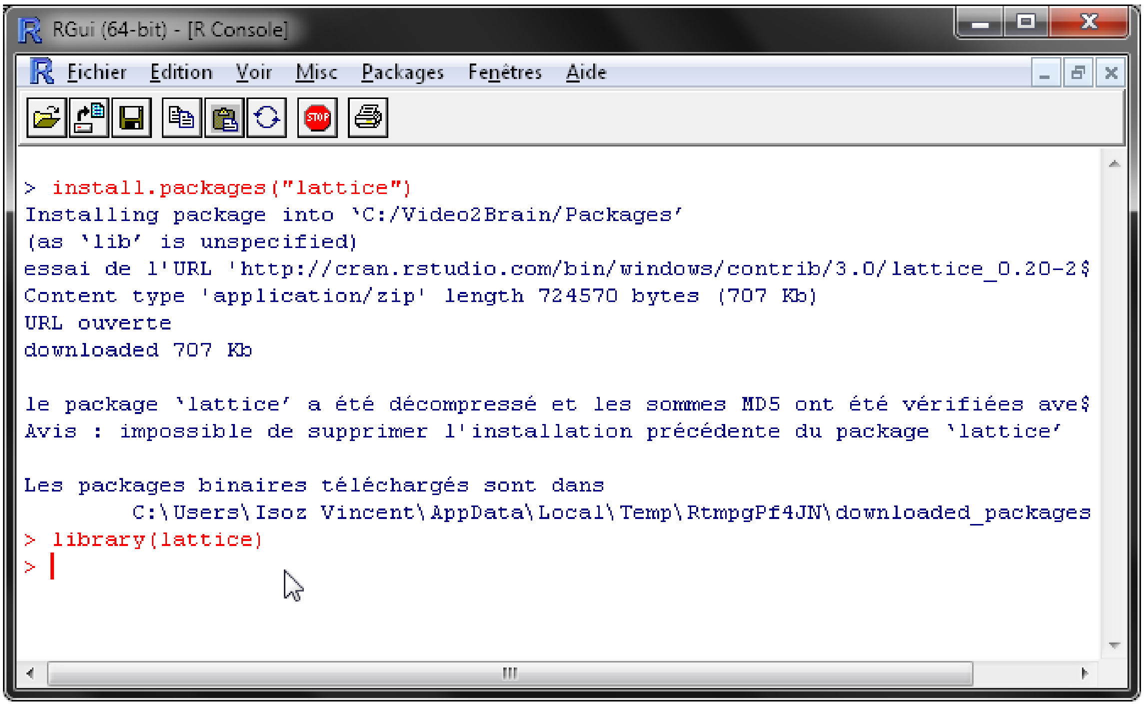
R 3.0.2

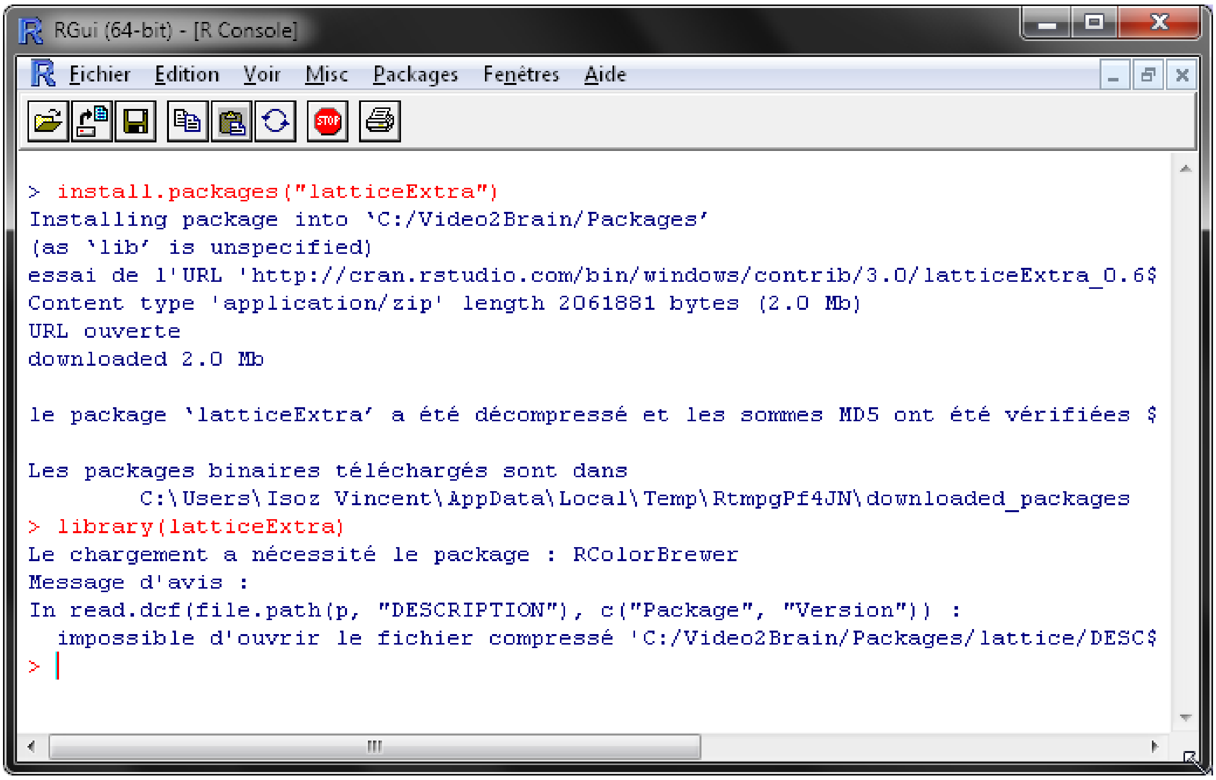
Un cas classique afin de voir si nous pouvons faire en gros la même chose que certains tableurs. Encore une fois c'est une excellente démonstration comme quoi dans R il faut appliquer des recettes de cuisine car la syntaxe est impossible à deviner intuitivement sans se référer longtemps à la documentation.

D'abord nous créons une table de contingence:



Nous installons ensuite deux packages qui sont **lattice** et **latticeExtra** (fallait deviner...):





et ensuite nous y allons joyeusement:

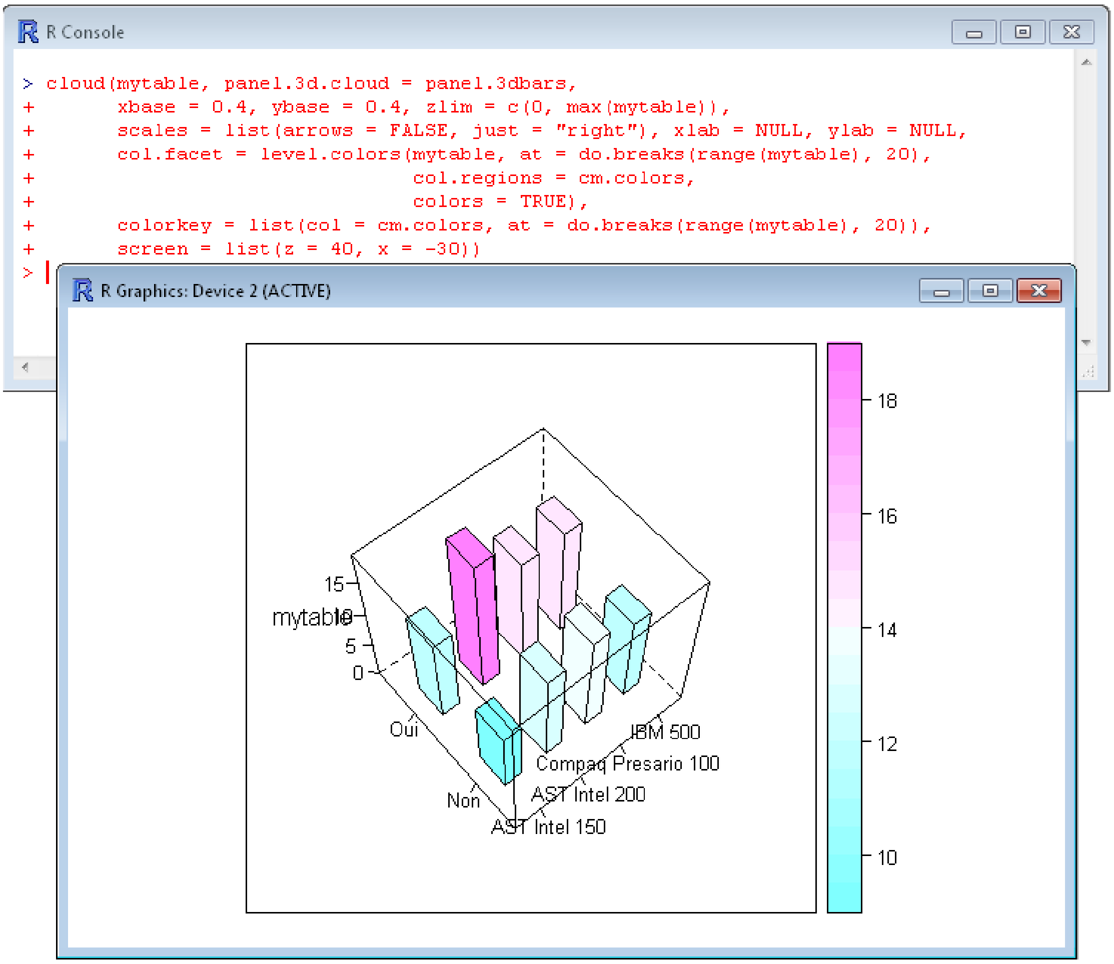
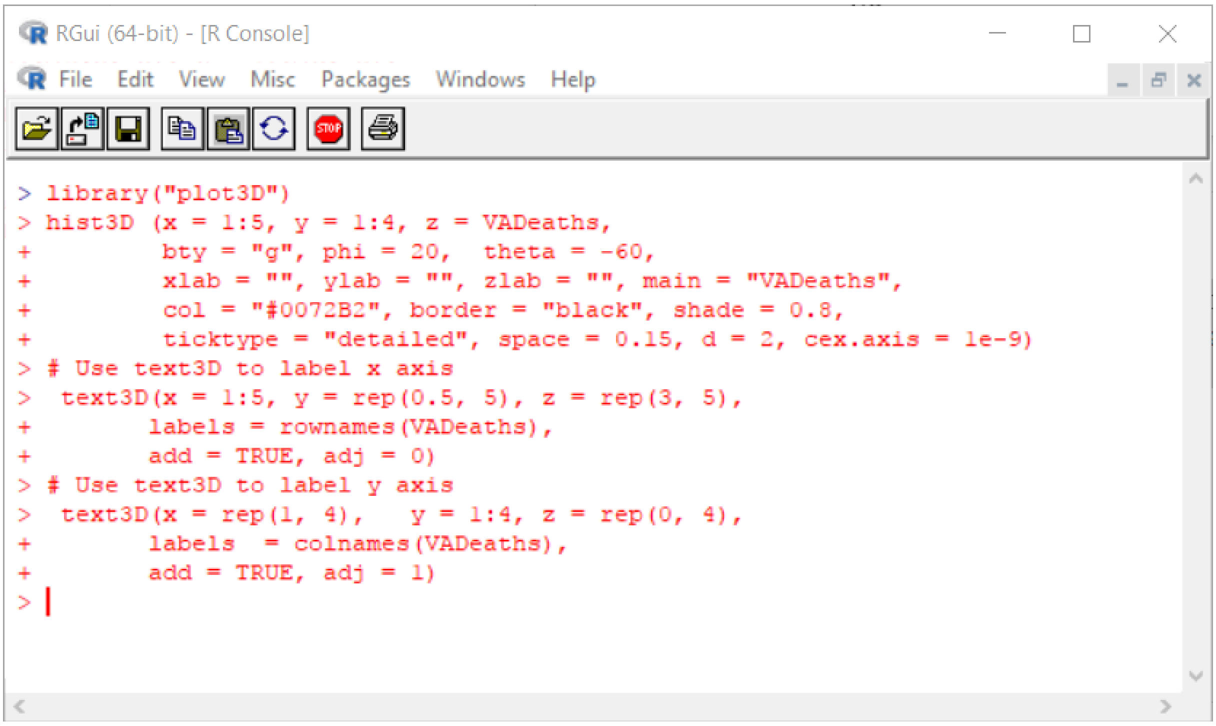
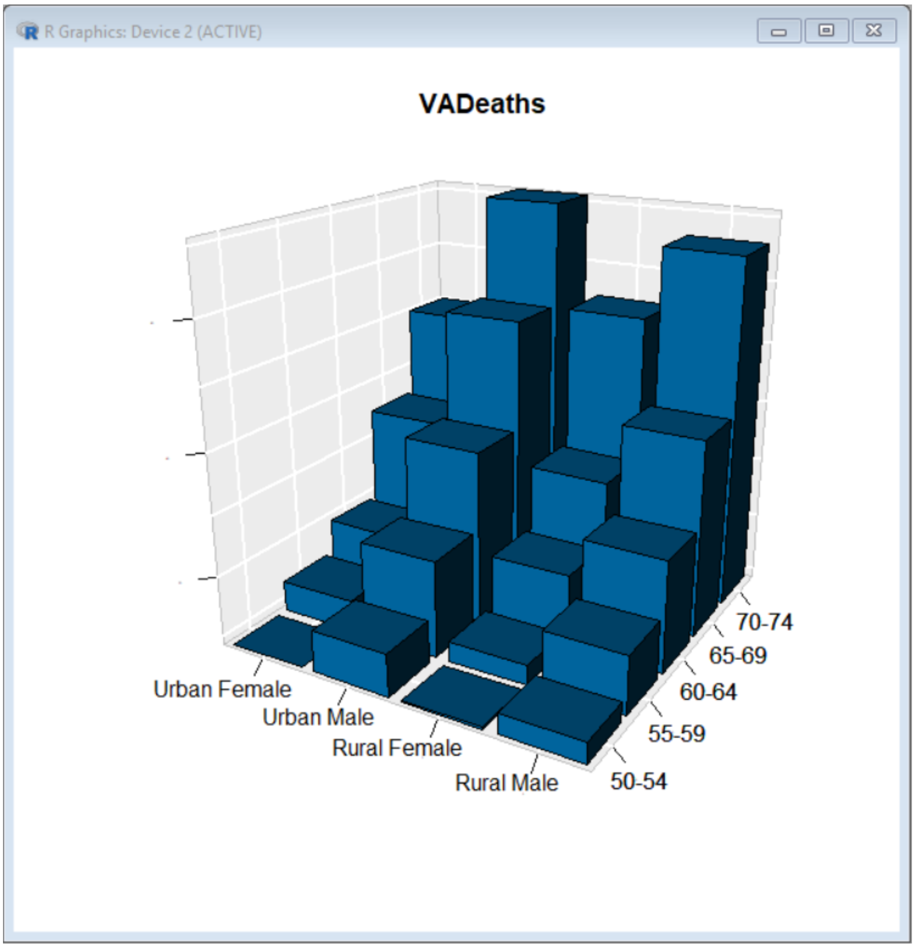


Figure 102 Graph à barres 3D

ou dans un tout autre style avec le package **plot3d** et la fonction **hist3d( )** :



Ce qui donne:



Exemple 121.: Diagrammes bullets (balles)

R 3.4.1

Les diagrammes bullets ("crées" par Stephen Few) sont des diagrammes à gauge (speedomètre/tachymètre) mais n'ayant pas subi de transformation polaires et dont la spécification complète est disponible ici:

http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf

Il n'existe pas à ce jour de package permettant des les créer mais voici un script R disponible sur Internet sous licence GPL reproduit ici tel quel[19]:

```
#----------------------------------------#
# Bullet Graph
#
# Bullet graphs have been devised by Stephen Few
# A complete specification of bullet graphs can be retrieved at:
# http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf
#
#   Author: Marco Torchiano
#           email: marco.torchiano@polito.it
#            twitter: @mtorchiano
#
#    Version: 1.1 (20 December 2014)
#
#    Copyright (c) 2014 by Marco Torchiano <marco.torchiano@polito.it>
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
#----------------------------------------#
#
# usage:
#       x: the value of the indicator to be plotted as the main bar
#     ref: the reference value to be plotted as a thick line
#  limits: the boundaries of the limits to be plotted as background
#             a four-element vector providing: base, 1st limit, 2nd limit,
upper limit.
#             The base and limit are used for defining the scale
#   width: the default width of the main indicator as a portion of the
graph width
#     col: the color of the indicator bar
# palette: the colors for the background limits
#
```

[19] http://softeng.polito.it/software/R/BulletGraph.R

```
bulletgraph.palette <- function(n,colored){
	if(colored){ # this colors are up to my personal taste ;-)
		if(n==2) cols = c("firebrick","dodgerblue")
		if(n==3) cols = c("firebrick","orange","olivedrab1")
		if(n==4) cols =
c("firebrick","orange","powderblue","olivedrab1")
		if(n==5) cols =
c("firebrick","orange","powderblue","plum","olivedrab1")
		if(n>5) cols = hsv((n:1 + n*0.3)/(n*1.3),.9,.8+(1:n/(n*5)))
	}else{ # these are the gray levels recommended by Stephen Few
		if(n==2) cols = hsv(0,0,c(.65,.9))
		if(n==3) cols = hsv(0,0,c(.6,.75,.9))
		if(n==4) cols = hsv(0,0,c(.5,.65,.8,.9))
		if(n==5) cols = hsv(0,0,c(.5,.65,.8,.9,.97))
		if(n>5) cols = hsv(1,0,seq(.4,.97,length.out=n))
	}
	return(cols)
}

bulletgraph <- function(x,ref,limits, name=NULL, subname="", width=0.4,
col=NULL,
					palette=NULL,colored=T){
	if(length(limits)<3){
		stop("limits must be a vector with at least three elements")
	}
	if(length(x)!=1){
		stop(paste("x must be a scalar",name))
	}
	limits=sort(limits)
	if(x<limits[1] | x>limits[4]){
		stop("x must be within outer limits")
	}
	if(ref<limits[1] | ref>limits[4]){
		stop("x must be within outer limits")
	}
	if(width<.01 | width>1){
		stop("width must be in the range [0 .. 1]")
	}
	if(is.null(name)) name = sys.call()[[2]]
	if(is.null(palette)){
		palette = bulletgraph.palette(length(limits)-1,colored)
	}
	if(is.null(col)){
		if(colored) col="steelblue3"
		else col = "black"
	}
	n = length(limits)
	ranges = matrix(tail(limits,-1)-c(0,head(tail(limits,-1),n-2)),n-1)
	barplot(ranges,col=palette,border=NA,horiz=T,
			xlim=c(min(limits),max(limits)),xpd=F)
	segments(ref,.3,ref,1.1,lwd=3)
	barplot(x[1],width= width,names.arg=name,cex.names=1,
			space=((1-width)/2+0.2)/width,
			add=T,horiz=T,border=NA,col=col,las=1,xpd=F)
	mtext(subname,side=2,line=1,at=0.4,cex=0.6,adj=1,las=2)
	if(limits[1]!=0){
		warning("Bars should be drawn on a zero-based scale: using a
jagged base to remark such non conformance.")
		jit = (limits[4] - limits[1])/100
		x = c(rep(c(limits[1],limits[1]+jit),6),0)
```

```
            y = c(2:13/10,1.2)
            polygon(x,y,col="white",border="white")
      }
}
```

Avec l'exemple correspondant à exécuter:

```
par(mfrow=c(3,1))
par(mar=c(2,9,.1,1))
# reproducing Stephen Few example in the specification document
bulletgraph(x=270,ref=260,limits=c(0,200,250,300),
name= "Revenue 2005 YTD",subname="(U.S. $ in thousands)",colored=F)
# colored version of the above
bulletgraph(x=270,ref=260,limits=c(0,200,250,300),
name= "Revenue 2005 YTD",subname="(U.S. $ in thousands)")

# an example with non-zero basis (no recommended)
x = 0.75
ref=0.83
limits=c(0.4,.6,.9,1)
bulletgraph(x,ref,limits,width=0.6)
```

Ce qui donne:

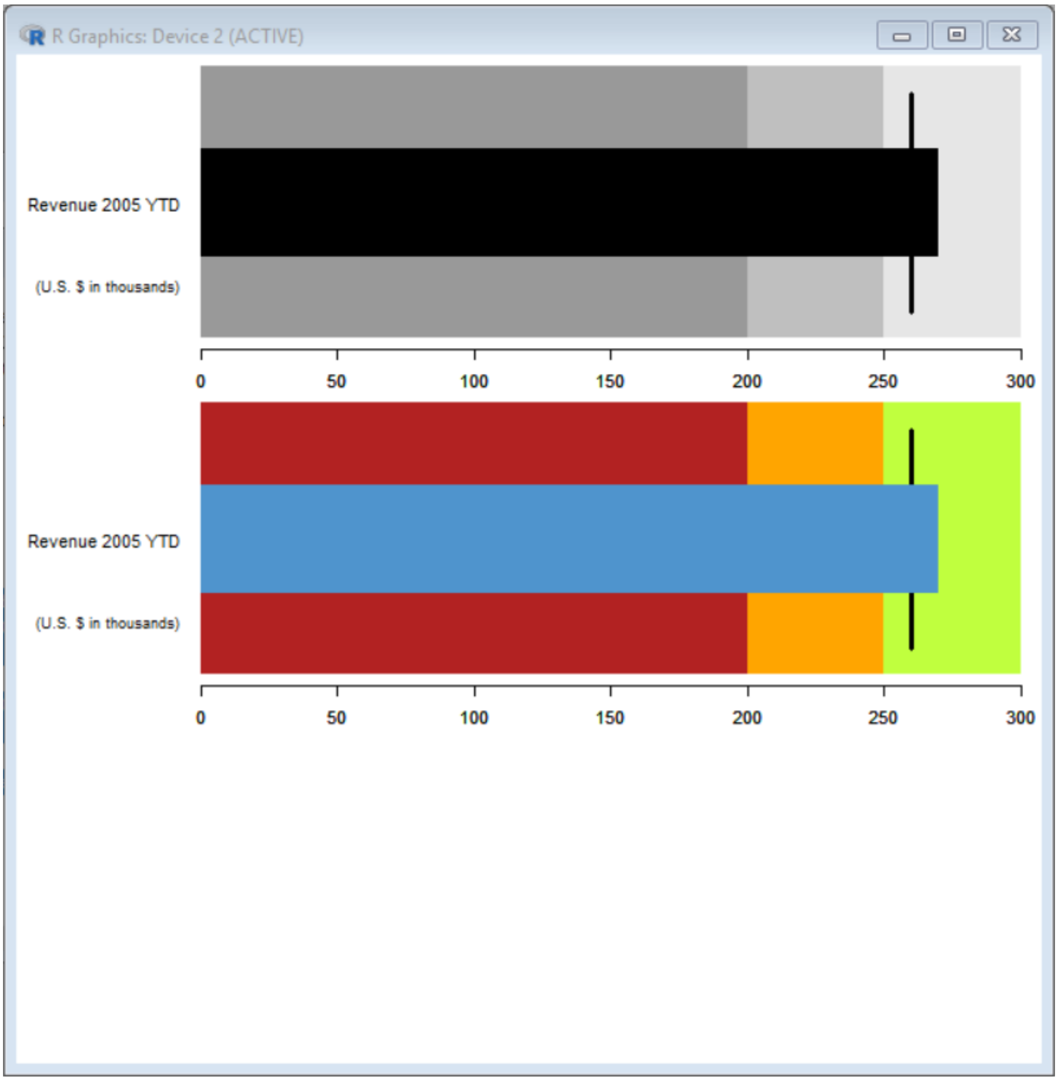


Figure 103 Graph Bullet (munition)

Exemple 122.: Diagrammes mosaïque à densité pour tables

R 3.2.1

Nous allons voir ici une représentation utilisée beaucoup dans le domaine du management (risques) et dans le domaine prédictif du data mining et utilisant le package **ggplot** et particulièrement la commande **scale_fill_gradient( )** .

Nous partons de notre jeu de données habituel mais résumé à l'aide de la fonction **group_by( )** du package **dplyr**:

```
> library(ggplot2)
> library(grid)
> library(reshape2)
> library(dplyr) #pour le regroupement
> mydata<-read.csv("C:/tmp/VentesClean.csv",header=T,sep=";")
> countGroup <- group_by(mydata, Activite, Article)
> count <- summarise(countGroup, count=sum(Quantite));count
Source: local data frame [29 x 3]
Groups: Activite [?]

      Activite              Article count
        (fctr)               (fctr) (int)
1  Alimentaire        AST Intel 200    12
2  Alimentaire Compaq Presario 100    23
3  Alimentaire              IBM 500    11
4   Assurances        AST Intel 150    27
5   Assurances        AST Intel 200     4
6   Assurances Compaq Presario 100    26
7   Assurances              IBM 500    15
8      Banques        AST Intel 150    10
9      Banques Compaq Presario 100    32
10     Banques              IBM 500    32
..         ...                  ...   ...
> 
```

Après il suffit de faire le graphe:

```
> ggplot(count, aes(x = Article, y = Activite)) +
+         scale_fill_gradient(name="Counts", low="white", high="violet") +
+         theme(axis.title.y = element_blank()) +
+         ggtitle("Ventes par Activités et Articles") +
+         geom_tile(aes(fill = count))
>
```

Ce qui donne:

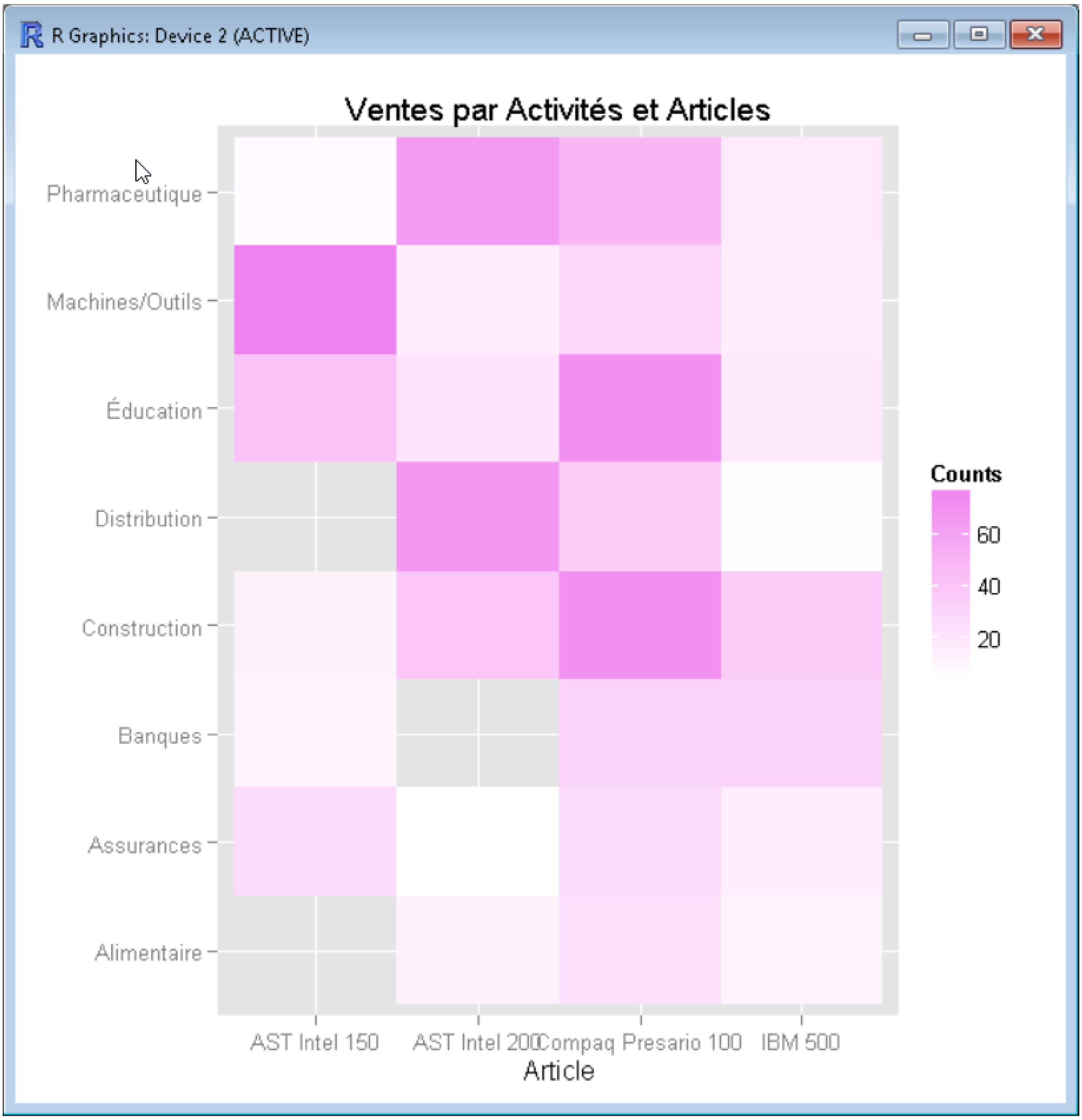
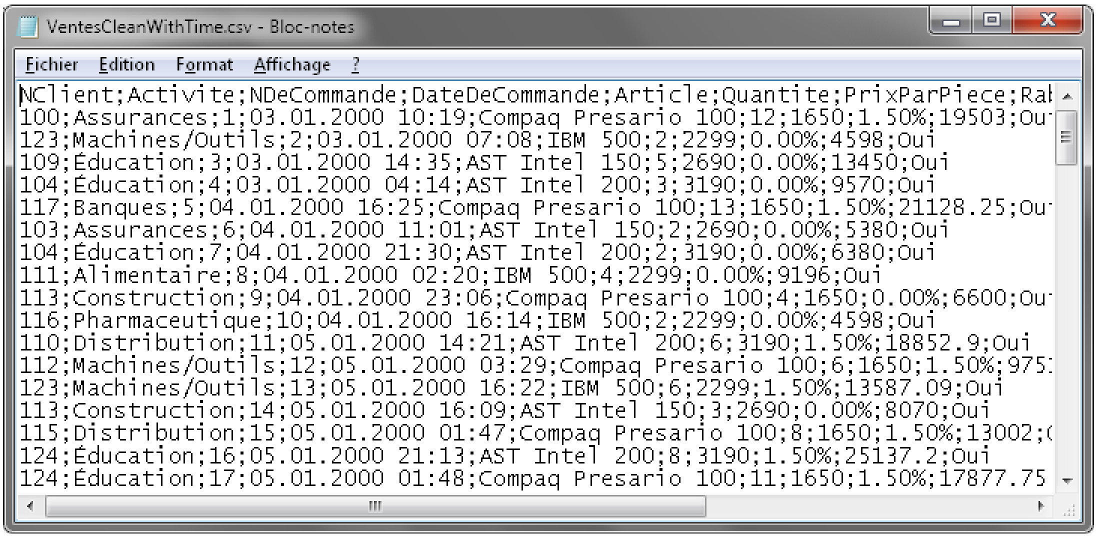


Figure 104 Graph à densité pour table de contingences (ggplot)

Exemple 123.: Histogramme (densité) avec des dates

R 3.0.2

Faisons maintenant un histogramme avec les dates (c'est plus difficile qu'avec de simples nombres) en utilisant le fichier *VentesCleanWithTime.csv* qui contient pour rappel:



```
VentesCleanWithTime.csv - Bloc-notes
Fichier Edition Format Affichage ?
NClient;Activite;NDeCommande;DateDeCommande;Article;Quantite;PrixParPiece;Ral
100;Assurances;1;03.01.2000 10:19;Compaq Presario 100;12;1650;1.50%;19503;Ou
123;Machines/Outils;2;03.01.2000 07:08;IBM 500;2;2299;0.00%;4598;Oui
109;Éducation;3;03.01.2000 14:35;AST Intel 150;5;2690;0.00%;13450;Oui
104;Éducation;4;03.01.2000 04:14;AST Intel 200;3;3190;0.00%;9570;Oui
117;Banques;5;04.01.2000 16:25;Compaq Presario 100;13;1650;1.50%;21128.25;Ou
103;Assurances;6;04.01.2000 11:01;AST Intel 150;2;2690;0.00%;5380;Oui
104;Éducation;7;04.01.2000 21:30;AST Intel 200;2;3190;0.00%;6380;Oui
111;Alimentaire;8;04.01.2000 02:20;IBM 500;4;2299;0.00%;9196;Oui
113;Construction;9;04.01.2000 23:06;Compaq Presario 100;4;1650;0.00%;6600;Ou
116;Pharmaceutique;10;04.01.2000 16:14;IBM 500;2;2299;0.00%;4598;Oui
110;Distribution;11;05.01.2000 14:21;AST Intel 200;6;3190;1.50%;18852.9;Oui
112;Machines/Outils;12;05.01.2000 03:29;Compaq Presario 100;6;1650;1.50%;975
123;Machines/Outils;13;05.01.2000 16:22;IBM 500;6;2299;1.50%;13587.09;Oui
113;Construction;14;05.01.2000 16:09;AST Intel 150;3;2690;0.00%;8070;Oui
115;Distribution;15;05.01.2000 01:47;Compaq Presario 100;8;1650;1.50%;13002;
124;Éducation;16;05.01.2000 21:13;AST Intel 200;8;3190;1.50%;25137.2;Oui
124;Éducation;17;05.01.2000 01:48;Compaq Presario 100;11;1650;1.50%;17877.75
```

Alors pour ce faire:

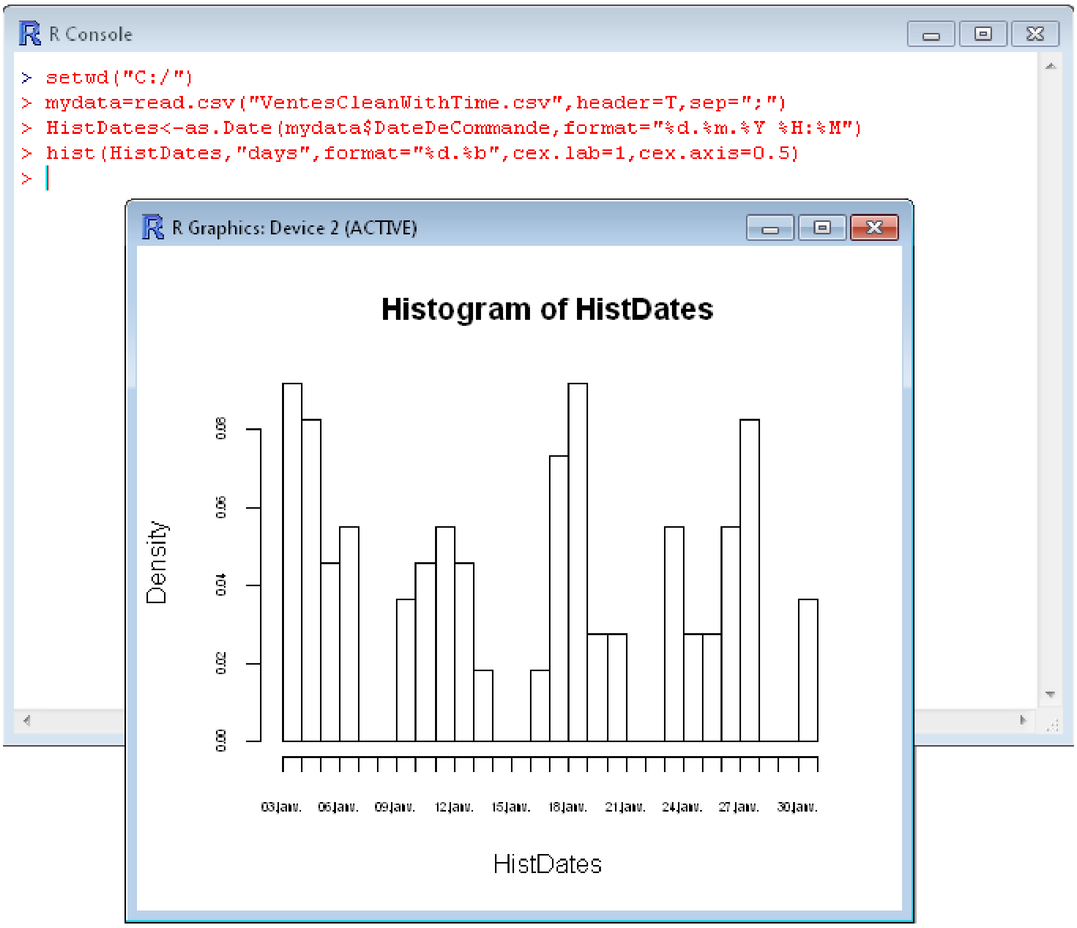


Figure 105 Graph à barres avec dates

Exemple 124.: Histogrammes superposés (fréquences)

R 3.0.2

Toujours dans les grands classiques, voyons comment faire de multiples histogrammes de distributions avec la commande **hist( )** et de plus superposés avec transparence:

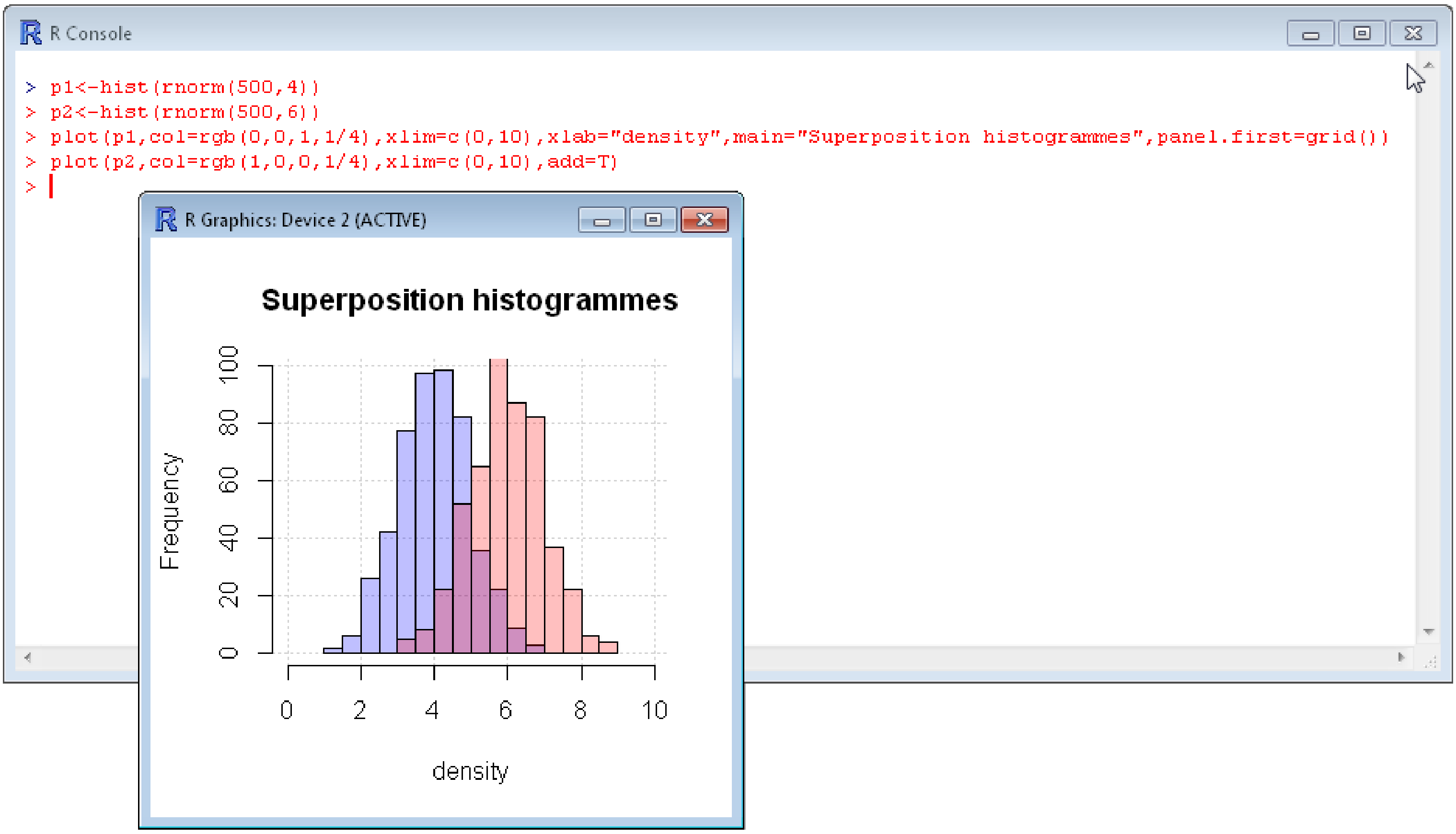
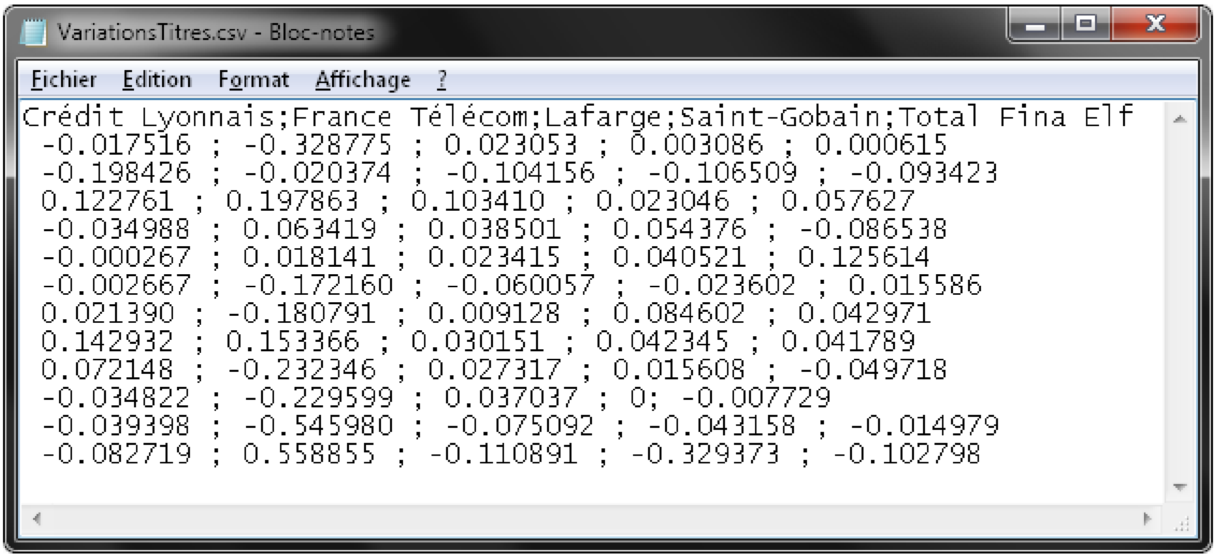


Figure 106 Graph histogrammes superposés (avec transparence)

Nous pouvons aussi nous amuser à combiner un plot avec des histogrammes. Considérons le jeu de données suivant:


VariationsTitres.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
Crédit Lyonnais;France Télécom;Lafarge;Saint-Gobain;Total Fina Elf
 -0.017516 ; -0.328775 ; 0.023053 ; 0.003086 ; 0.000615
 -0.198426 ; -0.020374 ; -0.104156 ; -0.106509 ; -0.093423
 0.122761 ; 0.197863 ; 0.103410 ; 0.023046 ; 0.057627
 -0.034988 ; 0.063419 ; 0.038501 ; 0.054376 ; -0.086538
 -0.000267 ; 0.018141 ; 0.023415 ; 0.040521 ; 0.125614
 -0.002667 ; -0.172160 ; -0.060057 ; -0.023602 ; 0.015586
 0.021390 ; -0.180791 ; 0.009128 ; 0.084602 ; 0.042971
 0.142932 ; 0.153366 ; 0.030151 ; 0.042345 ; 0.041789
 0.072148 ; -0.232346 ; 0.027317 ; 0.015608 ; -0.049718
 -0.034822 ; -0.229599 ; 0.037037 ; 0; -0.007729
 -0.039398 ; -0.545980 ; -0.075092 ; -0.043158 ; -0.014979
 -0.082719 ; 0.558855 ; -0.110891 ; -0.329373 ; -0.102798
```

En saisissant la série de commandes suivantes, nous obtenons:



```
R Console
> setwd("C:/")
> mydata<-read.csv("VariationsTitres.csv",header=T,sep=";")
> x<-mydata$Crédit.Lyonnais
> y<-mydata$France.Télécom
> xhist<-hist(x,plot=FALSE)
> yhist<-hist(y,plot=FALSE)
> top<-max(c(xhist$counts,yhist$counts))
> nf<-layout(matrix(c(2,0,1,3),2,2,byrow=TRUE),c(3,1),c(1,3),TRUE)
> par(mar=c(3,2,1,1))
> plot(x,y,xlab="",ylab="",col="red")
> grid()
> barplot(xhist$counts,axes=FALSE,ylim=c(0,top),space=0,col="light green")
> par(mar=c(3,0,1,1))
> barplot(yhist$counts,axes=FALSE,xlim=c(0,top),space=0,horiz=TRUE,col="light blue")
> |
```

ce qui donne (intéressant à comparer avec MATLAB...!):

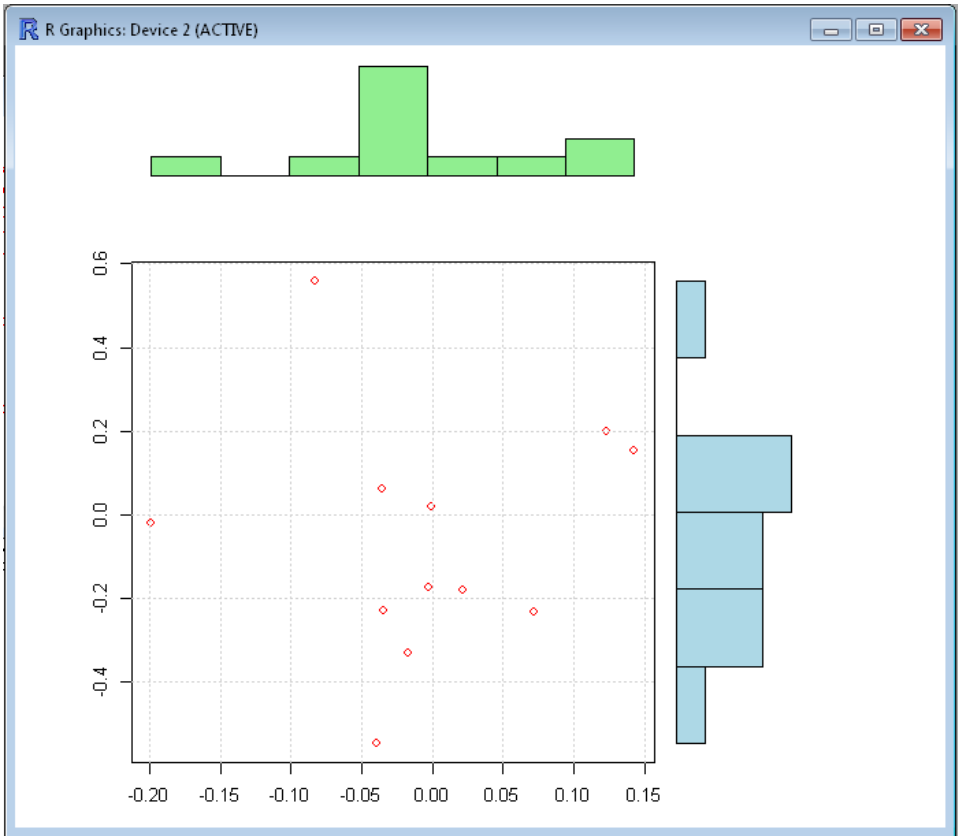


Figure 107 Graph X-Y avec histogrammes en marges

ou nous pouvons avoir une variante:

```
> setwd("C:/")
> mydata<-read.csv("VariationsTitres.csv",header=T,sep=";")
> x<-mydata$Crédit.Lyonnais
> y<-mydata$France.Télécom
> par(fig=c(0,0.9,0,0.9))
> plot(x,y,xlab="Crédit Lyonnais",ylab="Télécom",pch=18,col="blue")
> grid()
> par(fig=c(0,0.9,0.55,1),new=TRUE)
> boxplot(x,horizontal=TRUE,axes=FALSE,col="royal blue")
> par(fig=c(0.65,1,0,0.9),new=TRUE)
> boxplot(y,axes=FALSE,col="royal blue")
> 
```

ce qui donne:

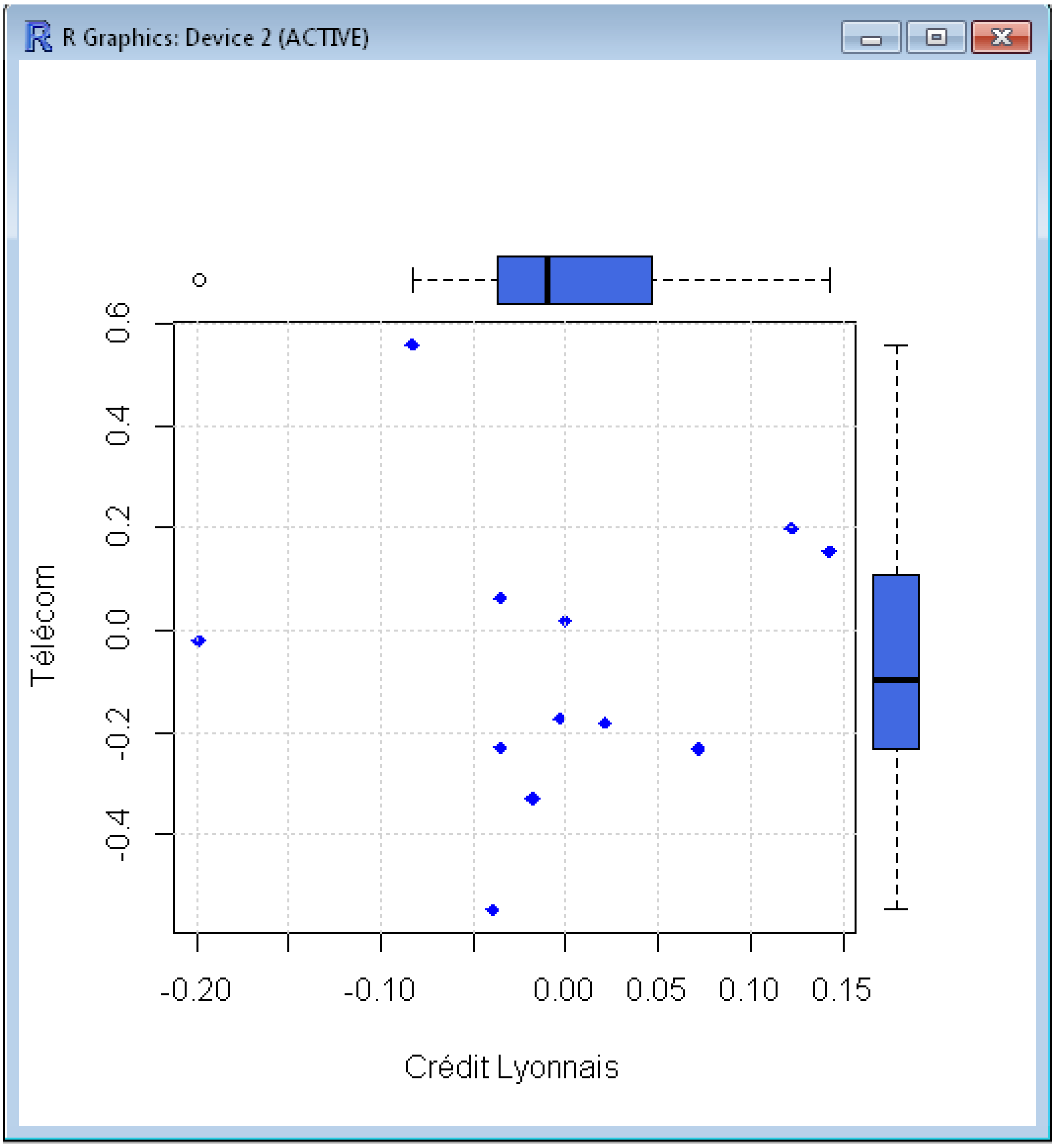


Figure 108 Graph X-Y avec Box-Whisker (boîtes à moustaches) en marges

ou encore une variante utilisant l'association de **rug( )** (densité marginale) et **jitter( )** :

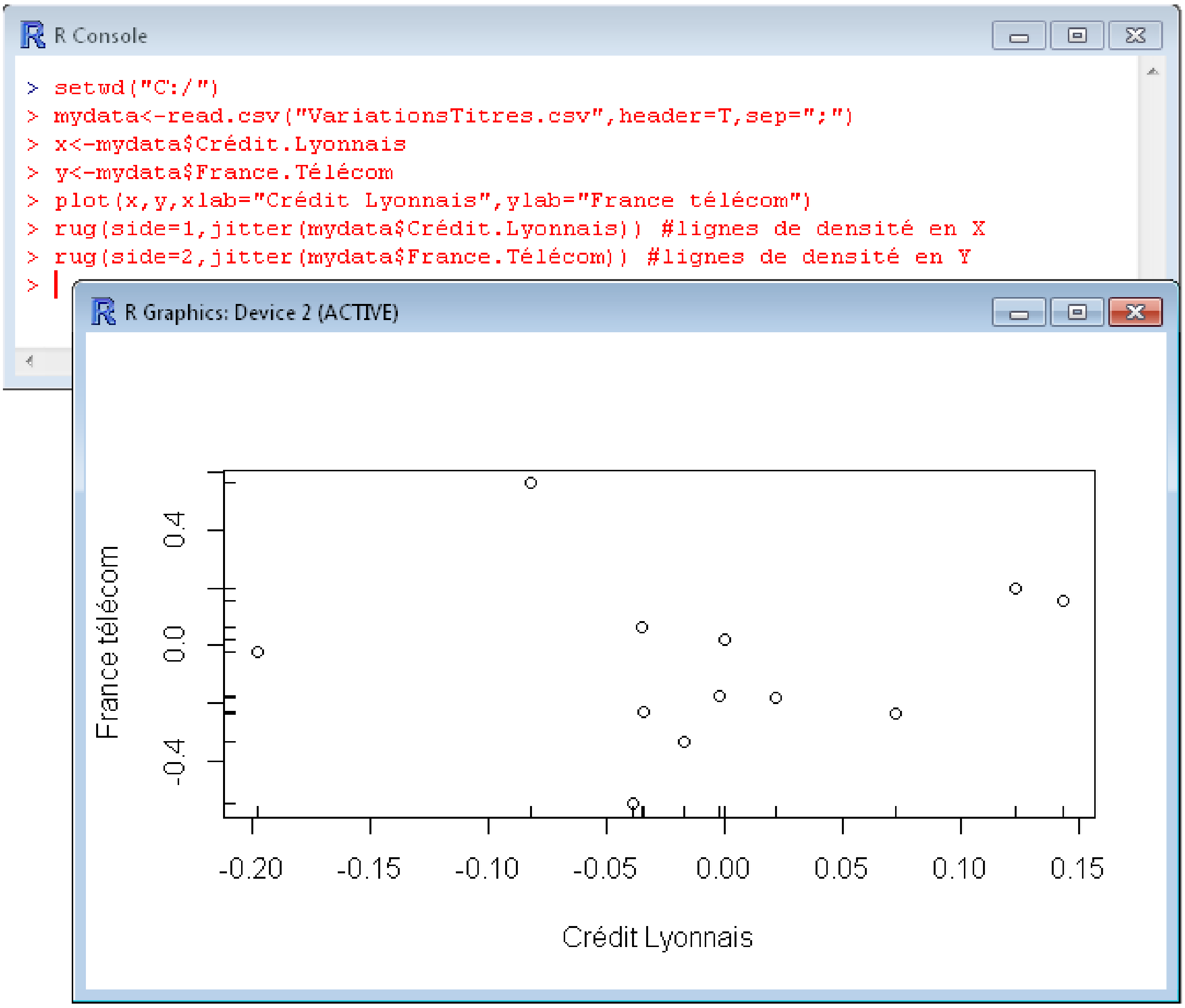


Figure 109 Graph X-Y avec jitter en marges

Exemple 125.: Diagrammes à aires

R 3.0.2

Un grand classique du monde des tableurs qui se passe de commentaires à part qu'à ce jour le package **ggplot2** avec la commande **geom_area( )** ne semblent pas gérer les variables catégorielles sur l'axe des abscisses (d'où le numéro des jours de la semaine au lieu de leur nom):

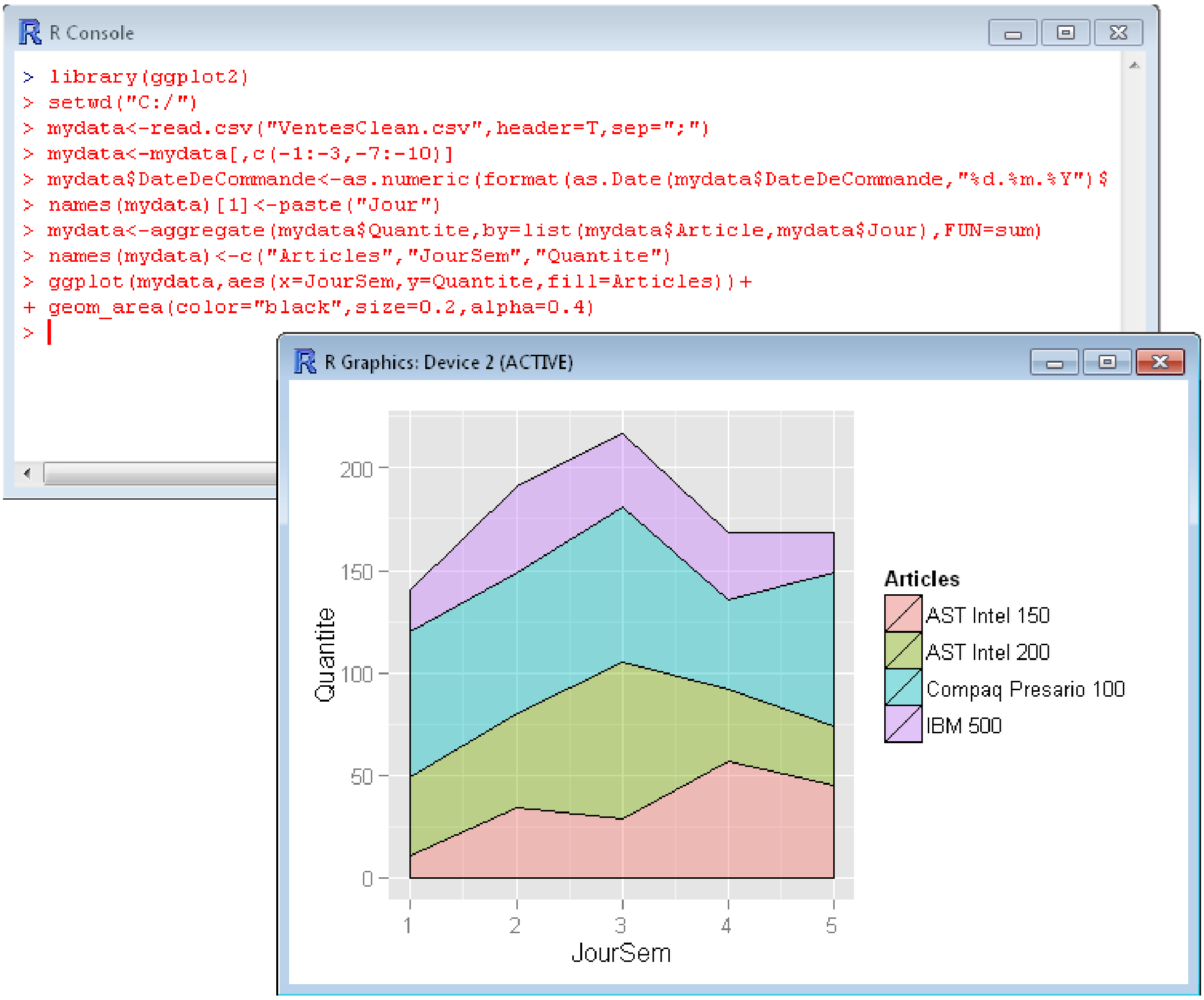


Figure 110 Graph à aires (ggplot)

Ou dans sa version normalisée en utilisant le package **plyr** avec la commande **ddply( )** :

```
R Console
> library(ggplot2);library(plyr)
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> mydata<-mydata[,c(-1:-3,-7:-10)]
> mydata$DateDeCommande<-as.numeric(format(as.Date(mydata$DateDeCommande,"%d.%m.%Y$
> names(mydata)[1]<-paste("Jour")
> mydata<-aggregate(mydata$Quantite,by=list(mydata$Article,mydata$Jour),FUN=sum)
> names(mydata)<-c("Articles","JourSem","Quantite")
> mydata_prop<-ddply(mydata,"JourSem",transform,Percent=Quantite/sum(Quantite)*100)
> mydata_prop
              Articles JourSem Quantite   Percent
1        AST Intel 150       1       11  7.857143
2        AST Intel 200       1       38 27.142857
3  Compaq Presario 100       1       71 50.714286
4              IBM 500       1       20 14.285714
5        AST Intel 150       2       34 17.801047
6        AST Intel 200       2       46 24.083770
7  Compaq Presario 100       2       69 36.125654
8              IBM 500       2       42 21.989529
9        AST Intel 150       3       29 13.364055
10       AST Intel 200       3       76 35.023041
11 Compaq Presario 100       3       76 35.023041
12             IBM 500       3       36 16.589862
13       AST Intel 150       4       57 33.928571
14       AST Intel 200       4       35 20.833333
15 Compaq Presario 100       4       44 26.190476
16             IBM 500       4       32 19.047619
17       AST Intel 150       5       45 26.785714
18       AST Intel 200       5       29 17.261905
19 Compaq Presario 100       5       75 44.642857
20             IBM 500       5       19 11.309524
> ggplot(mydata_prop,aes(x=JourSem,y=Percent,fill=Articles))+
+ geom_area(color="black",size=0.2,alpha=0.4)
> 
```

Ce qui donne:

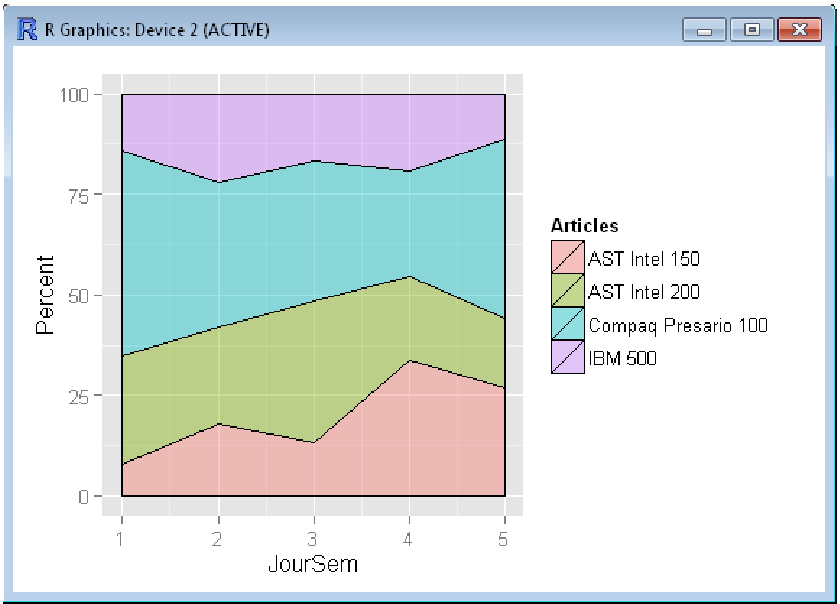
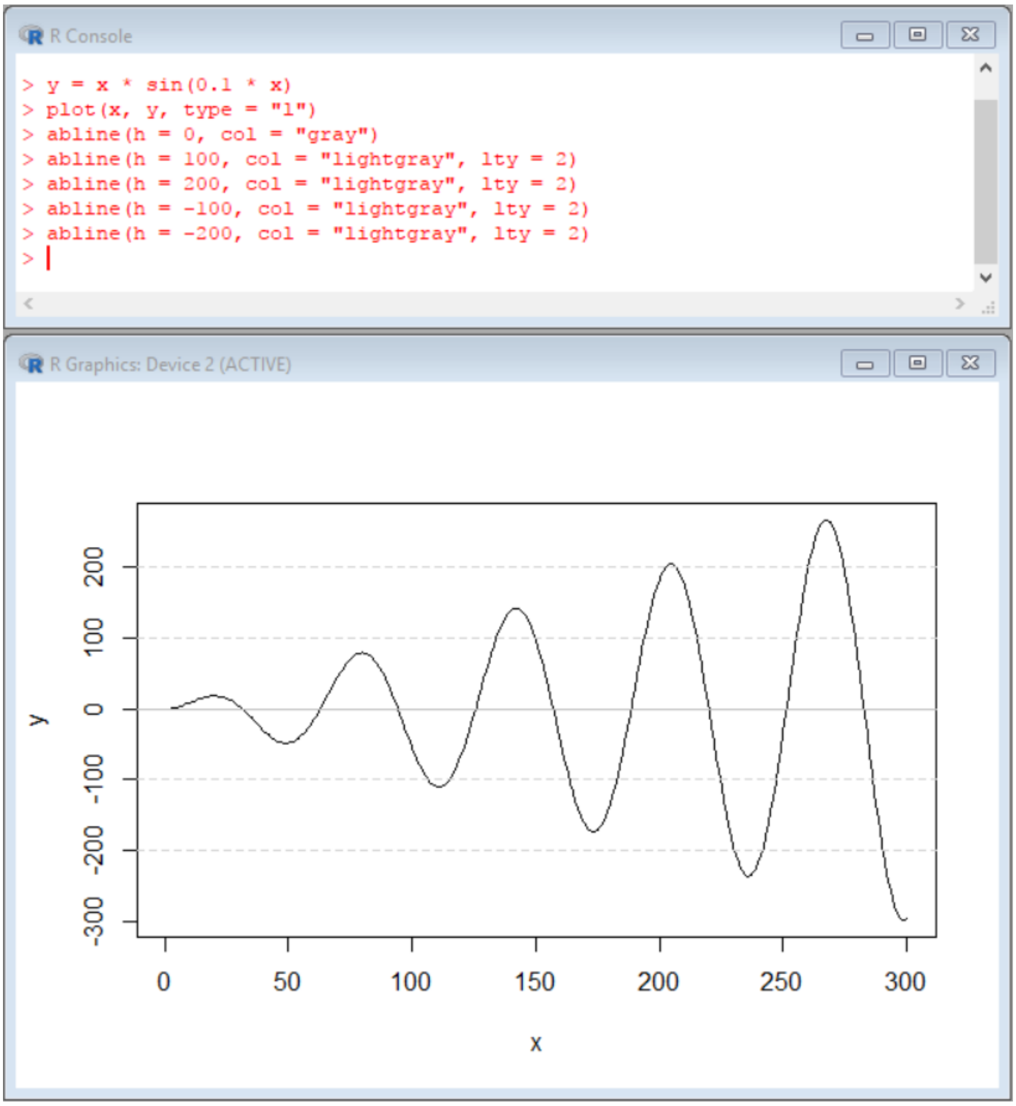


Figure 111 Graph à aires normalisé (ggplot)

Voyons aussi comment faire des "diagrammes horizons". Pour comprendre de quoi il s'agit, considérez d'abord le simple graphique suivant:



Avec la fonction **horizonplot( )** du package **extraLattice** le graphique devient:

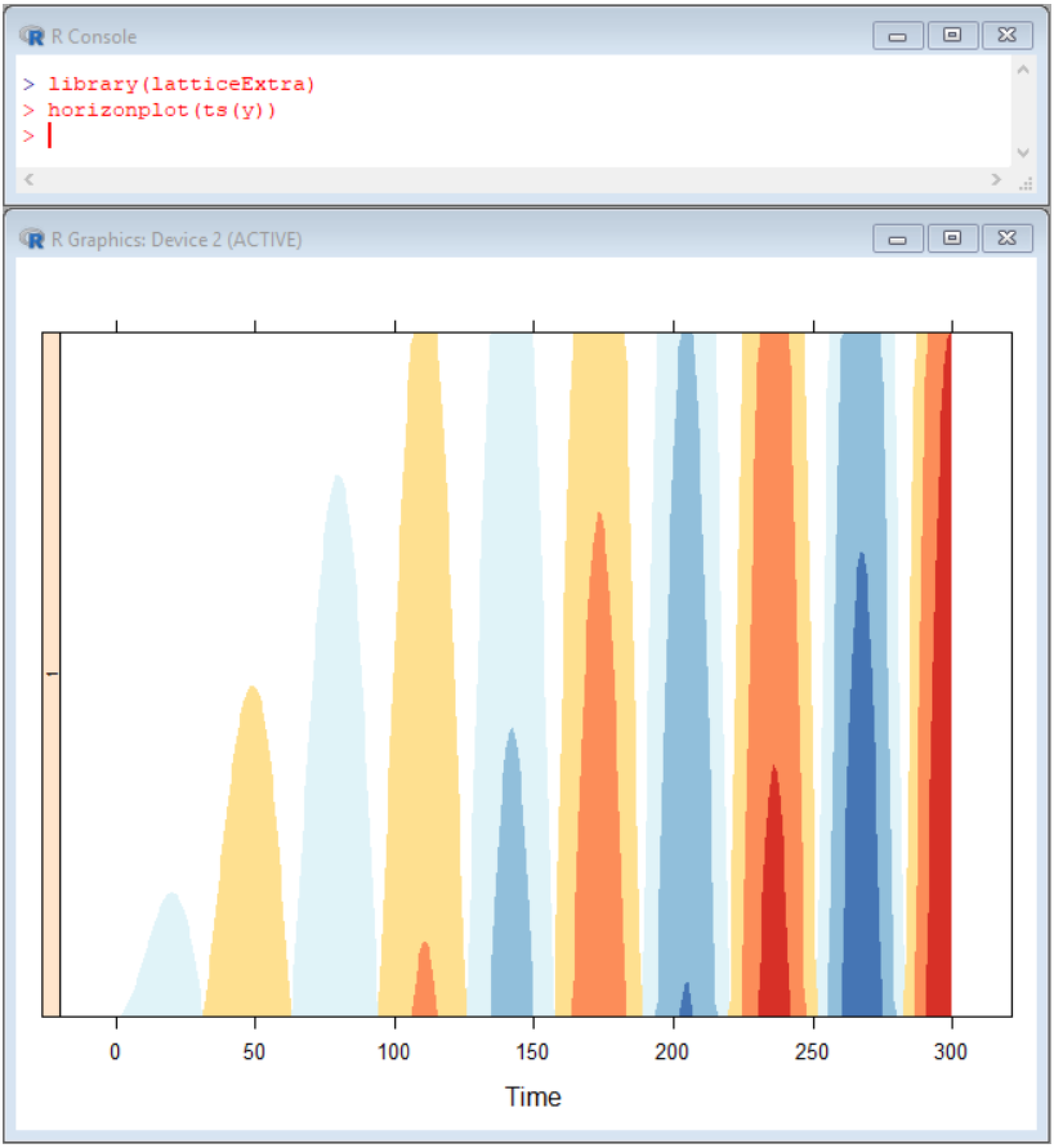


Figure 112 Graph de type Horizon

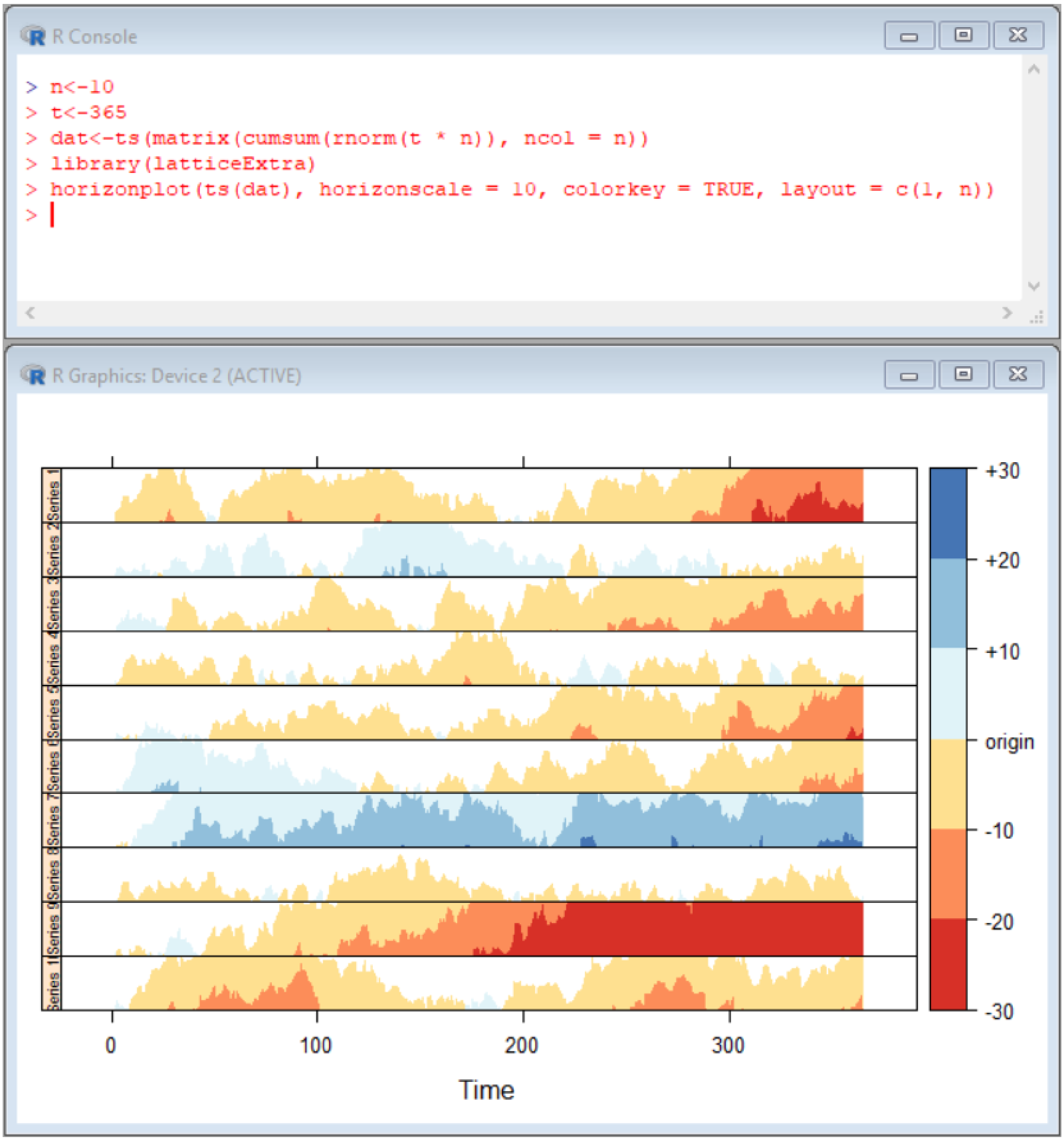
Donc nous voyons que l'idée est au fait de prendre:

1. La valeur absolue

2. Les points positifs sont bleus, les négatifs sont rouges

3. Les points en-dessous/au-dessus de change ligne grise traitillé est reportée dans une couleur différente

Donc voici un exemple plus complet qui montre que la lecture de ce type de graphique est loin d'être aisé:



ou encore plus élaboré:

```
> library(ggplot2)
> library(scales)
> colfill <- c('#7F3B08','#A50026','#40004B','#276419','#313695')
> data ("msleep")
> ggplot(msleep, aes(x=sleep_total,y=..count..,fill=vore))+
+       geom_density(position = position_fill()) +
+    scale_fill_manual(values = colfill,na.value ="black")+
+    theme_bw()+
+    scale_y_continuous(labels = percent_format(),
+                               expand = c(0,0))+
+    scale_x_continuous(expand = c(0,0))
> |
```

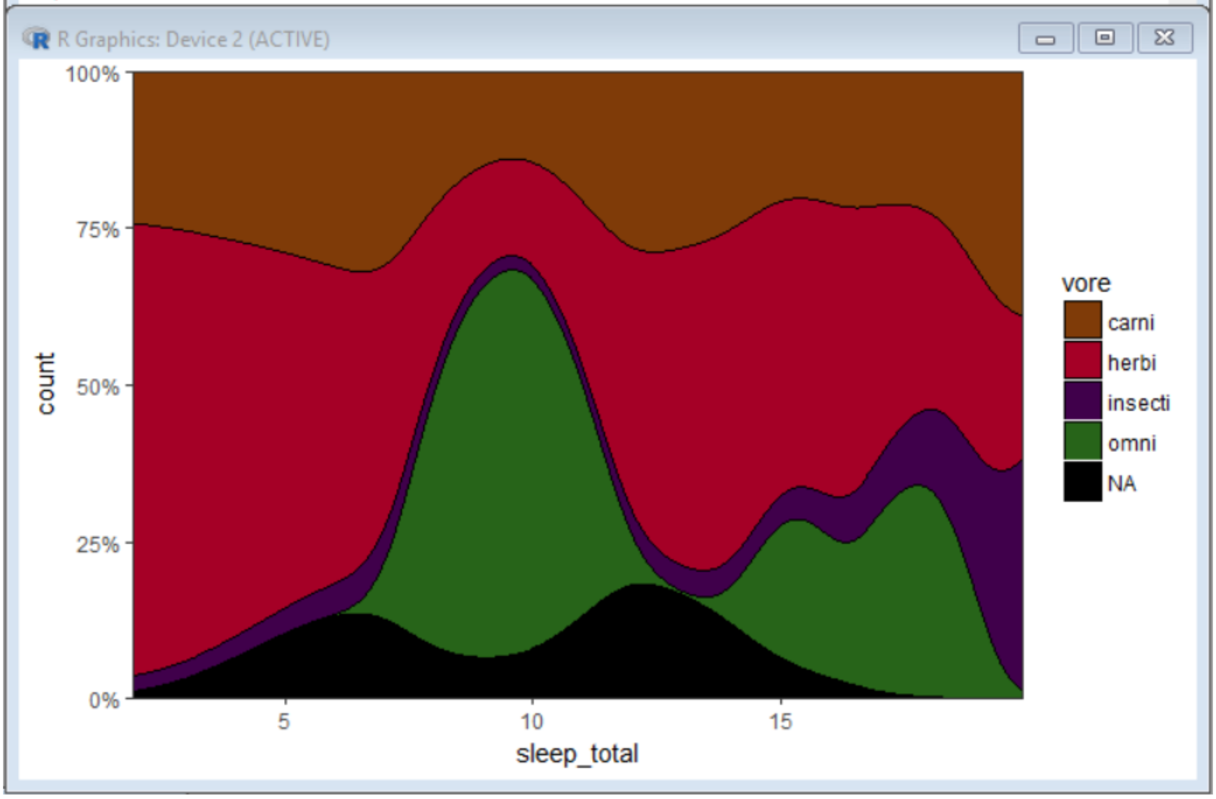


Figure 113 Graph de type Horizon normalisé

Exemple 126.: (Diagrammes Microsoft)

R 3.4.1

Nous venons de voir quelques graphiques simples. Il peut être considéré comme normal (même si c'est pénible et intellectuellement réducteur) que des gens souhaitent absolument avoir des graphiques Microsoft depuis R. Pour cela, Microsoft a créé un package nommé **mschart** qui fonctionne mains dans la main avec le package **officer** et nous allons ouvrir une brève **parenthèse** sur le sujet sachant que nous avons à ce jour 4 graphes à disposition:

```
ms_linechart(data, x, y, group = NULL)

ms_barchart(data, x, y, group = NULL)

ms_areachart(data, x, y, group = NULL)

ms_scatterchart(data, x, y, group = NULL)
```

Voici donc un exemple d'utilisation de ce package:

```
> library(officer)
> library(mschart)
> my_barchart <- ms_barchart(data = browser_data, x = "browser", y = "value", group = "serie")
> doc <- read_pptx()
> doc <- add_slide(doc, layout = "Title and Content", master = "Office Theme")
> doc <- ph_with_chart(doc, chart = my_barchart)
> print(doc, target = "c:/tmp/barchart.pptx")
[1] "c:/tmp/barchart.pptx"
>
```

Ce qui donnera:

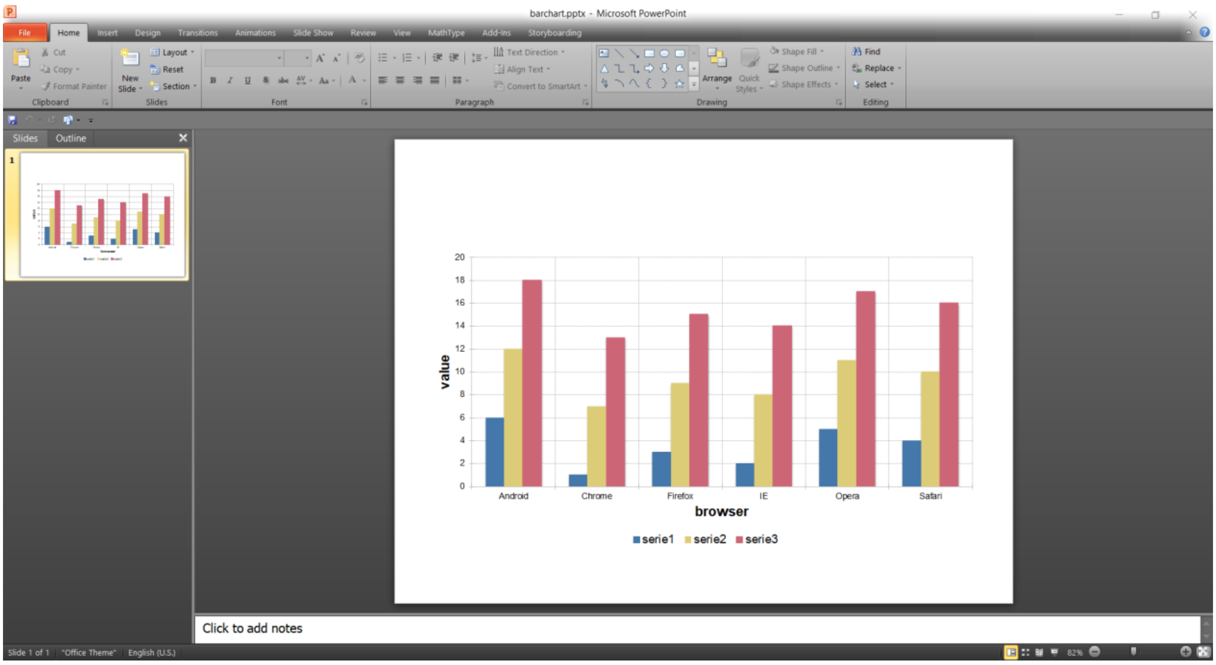


Figure 114 Graph générés dans Microsoft PowerPoint

Et voici un exemple avec Microsoft Word:

```
> library(officer)
> library(mschart)
> my_barchart <- ms_barchart(data = browser_data, x = "browser", y = "value", group = "serie")
> doc <- read_docx()
> doc <- body_add_chart(doc, chart = my_barchart, style = "centered")
> print(doc, target = "c:/tmp/barchart.docx")
[1] "c:/tmp/barchart.docx"
>
```

Ce qui donne:

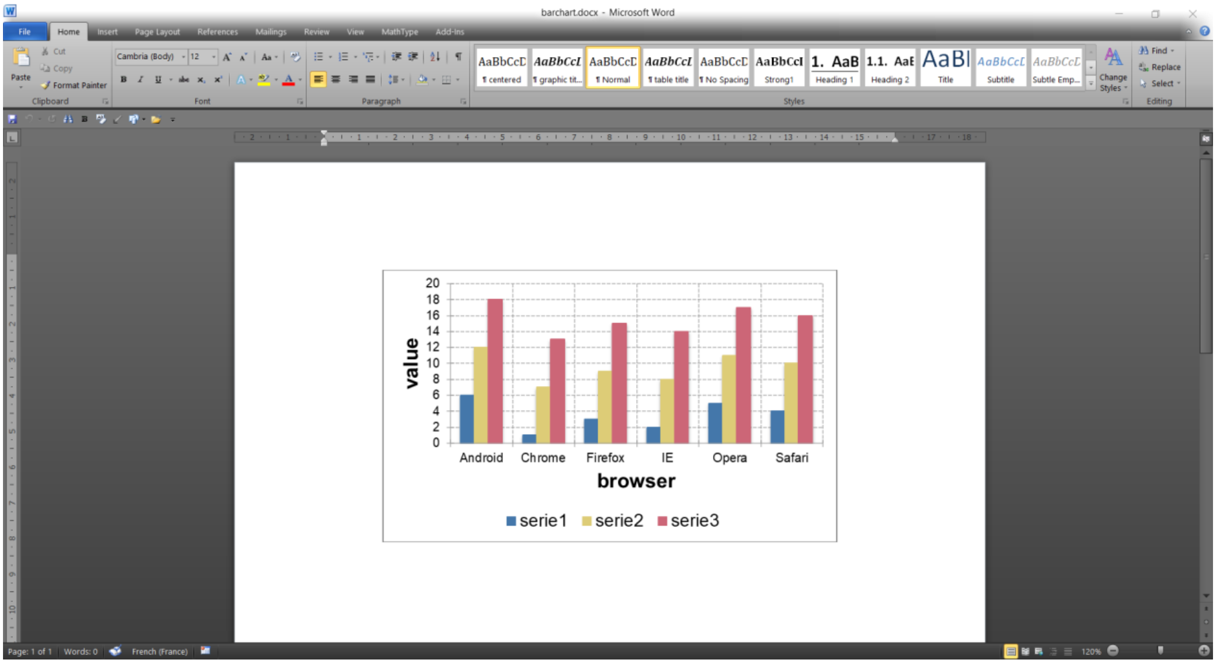
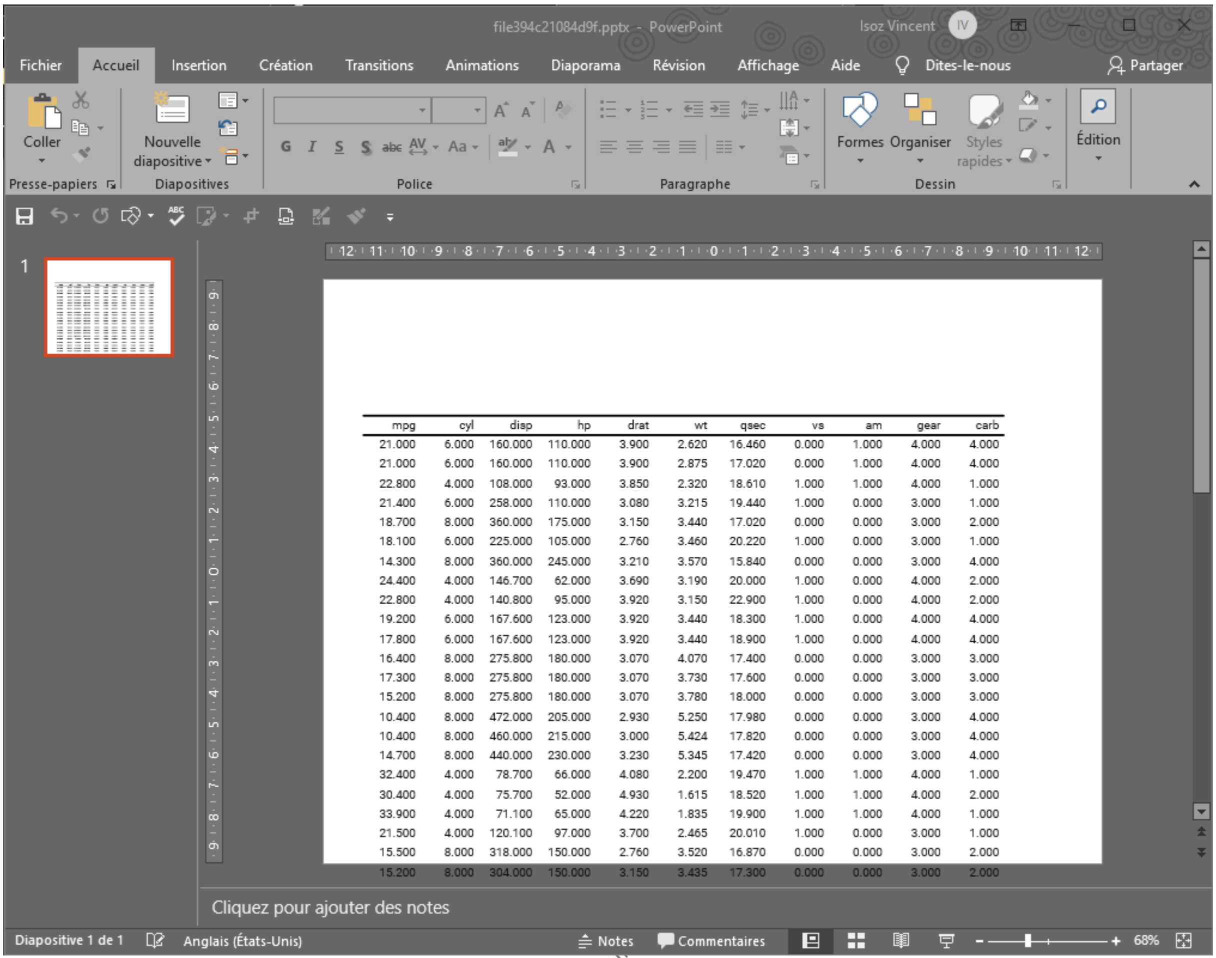


Figure 115 Graph générés dans Microsoft Word

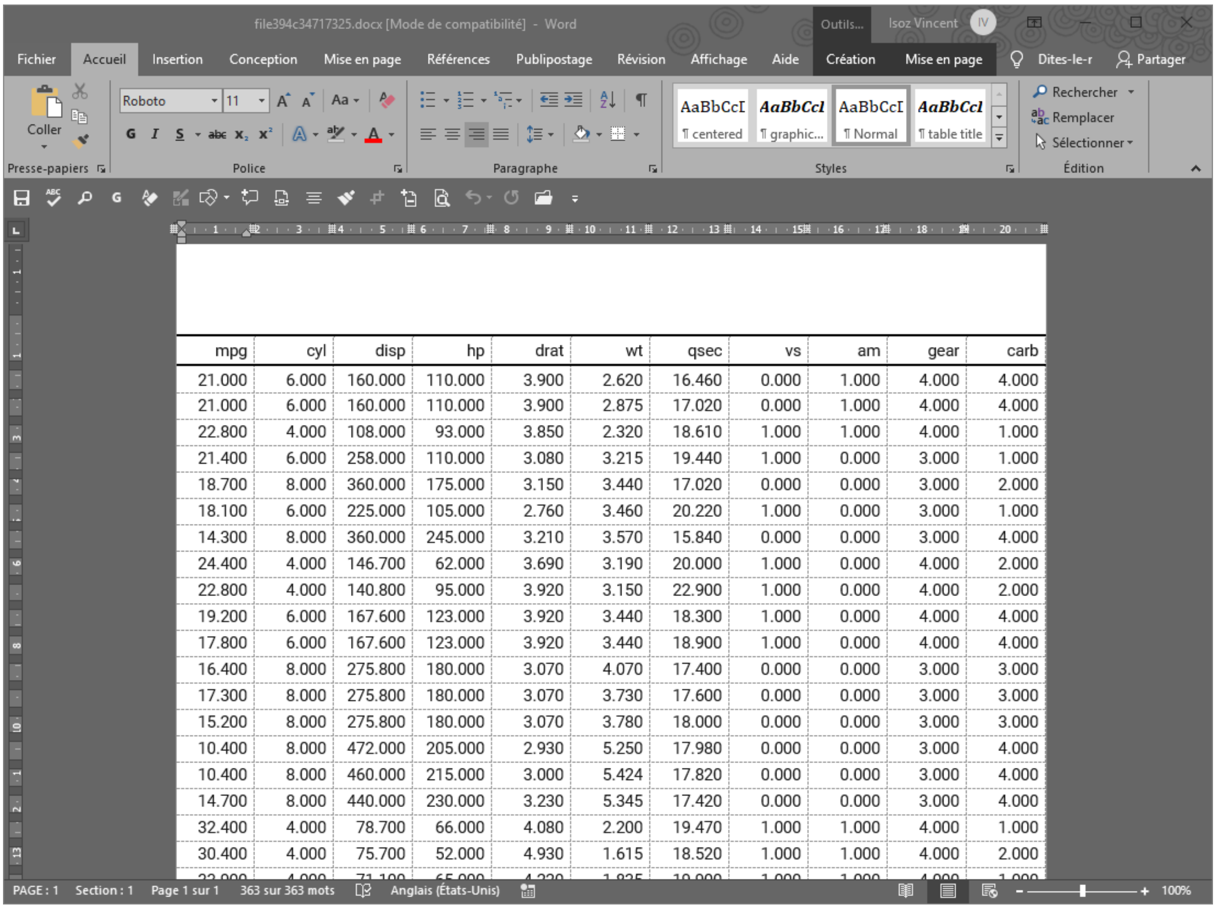
Notez qu'on peut faire de même en exportant des tables plus ou moins complexes et personnalisés avec le package **flextable** et sa fonction **flextable( )**:

```
> library("flextable")
> ft <- flextable(mtcars)
> print(flextable(mtcars),"pptx")
> print(flextable(mtcars),"docx")
```

Ce qui donnera:

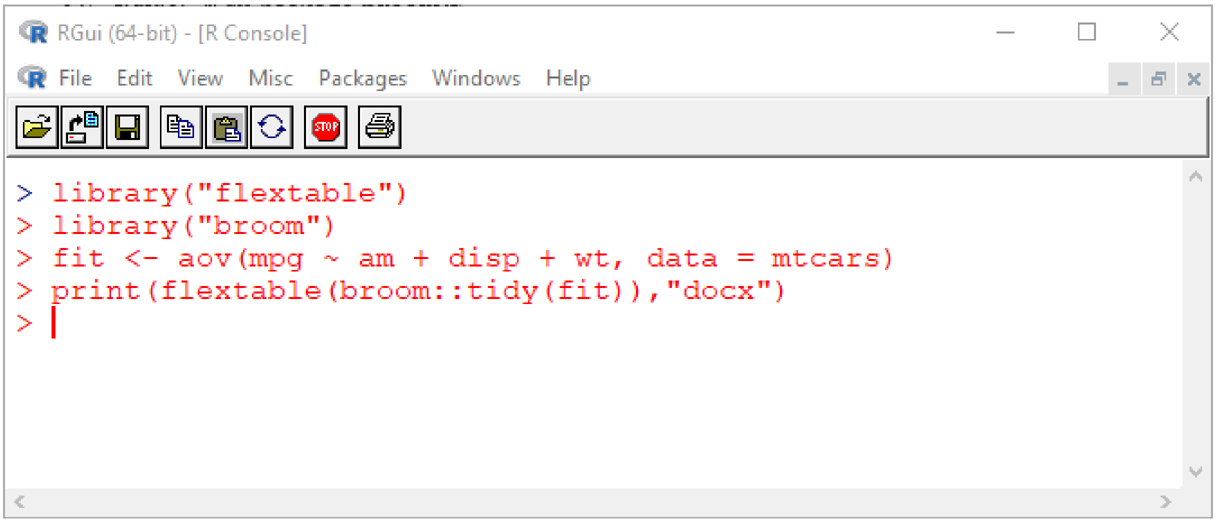


Et respectivement:

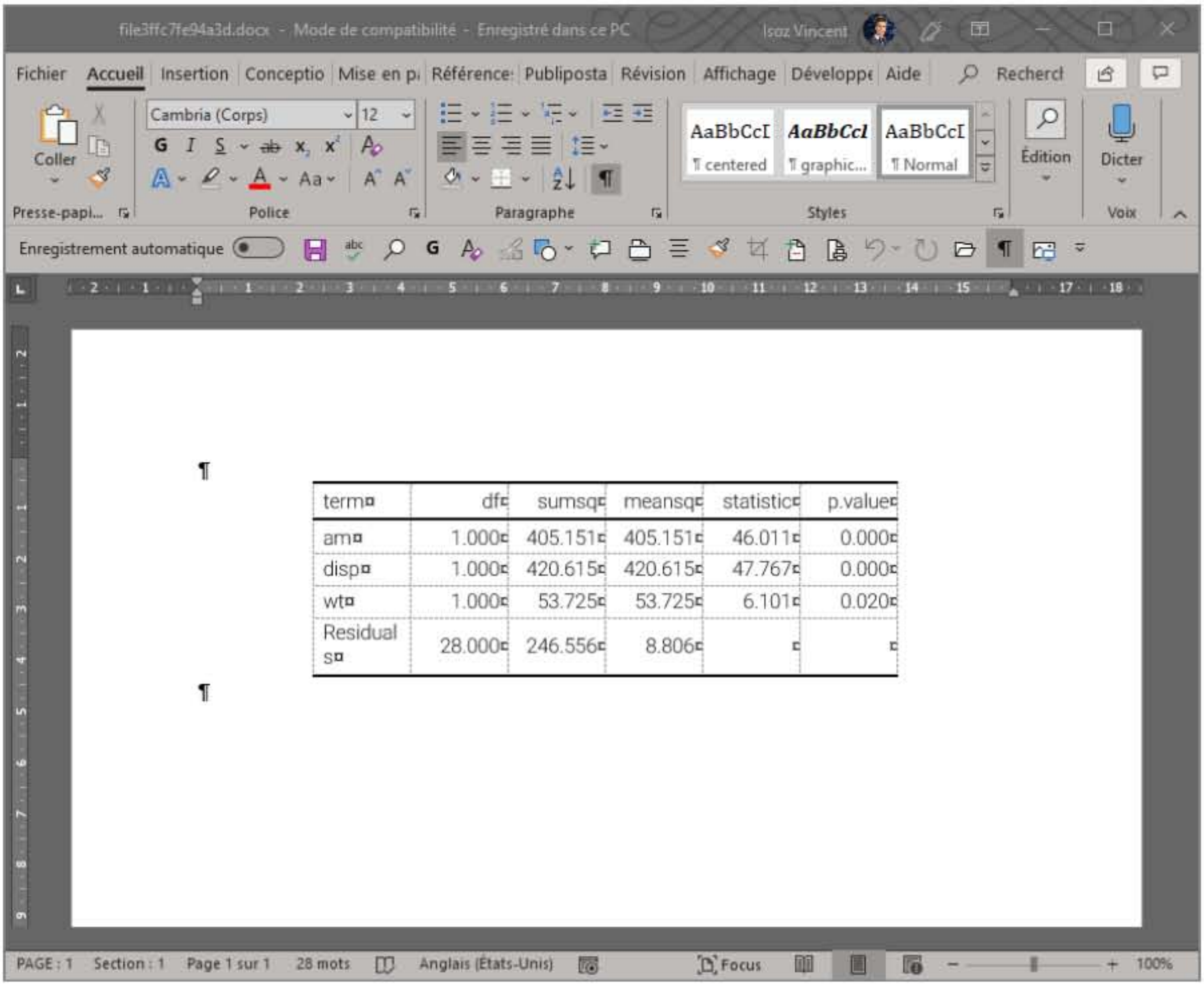


mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.000	6.000	160.000	110.000	3.900	2.620	16.460	0.000	1.000	4.000	4.000
21.000	6.000	160.000	110.000	3.900	2.875	17.020	0.000	1.000	4.000	4.000
22.800	4.000	108.000	93.000	3.850	2.320	18.610	1.000	1.000	4.000	1.000
21.400	6.000	258.000	110.000	3.080	3.215	19.440	1.000	0.000	3.000	1.000
18.700	8.000	360.000	175.000	3.150	3.440	17.020	0.000	0.000	3.000	2.000
18.100	6.000	225.000	105.000	2.760	3.460	20.220	1.000	0.000	3.000	1.000
14.300	8.000	360.000	245.000	3.210	3.570	15.840	0.000	0.000	3.000	4.000
24.400	4.000	146.700	62.000	3.690	3.190	20.000	1.000	0.000	4.000	2.000
22.800	4.000	140.800	95.000	3.920	3.150	22.900	1.000	0.000	4.000	2.000
19.200	6.000	167.600	123.000	3.920	3.440	18.300	1.000	0.000	4.000	4.000
17.800	6.000	167.600	123.000	3.920	3.440	18.900	1.000	0.000	4.000	4.000
16.400	8.000	275.800	180.000	3.070	4.070	17.400	0.000	0.000	3.000	3.000
17.300	8.000	275.800	180.000	3.070	3.730	17.600	0.000	0.000	3.000	3.000
15.200	8.000	275.800	180.000	3.070	3.780	18.000	0.000	0.000	3.000	3.000
10.400	8.000	472.000	205.000	2.930	5.250	17.980	0.000	0.000	3.000	4.000
10.400	8.000	460.000	215.000	3.000	5.424	17.820	0.000	0.000	3.000	4.000
14.700	8.000	440.000	230.000	3.230	5.345	17.420	0.000	0.000	3.000	4.000
32.400	4.000	78.700	66.000	4.080	2.200	19.470	1.000	1.000	4.000	1.000
30.400	4.000	75.700	52.000	4.930	1.615	18.520	1.000	1.000	4.000	2.000

Concernant l'export de nombreux modèles), vous pouvez/devez utiliser la fonction **tidy( )** du package **broom**:



Ce qui donnera:



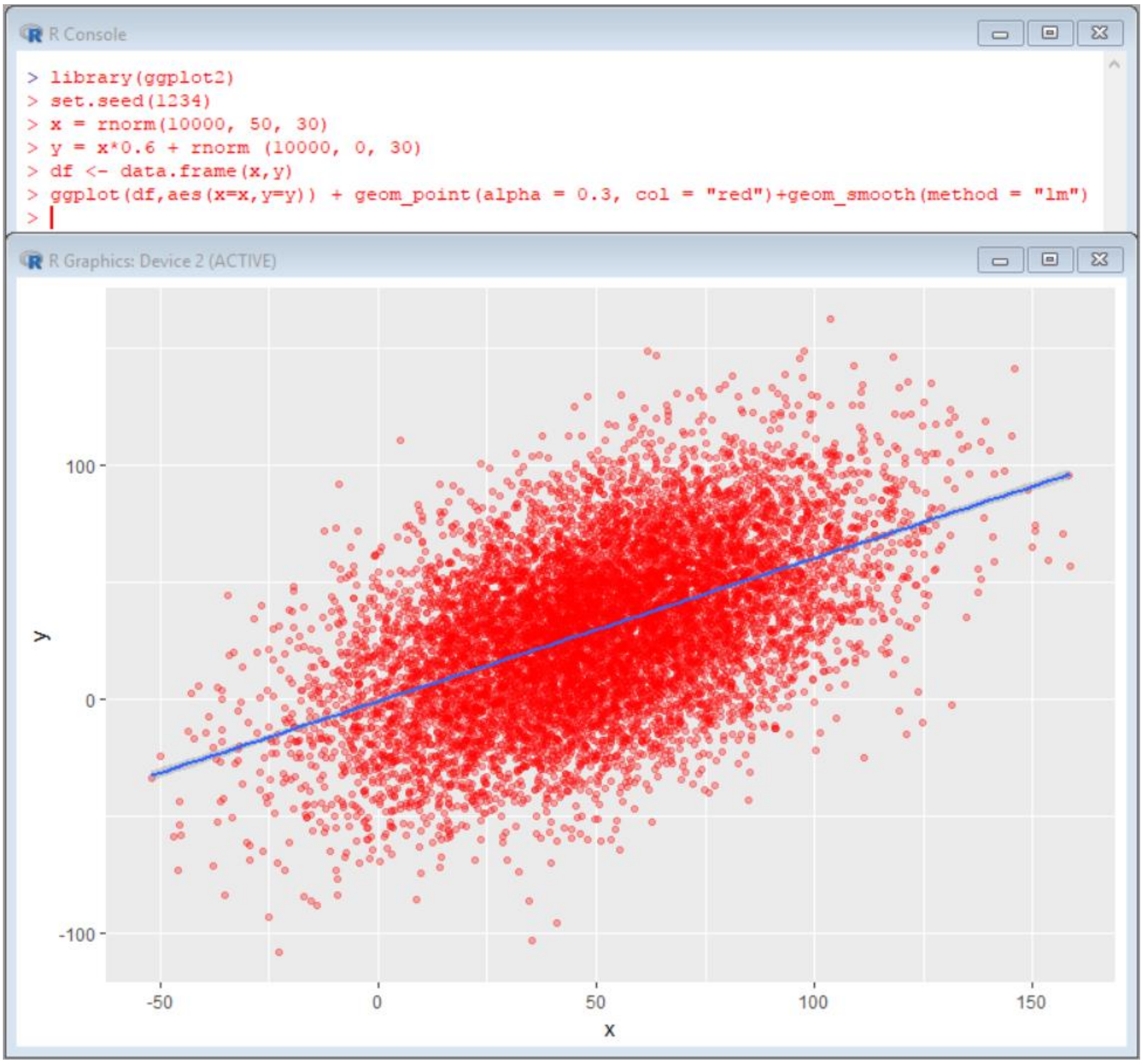
term	df	sumsq	meansq	statistic	p.value
am	1.000	405.151	405.151	46.011	0.000
disp	1.000	420.615	420.615	47.767	0.000
wt	1.000	53.725	53.725	6.101	0.020
Residuals	28.000	246.556	8.806		

Exemple 127.: Hexabins

R 3.4.1

Les diagrammes hexabins sont des diagrammes qui regroupent une densité de points sous la forme d'hexagones dont la couleur change avec la fréquence des points qui s'y trouvent. C'est une technique comme pour analyser des nuages de points lorsque le nombre de points est trop élevé.

Voyons d'abord un exemple avec ggplot2 sans hexabin:



Donc au vu de la densité de points ce n'est pas l'idéal pour un humain…

Maintenant avec hexabin de ggplot2:

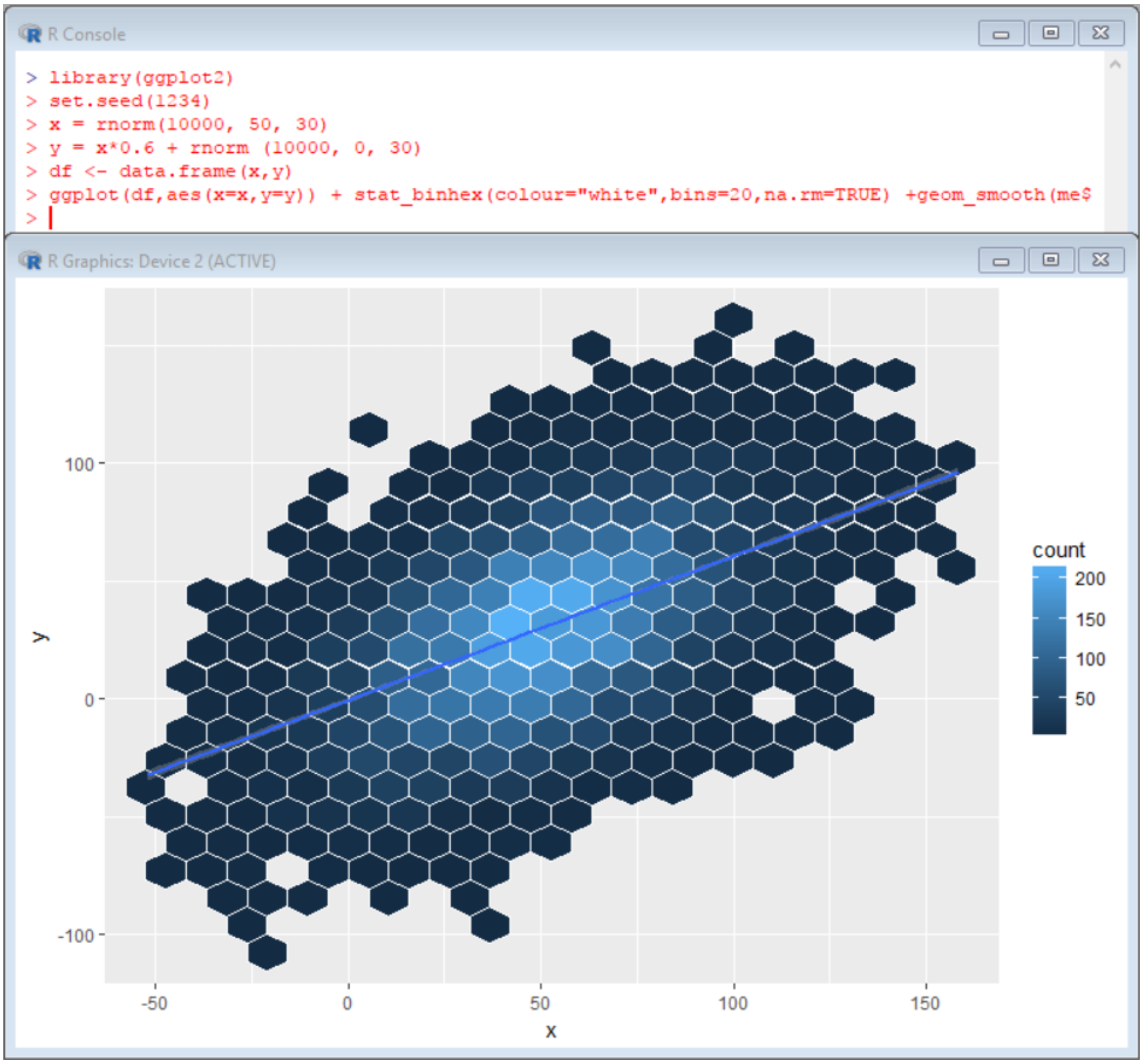


Figure 116 Graph Hexabin

Nous identifions alors beaucoup mieux en tant qu'humains où se trouve la plus grande concentration de points.

Ou sans ggplot2 avec le package **hexabin** et la fonction **hexbin( )** :

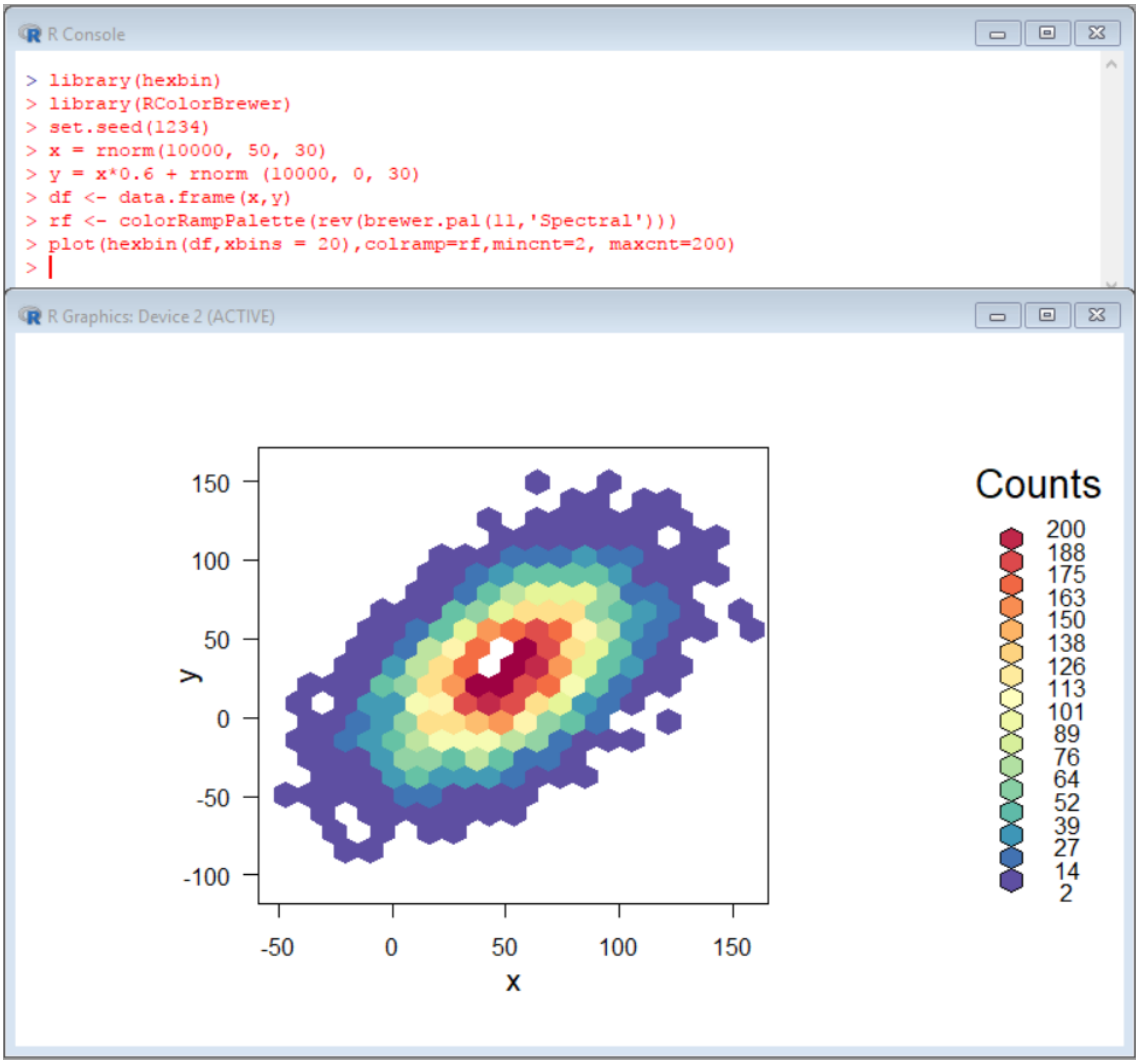


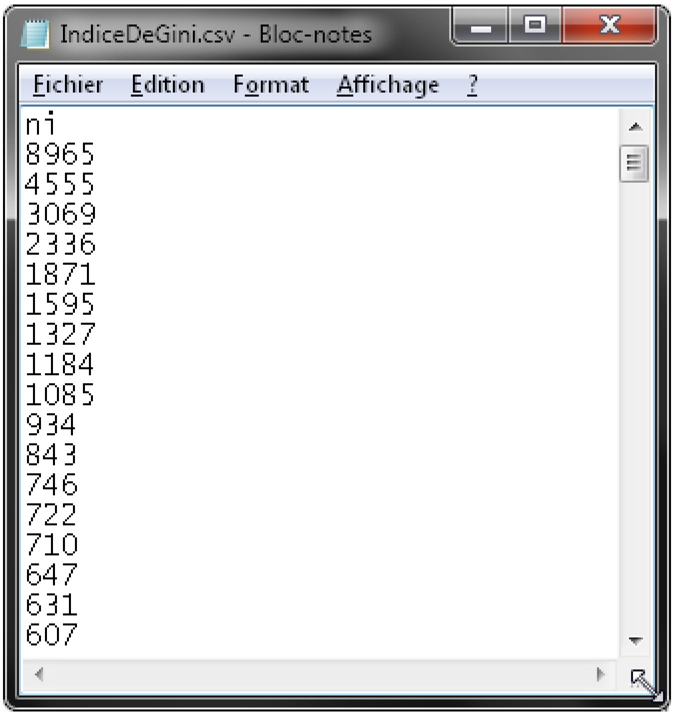
Figure 117 Graph Hexabin avec gradient de couleurs de densité

Exemple 128.: Courbe de Lorenz et indice de Gini

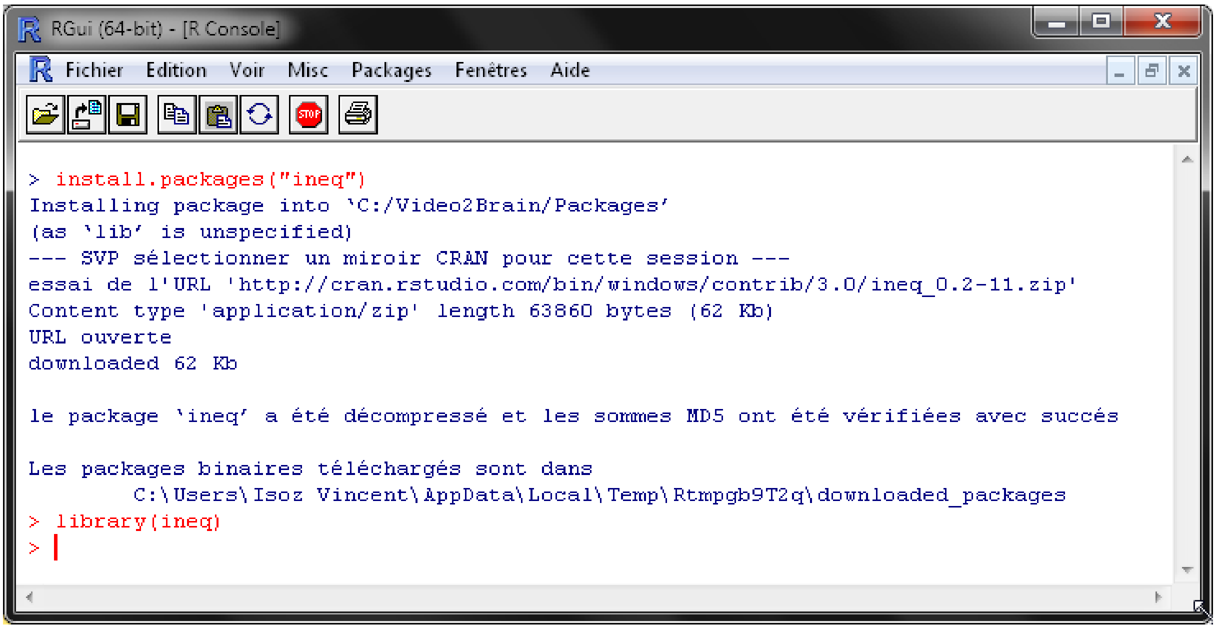
R 3.0.2

Le but ici va être de vérifier que nous retrouvons la même valeur de l'indice de Gini et la même courbe de Lorenz que celle calculée à la main dans le cours théorique de Techniques de Gestion.

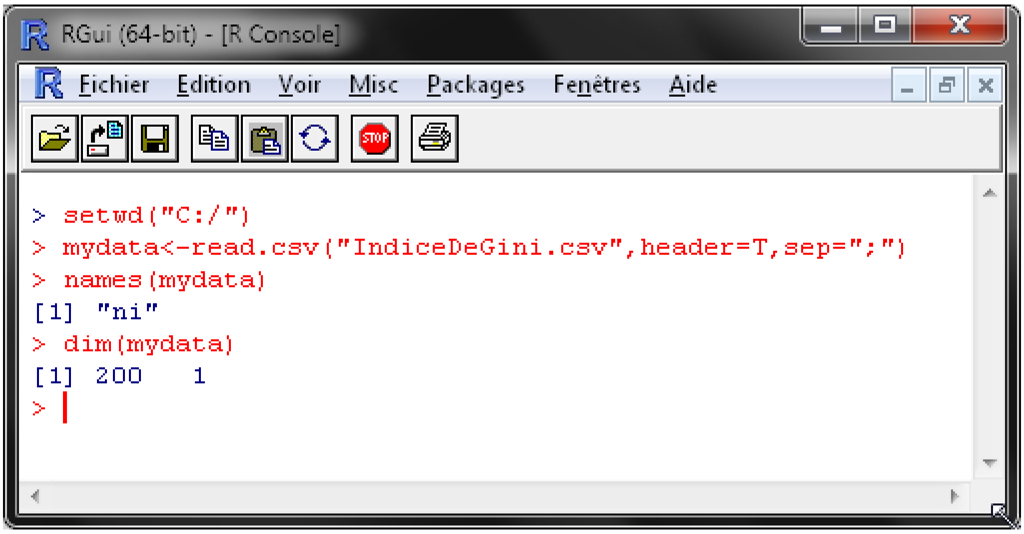
Nous partons alors des mêmes données:



Nous installons et chargeons-le package **ineq**:



et nous chargeons le fichier de données:



et ensuite nous utilisons le package **ineq** avec les commandes **Gini( )** et **Lc( )** :

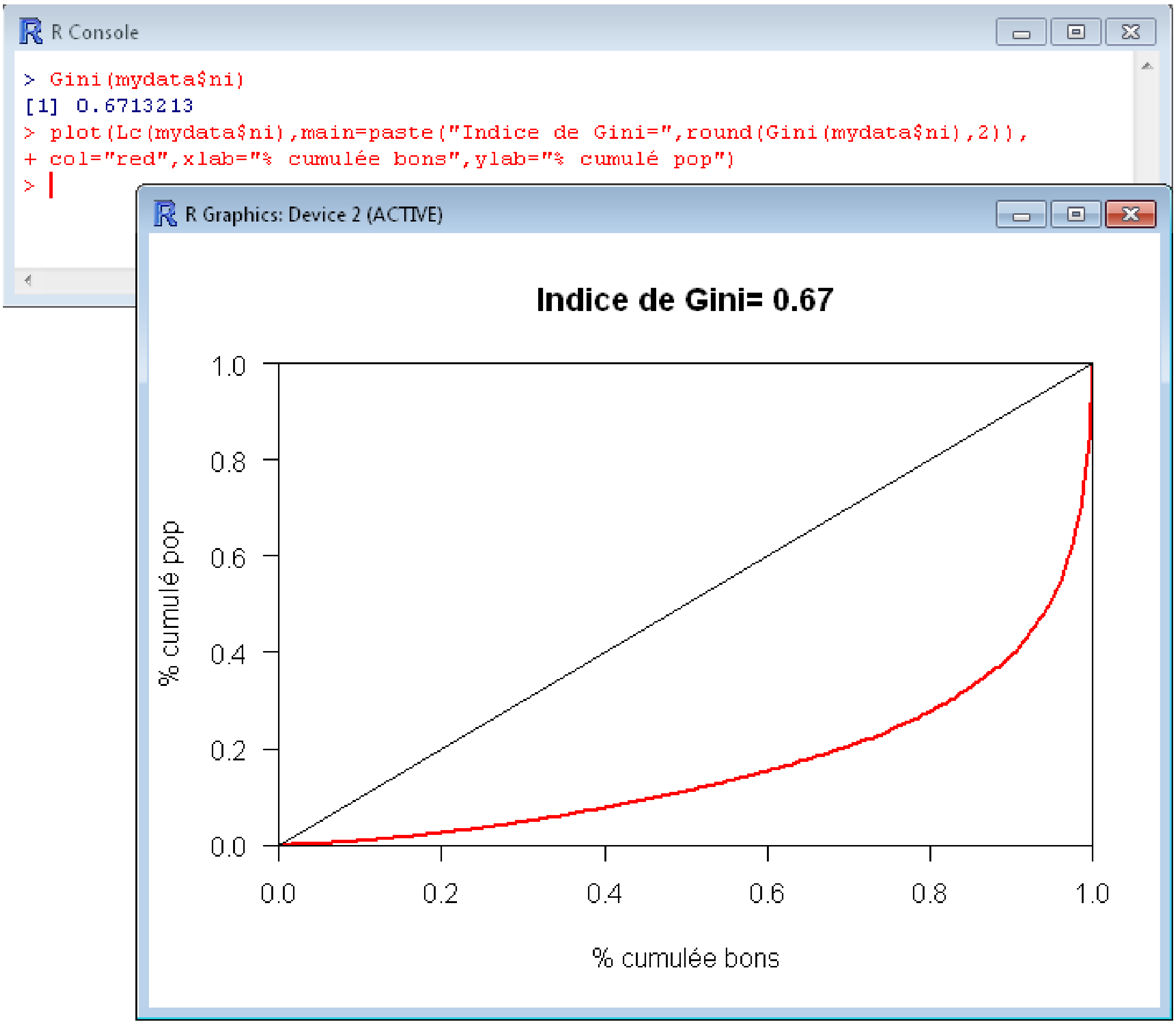


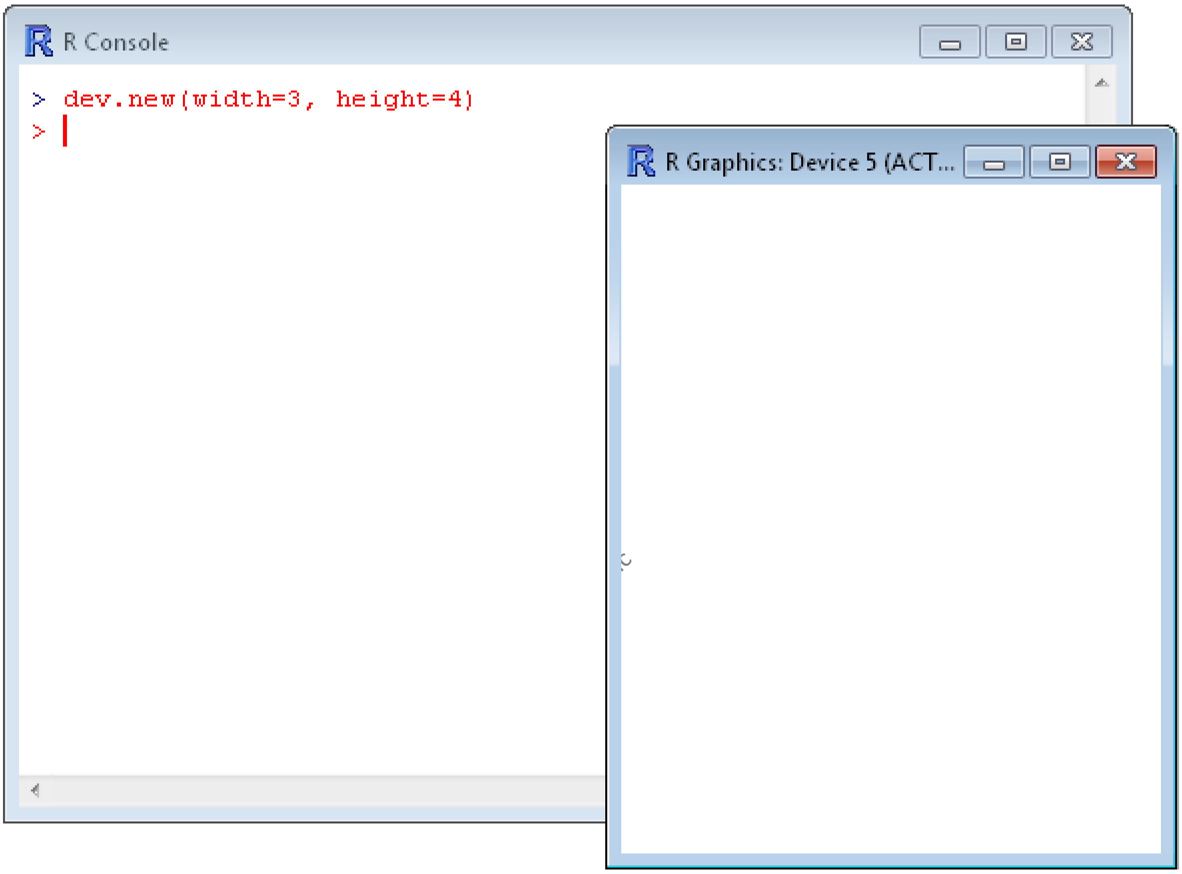
Figure 118 Graph avec courbe et indice de Gini

Exemple 129.: Combinaisons de graphiques (subplots)

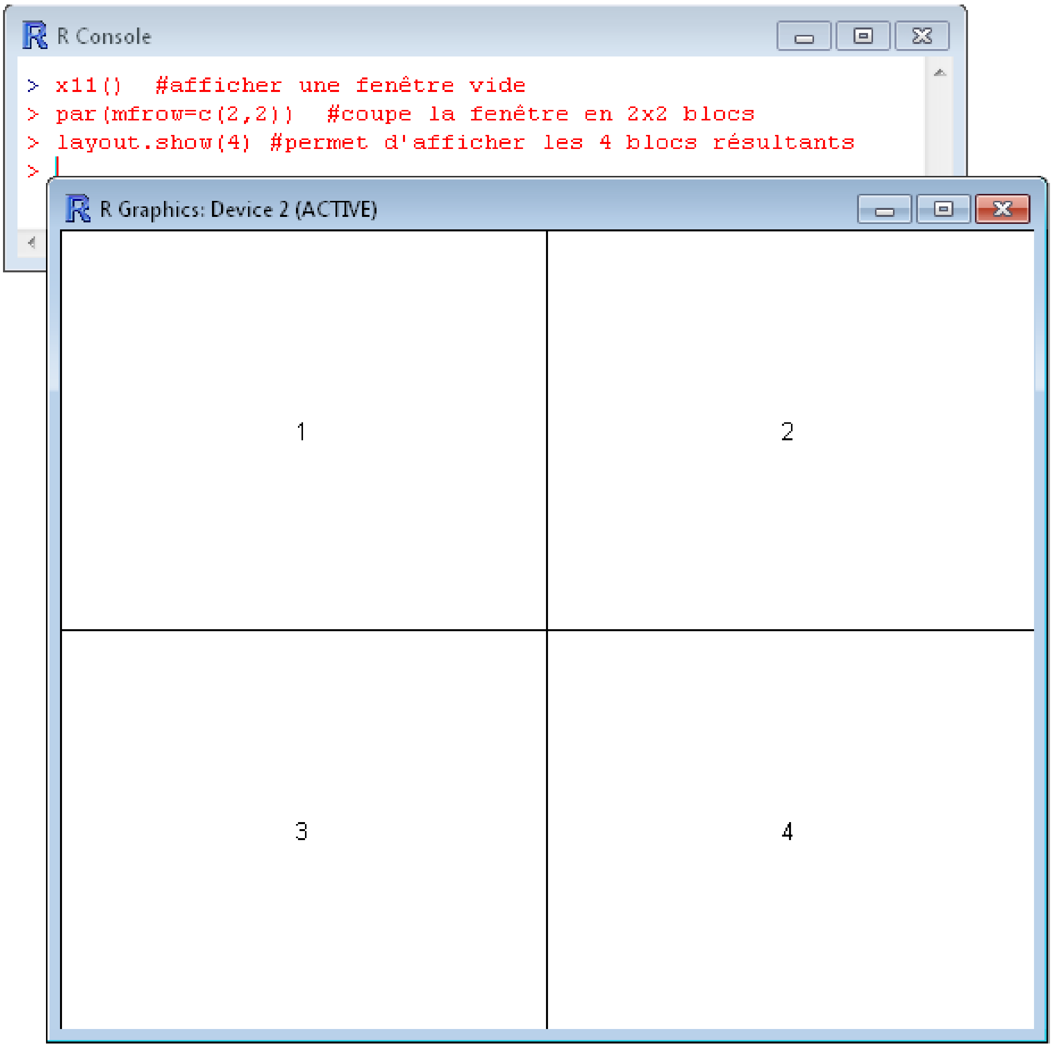
R 3.0.2

Avant de créer des graphiques multiples dans une même fenêtre il faut comprendre la structure de ces dernières.

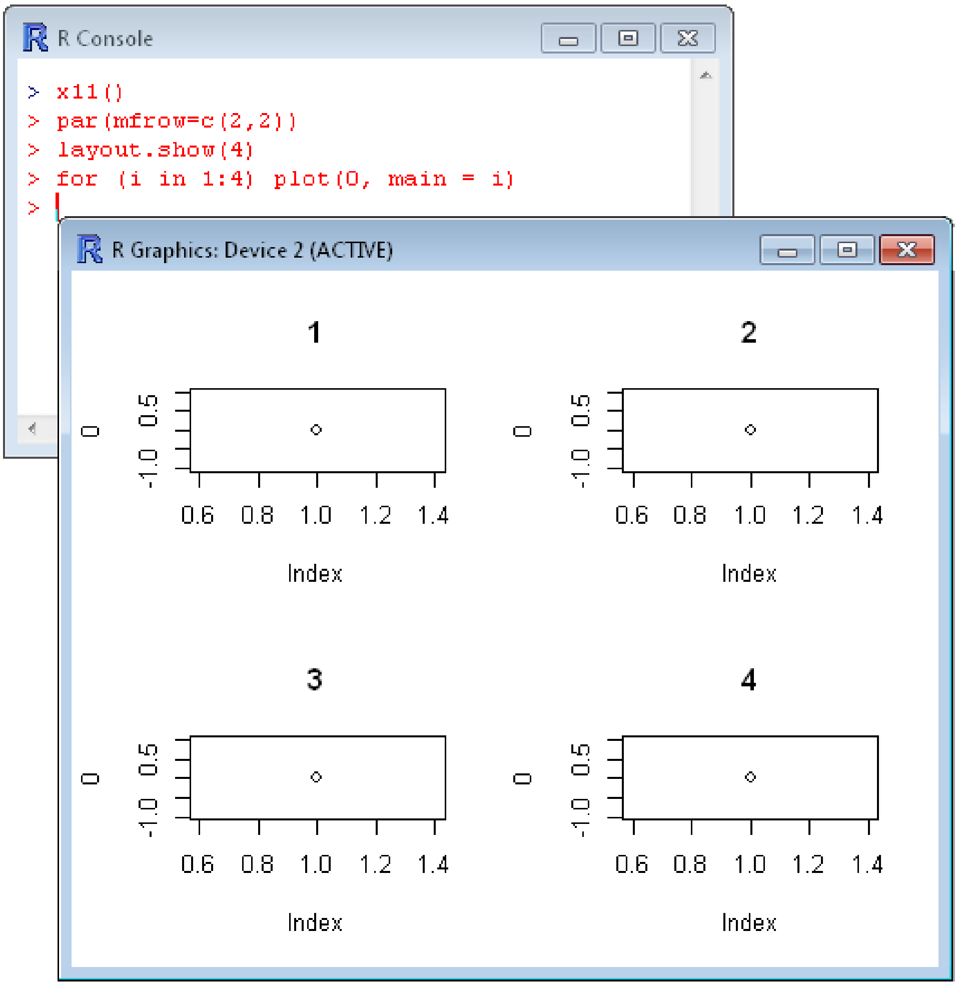
Mais d'abord, comme les combinaisons de graphiques peuvent prendre de la place, il est bon de savoir comment contrôler la taille de la fenêtre de plot avant d'y mettre quelque chose:



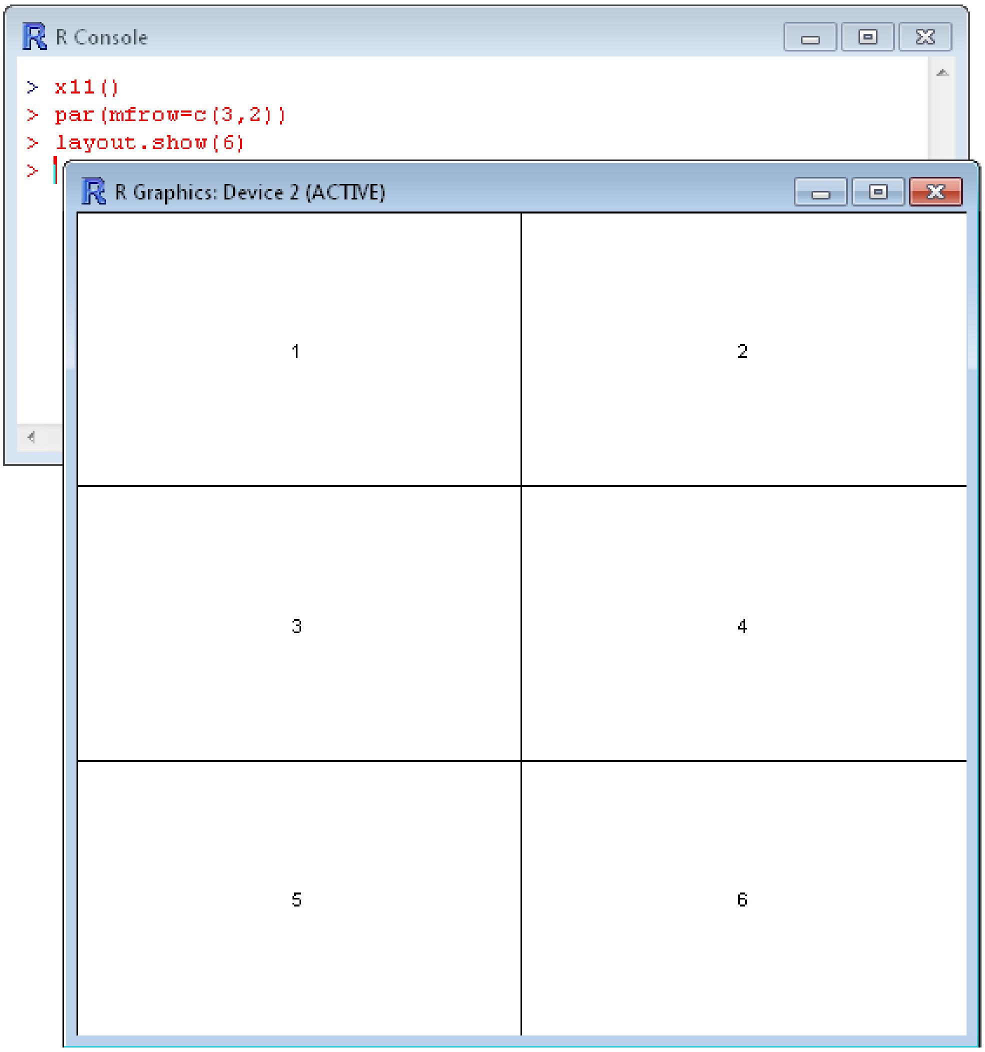
Voici comment on peut s'entraîner à comprendre les choses en utilisant les commandes **par( )** et **layout.show( )** :



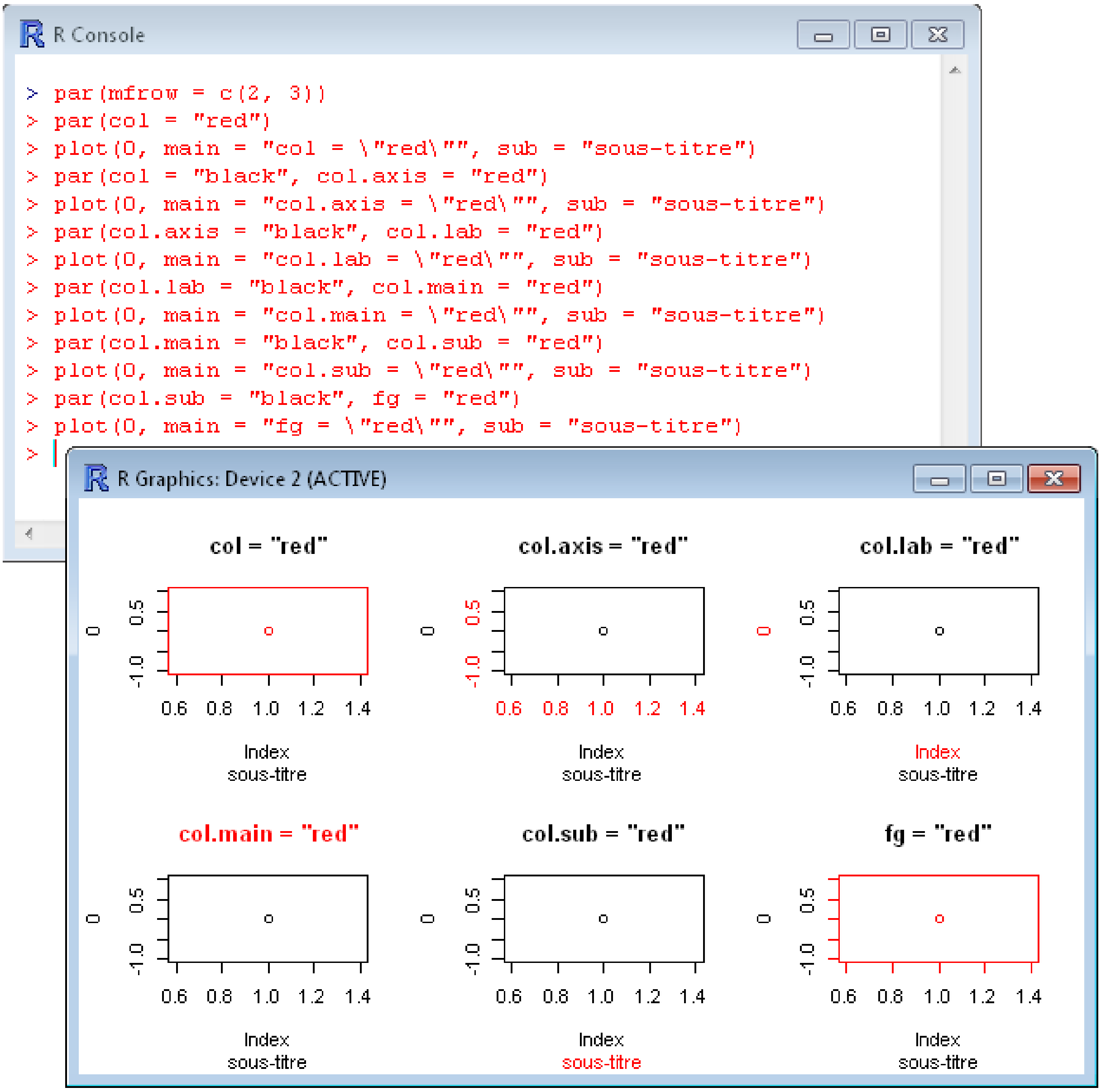
Que l'on peut déjà s'amuser à remplir par quelques plots vides:



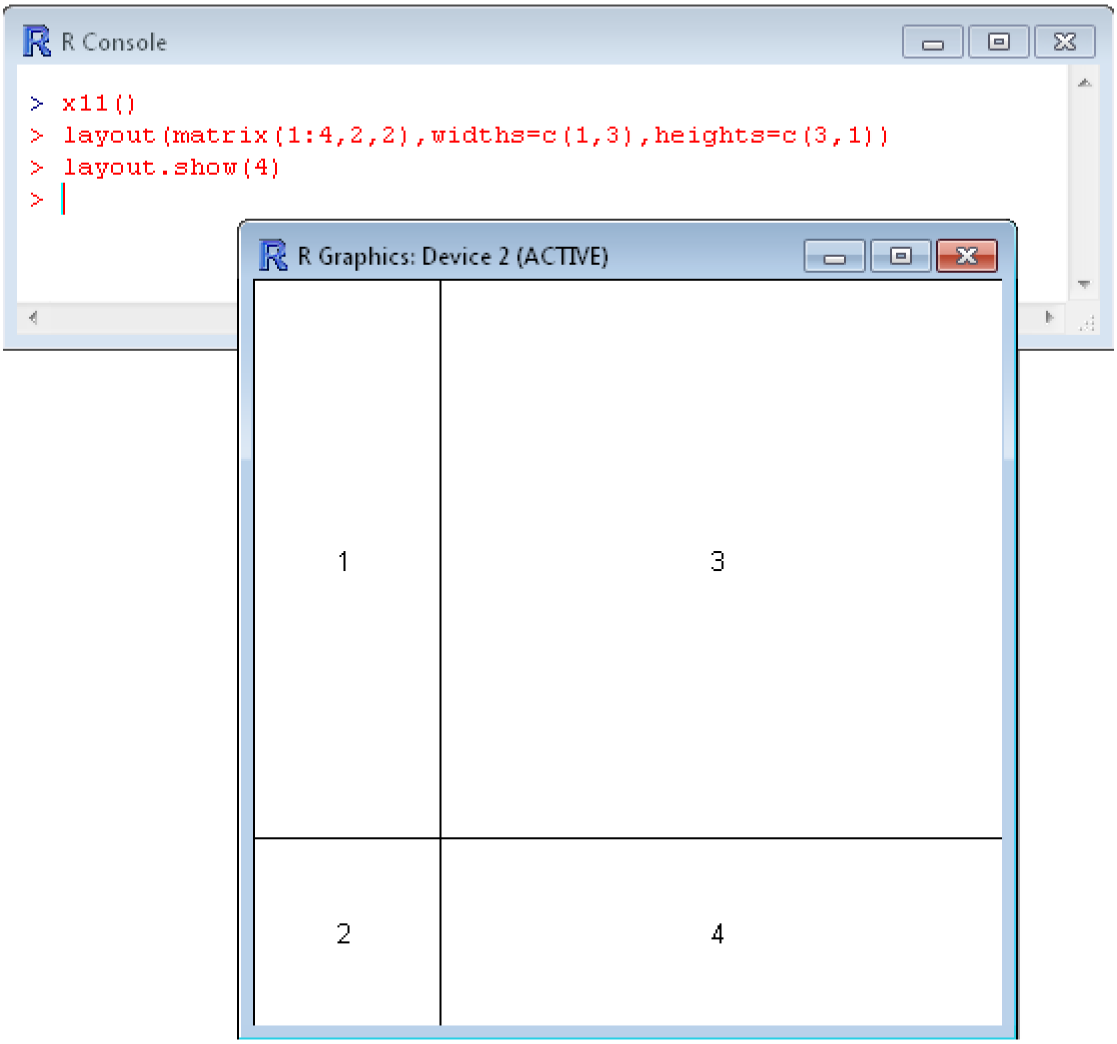
ou toujours dans la même idée:



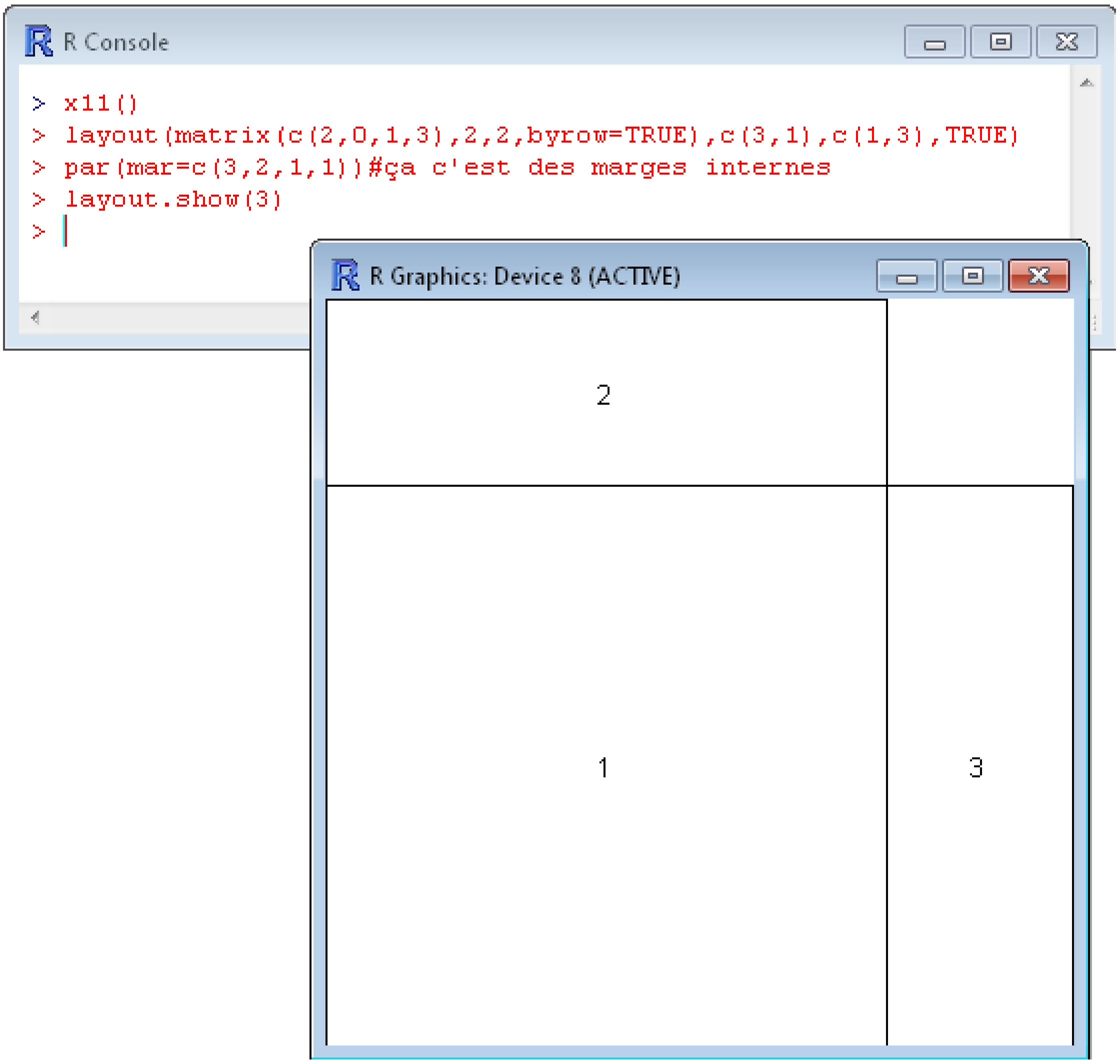
Avec une petite complication bien utile pour comprendre les choses:



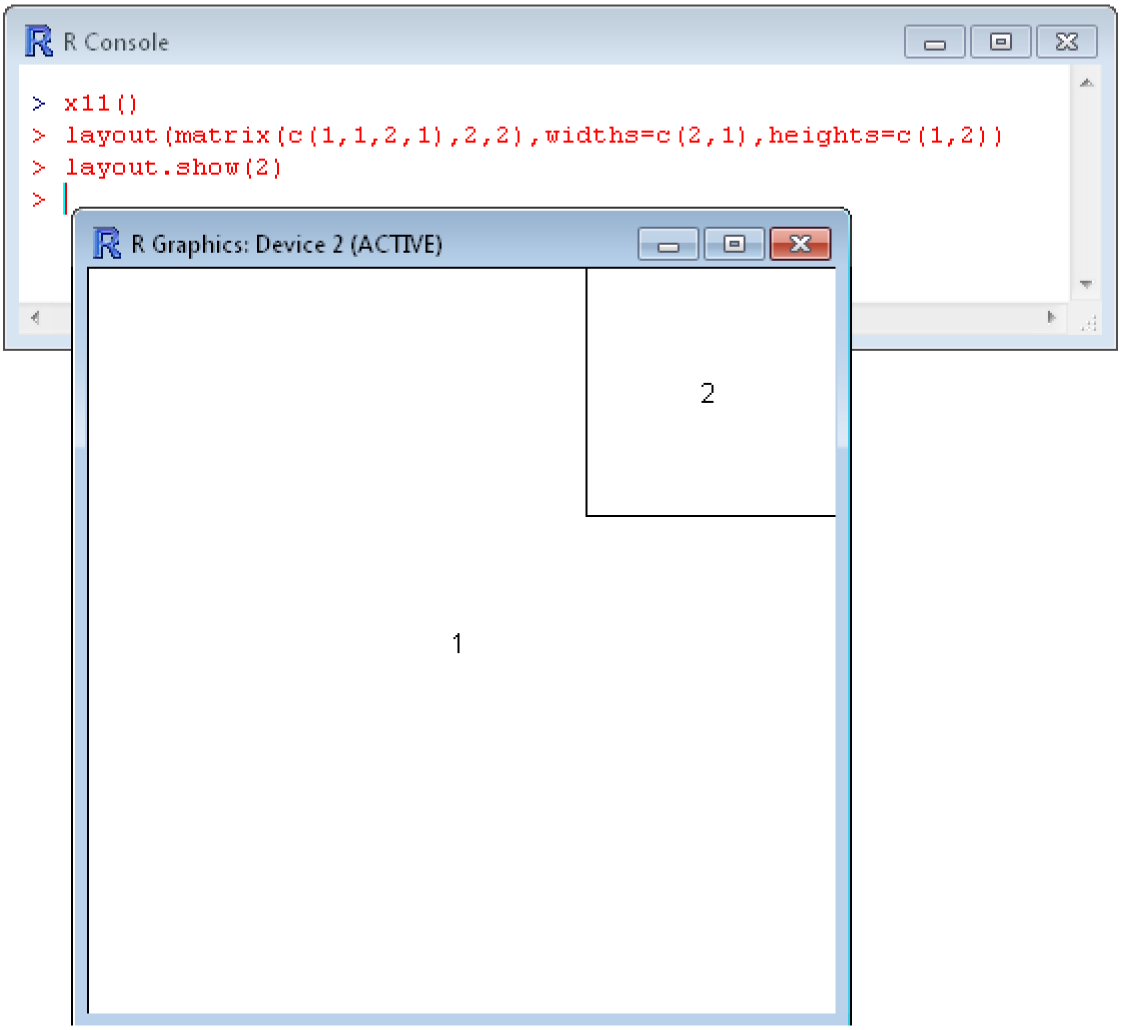
ou encore en un peu plus compliqué:



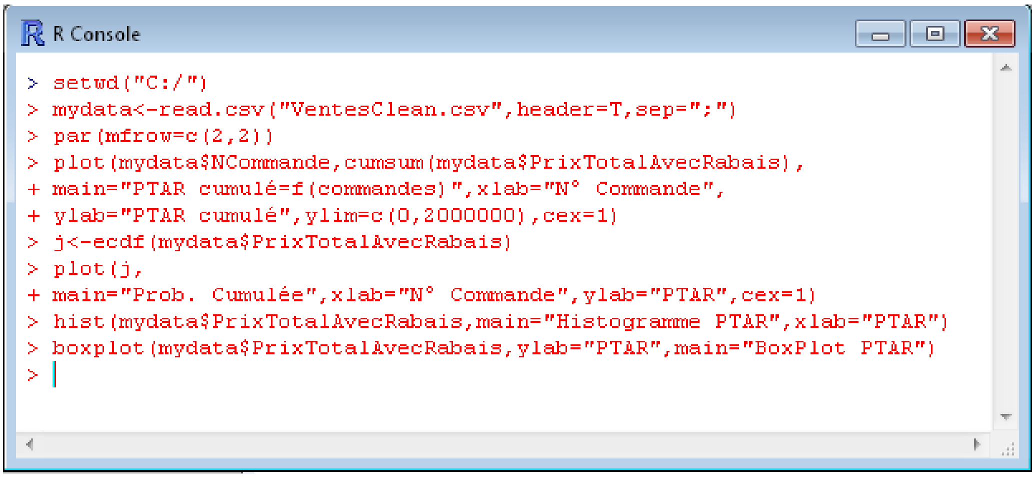
ou encore en un peu plus compliqué:



ou encore (on laisse tomber les marges car tant qu'on y met pas des graphiques, nous ne verrons pas la différence):



Dans ce qui suivra nous allons retrouver plusieurs fois des combinaisons de graphiques, voyons un exemple simple:



ce qui donnera:

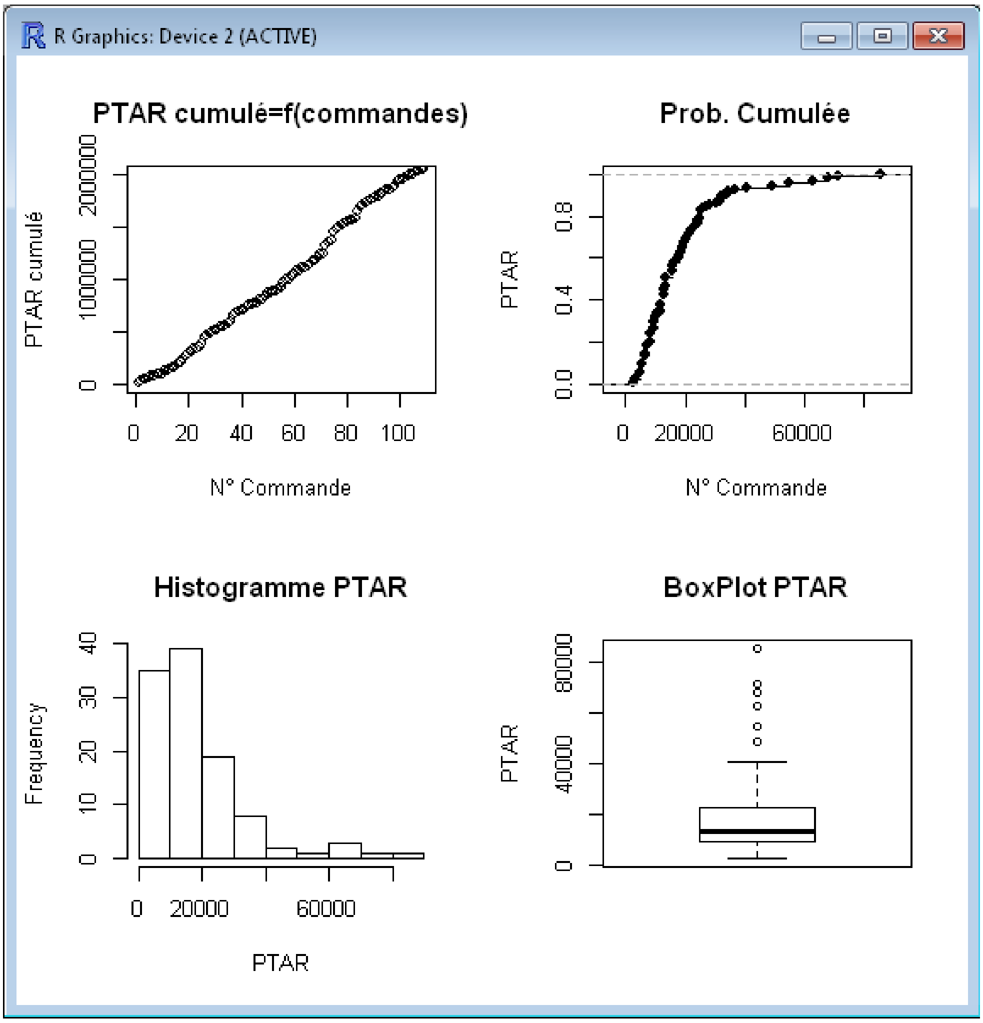
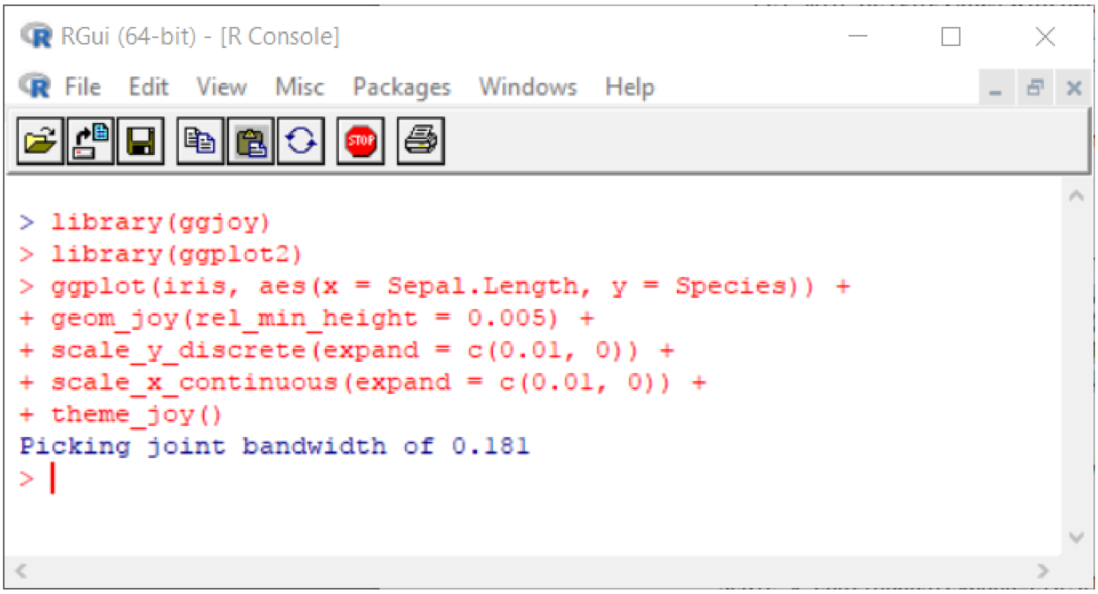


Figure 119 Combinaison de graphiques (points, répartition, histogramme, box-whisker)

Exemple 130.: Joy Plots

R 3.4.1

Il s'agit ici d'utiliser le package **ggjoy** et le paramètre geom_joy dans ggplot2 pour faire des graphiques qui se passent de commentaires:



ce qui donne:

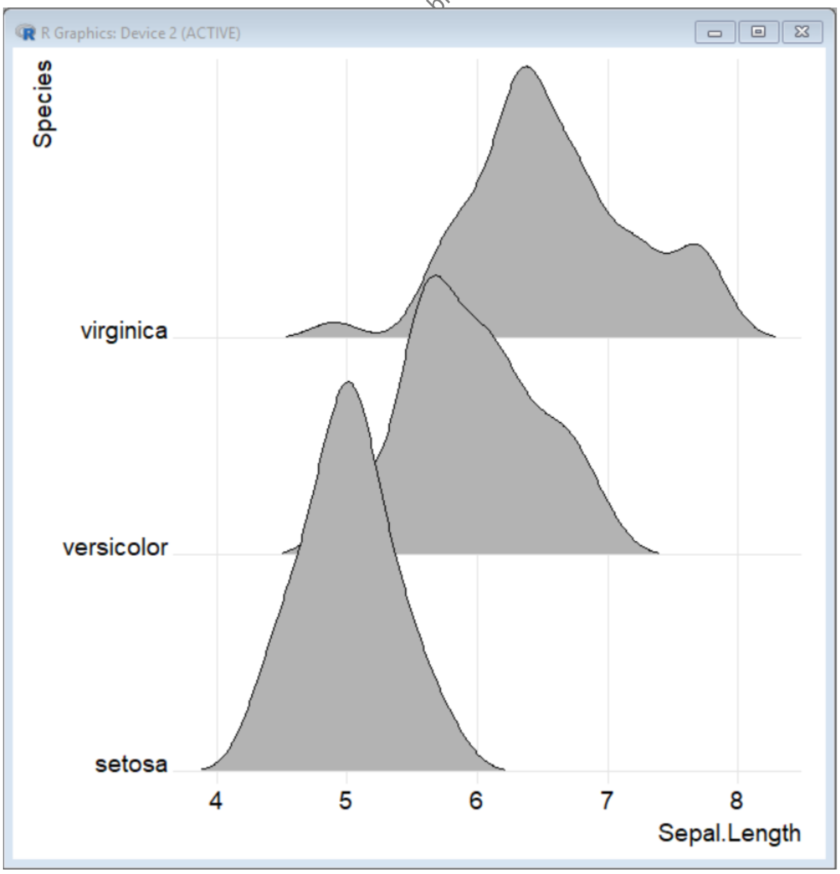
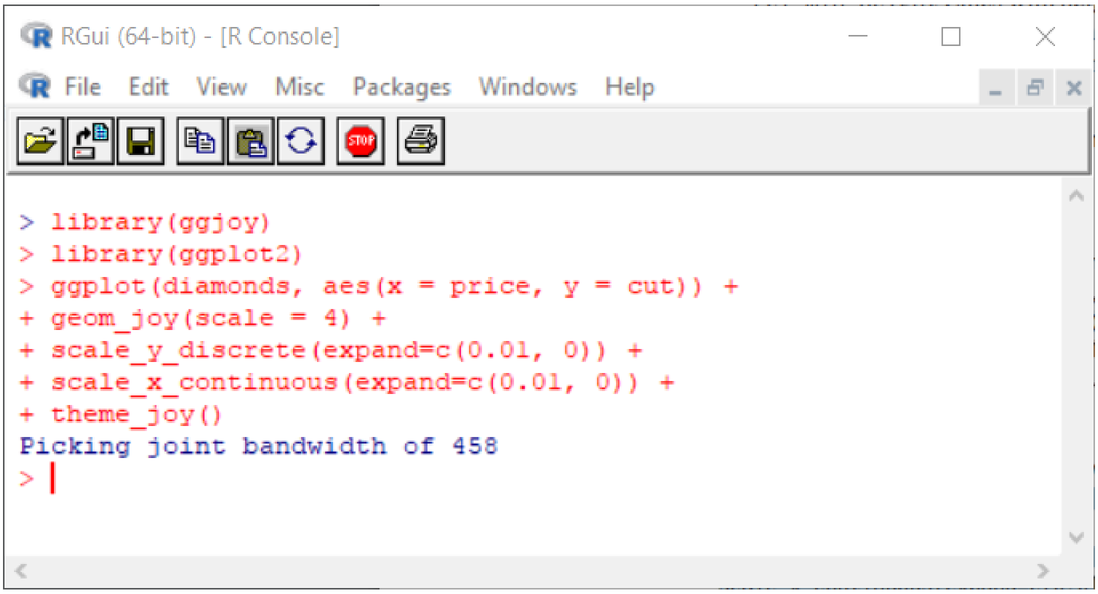
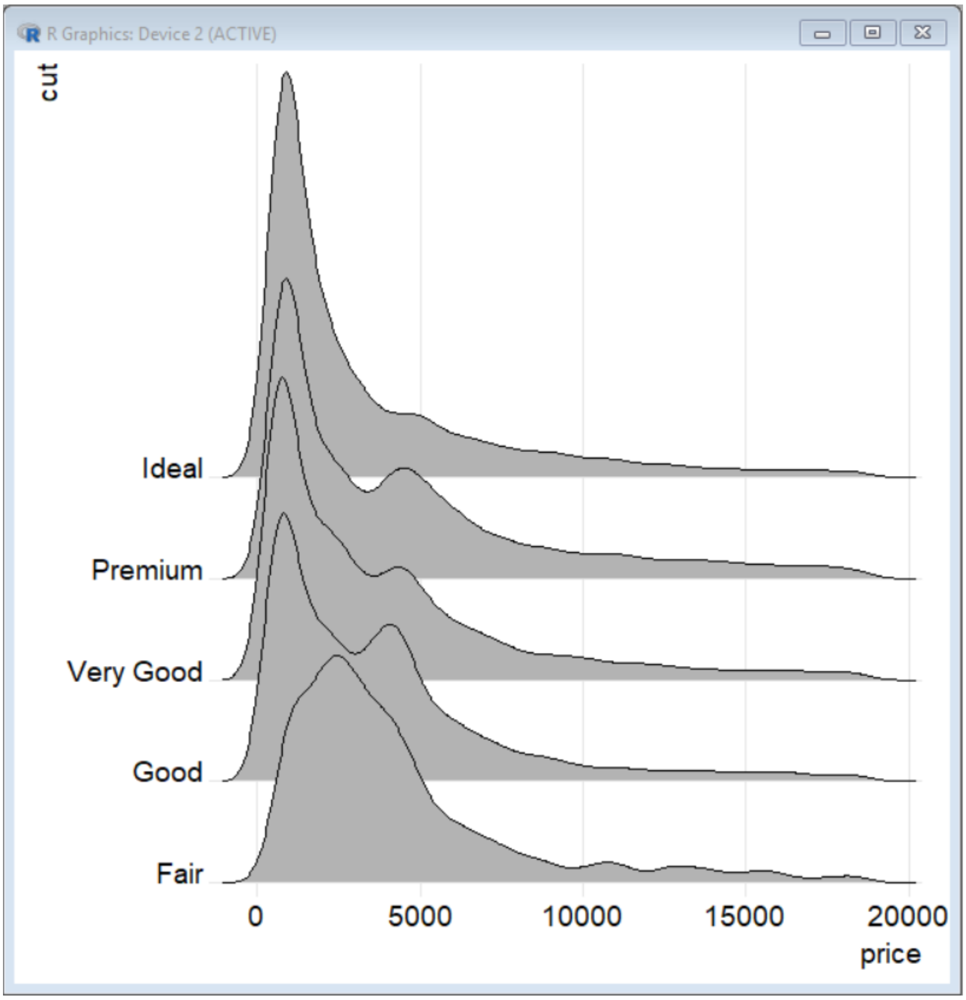


Figure 120 Graph Joy plot

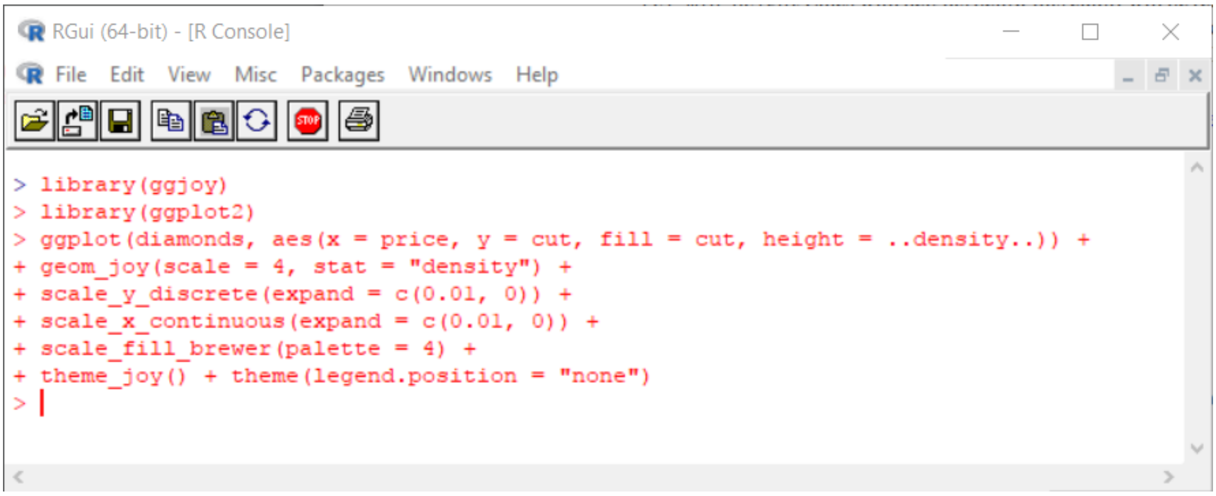
ou encore:



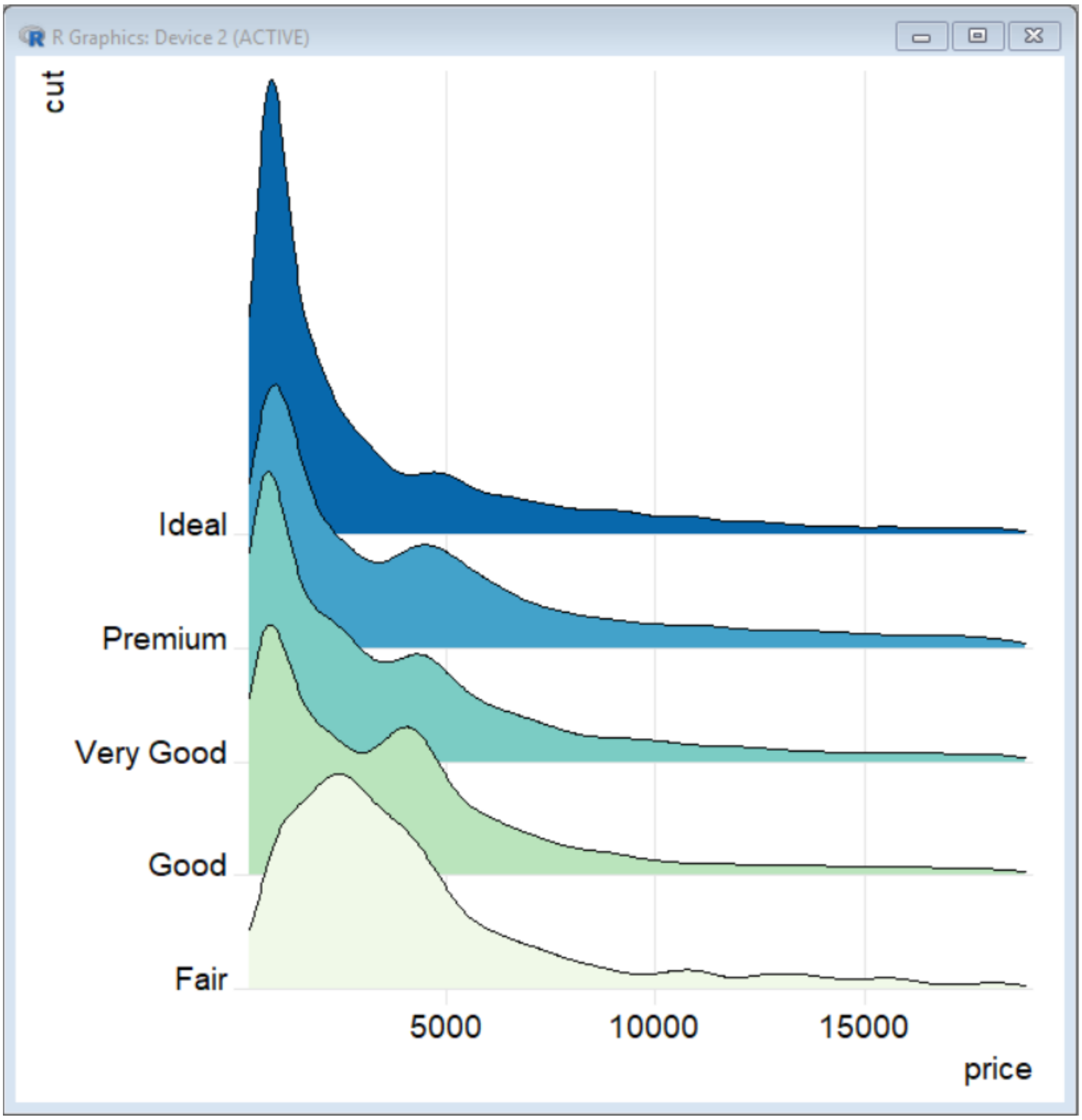
Ce qui donne:



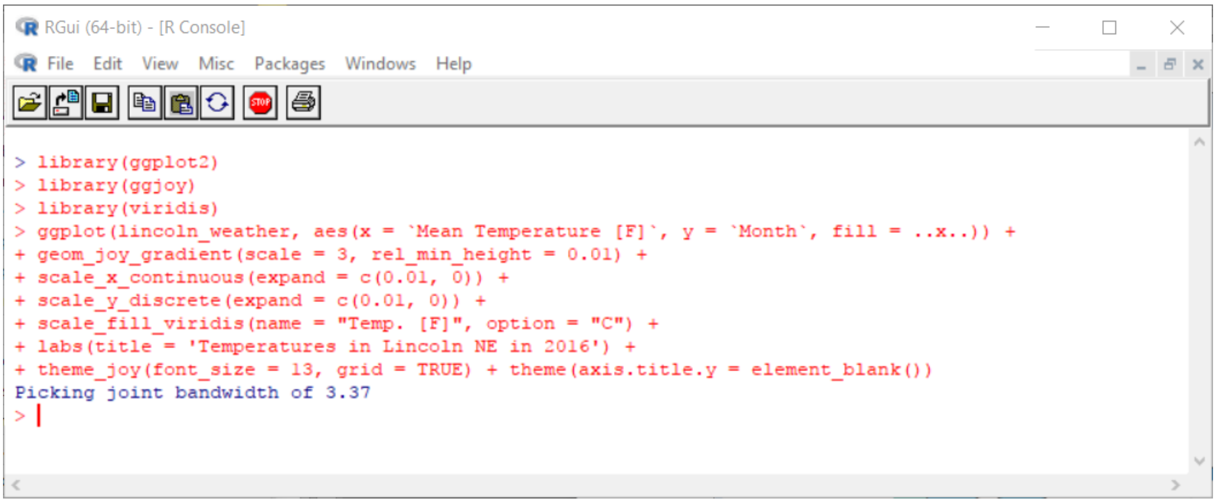
ou encore:



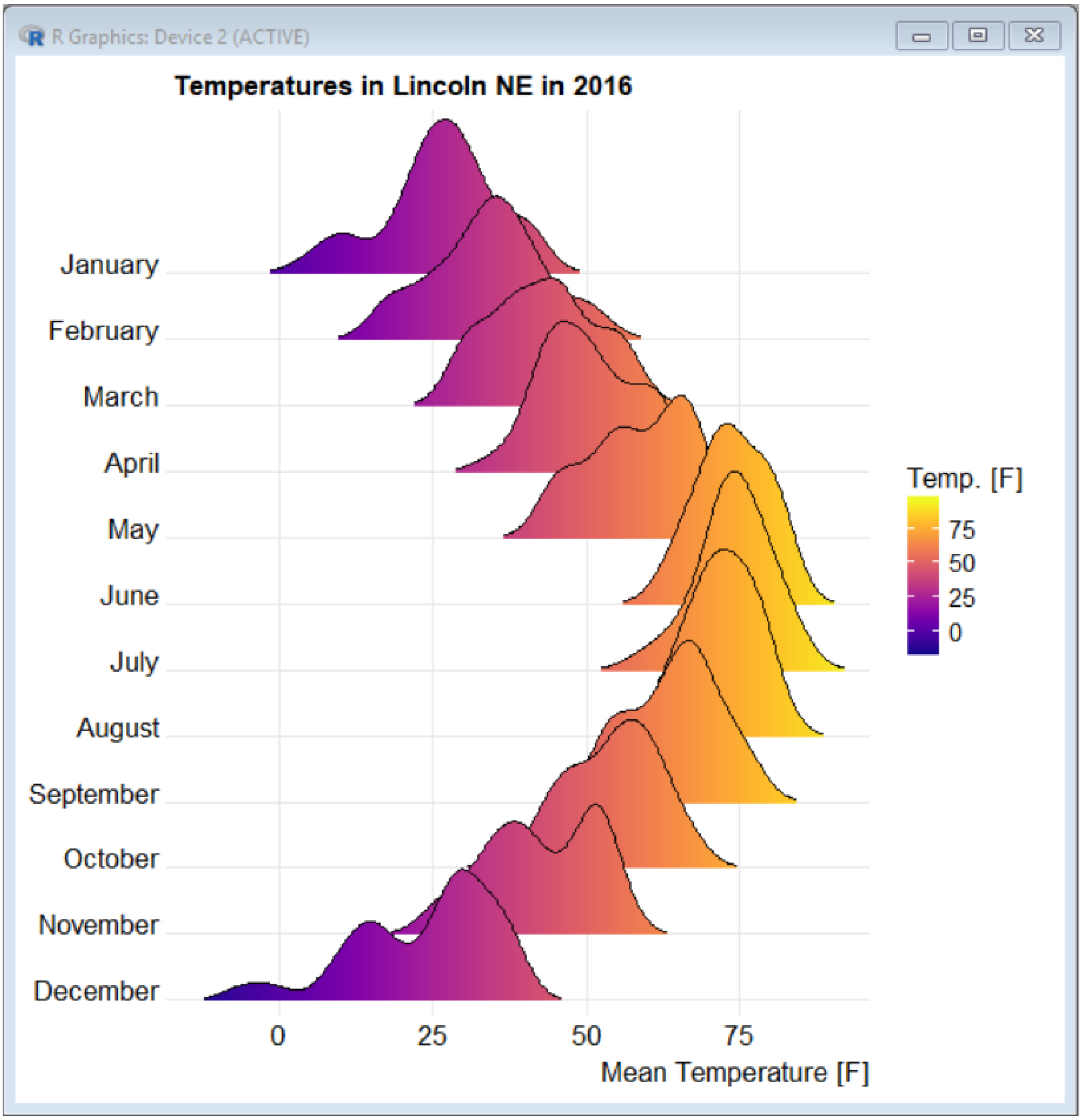
Ce qui donne:



ou encore:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(ggplot2)
> library(ggjoy)
> library(viridis)
> ggplot(lincoln_weather, aes(x = `Mean Temperature [F]`, y = `Month`, fill = ..x..)) +
+ geom_joy_gradient(scale = 3, rel_min_height = 0.01) +
+ scale_x_continuous(expand = c(0.01, 0)) +
+ scale_y_discrete(expand = c(0.01, 0)) +
+ scale_fill_viridis(name = "Temp. [F]", option = "C") +
+ labs(title = 'Temperatures in Lincoln NE in 2016') +
+ theme_joy(font_size = 13, grid = TRUE) + theme(axis.title.y = element_blank())
Picking joint bandwidth of 3.37
>

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Temperatures in Lincoln NE in 2016
January
February
March
April
May
June
July
August
September
October
November
December
0
25
50
75
Mean Temperature [F]
Temp. [F]
75
50
25
0

Exemple 131.: Sparklines

R 3.0.2

Maintenant que nous savons faire des combinaisons graphiques (lattices), nous pouvons jouer avec R pour faire des simili-sparklines (il existe un package pour qui se nomme **sparkTable** mais malheureusement à ce jour il ne peut qu'exporter le résultat en un fichier *.html, *.tex ou *.pdf et non pas l'afficher dans R même).

Voici un exemple que nous laisserons au lecteur le soin de personnaliser ou améliorer (nous mettons en entrée un vecteur aléatoire si l'utilisateur ne précise rien):

C:\Users\Isoz Vincent\Desktop\SparkLines.R - Editeur R

```
sparkline<-function(ydata=rnorm(100,500,50),width=1.5,height=0.5,sigfigs=4) {

      # ydata = vector of data to be plotted
      # width = width of sparlkline in inches, including text
      # height = height of sparkline in inches
      # sigfigs = number of significant figures to round min, max, and last values to
    len<-length(ydata) # determine the length of the data set
    ymin<-min(ydata) # determine the minimum
    tmin<-which.min(ydata) # and its index
    ymax<-max(ydata) # determine the maximum
    tmax<-which.max(ydata) # and its index
    yfin<-signif(ydata[len],sigfigs) #determine most recent data point
      # define plot range to leave enough room for min and max circles and text
    plotrange=c(ymin-0.1*(ymax-ymin),ymax+0.1*(ymax-ymin))
    plot(x=1:len,y=ydata,type="l",xlim=c(1,len*1.2),ylim=plotrange,col="gray",
    lwd=0.5,ann=FALSE,axes=FALSE) # plot sparkline
    points(x=c(tmin,tmax),y=c(ymin,ymax),pch=19,col=c("red","blue"),cex=1) # plot min and max points
    text(x=len,y=ymin,labels=signif(ymin,sigfigs),cex=0.5,pos=4,col="red") # show minimum value
    text(x=len,y=ymax,labels=signif(ymax,sigfigs),cex=0.5,pos=4,col="blue") # show maximum value
    text(x=len,y=(ymin+ymax)/2,labels=yfin,cex=0.5,pos=4) # show most recent value
}
```

Et si nous appelons la fonction 4 fois:

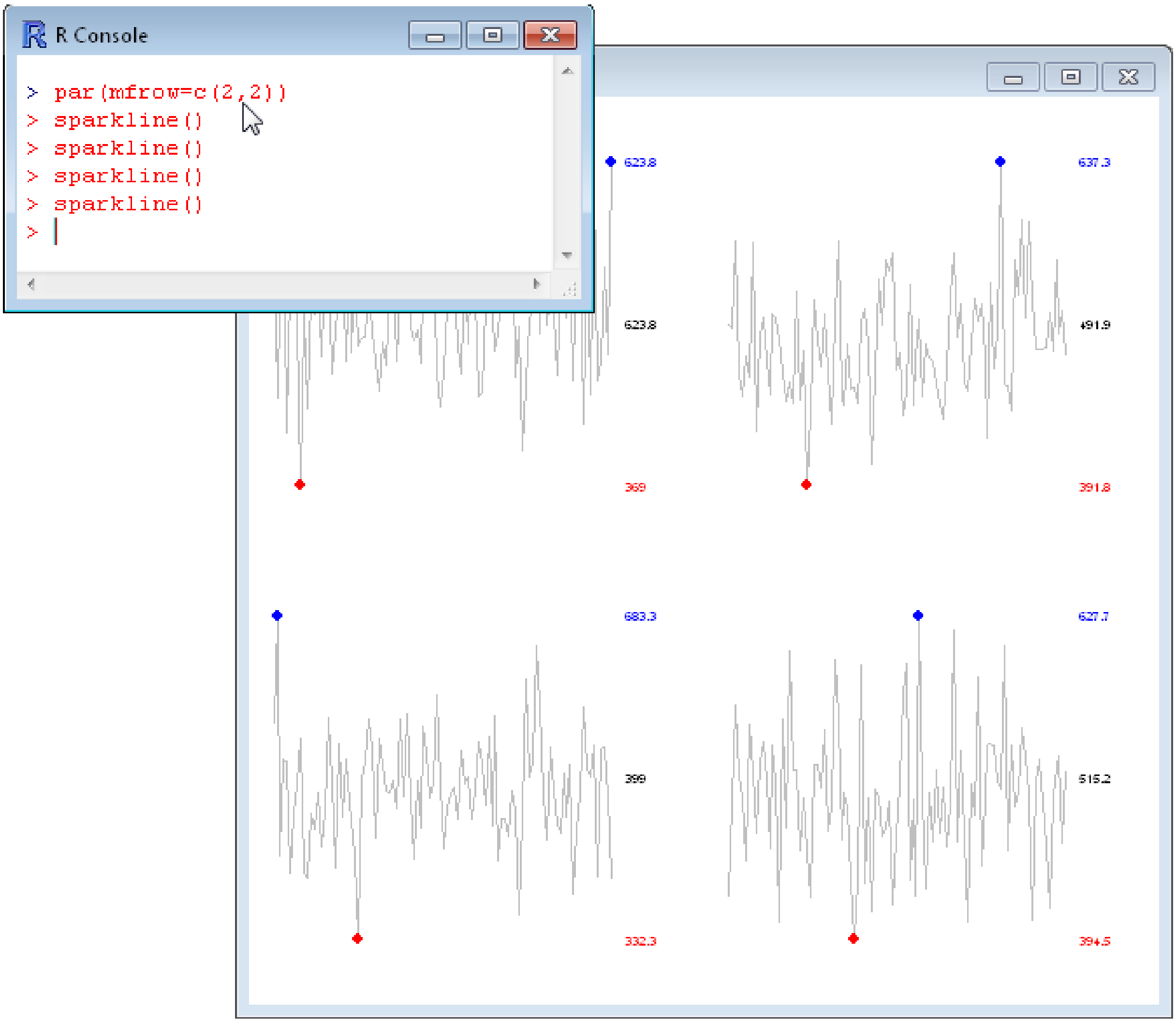


Figure 121 Graph Sparkline

Exemple 132.: Fonction de répartition empirique (FRE)

R 3.0.2

Une analyse de la FRE n'a pas le problème de l'analyse par densité où se pose la question des largeurs des intervalles. Il vaut donc mieux prendre l'habitude de lire les FRE plutôt que les FDP (fonctions de densité de probabilité).

Voici un exemple d'utilisation de la commande **ecdf( )** :

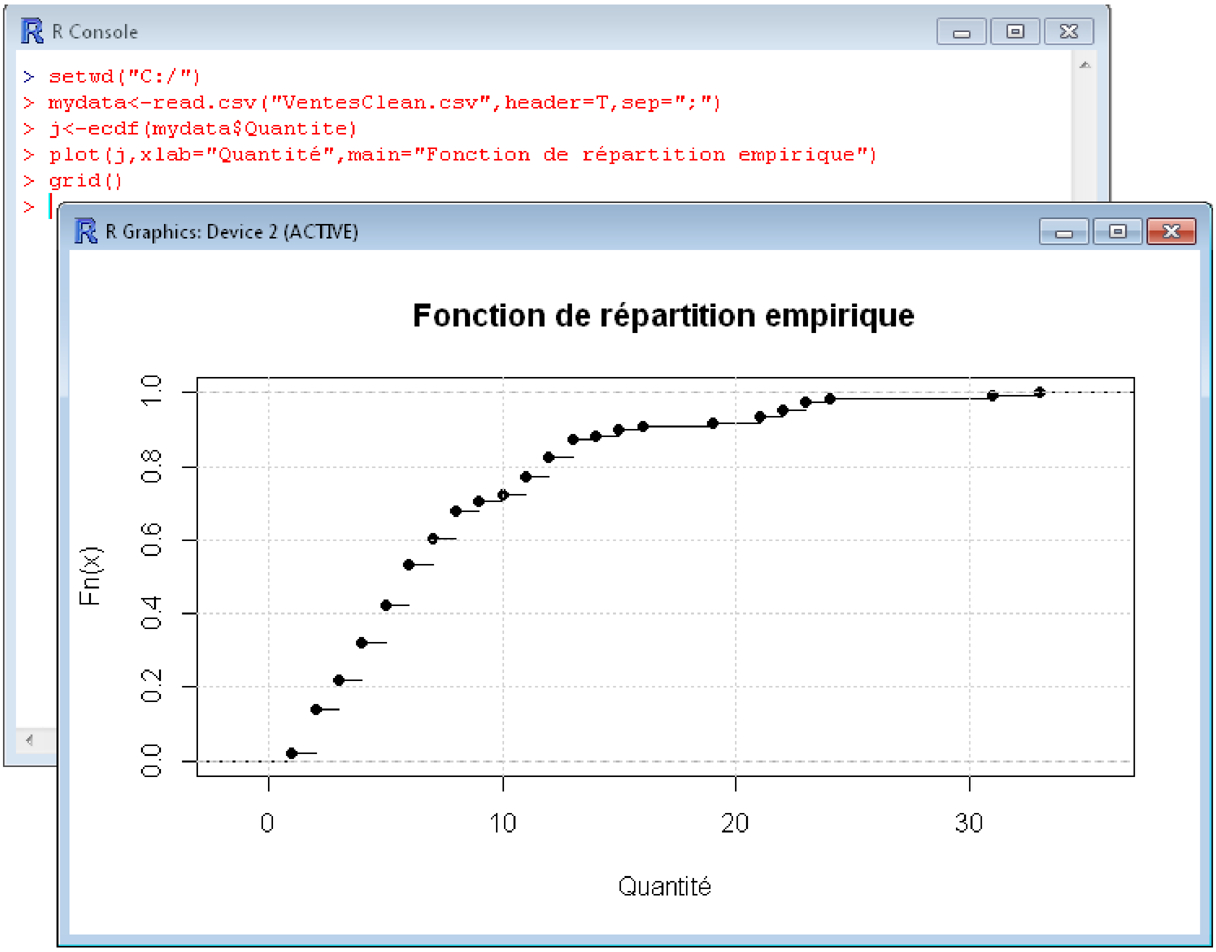
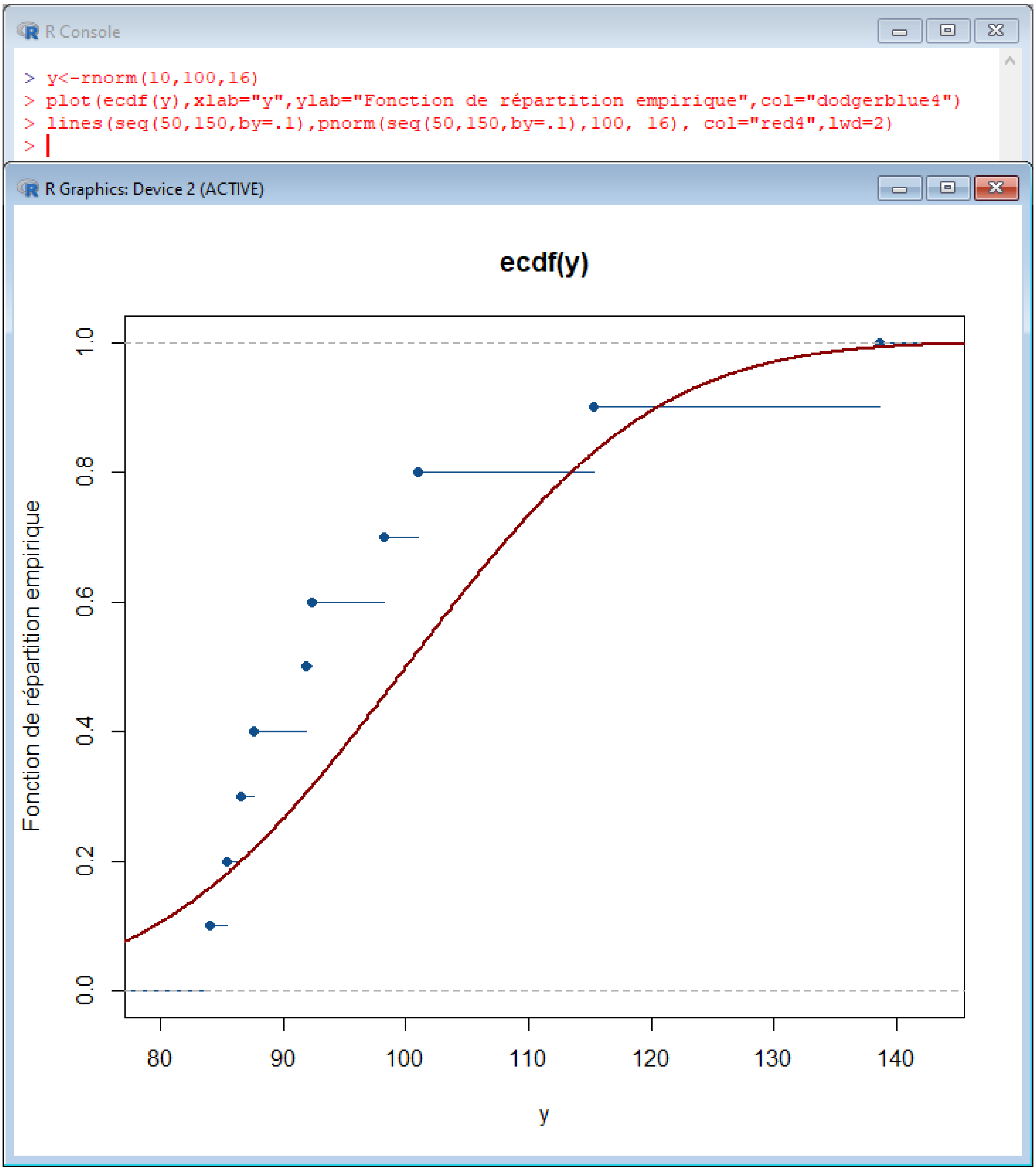
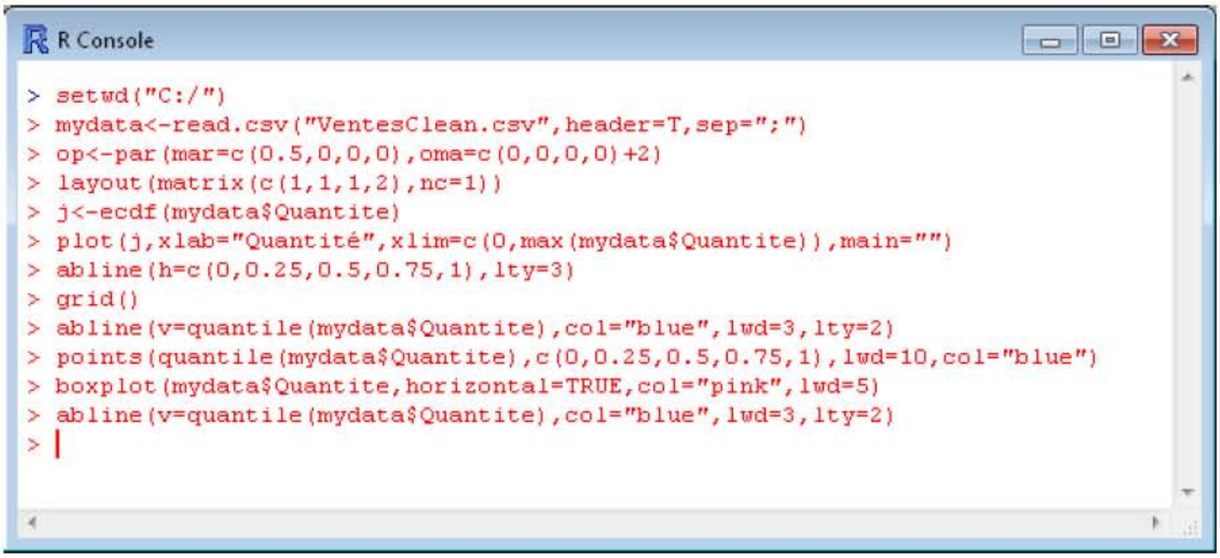


Figure 122 Graph de répartition empirique

Nous pouvons cependnat faire mieux:



Nous pouvons cependant faire cependant encore mieux (je ne suis pas encore arrivé à résoudre le problème du décalage entre la box plot et la ECDF) en utilisant la fonction **ecdf( )**:



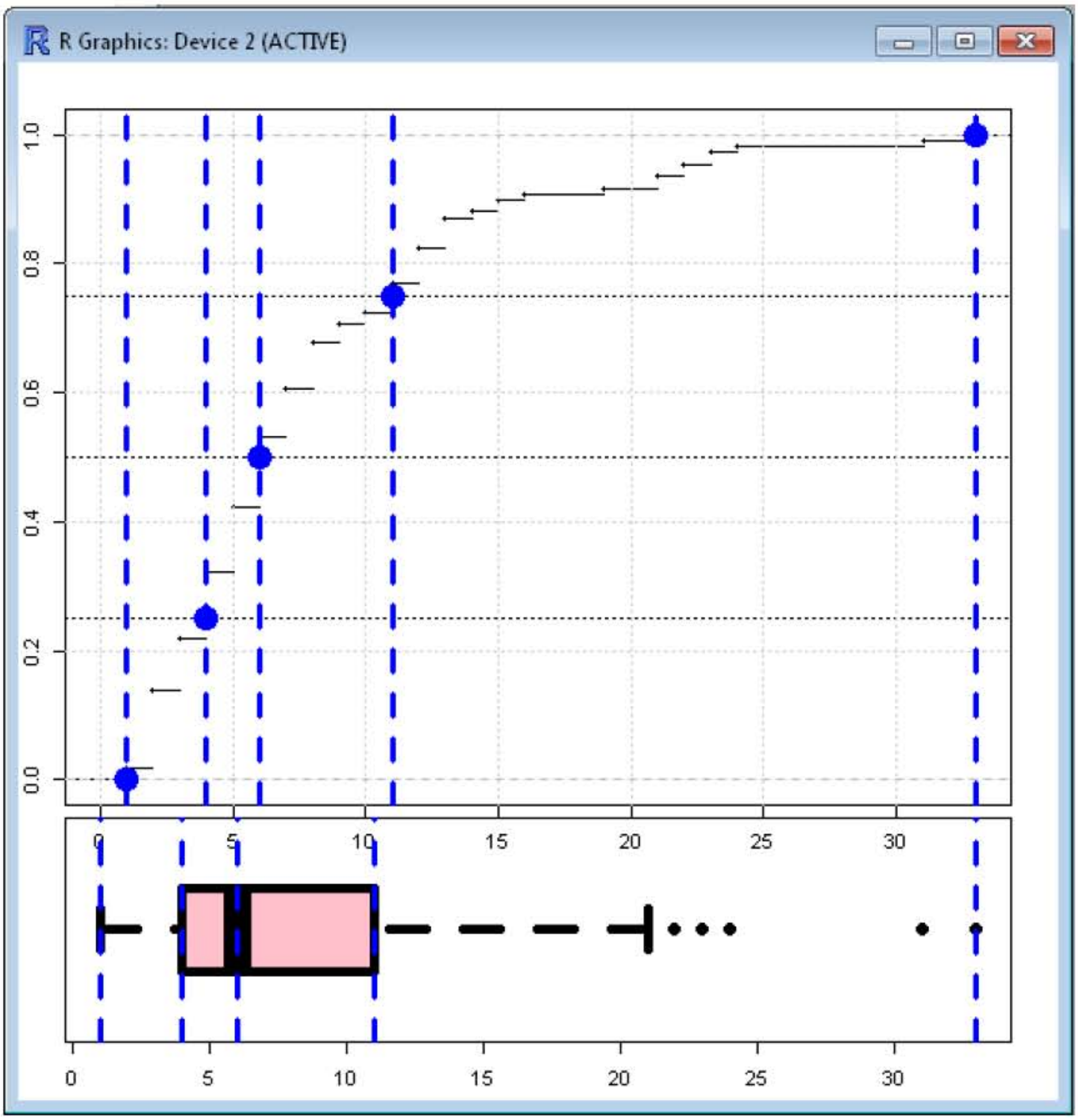
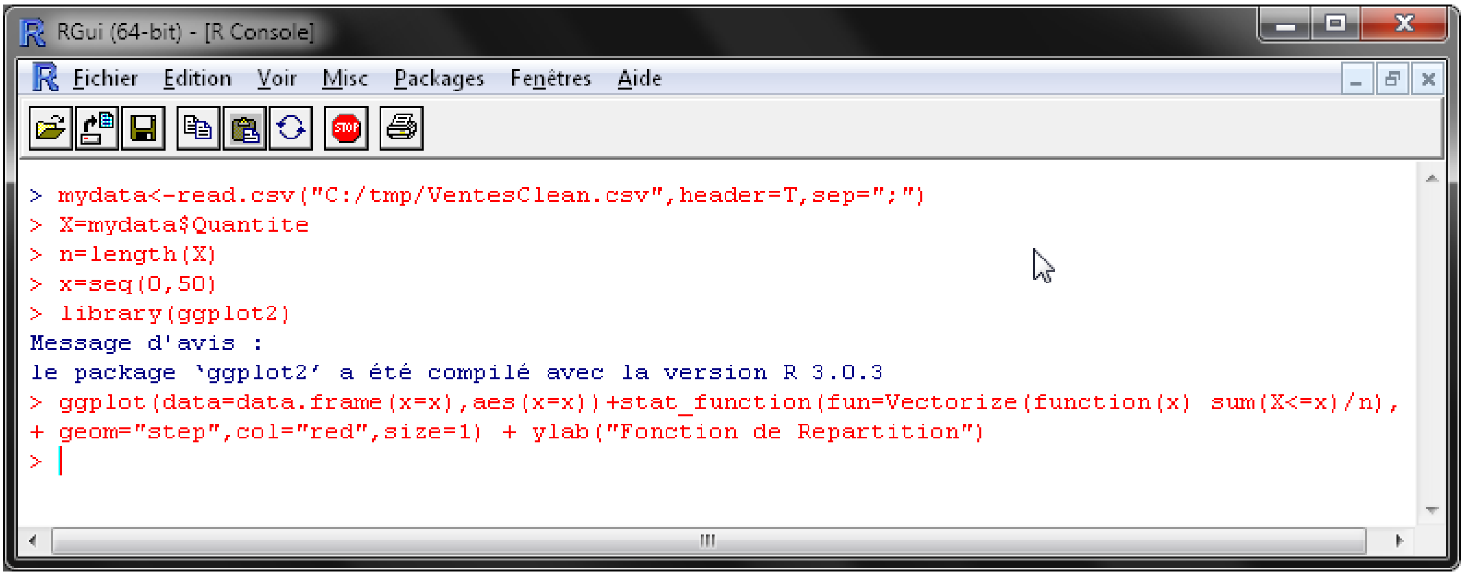


Figure 123 Graph avec fonction de répartition empirique avec box-plot en marge

Ou pour ceux qui préfèrent avec la package **ggplot**:



Ce qui donnera:

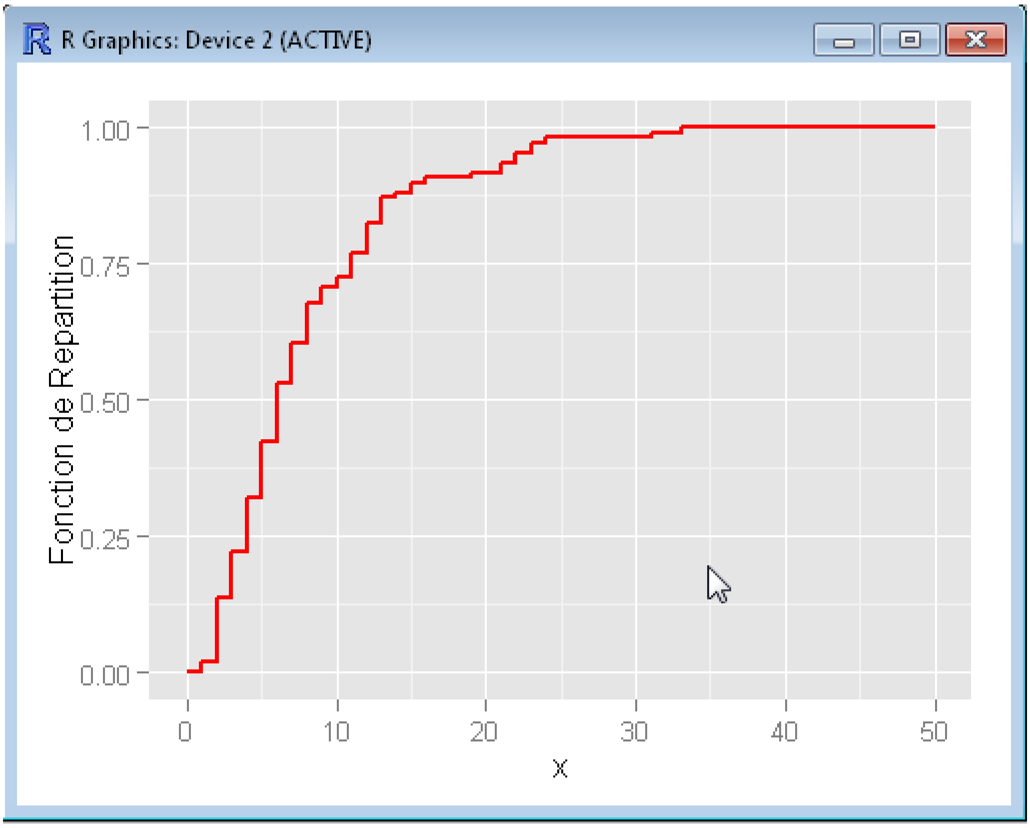


Figure 124 Graph avec fonction de répartition empirique (ggplot)

Il est possible aussi à l'aide du package classInt et de la fonction **classIntervals( )** d'avoir un résumé numérique qui parfois est plus utile qu'un graphique:

```
> mydata<-read.csv("C:/tmp/Ventes.csv",header=T,sep=";")
> library(classInt)
> classIntervals(mydata$Quantité,7,style="equal")
style: equal
  one of 74,613 possible partitions of this variable into 7 classes
       [1,5.571429) [5.571429,10.14286) [10.14286,14.71429) [14.71429,19.28571) [19.28571,23.85714)
                 46                  33                  17                   4                   6
[23.85714,28.42857)       [28.42857,33]
                  1                   2
>
```

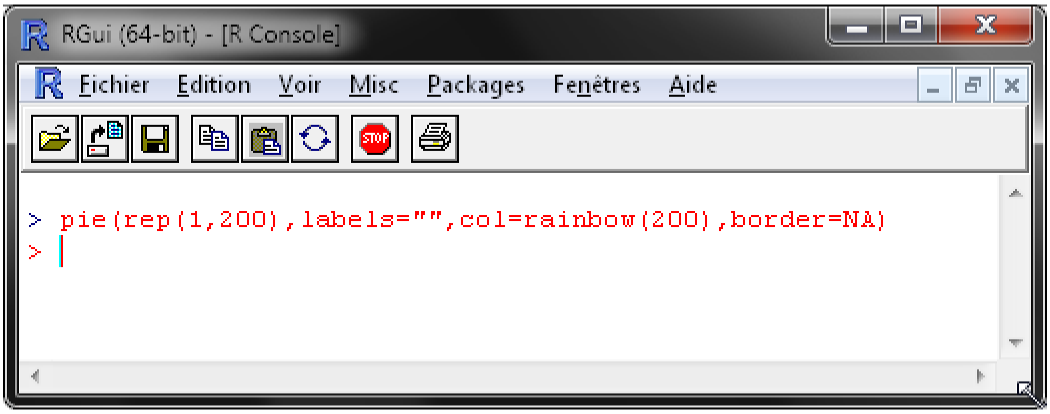
ou en se basant sur les quantiles:

```
> classIntervals(mydata$Quantité,7,style="quantile")
style: quantile
  one of 74,613 possible partitions of this variable into 7 classes
  [1,3)   [3,4)   [4,6)   [6,7)  [7,10) [10,13) [13,33]
     15       9      22      12      19      13      19
>
```

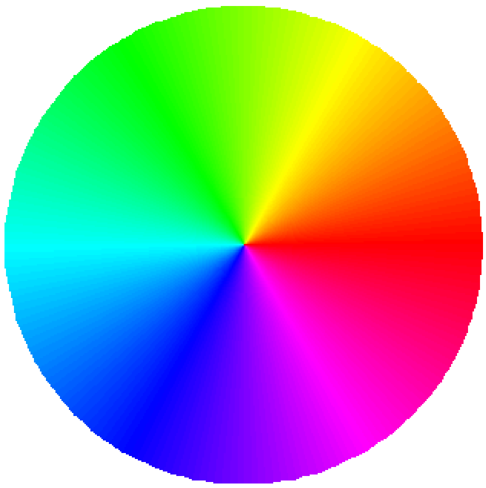
Exemple 133.: Camemberts (fromages) 2D et 3D

R 3.0.2

Commençons par un camembert tout simple en utilisant la commande **pie( )** :



cela donnera:



Ensuite, faisons un camembert un peu plus intéressant. Résumons d'abord nos données de ventes par article:

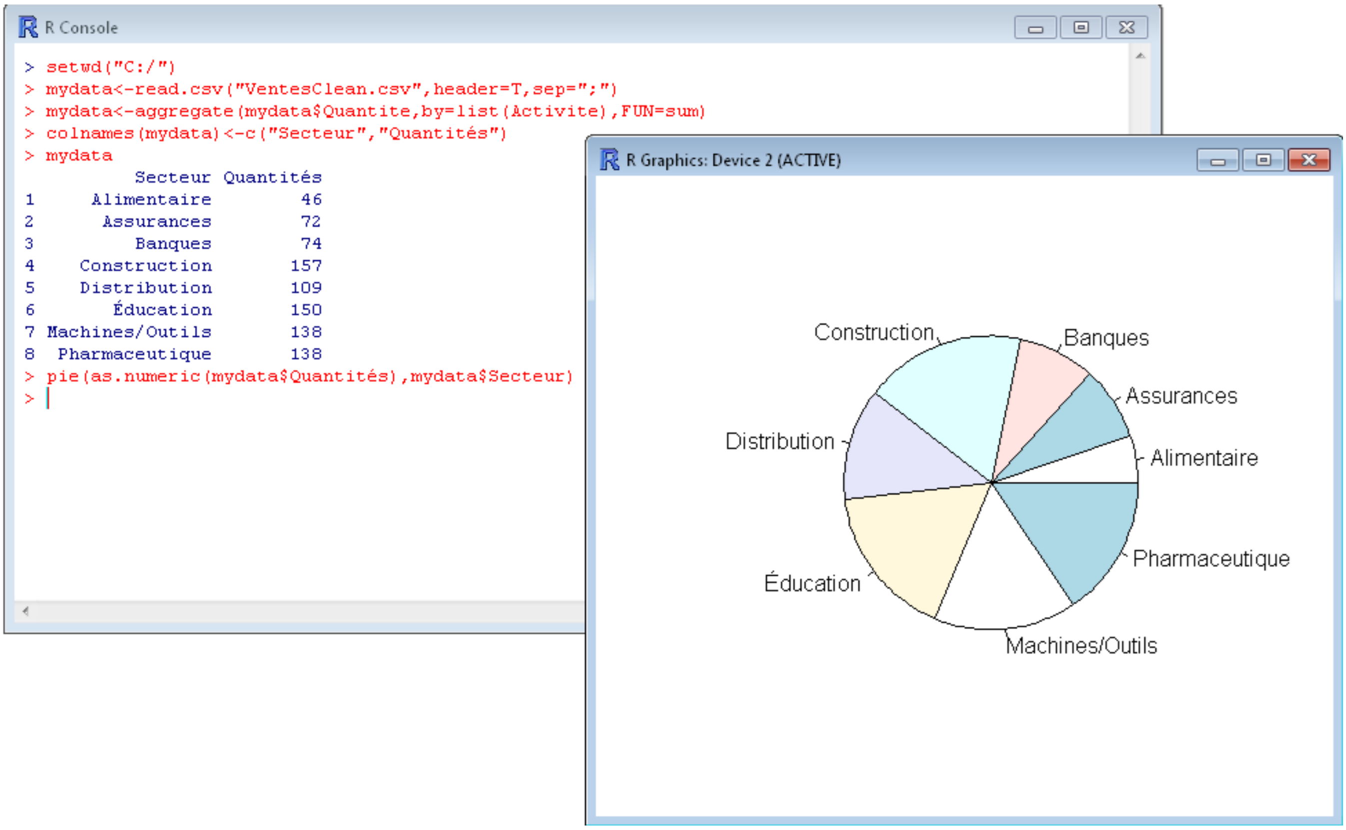
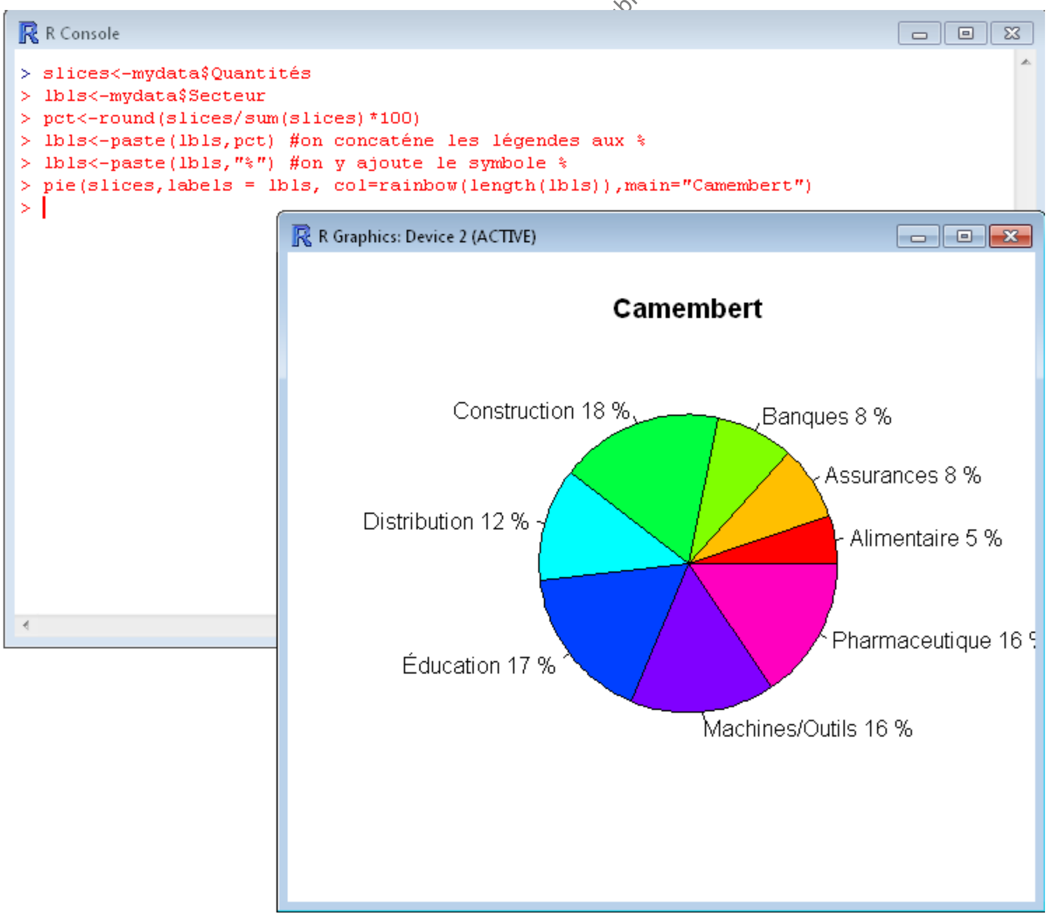


Figure 125 Graph de type camembert

Nous pouvons faire mieux heureusement:



ou encore mieux avec ggplot2:

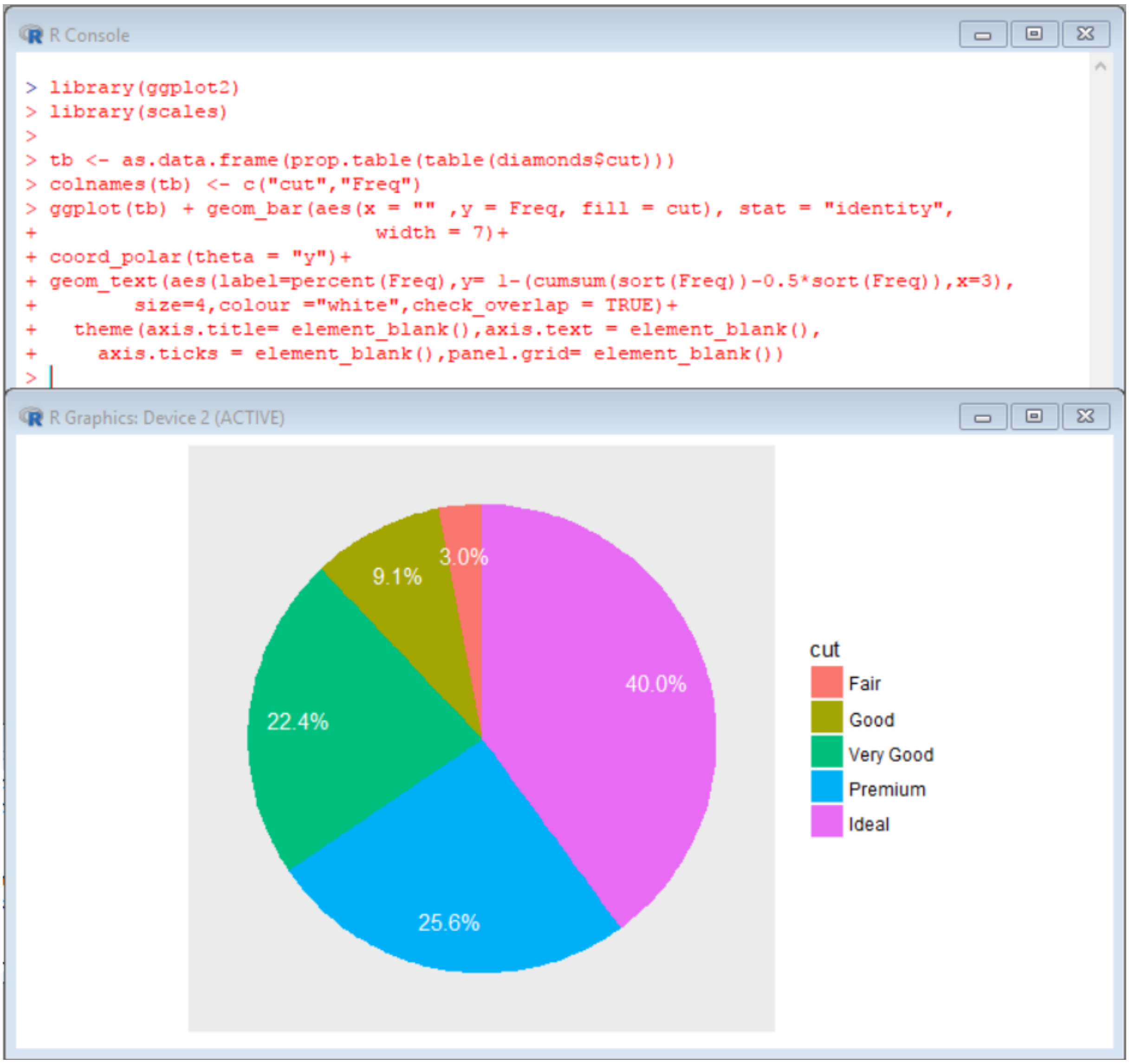
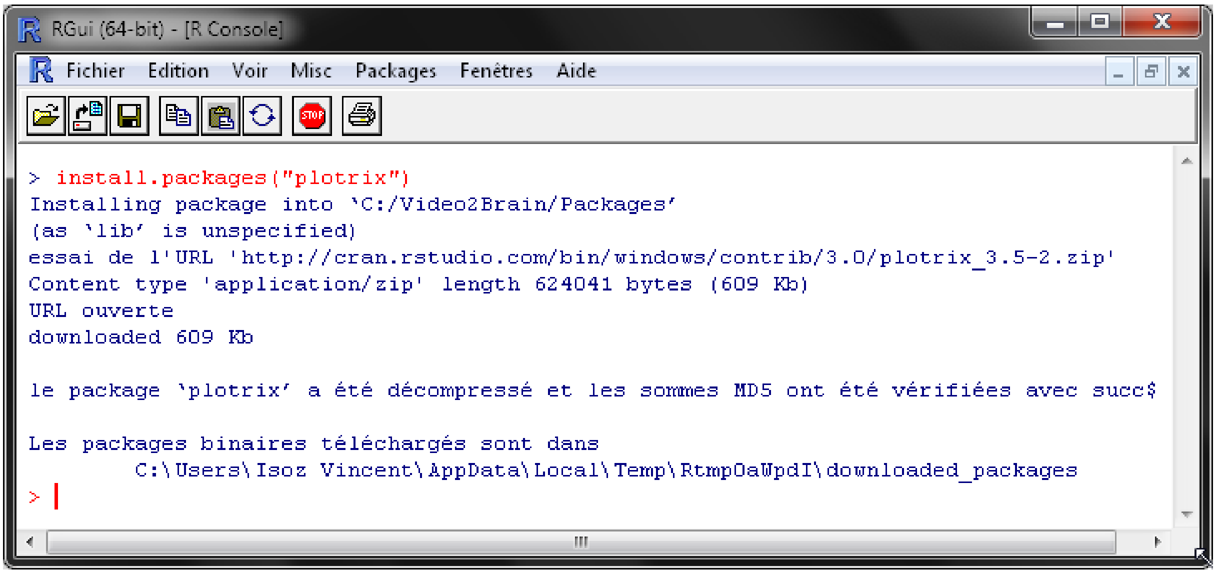


Figure 126 Graph de type camembert (ggplot)

Pour faire un camembert 3D, nous chargeons installons le package **plotrix**:



ce qui nous donnera avec la commande **pie3D( )** :

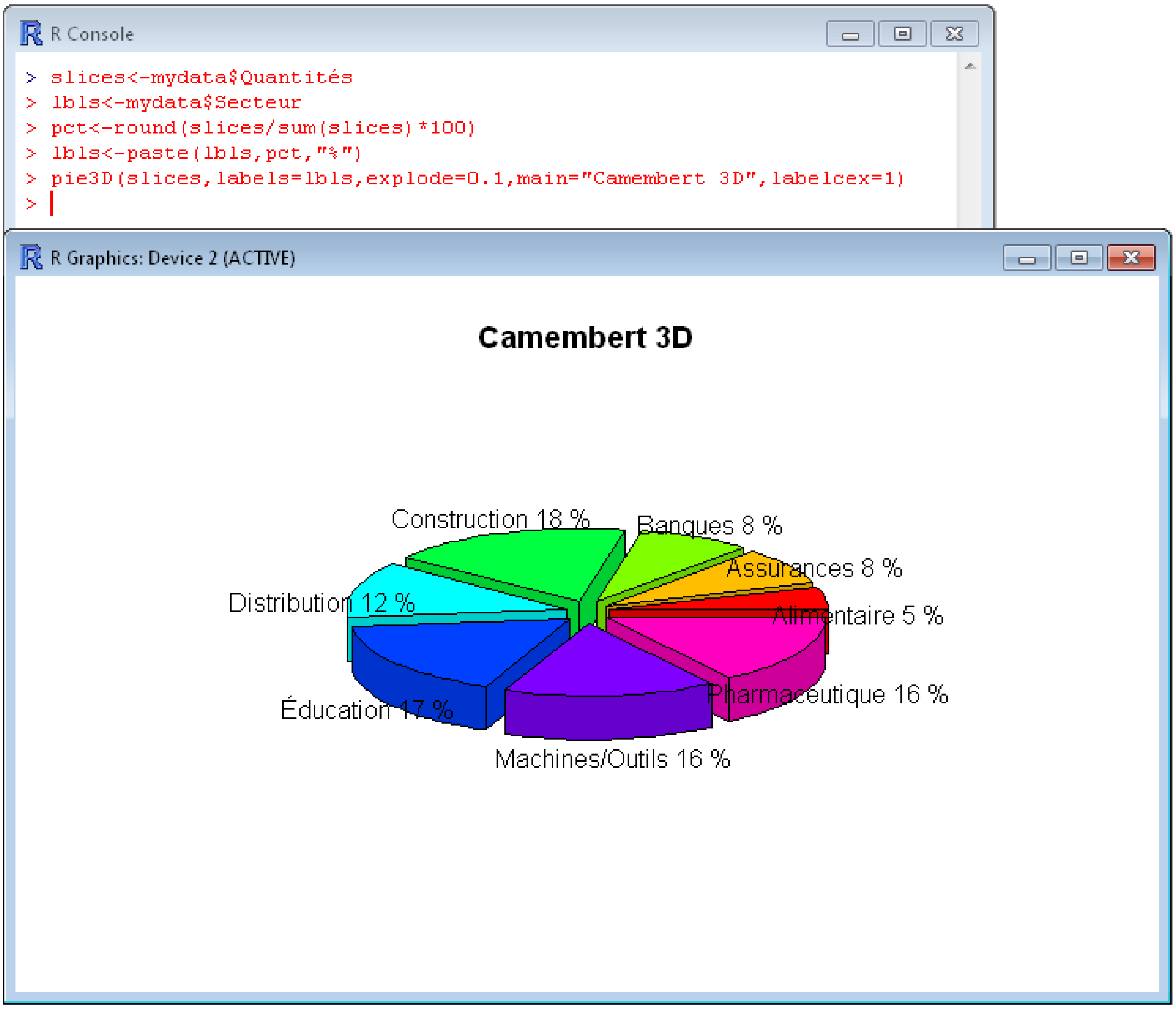


Figure 127 Graph de type camembert en 3D

Exemple 134.: Diagramme en anneaux

R 3.0.2

Pour faire un diagramme en anneaux la procédure est très loin d'être intuitive. Il faut dans un premier temps avoir installé le package **ggplot2**.

D'abord, nous allons préparer notre jeu des ventes en conséquence pour faire un exemple sympa:

```
> mydata<-read.csv("c:/VentesClean.csv",header=T,sep=";")
> #On agrège les factures payées
> mydataYes<-mydata[which(mydata$FacturePayee=="Oui"),]
> mydataYes<-aggregate(mydataYes$Quantite,by=list(mydataYes$Activite),FUN=length)
> colnames(mydataYes)<-c("Secteur","NombresYes")
> #On agrège les factures non payées
> mydataNo<-mydata[which(mydata$FacturePayee=="Non"),]
> mydataNo<-aggregate(mydataNo$Quantite,by=list(mydataNo$Activite),FUN=length)
> colnames(mydataNo)<-c("Secteur","NombresNo")
> #on fusionne les deux pour garder que le ratio des payées sur la totalité
> mydata<-cbind(mydataYes[,1:2],NombresNo=mydataNo[,2],RatioPaied=
+ mydataYes[,2]/(mydataNo[,2]+mydataYes[,2])*100);mydata
          Secteur NombresYes NombresNo RatioPaied
1     Alimentaire          5         2   71.42857
2      Assurances          7         3   70.00000
3         Banques          4         5   44.44444
4    Construction         12         9   57.14286
5    Distribution          8         5   61.53846
6       Éducation         13         6   68.42105
7 Machines/Outils          9         9   50.00000
8  Pharmaceutique          3         9   25.00000
> myd<-mydata[,c(1,4)]
> colnames(myd)<-c("category","percent")
> myd$category<-factor(myd$category,levels=rev(myd$category))
> myd
         category  percent
1     Alimentaire 71.42857
2      Assurances 70.00000
3         Banques 44.44444
4    Construction 57.14286
5    Distribution 61.53846
6       Éducation 68.42105
7 Machines/Outils 50.00000
8  Pharmaceutique 25.00000
> 
```

Ensuite, nous écrivons le code pour faire le graphe:

```
> library(ggplot2)
> plt<-ggplot(myd,aes(x=category,y=percent,fill=category)) +
+ geom_bar(width=0.85,stat="identity")+coord_polar(theta="y") +
+ xlab("")+ ylab("")+ylim(c(0,100))+ggtitle("Graph Anneaux") +
+ geom_text(data=myd,hjust=1,size=3,aes(x=category,y=0,label=category))
>
> plt + scale_fill_manual(values=c(rainbow(8))) + theme_minimal() +
+ theme(legend.position="none",panel.grid.major=element_blank()
+ ,panel.grid.minor=element_blank(),axis.line=element_blank(),axis.text.y=element_blank()
+ ,axis.text.x=element_blank(),axis.ticks = element_blank())
>
```

Ce qui donne (pour l'esthétique par rapport à Microsoft Excel on se passera de commentaires là aussi...):

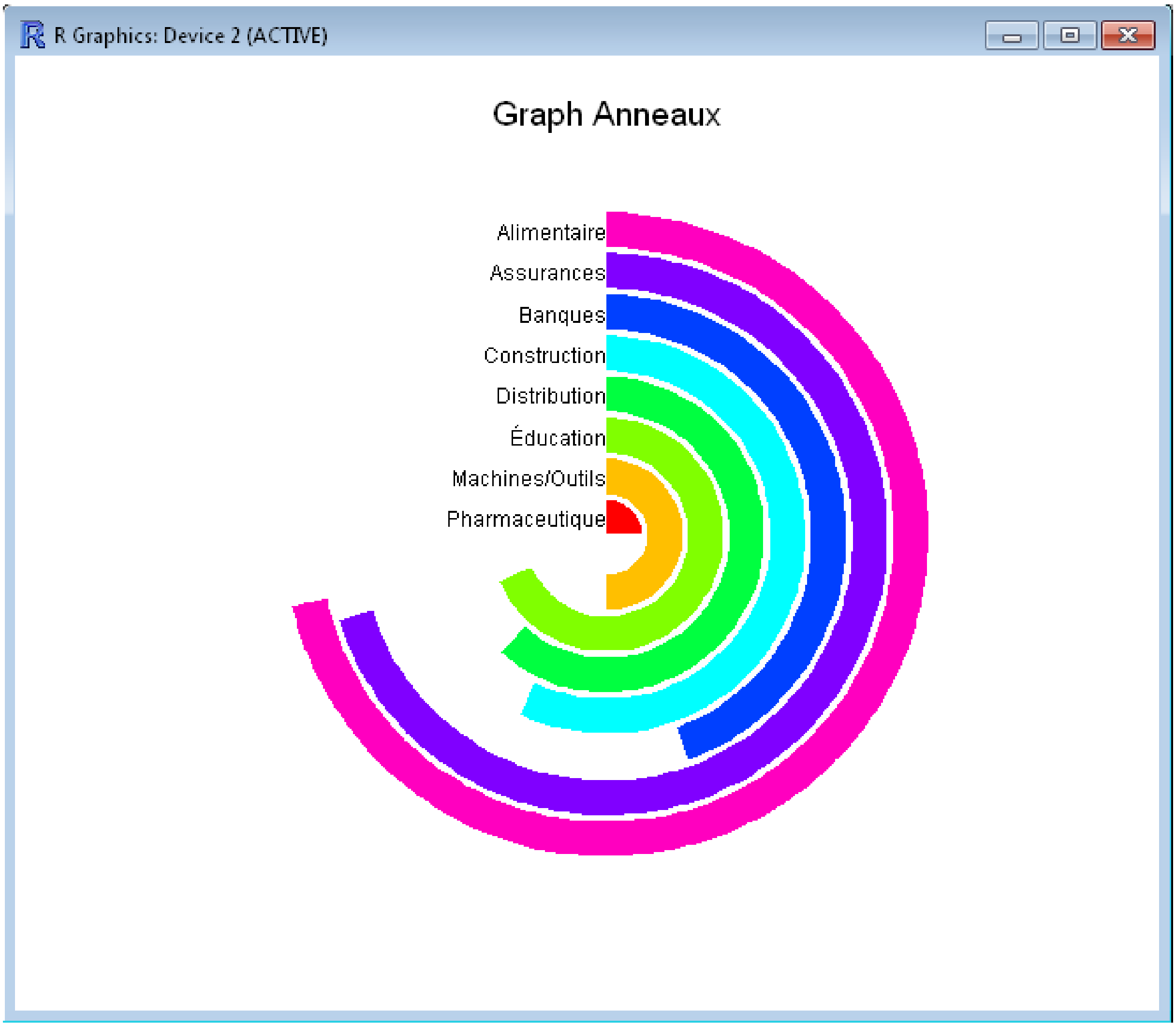
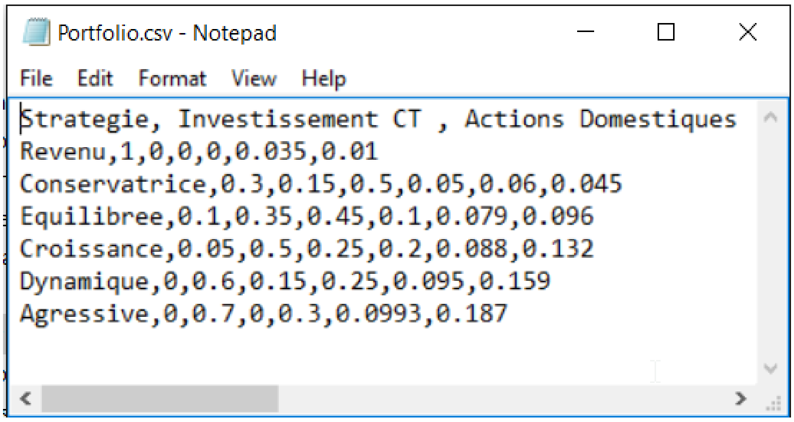


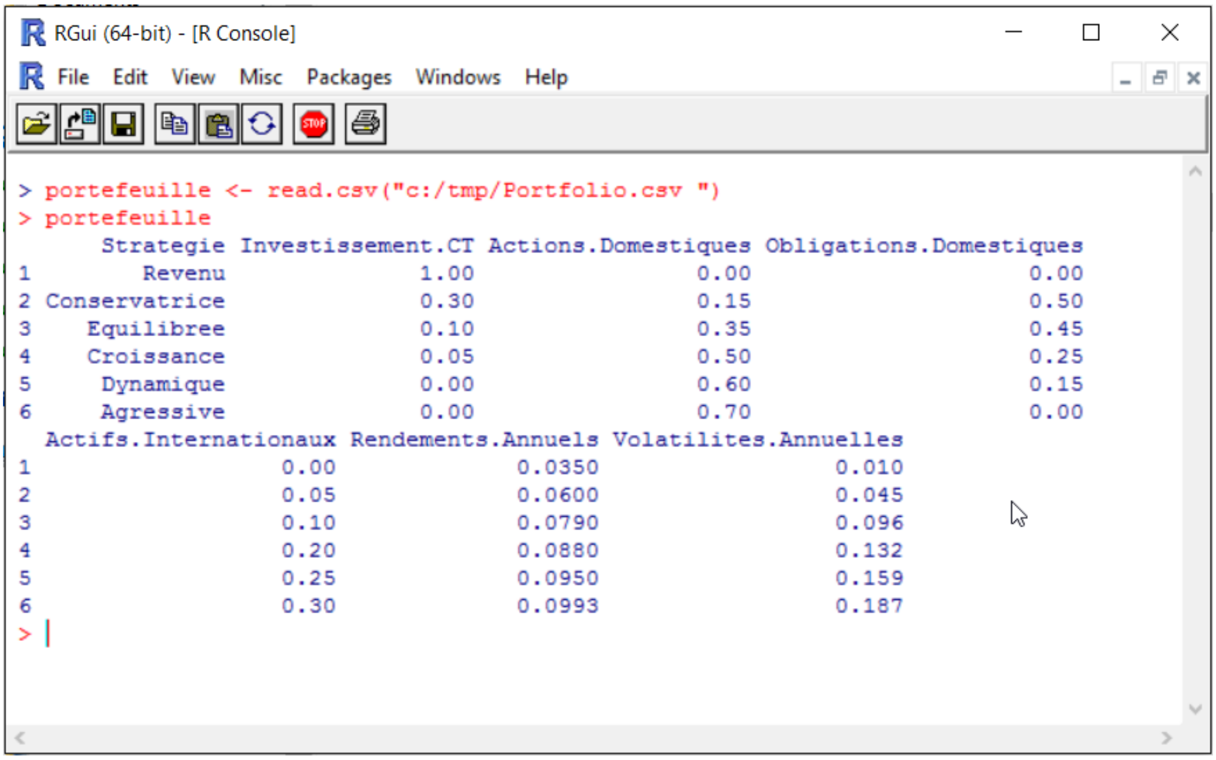
Figure 128 Graph en anneaux

Enfin voyons une autre méthode (merci à Daname) ne nécessitant pas de package. Considérons le fichier suivant:



```
Strategie, Investissement CT , Actions Domestiques
Revenu,1,0,0,0,0.035,0.01
Conservatrice,0.3,0.15,0.5,0.05,0.06,0.045
Equilibree,0.1,0.35,0.45,0.1,0.079,0.096
Croissance,0.05,0.5,0.25,0.2,0.088,0.132
Dynamique,0,0.6,0.15,0.25,0.095,0.159
Agressive,0,0.7,0,0.3,0.0993,0.187
```

et importons-le:



```
> portefeuille <- read.csv("c:/tmp/Portfolio.csv ")
> portefeuille
      Strategie Investissement.CT Actions.Domestiques Obligations.Domestiques
1        Revenu              1.00                0.00                    0.00
2 Conservatrice              0.30                0.15                    0.50
3    Equilibree              0.10                0.35                    0.45
4    Croissance              0.05                0.50                    0.25
5     Dynamique              0.00                0.60                    0.15
6     Agressive              0.00                0.70                    0.00
  Actifs.Internationaux Rendements.Annuels Volatilites.Annuelles
1                  0.00             0.0350                 0.010
2                  0.05             0.0600                 0.045
3                  0.10             0.0790                 0.096
4                  0.20             0.0880                 0.132
5                  0.25             0.0950                 0.159
6                  0.30             0.0993                 0.187
> 
```

On restructure un peu:

```
> #distinguer les indicateurs des parts
> #grouper les actifs par stratégie de portfeuille
> strategie<-split(portefeuille[2:5],portefeuille$Strategie)
> strategie
$Agressive
  Investissement.CT Actions.Domestiques Obligations.Domestiques
6                 0                 0.7                       0
  Actifs.Internationaux
6                   0.3

$Conservatrice
  Investissement.CT Actions.Domestiques Obligations.Domestiques
2               0.3                0.15                     0.5
  Actifs.Internationaux
2                  0.05

$Croissance
  Investissement.CT Actions.Domestiques Obligations.Domestiques
4              0.05                 0.5                    0.25
  Actifs.Internationaux
4                   0.2

$Dynamique
  Investissement.CT Actions.Domestiques Obligations.Domestiques
5                 0                 0.6                    0.15
  Actifs.Internationaux
5                  0.25

$Equilibree
  Investissement.CT Actions.Domestiques Obligations.Domestiques
3               0.1                0.35                    0.45
  Actifs.Internationaux
3                   0.1

$Revenu
  Investissement.CT Actions.Domestiques Obligations.Domestiques
1                 1                   0                       0
  Actifs.Internationaux
1                     0

> |
```

On choisit une des stratégies:

RGui (64-bit) - [R Console]

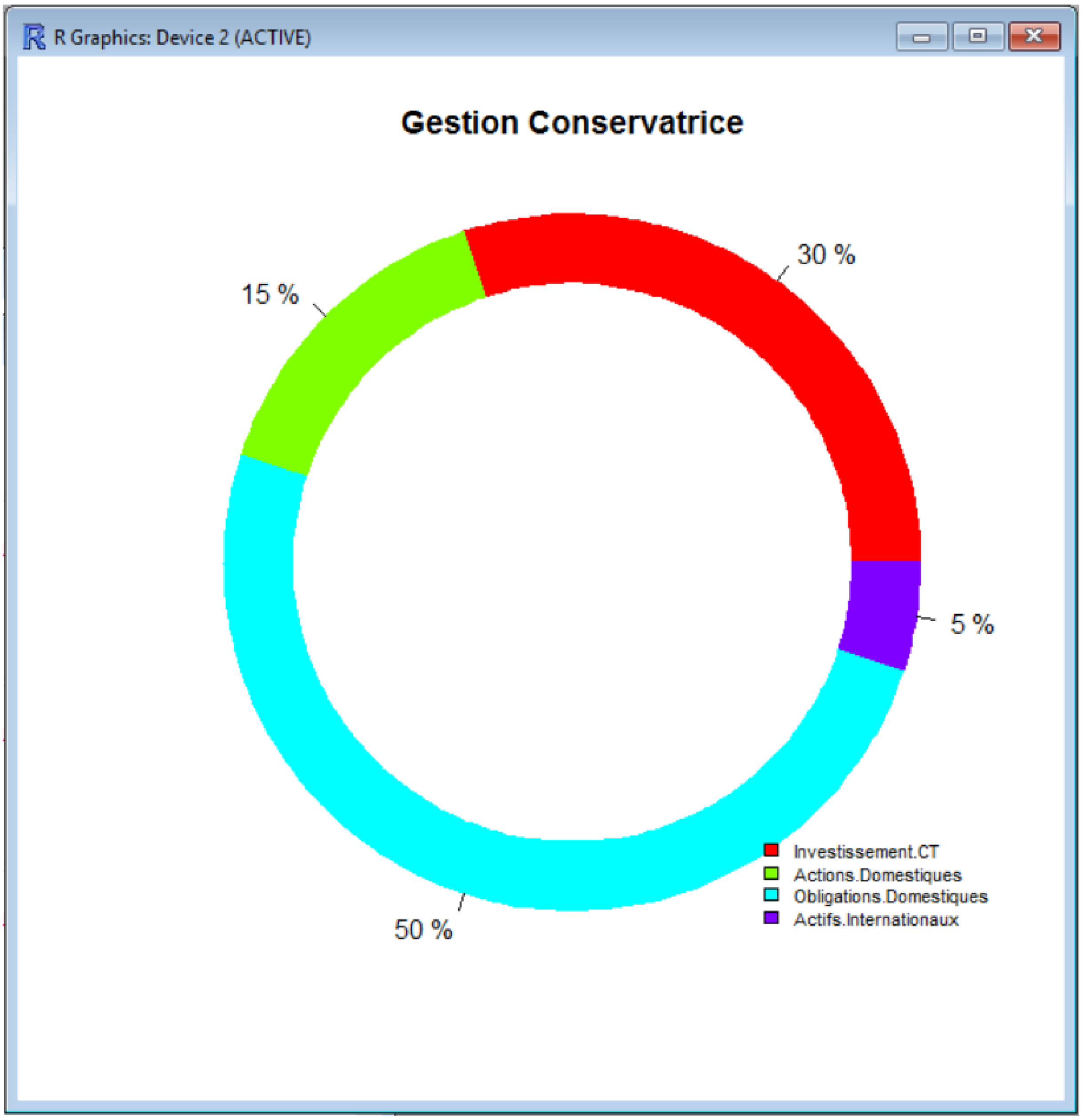
```
> strategie$Conservatrice
  Investissement.CT Actions.Domestiques Obligations.Domestiques Actifs.Internationaux
2               0.3                0.15                     0.5                  0.05
> |
```

Et maintenant allons-y pour l'astuce simple consistant à faire un fromage blanc sans bords dans un fromage normal sans bords lui aussi:

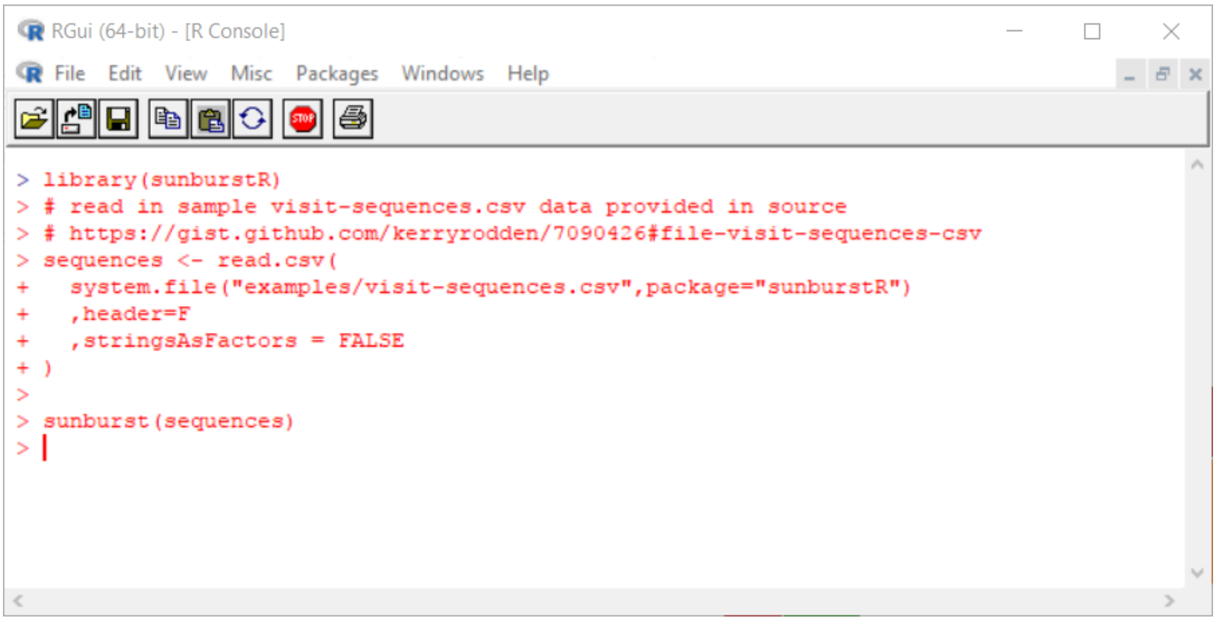
RGui (64-bit) - [R Console]

```
> labs<-paste(round(unlist(strategie$Conservatrice)*100,2),"%")
> couleurs <- rainbow(4)
> pie(unlist(strategie$Conservatrice),col=couleurs,labels=labs,
+ main="Gestion Conservatrice",border=NA,radius=1)
> legend("bottomright",gsub(" \\. "," ",names(strategie$Conservatrice)),cex=0.7,fill=couleurs,bty="n")
> par(new =T)
> pie(unlist(strategie$Conservatrice),radius=0.8,col="white",border=NA,labels=NA)
> |
```

Ce qui donne:



Sinon dans la famille des diagrammes en anneaux nous avons aussi les graphiques en rayons de soleil (Sunburst) avec le package **sunburstR** et sa fonction **sunburst( ) :**



Ce qui donne:

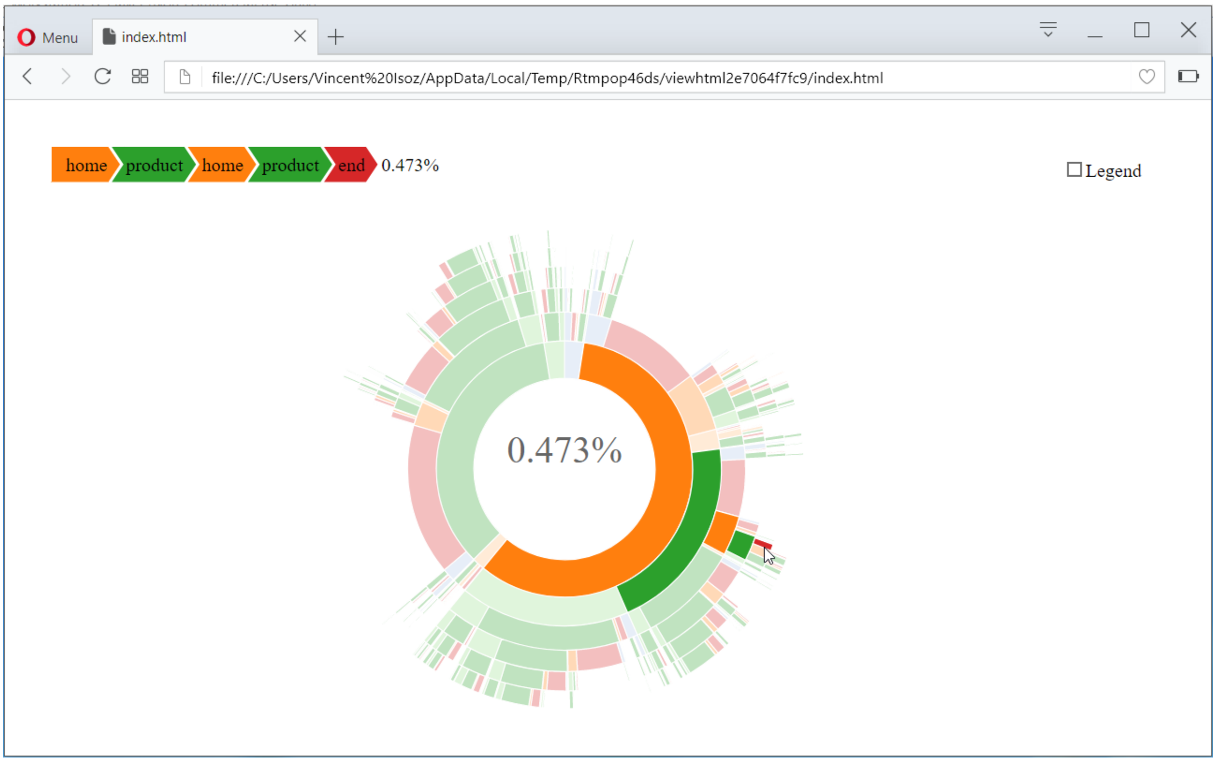


Figure 129 Graph de type Sunburst (rayon de soleil)

Exemple 135.: Diagramme radar (Kiviat)

R 3.0.2

Le diagramme radar va être créé à l'aide du package **plotrix** et de la commande **diamondplot( )** . Il n'est pas évident de préparer la source de données pour ce type de diagramme car l'entrée doit être un data frame dont les noms des catégories doivent se trouver sur l'index de numérotation de ce même data frame (d'où l'utilisation de la commande **row.names( )**):

```
> library(plotrix)
> setwd("C:/")
> mydataMain<-read.csv("Ventes.csv",header=T,sep=";")
> mydata1<-aggregate(mydataMain$Quantité,by=list(mydataMain$Activité),FUN=sum)
> colnames(mydata1)<-c("Secteur","Somme Quantité")
> rownames(mydata1)<-mydata1$Secteur
> mydata1[1]<-NULL
>
> mydata2<-aggregate(mydataMain$Quantité[which(mydataMain$Facture.payée=="Oui")],
+ by=list(mydataMain$Activité[which(mydataMain$Facture.payée=="Oui")]),FUN=sum)
> colnames(mydata2)<-c("Secteur","Somme Quantité Payées")
>
> mydata3<-aggregate(mydataMain$Quantité[which(mydataMain$Facture.payée=="Non")],
+ by=list(mydataMain$Activité[which(mydataMain$Facture.payée=="Non")]),FUN=sum)
> colnames(mydata3)<-c("Secteur","Somme Quantité Payées")
> mydata<-cbind(mydata1,mydata2[2],mydata3[2])
>
> mydata
                Somme Quantité Somme Quantité Payées Somme Quantité Payées
Alimentaire                 46                    36                    10
Assurances                  72                    47                    25
Banques                     74                    29                    45
Construction               157                    84                    73
Distribution               109                    55                    54
Éducation                  150                   100                    50
Machines/Outils            138                    68                    70
Pharmaceutique             138                    13                   125
>
> diamondplot(mydata,bg="white")
>
```

Ce qui donne:

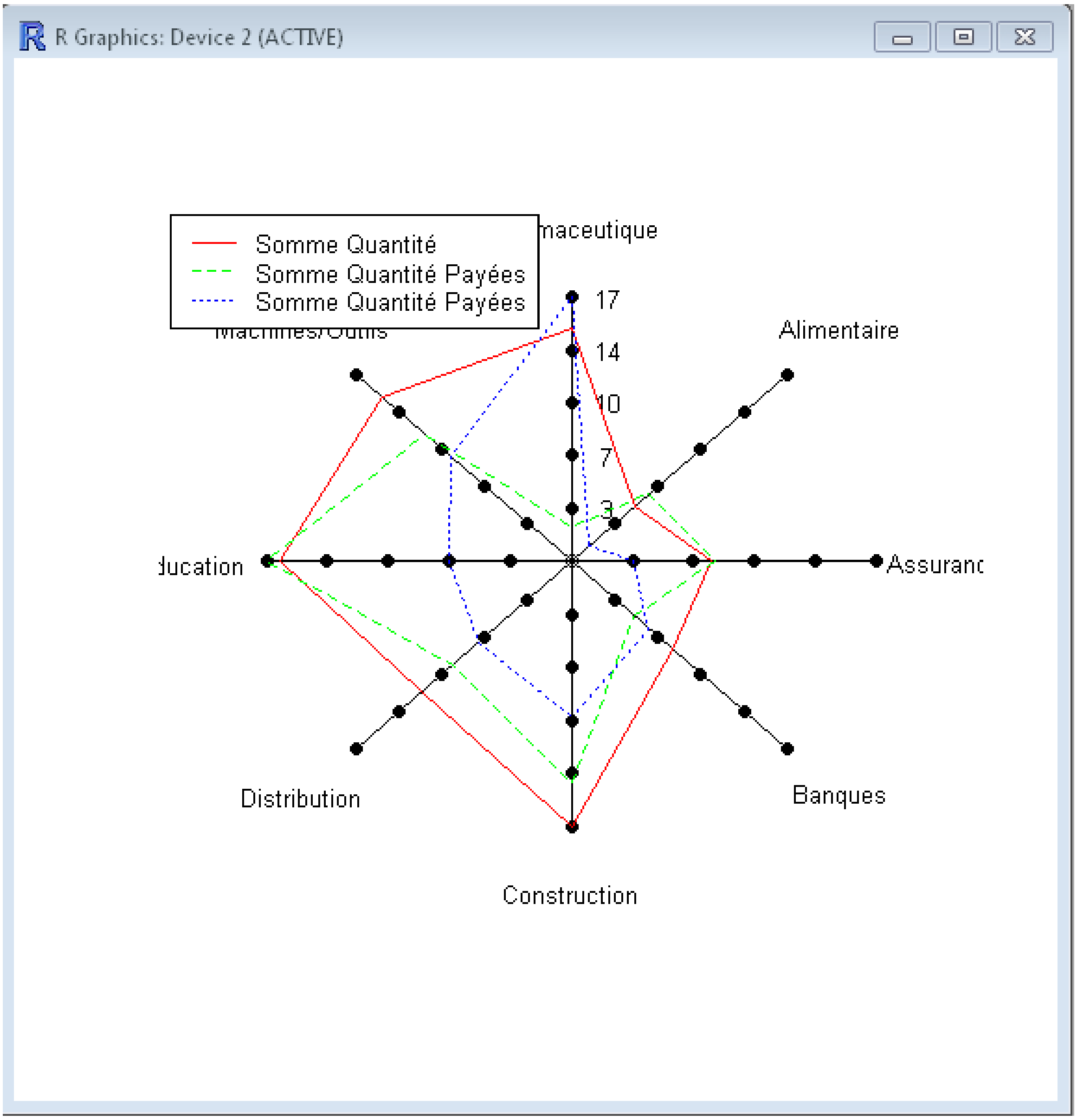
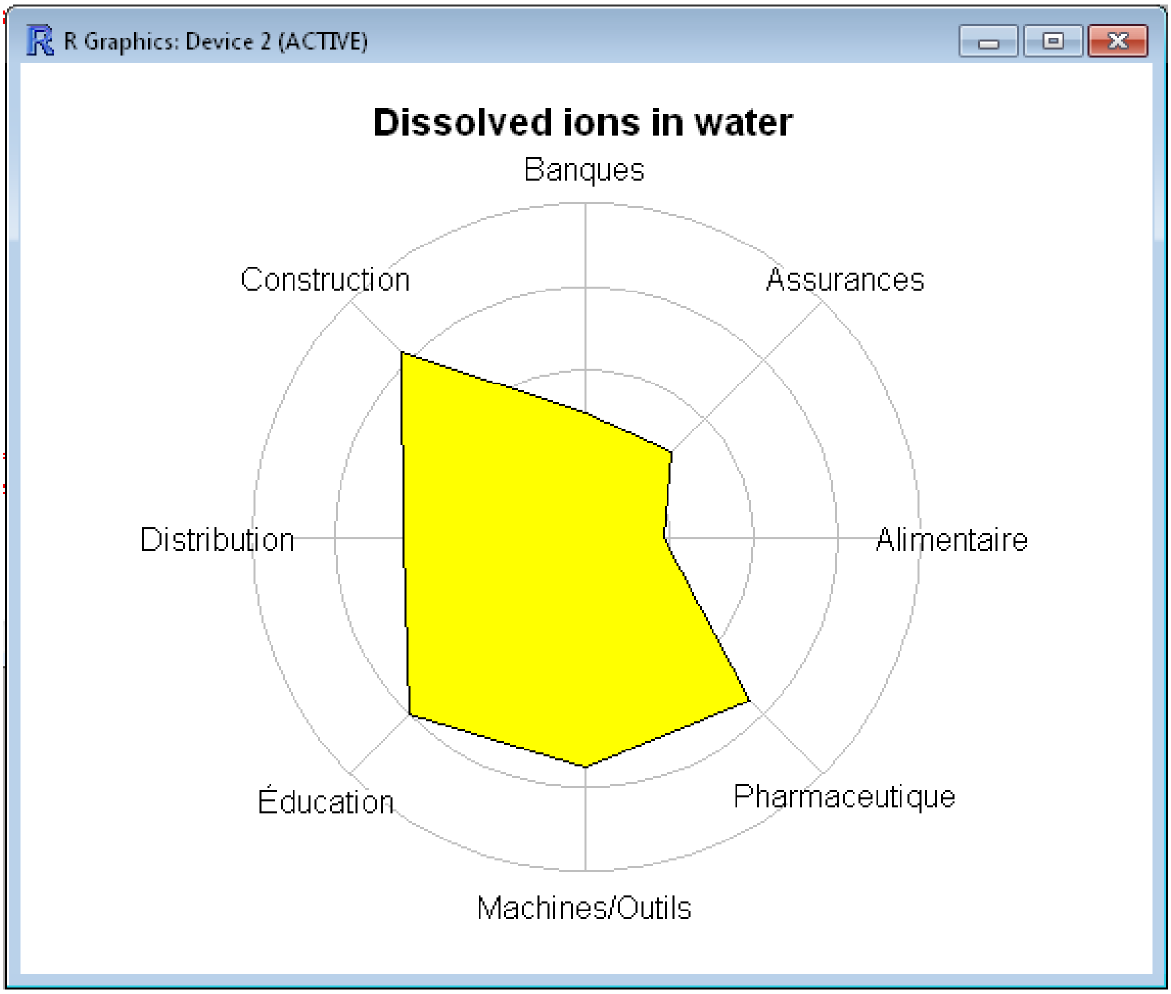


Figure 130 Graph de type radar

Ou encore, toujours avec le même package:

```
> library(plotrix)
> setwd("C:/")
> mydataMain<-read.csv("Ventes.csv",header=T,sep=";")
> mydata<-aggregate(mydataMain$Quantité,by=list(mydataMain$Activité),FUN=sum)
> colnames(mydata)<-c("Secteur","Somme Quantité")
> mydata
         Secteur Somme Quantité
1      Alimentaire             46
2       Assurances             72
3          Banques             74
4     Construction            157
5     Distribution            109
6        Éducation            150
7 Machines/Outils            138
8   Pharmaceutique            138
> SectorValues<-mydata[,2]
> SectorNames<-mydata[,1]
> MaxValue<-max(mydata[,2])
> radial.plot(SectorValues,labels=SectorNames,rp.type="p",main="Dissolved ions in water"
+ ,radial.lim=c(0,MaxValue),poly.col="yellow")
0 157
0 50 100 150 200
> |
```

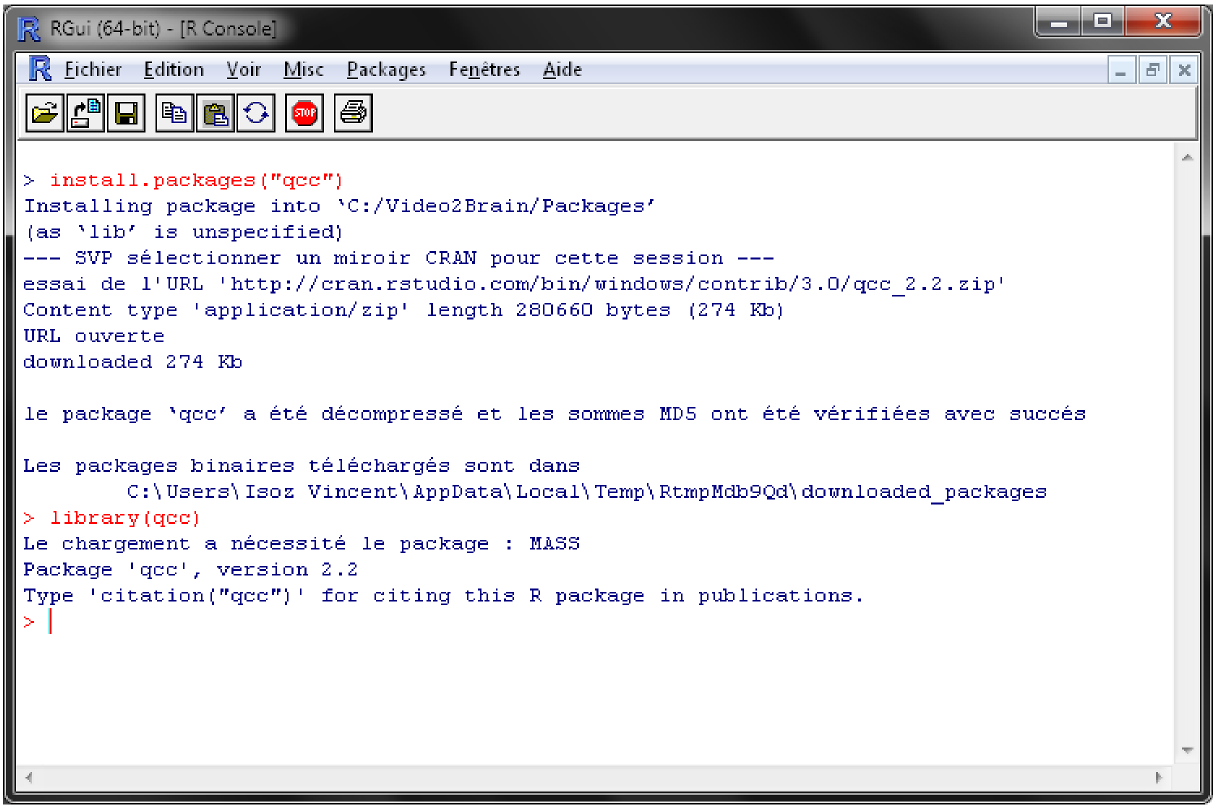
Ce qui donne:



Exemple 136.: Diagramme d'Ishikawa (cause/effets)

R 3.0.2

Pour faire un diagramme d'Ishikawa nous allons d'abord charger le package **qcc** qui nous sera très utile dans le chapitre de la maîtrise statistique de la qualité:



```
> install.packages("qcc")
Installing package into 'C:/Video2Brain/Packages'
(as 'lib' is unspecified)
--- SVP sélectionner un miroir CRAN pour cette session ---
essai de l'URL 'http://cran.rstudio.com/bin/windows/contrib/3.0/qcc_2.2.zip'
Content type 'application/zip' length 280660 bytes (274 Kb)
URL ouverte
downloaded 274 Kb

le package 'qcc' a été décompressé et les sommes MD5 ont été vérifiées avec succés

Les packages binaires téléchargés sont dans
        C:\Users\Isoz Vincent\AppData\Local\Temp\RtmpMdb9Qd\downloaded_packages
> library(qcc)
Le chargement a nécessité le package : MASS
Package 'qcc', version 2.2
Type 'citation("qcc")' for citing this R package in publications.
>
```

Ensuite, nous faisons un exemple (simplement reprise de l'aide du package):

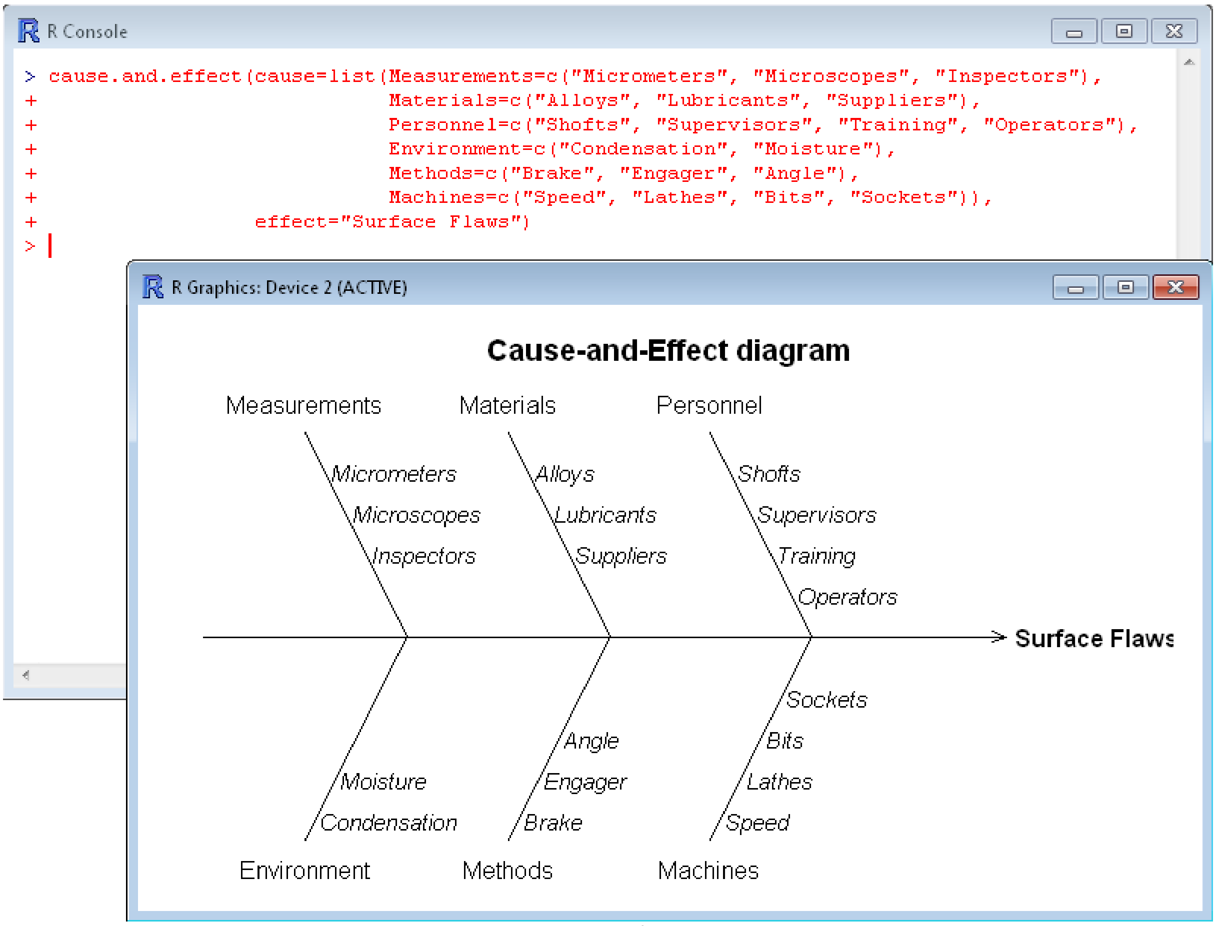


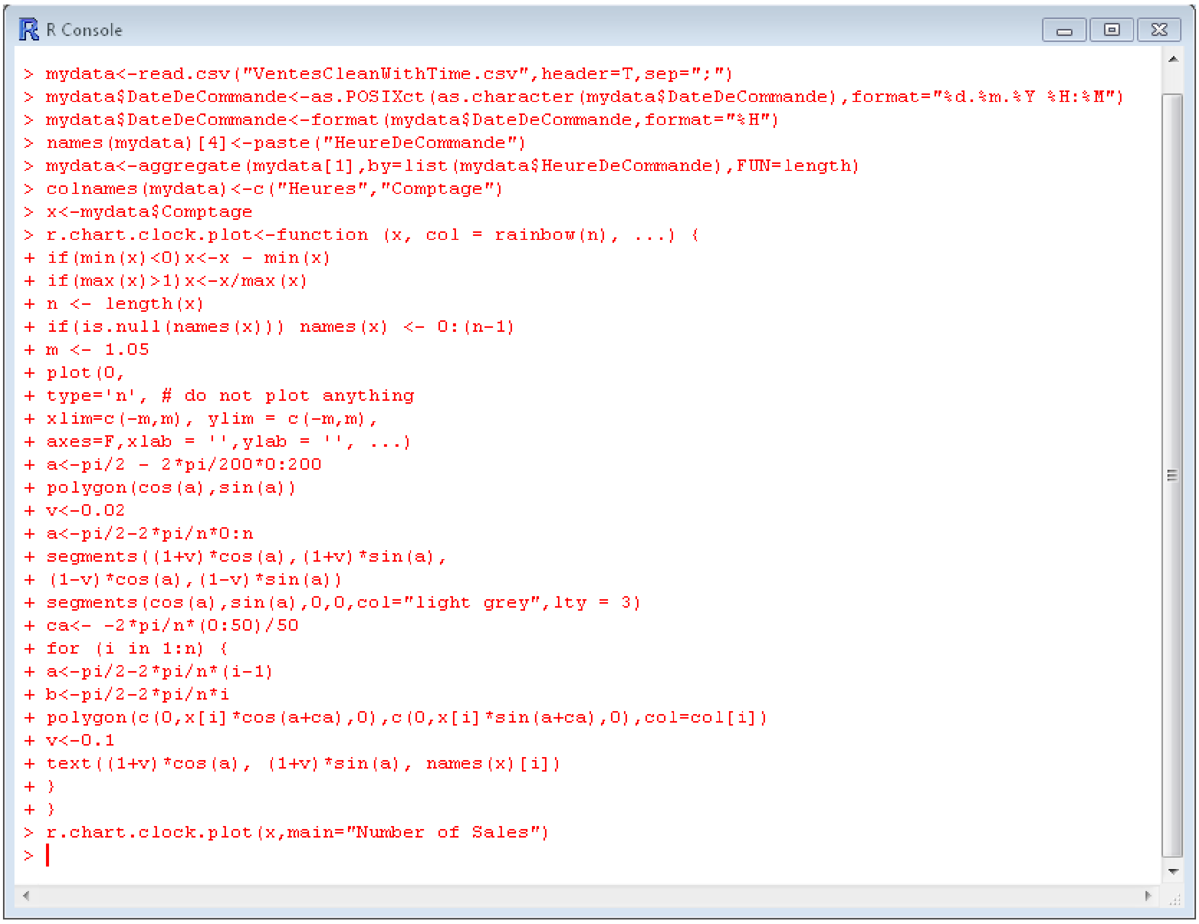
Figure 131 Graph de type Ishikawa

Exemple 137.: Diagramme radial (rose des vents)

R 3.0.2

Voici un diagramme radial fait pour représenter des quantités sur un intervalle de 24 heures (le script est inclus dans le framework des participants aux cours).

PS: Merci à Vincent ZOONEKYND



```
R Console
> mydata<-read.csv("VentesCleanWithTime.csv",header=T,sep=";")
> mydata$DateDeCommande<-as.POSIXct(as.character(mydata$DateDeCommande),format="%d.%m.%Y %H:%M")
> mydata$DateDeCommande<-format(mydata$DateDeCommande,format="%H")
> names(mydata)[4]<-paste("HeureDeCommande")
> mydata<-aggregate(mydata[1],by=list(mydata$HeureDeCommande),FUN=length)
> colnames(mydata)<-c("Heures","Comptage")
> x<-mydata$Comptage
> r.chart.clock.plot<-function (x, col = rainbow(n), ...) {
+ if(min(x)<0)x<-x - min(x)
+ if(max(x)>1)x<-x/max(x)
+ n <- length(x)
+ if(is.null(names(x))) names(x) <- 0:(n-1)
+ m <- 1.05
+ plot(0,
+ type='n', # do not plot anything
+ xlim=c(-m,m), ylim = c(-m,m),
+ axes=F,xlab = '',ylab = '', ...)
+ a<-pi/2 - 2*pi/200*0:200
+ polygon(cos(a),sin(a))
+ v<-0.02
+ a<-pi/2-2*pi/n*0:n
+ segments((1+v)*cos(a),(1+v)*sin(a),
+ (1-v)*cos(a),(1-v)*sin(a))
+ segments(cos(a),sin(a),0,0,col="light grey",lty = 3)
+ ca<- -2*pi/n*(0:50)/50
+ for (i in 1:n) {
+ a<-pi/2-2*pi/n*(i-1)
+ b<-pi/2-2*pi/n*i
+ polygon(c(0,x[i]*cos(a+ca),0),c(0,x[i]*sin(a+ca),0),col=col[i])
+ v<-0.1
+ text((1+v)*cos(a), (1+v)*sin(a), names(x)[i])
+ }
+ }
> r.chart.clock.plot(x,main="Number of Sales")
>
```

Ce qui donne:

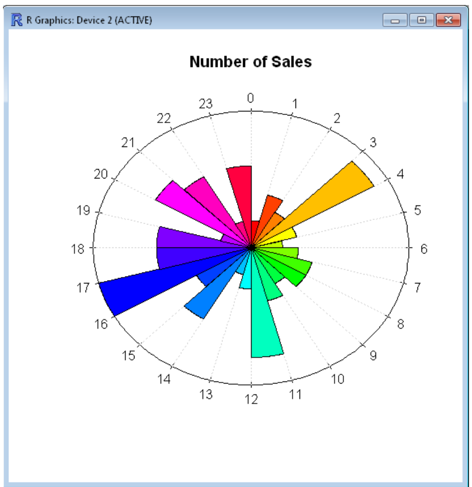


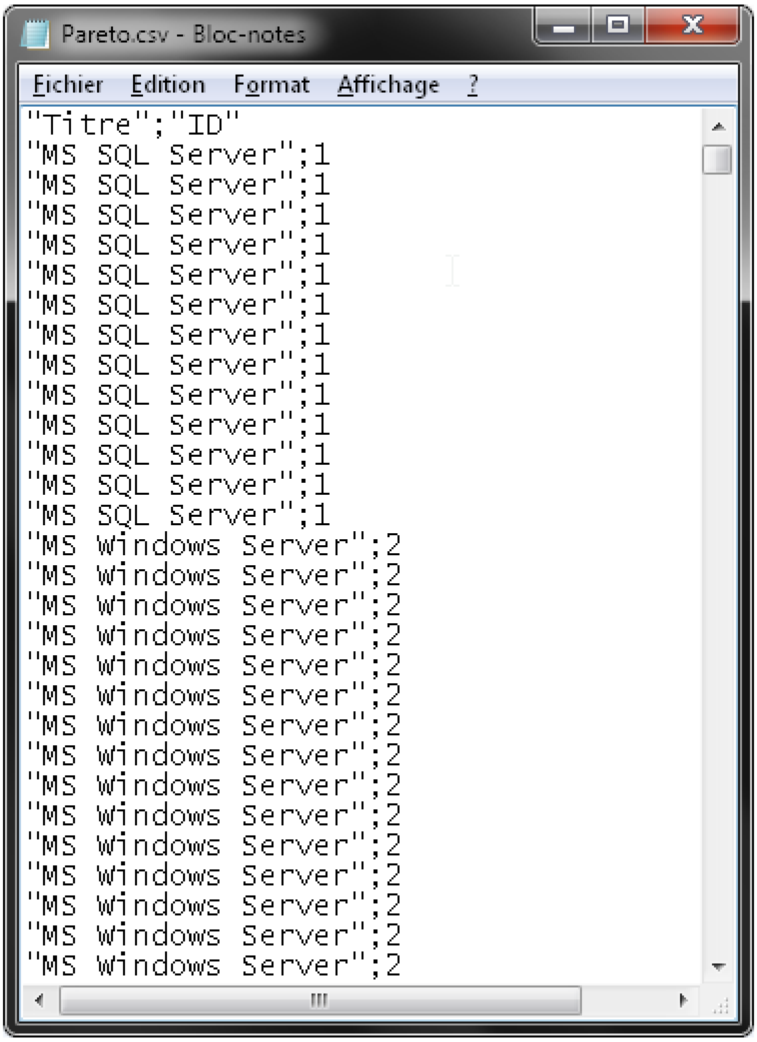
Figure 132 Graph de type Rose des vents

Exemple 138.: Diagramme de Pareto et analyse ABC

R 3.0.2

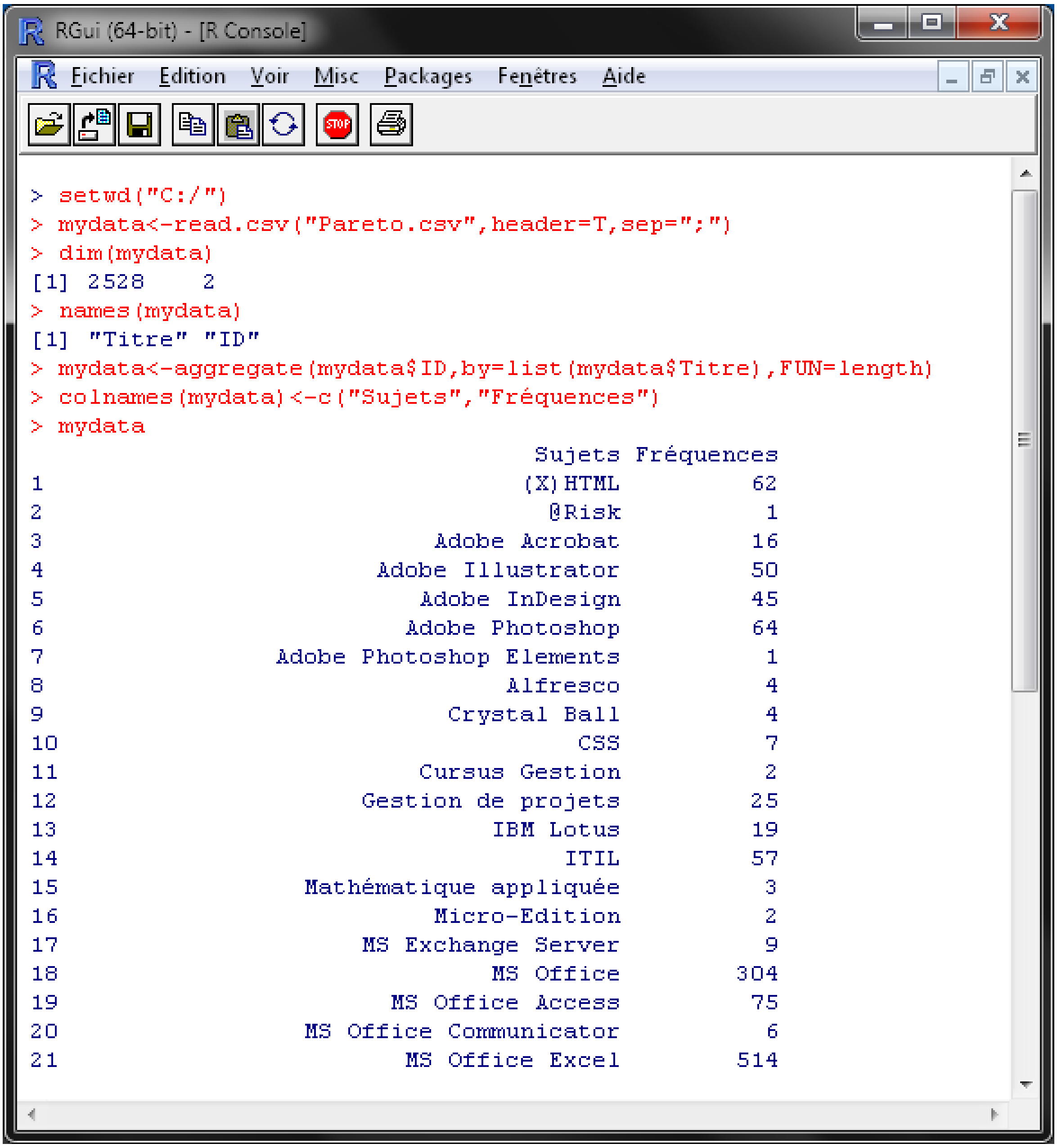
Le but ici va être de vérifier que nous pouvons reproduire le diagramme de Pareto que nous avions fait dans R.

Nous partons du fichier *.csv suivant:



```
"Titre";"ID"
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS SQL Server";1
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
"MS Windows Server";2
```

et d'abord nous l'importons et le résumons:




```
> setwd("C:/")
> mydata<-read.csv("Pareto.csv",header=T,sep=";")
> dim(mydata)
[1] 2528    2
> names(mydata)
[1] "Titre" "ID"
> mydata<-aggregate(mydata$ID,by=list(mydata$Titre),FUN=length)
> colnames(mydata)<-c("Sujets","Fréquences")
> mydata
                                 Sujets Fréquences
1                              (X)HTML         62
2                                @Risk          1
3                        Adobe Acrobat         16
4                    Adobe Illustrator         50
5                       Adobe InDesign         45
6                      Adobe Photoshop         64
7             Adobe Photoshop Elements          1
8                            Alfresco           4
9                        Crystal Ball           4
10                                CSS           7
11                     Cursus Gestion           2
12                 Gestion de projets          25
13                          IBM Lotus          19
14                               ITIL          57
15            Mathématique appliquée           3
16                      Micro-Edition           2
17                 MS Exchange Server           9
18                          MS Office         304
19                   MS Office Access          75
20             MS Office Communicator           6
21                    MS Office Excel         514
```

Remarque: Pour une analyse ABC il suffit de calculer une colonne en pourcents du total des fréquences et d'assigner une lettre à ABC à chaque intervalle de notre choix:

```
> mydata<-read.csv("C:/tmp/Pareto.csv",header=T,sep=";")
> mydata<-aggregate(mydata$ID,by=list(mydata$Titre),FUN=length)
> colnames(mydata)<-c("Sujets","Fréquences")
> mydata$Cumulé<-mydata$Fréquences/sum(mydata$Fréquences)
> mydata$ABC <- ifelse(mydata$Cumulé>=0.05,"C",
+           ifelse(mydata$Cumulé<0.01, "A", "B"))
> mydata[order(mydata$Cumulé,decreasing = TRUE),]
                                 Sujets Fréquences       Cumulé ABC
21                      MS Office Excel        514 0.2033227848   C
18                            MS Office        304 0.1202531646   C
29                       MS Office Word        229 0.0905854430   C
32                  MS SharePoint Server        223 0.0882120253   C
25                    MS Office Project        183 0.0723892405   C
35                           MS Windows        134 0.0530063291   C
24                 MS Office PowerPoint        113 0.0446993671   B
23                    MS Office Outlook        101 0.0399525316   B
19                     MS Office Access         75 0.0296677215   B
26              MS Office Project Server         72 0.0284810127   B
6                       Adobe Photoshop         64 0.0253164557   B
1                               (X)HTML         62 0.0245253165   B
14                                 ITIL         57 0.0225474684   B
4                     Adobe Illustrator         50 0.0197784810   B
5                        Adobe InDesign         45 0.0178006329   B
28                      MS Office Visio         35 0.0138449367   B
43                      XML/XSL/XSD/DTD         32 0.0126582278   B
36                    MS Windows Server         28 0.0110759494   B
41                     VBA (MS Office)          27 0.0106803797   B
12                   Gestion de projets         25 0.0098892405   A
37                      Oracle Database         23 0.0090981013   A
13                            IBM Lotus         19 0.0075158228   A
3                         Adobe Acrobat         16 0.0063291139   A
39                             PRINCE2          16 0.0063291139   A
33                        MS SQL Server         13 0.0051424051   A
17                   MS Exchange Server          9 0.0035601266   A
10                                  CSS          7 0.0027689873   A
```

Enfin nous générons le diagramme de Pareto avec la commande **pareto.chart( )** :

```
> dim(mydata)
[1] 43  2
> defect<-mydata$Fréquences
> names(defect)<-mydata$Sujets
> pareto.chart(defect, cumperc = seq(0, 100, by = 5), ylab2 = "A finer tickmarks grid")

Pareto chart analysis for defect
                                                    Frequency    Cum.Freq.   Percentage
  MS Office Excel                                5.140000e+02 5.140000e+02 2.033228e+01
  MS Office                                      3.040000e+02 8.180000e+02 1.202532e+01
  MS Office Word                                 2.290000e+02 1.047000e+03 9.058544e+00
  MS SharePoint Server                           2.230000e+02 1.270000e+03 8.821203e+00
  MS Office Project                              1.830000e+02 1.453000e+03 7.238924e+00
  MS Windows                                     1.340000e+02 1.587000e+03 5.300633e+00
  MS Office PowerPoint                           1.130000e+02 1.700000e+03 4.469937e+00
  MS Office Outlook                              1.010000e+02 1.801000e+03 3.995253e+00
  MS Office Access                               7.500000e+01 1.876000e+03 2.966772e+00
  MS Office Project Server                       7.200000e+01 1.948000e+03 2.848101e+00
  Adobe Photoshop                                6.400000e+01 2.012000e+03 2.531646e+00
  (X)HTML                                        6.200000e+01 2.074000e+03 2.452532e+00
  ITIL                                           5.700000e+01 2.131000e+03 2.254747e+00
  Adobe Illustrator                              5.000000e+01 2.181000e+03 1.977848e+00
  Adobe InDesign                                 4.500000e+01 2.226000e+03 1.780063e+00
  MS Office Visio                                3.500000e+01 2.261000e+03 1.384494e+00
  XML/XSL/XSD/DTD                                3.200000e+01 2.293000e+03 1.265823e+00
  MS Windows Server                              2.800000e+01 2.321000e+03 1.107595e+00
  VBA (MS Office)                                2.700000e+01 2.348000e+03 1.068038e+00
  Gestion de projets                             2.500000e+01 2.373000e+03 9.889241e-01
  Oracle Database                                2.300000e+01 2.396000e+03 9.098101e-01
  IBM Lotus                                      1.900000e+01 2.415000e+03 7.515823e-01
  Adobe Acrobat                                  1.600000e+01 2.431000e+03 6.329114e-01
```

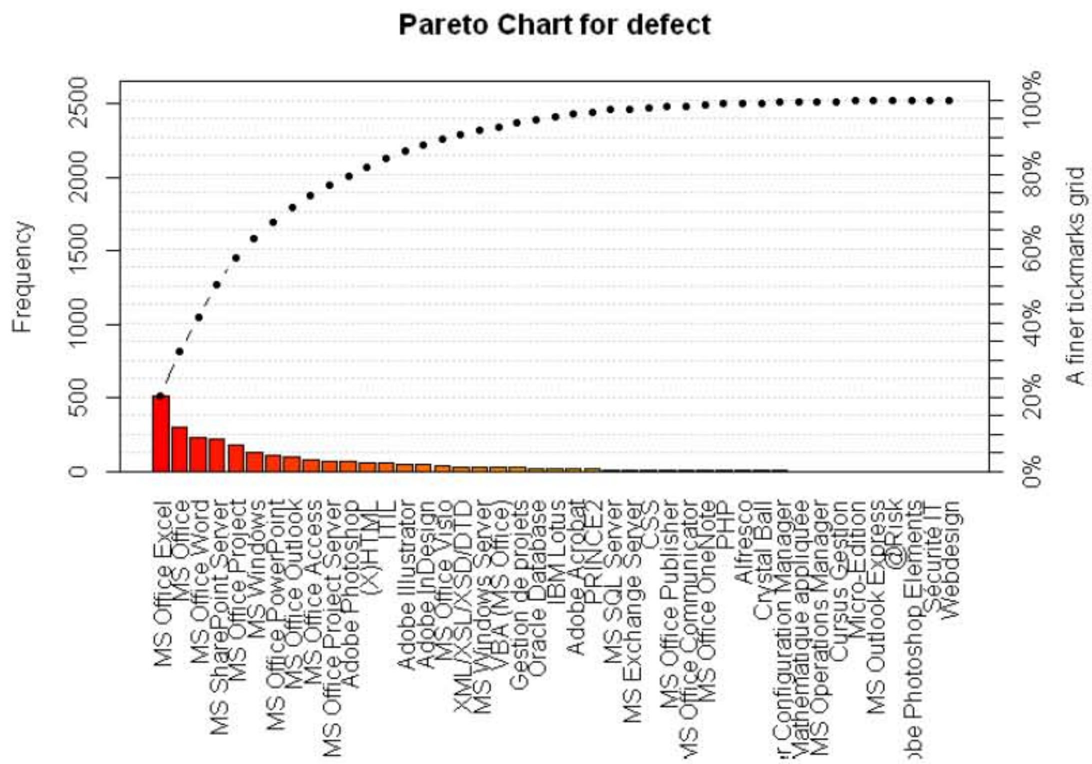
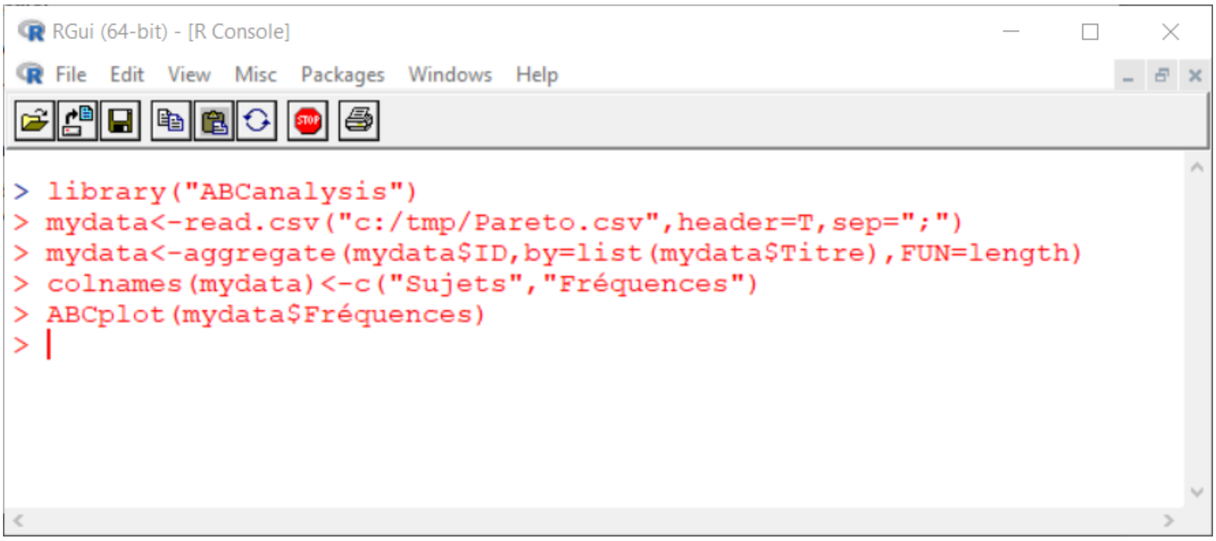


Figure 133 Graph de Pareto avec analyse ABC

Et pour ceux qui tiennent vraiment au diagramme ABC, en voici une première version avec la fonction **ABCplot( )** du package **ABCAnalysis**:



Ce qui donne:

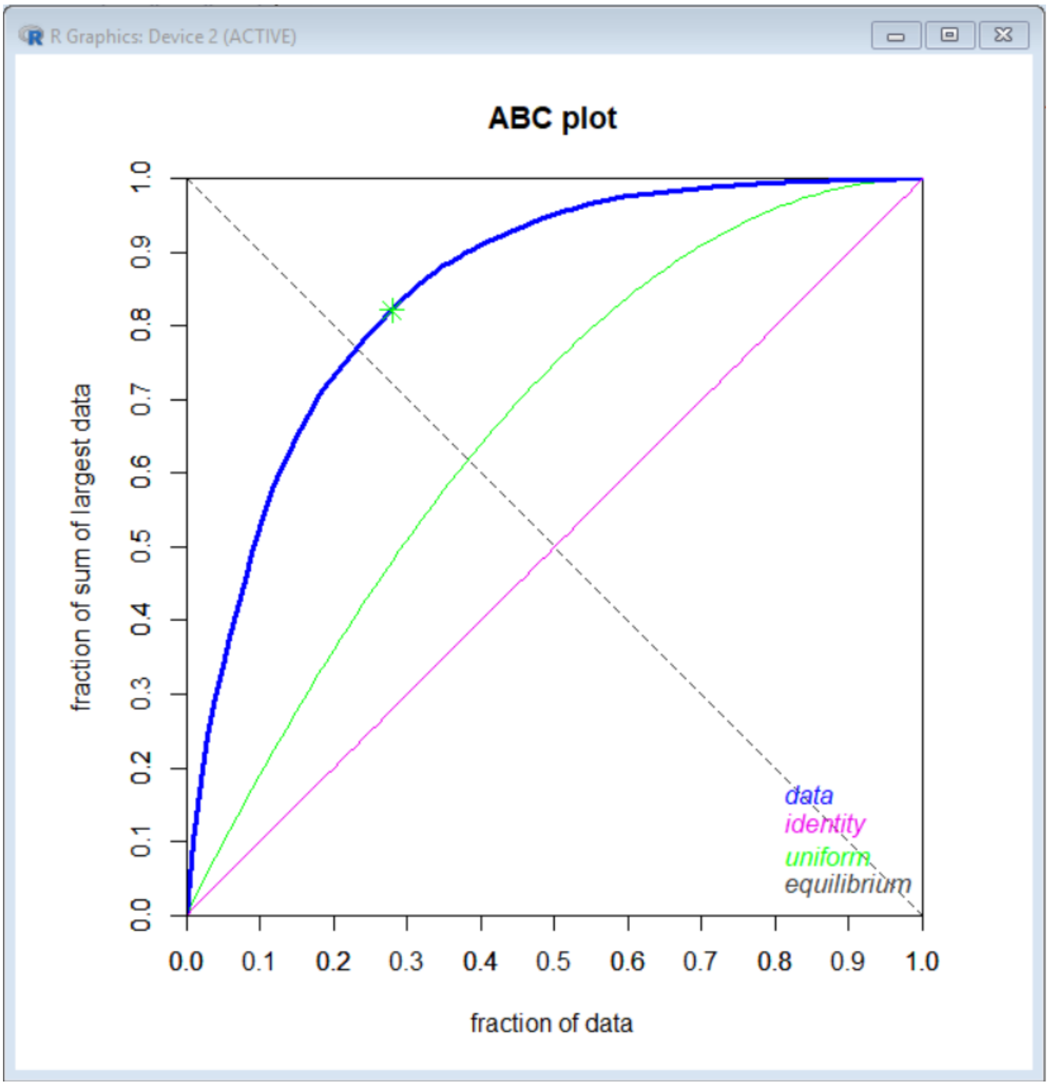
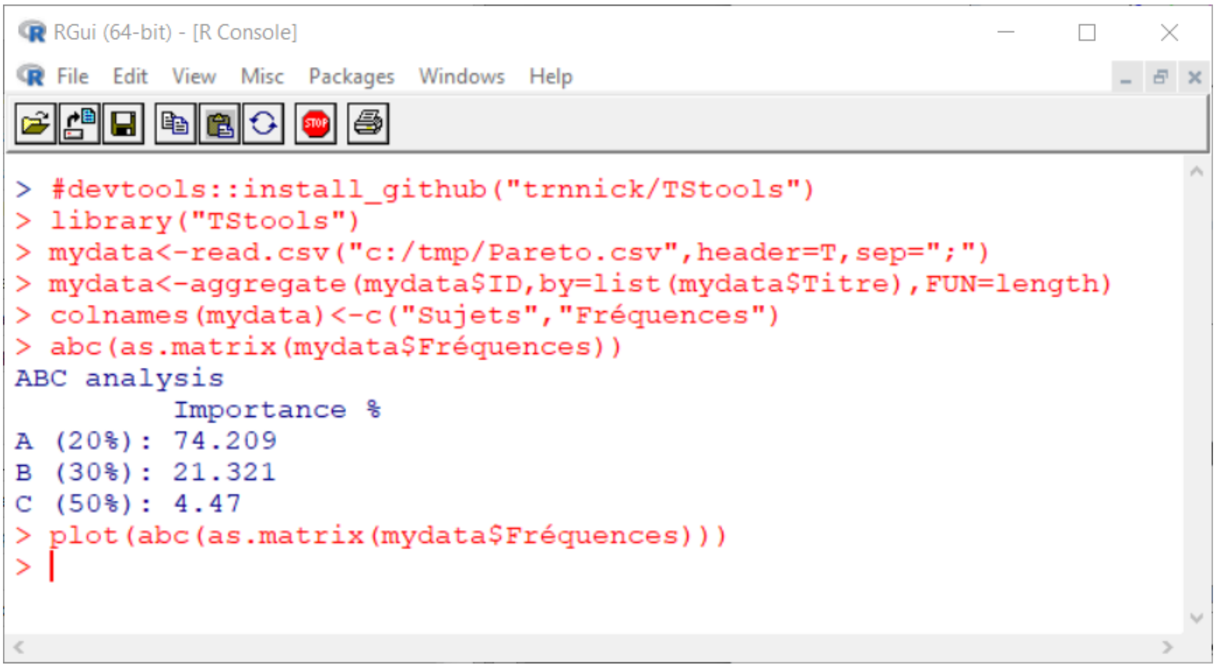
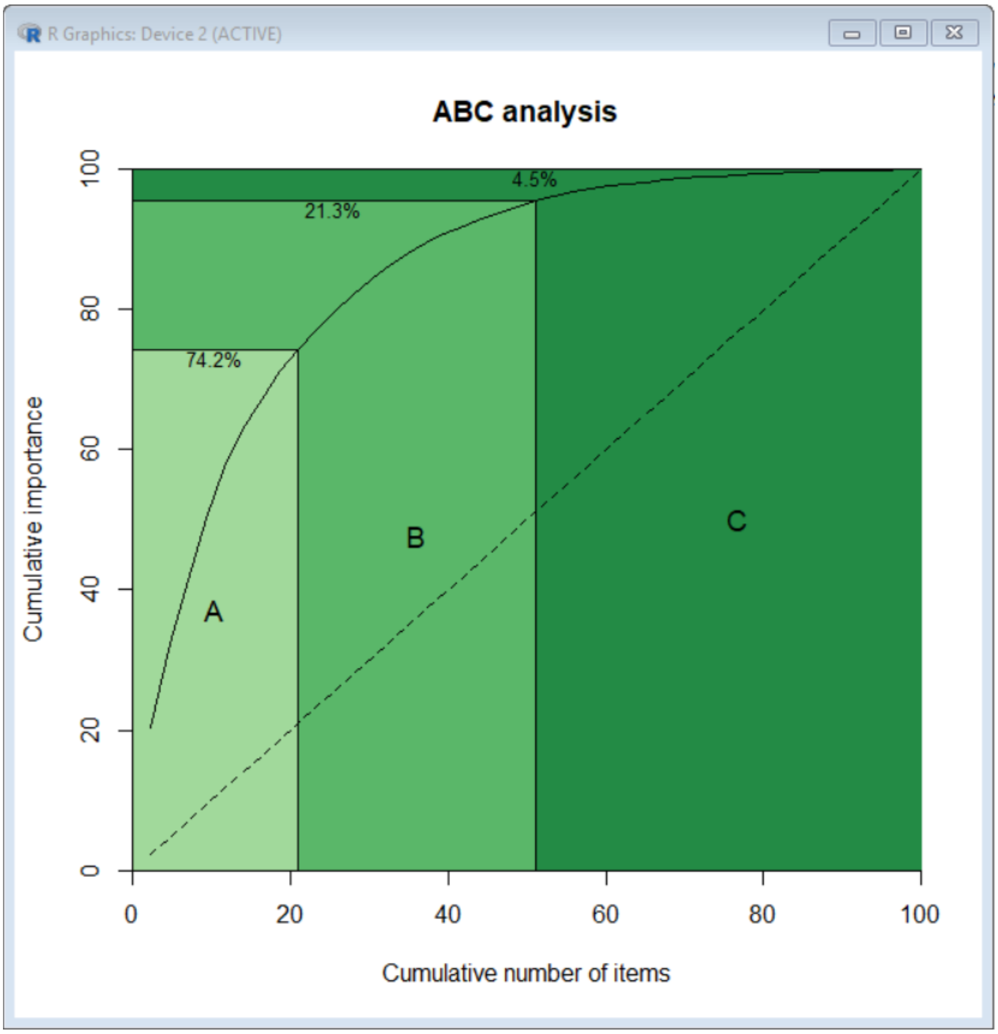


Figure 134 Graph de type Analyse ABC

Ou avec le package **TSTools** et sa fonction **abc( )** :



Ce qui donne graphiquement:



Exemple 139.: Diagrammes Box Plots

R 3.0.2

Il ne nous faut pas manquer le traditionnel Box Plot avec la commande **boxplot( )** et nous y ajoutons une touche de **rug( )** ...:

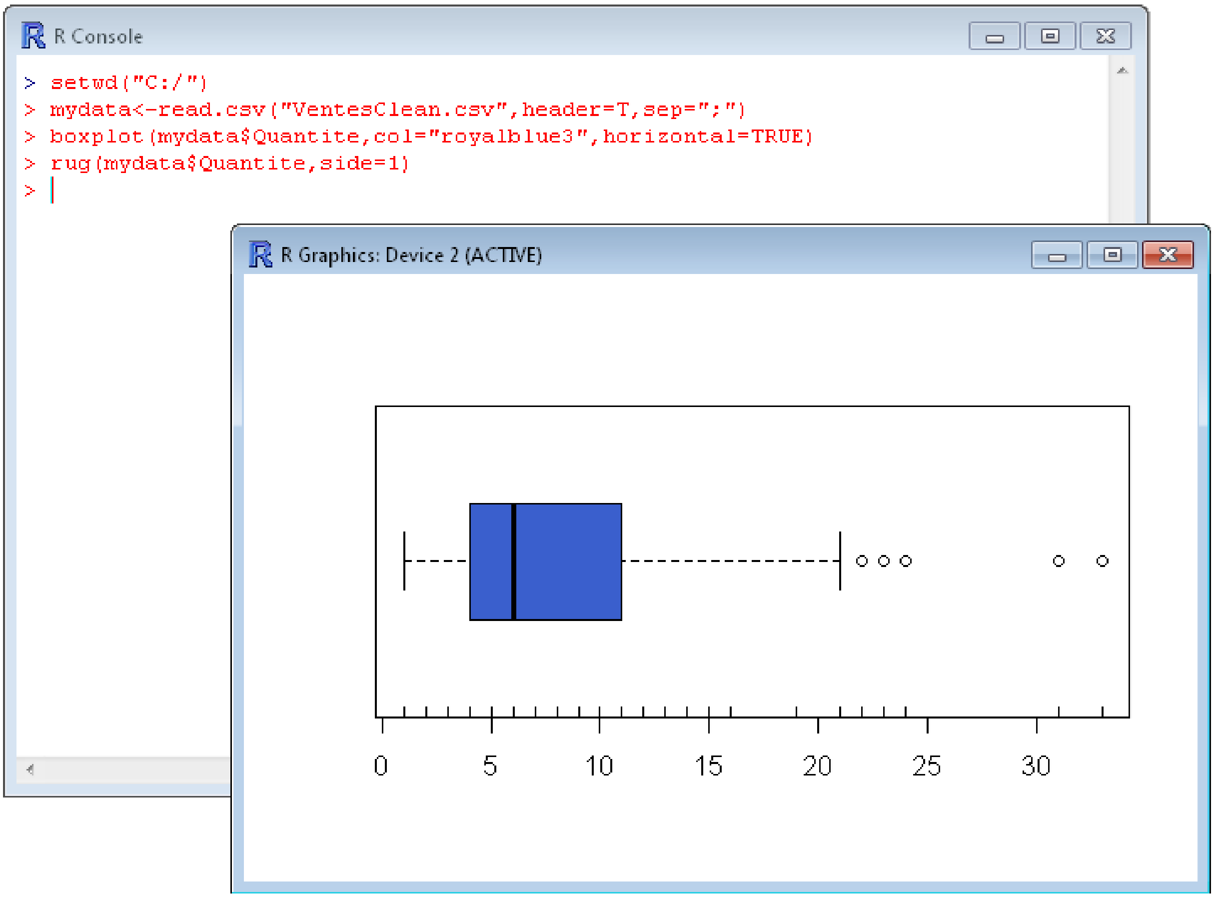
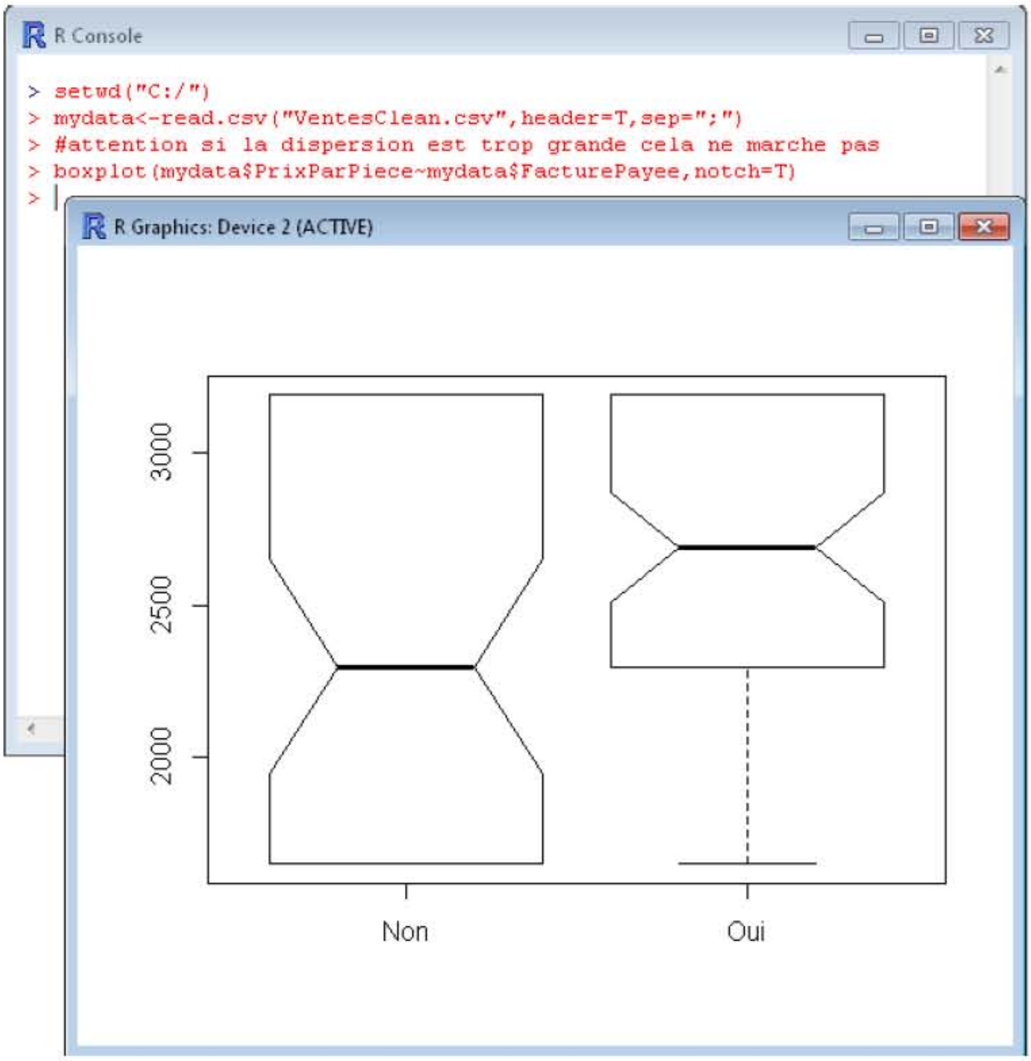
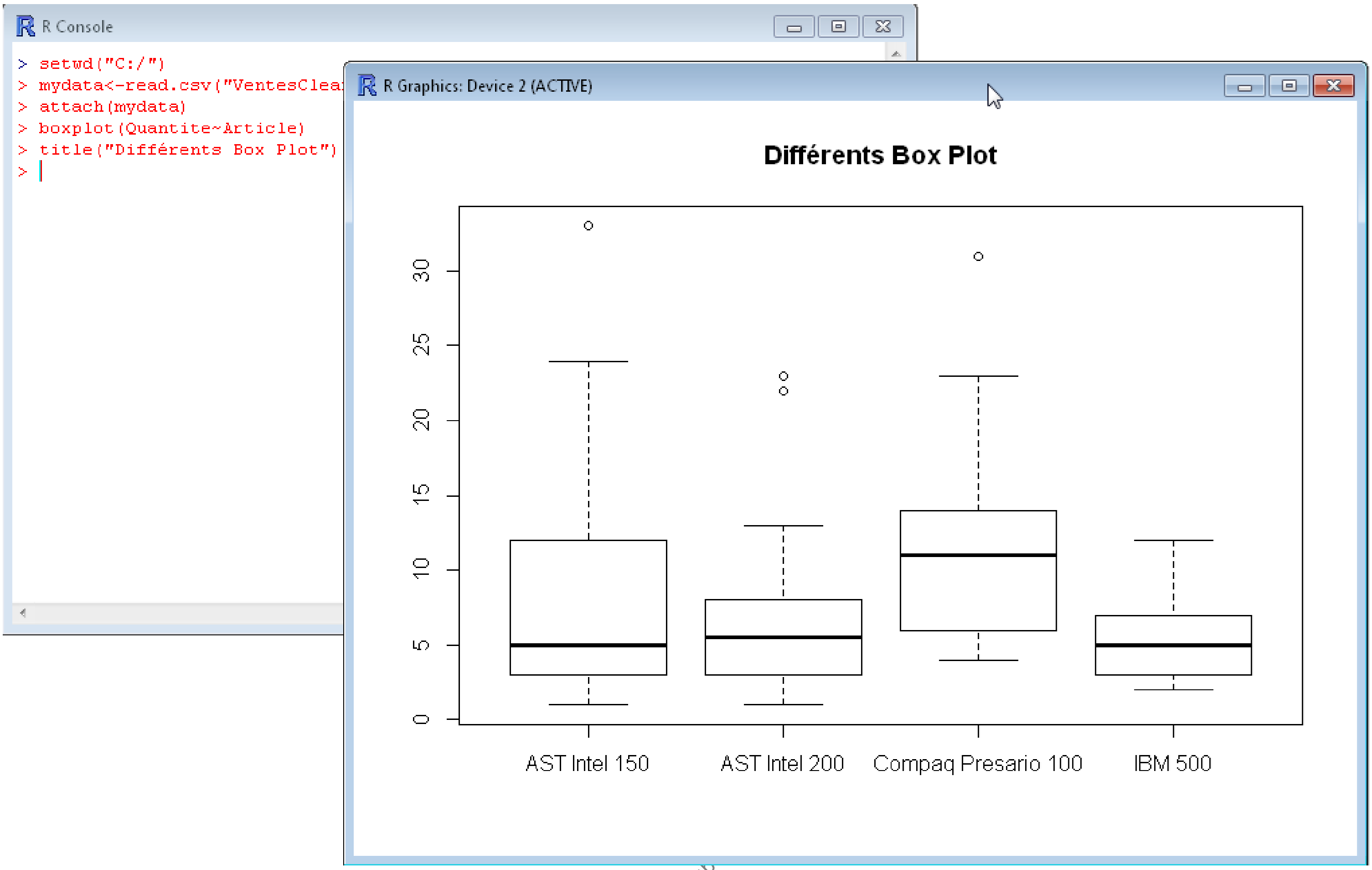


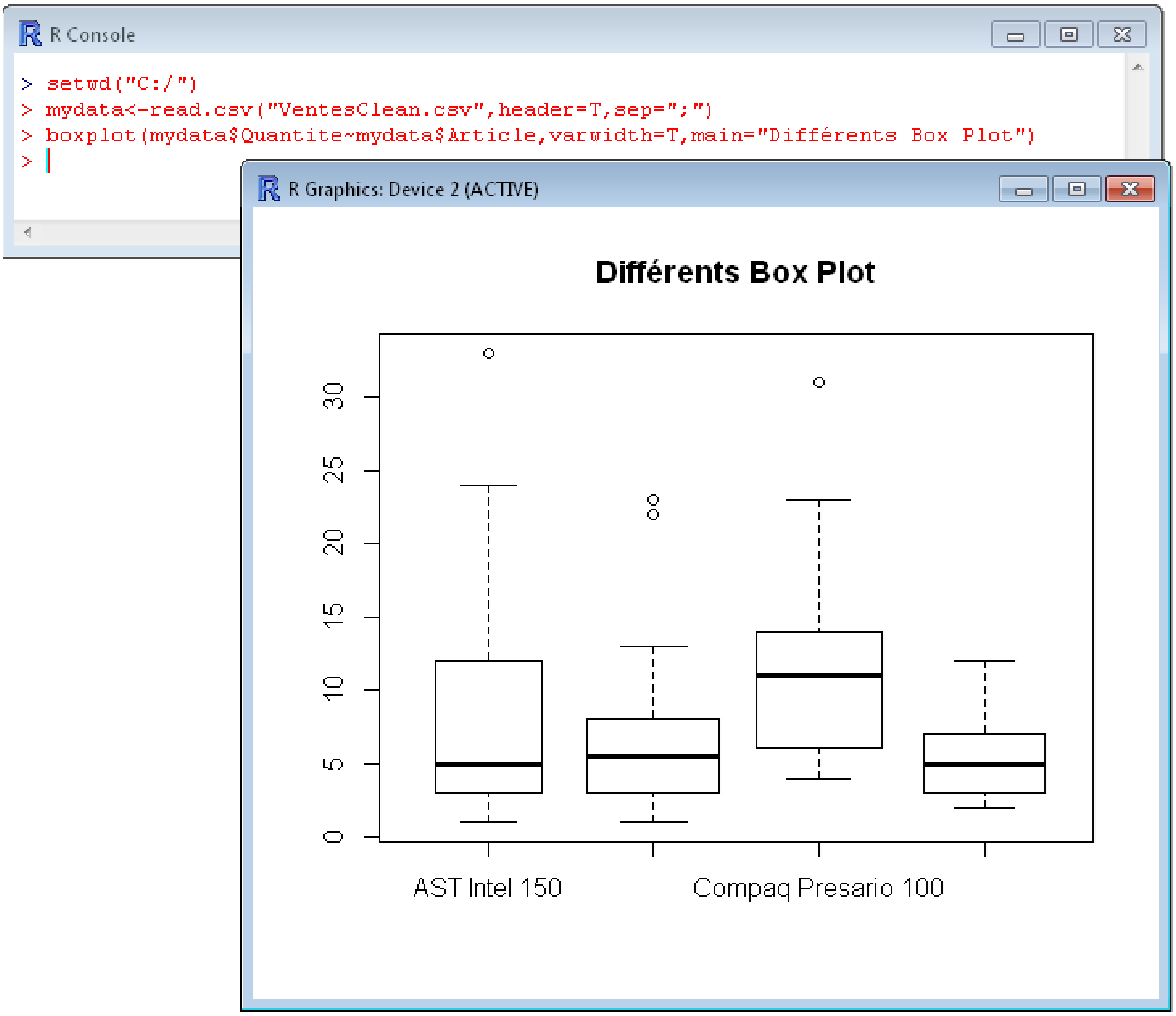
Figure 135 Graph de type Box-Whisker (boîte à moustaches) simple

ou encore (avec des intervalles de confiance de la médiane à 95%):


R Console
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> #attention si la dispersion est trop grande cela ne marche pas
> boxplot(mydata$PrixParPiece~mydata$FacturePayee,notch=T)
R Graphics: Device 2 (ACTIVE)
3000
2500
2000
Non
Oui



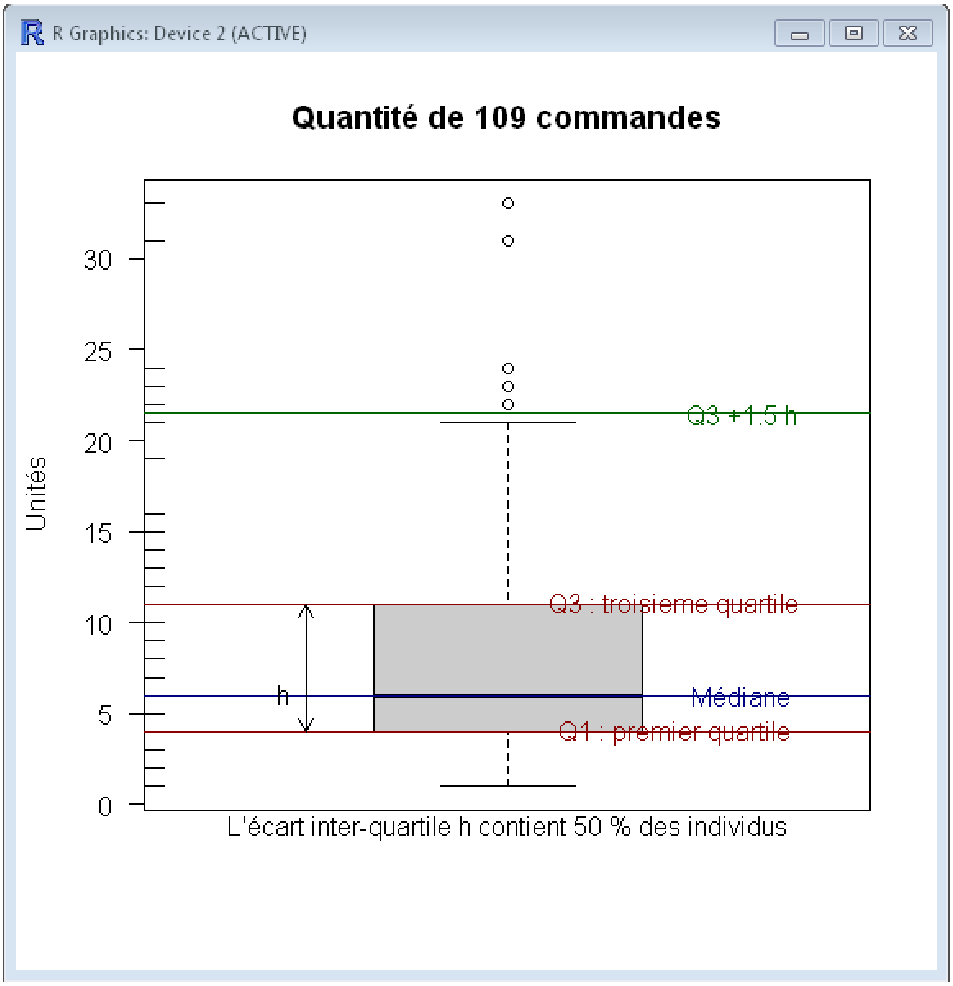
Nous pouvons aussi faire varier l'épaisseur des box plot en fonction des effectifs avec le paramètre *varwidth*:



Nous pouvons faire plus technique aussi:

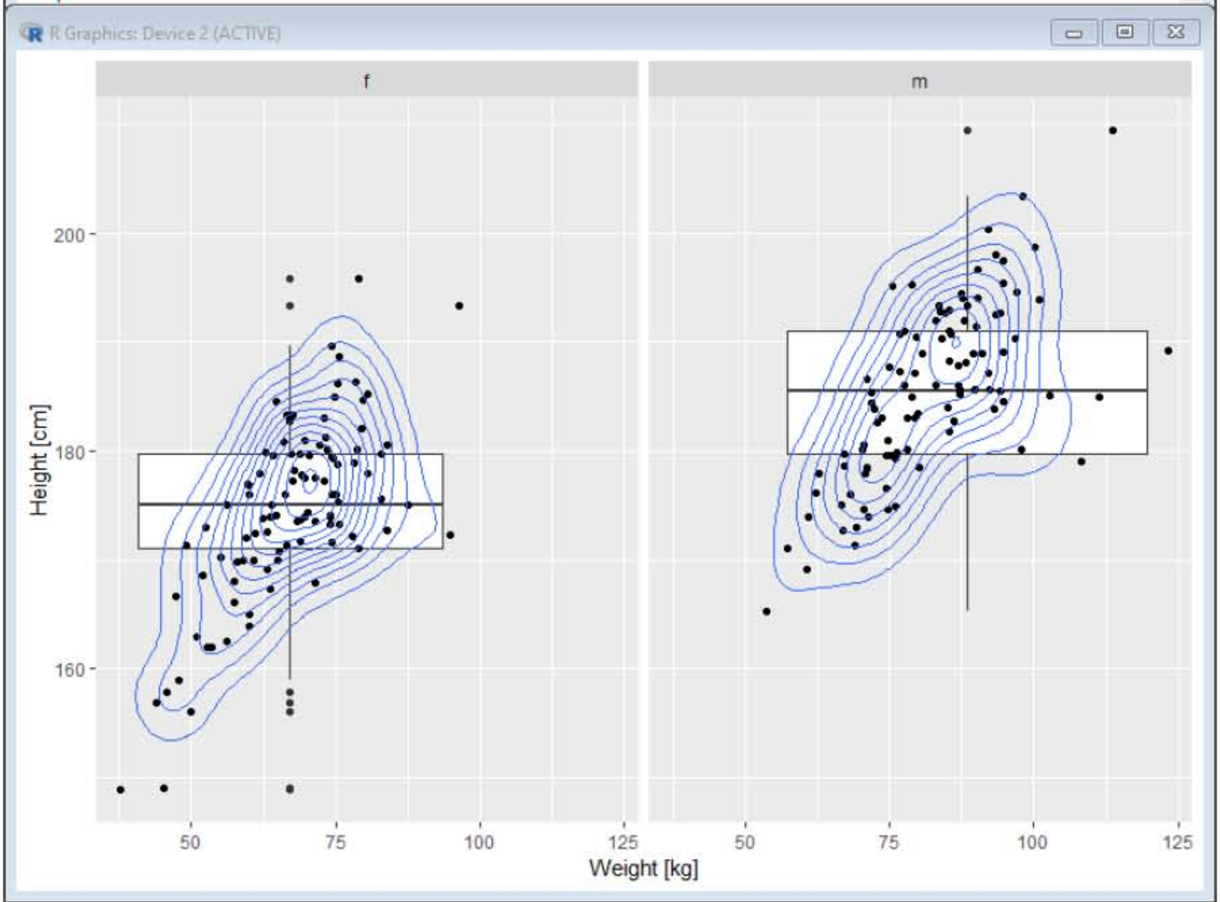
```
> boxplot(mydata$Quantite, col = grey(0.8), main = paste("Quantité de",
+ nrow(mydata), "commandes"), ylab = "Unités",
+ las = 1)
> rug(mydata$Quantite, side = 2)
> abline(h = median(mydata$Quantite), col = "navy")
> text(1.35, median(mydata$Quantite), "Médiane", col = "navy")
> Q1 <- quantile(mydata$Quantite,0.25)
> abline(h = Q1, col = "darkred")
> text(1.25, quantile(mydata$Quantite,0.25), "Q1 : premier quartile", col = "darkred")
> Q3 <- quantile(mydata$Quantite, 0.75)
> abline(h = Q3, col = "darkred")
> text(1.25, quantile(mydata$Quantite,0.75), "Q3 : troisieme quartile", col = "darkred")
> arrows(x0 = 0.7, y0 = quantile(mydata$Quantite, 0.75), x1 = 0.7, y1 = quantile(mydata$Quantite,0.25), length = 0.1, code = 3)
> text(0.7, median(mydata$Quantite), "h", pos = 2)
> mtext("L'écart inter-quartile h contient 50 % des individus",side = 1)
> abline(h = Q1 - 1.5*IQR(mydata$Quantite), col = "darkgreen")
> text(1.35, Q1 - 1.5*IQR(mydata$Quantite), "Q1 -1.5 h", col = "darkgreen")
> abline(h = Q3 + 1.5*IQR(mydata$Quantite), col = "darkgreen")
> text(1.35, Q3 + 1.5*IQR(mydata$Quantite), "Q3 +1.5 h", col = "darkgreen")
>
```

ce qui donne:



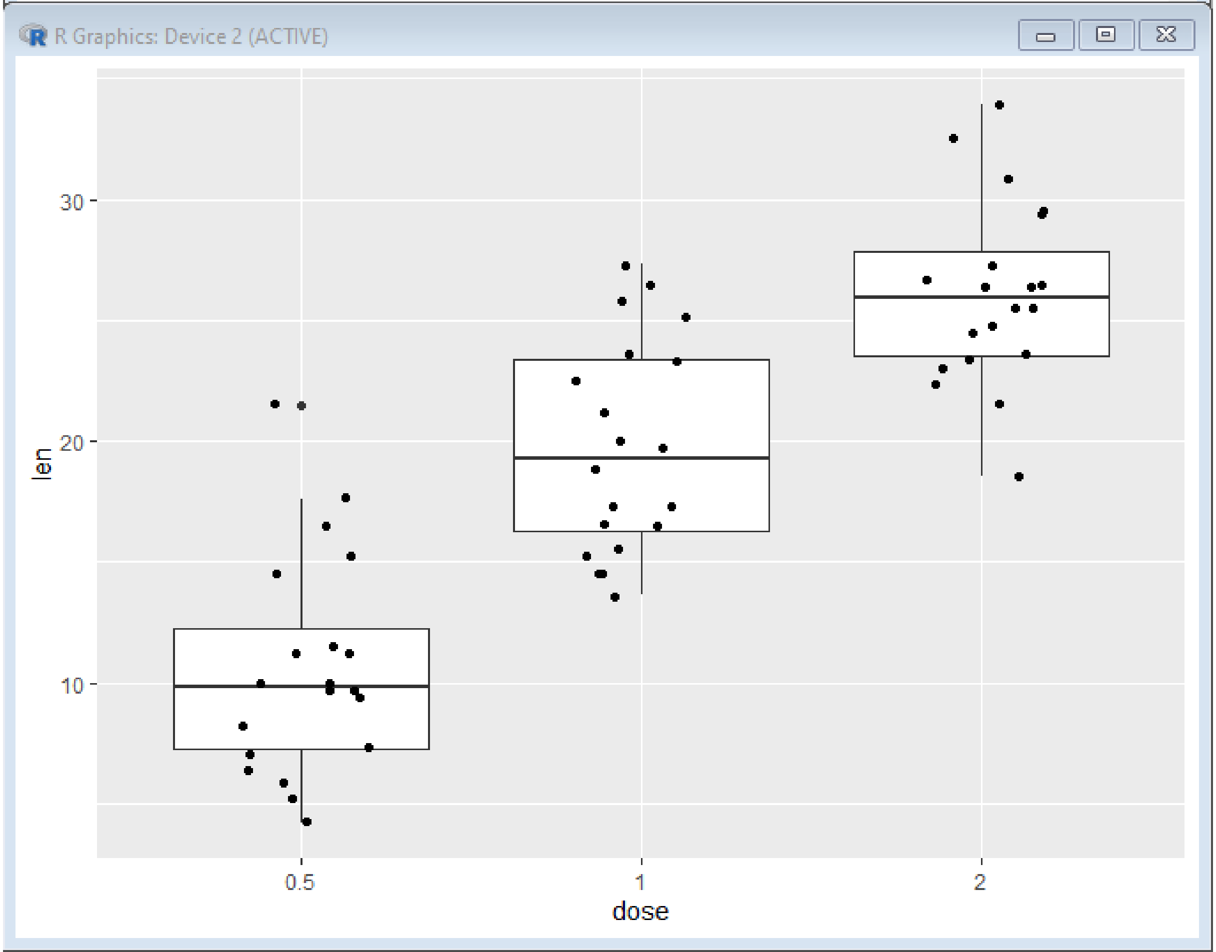
Ou encore avec le package ggplot on peut mélanger des boxplot avec des courbes de densité:

```
> library("ggplot2")
> library("DAAG") #pour le jeu de données
> data(ais)
> head(ais)
   rcc wcc   hc   hg ferr   bmi   ssf pcBfat   lbm    ht   wt sex  sport
1 3.96 7.5 37.5 12.3   60 20.56 109.1  19.75 63.32 195.9 78.9   f B_Ball
2 4.41 8.3 38.2 12.7   68 20.67 102.8  21.30 58.55 189.7 74.4   f B_Ball
3 4.14 5.0 36.4 11.6   21 21.86 104.6  19.88 55.36 177.8 69.1   f B_Ball
4 4.11 5.3 37.3 12.6   69 21.88 126.4  23.66 57.18 185.0 74.9   f B_Ball
5 4.45 6.8 41.5 14.0   29 18.96  80.3  17.64 53.20 184.6 64.6   f B_Ball
6 4.10 4.4 37.4 12.5   42 21.04  75.2  15.58 53.77 174.0 63.7   f B_Ball
> quickplot(wt,ht,xlab="Weight [kg]",ylab="Height [cm]",data=ais,
+ geom = c ("boxplot" , "point" , "density2d" ),facets=.~sex)
Warning message:
Continuous x aesthetic -- did you forget aes(group=...)?
> 
```

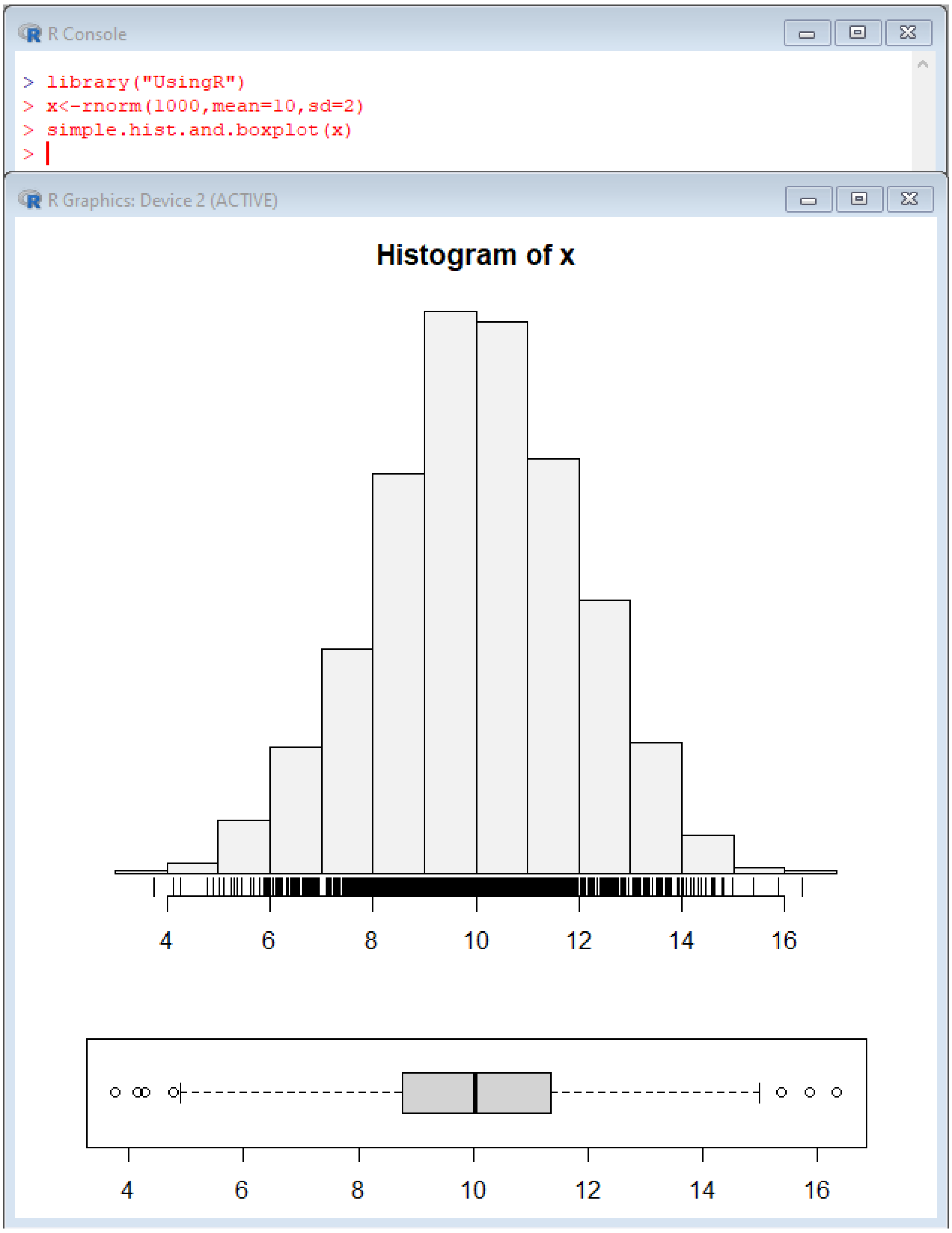


ou:

```
> data(ToothGrowth)
> head(ToothGrowth)
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
> ToothGrowth$dose <- as.factor(ToothGrowth$dose)
> library("ggplot2")
> ggplot(ToothGrowth, aes(x=dose, y=len)) +
+   geom_boxplot()+
+   geom_jitter(position=position_jitter(0.2))
>
```



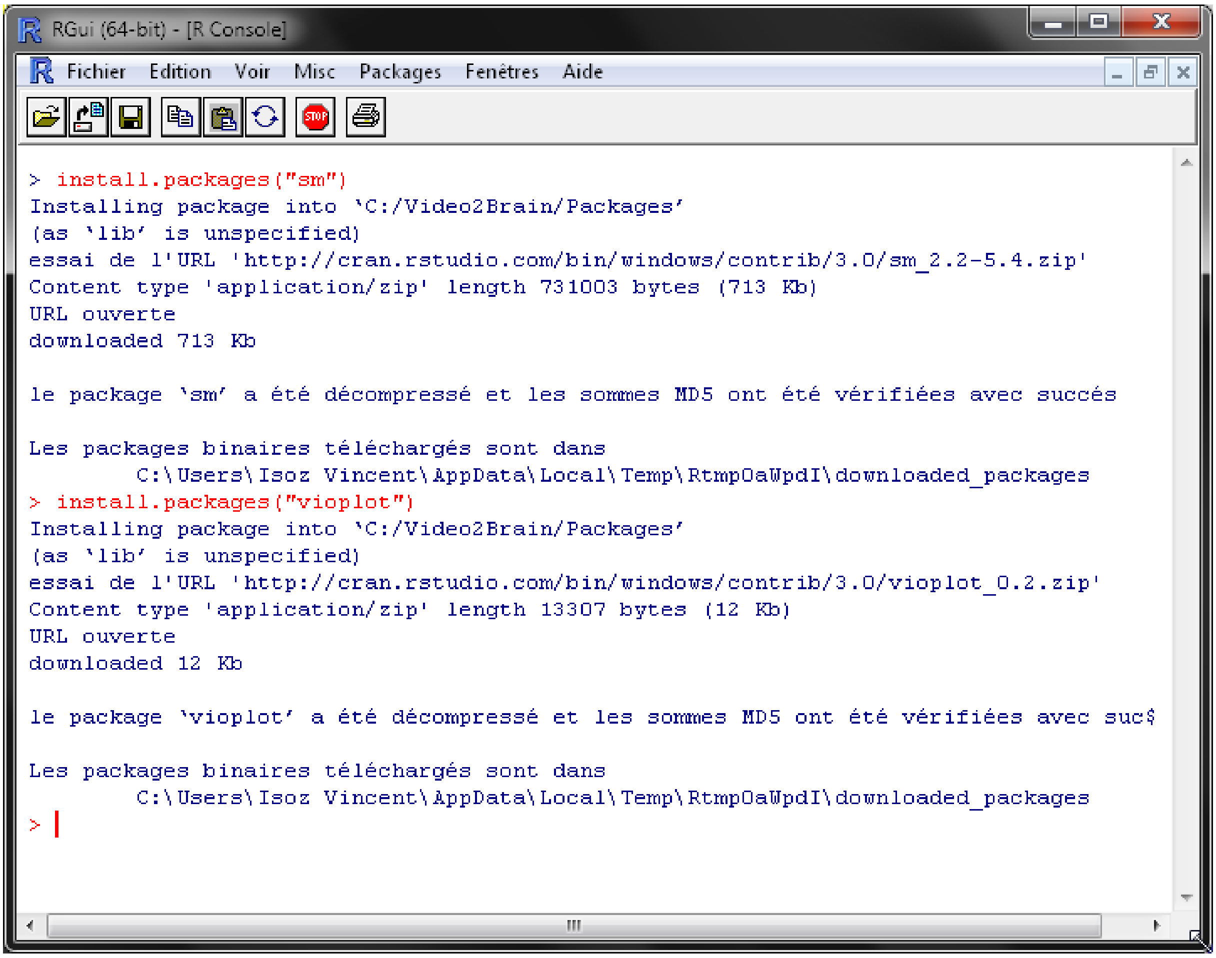
Voyons encore un autre exemple relativement famaux avec la fonction **simple.hist.and.boxplot( )** du package **UsingR**:


R Console
> library("UsingR")
> x<-rnorm(1000,mean=10,sd=2)
> simple.hist.and.boxplot(x)
>
R Graphics: Device 2 (ACTIVE)
Histogram of x
4 6 8 10 12 14 16
4 6 8 10 12 14 16

Diagrammes en violons (violin plot)

Les diagrammes en violon essayent de combiner les avantages des boîtes à moustaches et des estimateurs de la densité locale.

Nous installons d'abord deux packages qui sont **sm** et **vioplot**:



et nous y allons en utilisant la commande **vioplot( )** :

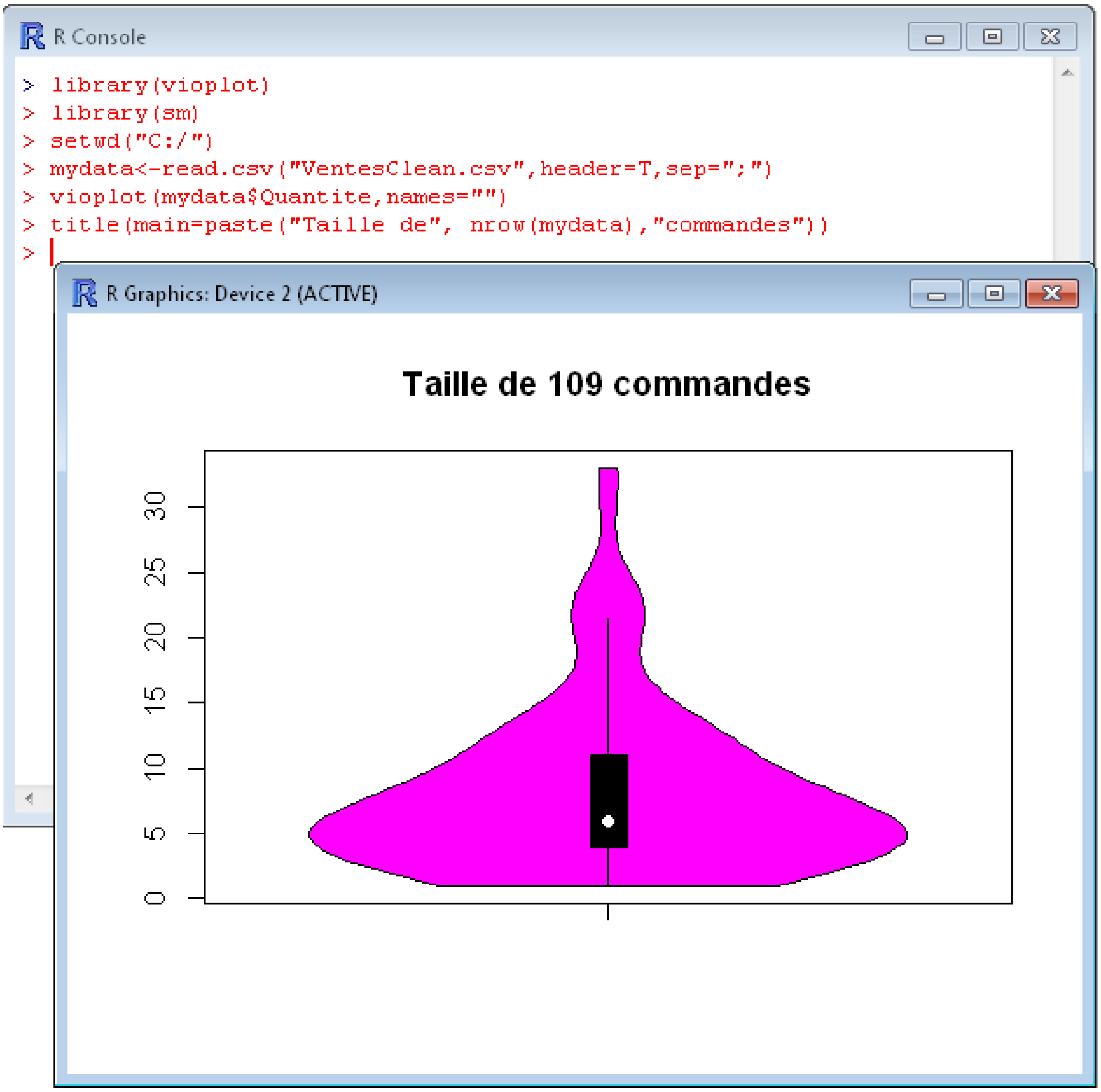
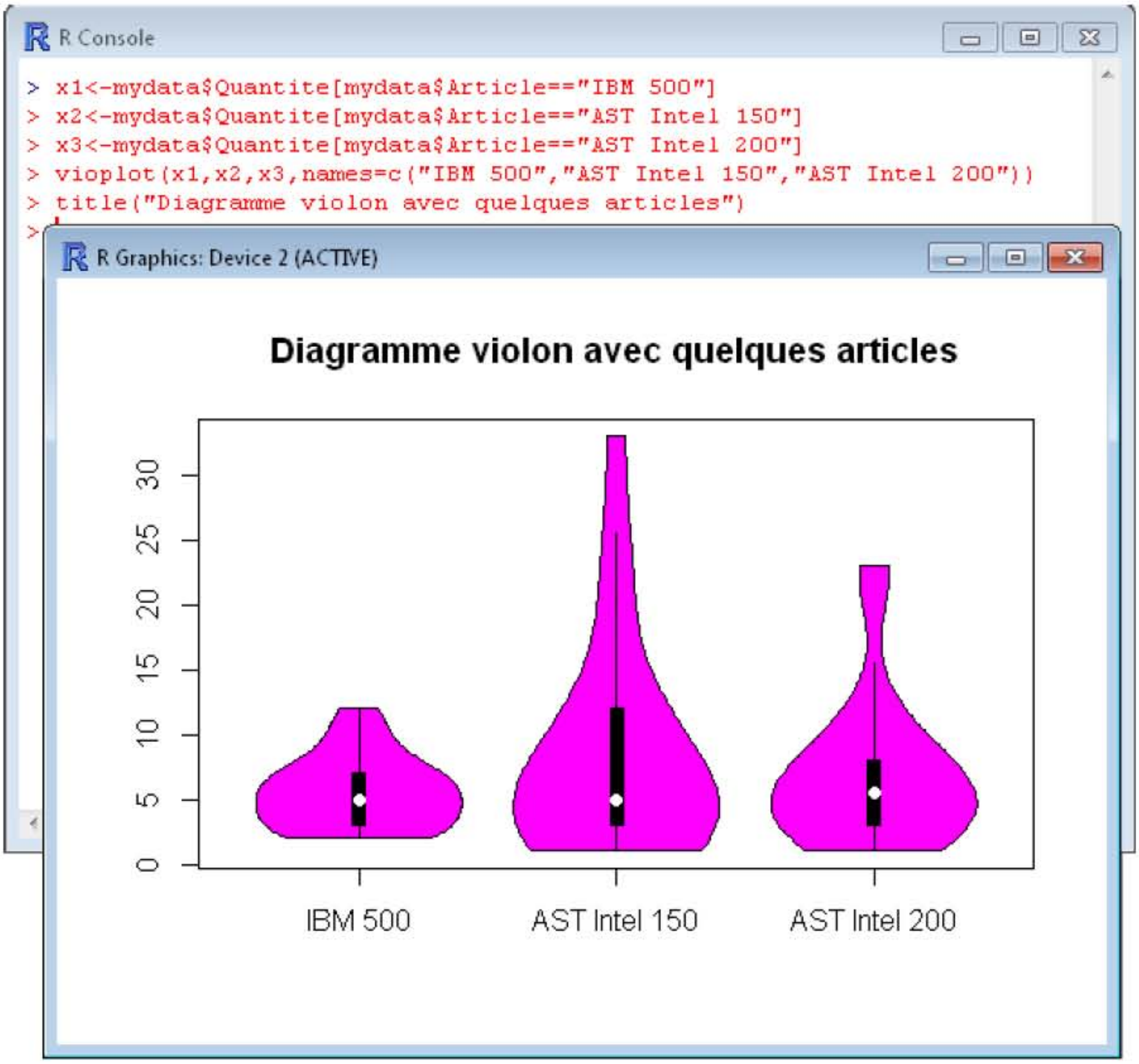


Figure 136 Graph Box-Whisker de type violon

Nous pouvons faire mieux aussi:


R Console
> x1<-mydata$Quantite[mydata$Article=="IBM 500"]
> x2<-mydata$Quantite[mydata$Article=="AST Intel 150"]
> x3<-mydata$Quantite[mydata$Article=="AST Intel 200"]
> vioplot(x1,x2,x3,names=c("IBM 500","AST Intel 150","AST Intel 200"))
> title("Diagramme violon avec quelques articles")
R Graphics: Device 2 (ACTIVE)
Diagramme violon avec quelques articles
0
5
10
15
20
25
30
IBM 500
AST Intel 150
AST Intel 200

Diagrammes de Gardner-Altman

Voir page 1312.

Diagramme pirate

Un diagramme pirate est une façon moderne de visualiser la relation entre une variable indépendante catégorielle et une variable dépendante continue. Contrairement aux méthodes de traçage traditionnelles, comme les barplots et les boxplots, un diagramme pirate (pirateplot) est un triptyque qui présente des données brutes (toutes les données comme des points), des statistiques descriptives (comme une ligne horizontale à la moyenne ou toute autre fonction que vous souhaitez) et un intervalle de densité de Bayesian à 95% lissé.

Avec nos données cela donne en utilisant le package **yarrr** et la fonction **pirateplot( )** correspondante:

```
> library(yarrr)
> mydata<-read.csv("c:/tmp/VentesClean.csv",header=T,sep=";")
> pirateplot(formula = Quantite ~ Article,
+            data = mydata,
+            main = "Quantités par Article")
>
```

Ce qui donne:

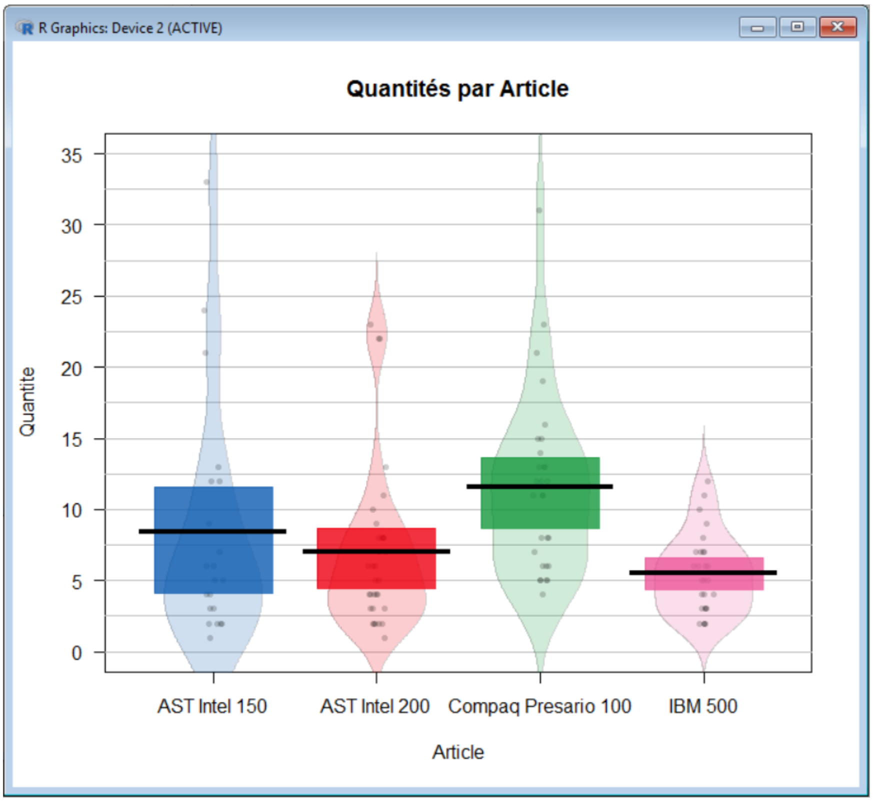
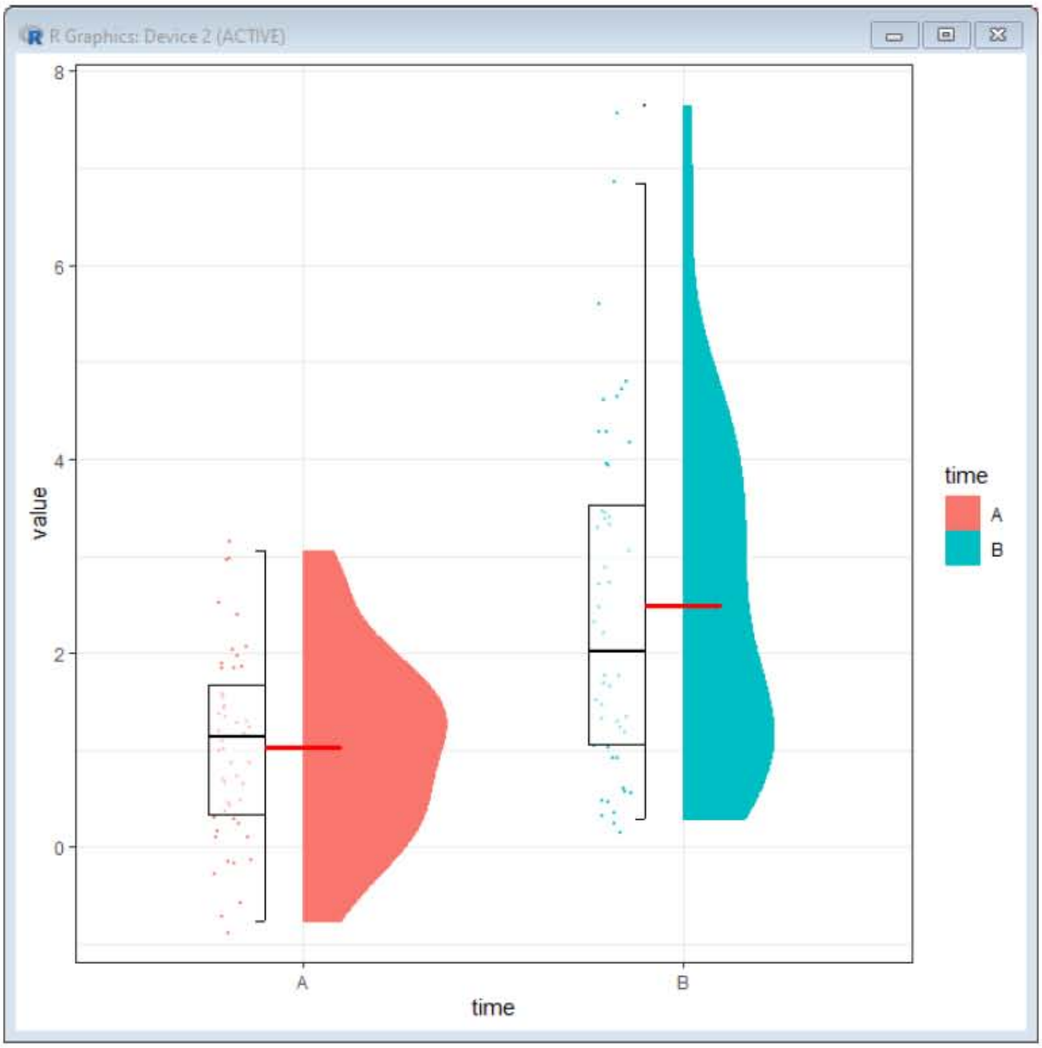


Figure 137 Graph de type Pirate

Ou encore:

```
> library(gghalves)
> library(dplyr)
> d <- data.frame(time = rep(LETTERS[1:2], each=50), value = c(rnorm(50, 1), rchisq(50, 2)))
>
> ggplot(data = d, aes(x=time, y=value)) +
+  theme_bw() +
+  geom_half_point(aes(fill = time, color=time),side = "l", size = 0.3, alpha=0.8) +
+  geom_half_boxplot(color="black", fill="white", side = "l", width = 0.5, alpha = 0.5, nudge = 0.1, outlier.size = 0) +
+  geom_half_violin(aes(fill = time,colour=time),side = "r") +
+  stat_summary(fun.data="mean_sdl", fun.args = list(mult=0), geom="crossbar", color = "red", width=.2)
>
```

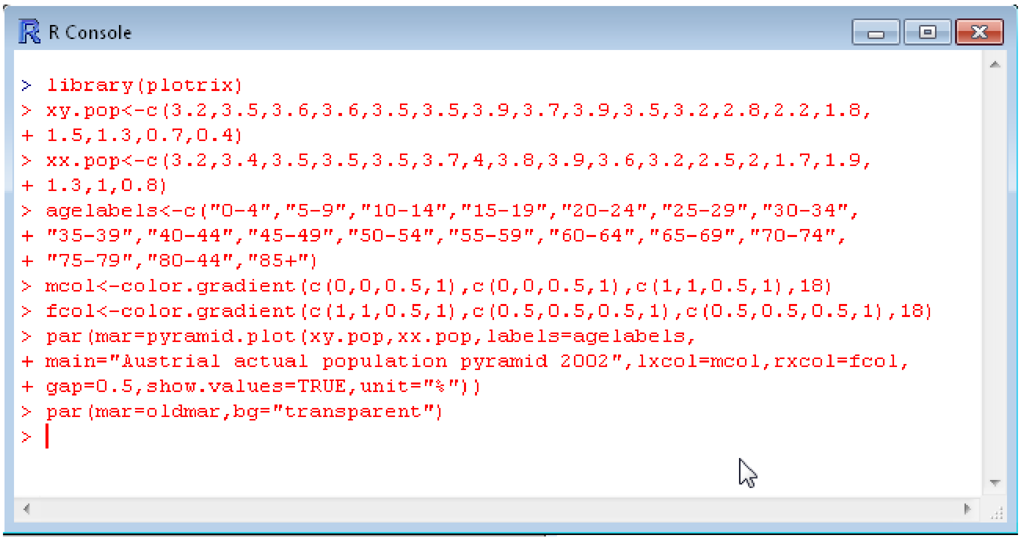
Ce qui donne:


R Graphics: Device 2 (ACTIVE)
value
time
A
B
time
A
B
0
2
4
6
8

Exemple 140.: Pyramide des âges

R 3.0.2

Un grand classique dans les tableurs aussi. Dans R il faut encore une fois utiliser la package **plotrix** mais cette fois-ci avec la commande **pyramide.plot( )** . Voici un exemple repris tel quel du PDF du package:



R Console

```
> library(plotrix)
> xy.pop<-c(3.2,3.5,3.6,3.6,3.5,3.5,3.9,3.7,3.9,3.5,3.2,2.8,2.2,1.8,
+ 1.5,1.3,0.7,0.4)
> xx.pop<-c(3.2,3.4,3.5,3.5,3.5,3.7,4,3.8,3.9,3.6,3.2,2.5,2,1.7,1.9,
+ 1.3,1,0.8)
> agelabels<-c("0-4","5-9","10-14","15-19","20-24","25-29","30-34",
+ "35-39","40-44","45-49","50-54","55-59","60-64","65-69","70-74",
+ "75-79","80-44","85+")
> mcol<-color.gradient(c(0,0,0.5,1),c(0,0,0.5,1),c(1,1,0.5,1),18)
> fcol<-color.gradient(c(1,1,0.5,1),c(0.5,0.5,0.5,1),c(0.5,0.5,0.5,1),18)
> par(mar=pyramid.plot(xy.pop,xx.pop,labels=agelabels,
+ main="Austrial actual population pyramid 2002",lxcol=mcol,rxcol=fcol,
+ gap=0.5,show.values=TRUE,unit="%"))
> par(mar=oldmar,bg="transparent")
>
```

Ce qui donne:

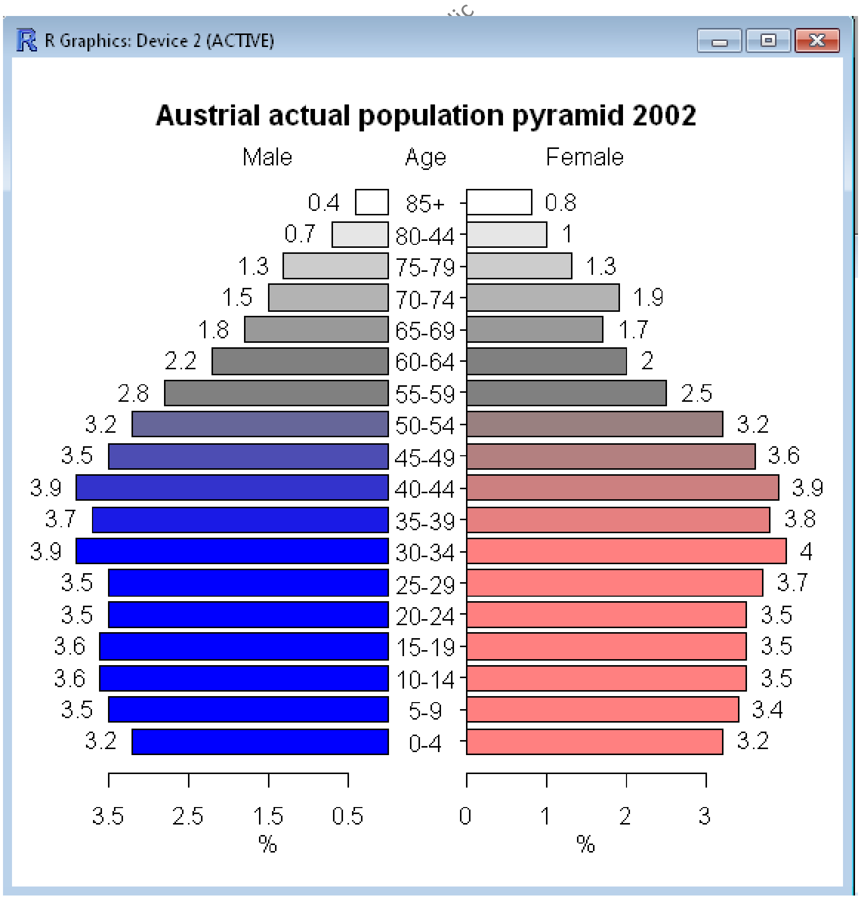
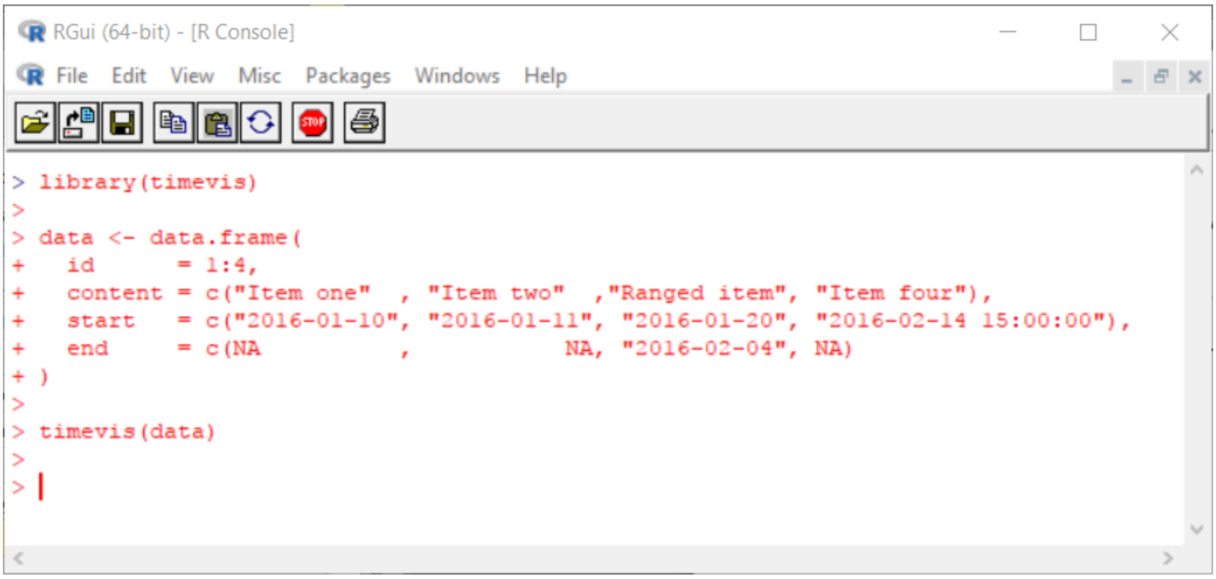


Figure 138 Graph de type pyramide des âges

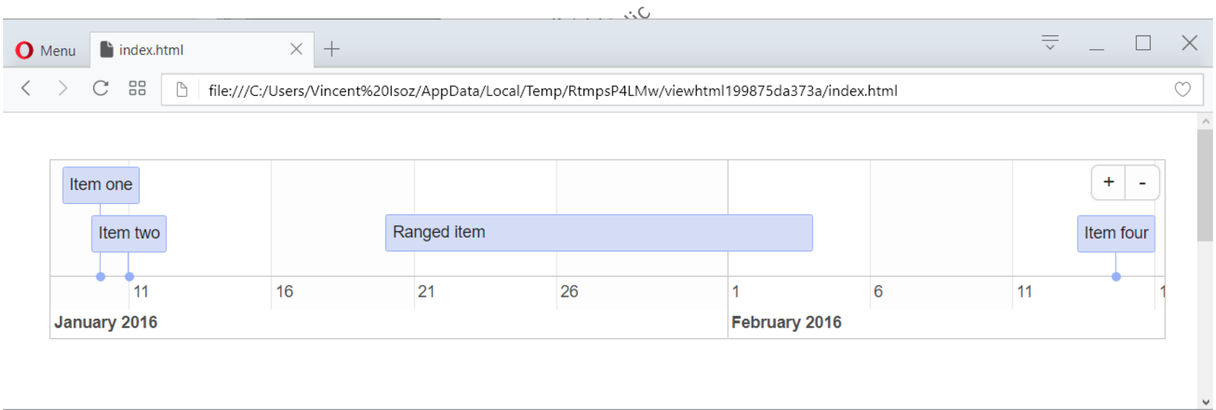
Exemple 141.: Chronologie et Diagramme de Gantt

R 3.4.1

Voyons comment créer d'abord une chronologie avec la fonction **timevis( )** du package **timevis**:



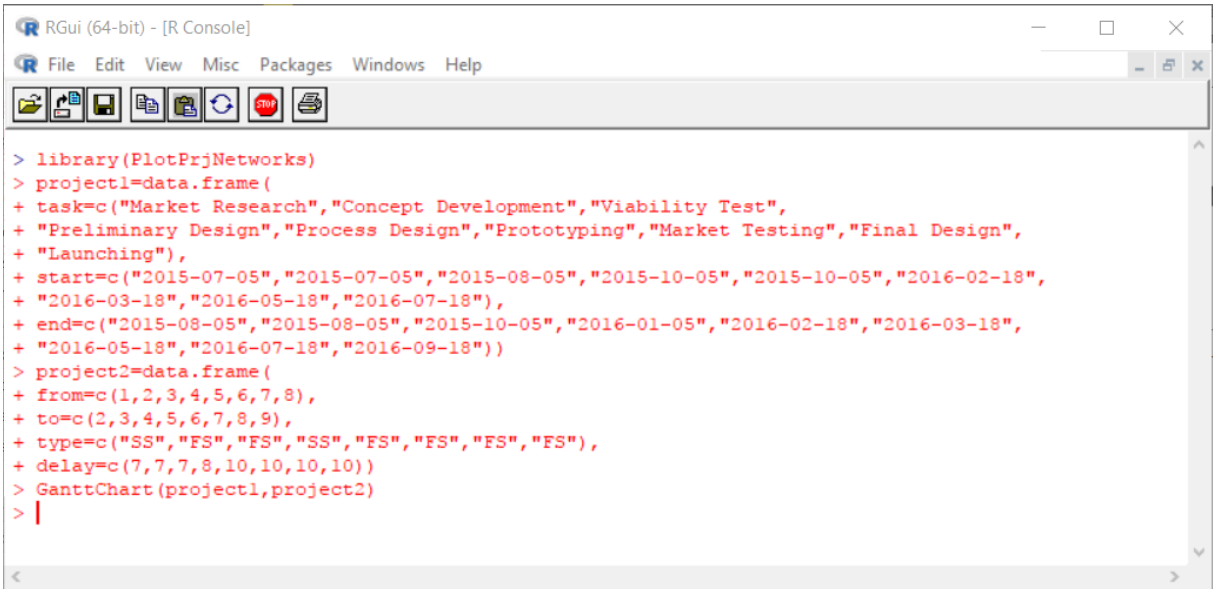
Cela crée alors dans le navigateur:



Nous sommes évidemment très loin de ce que peut faire un outil comme Microsoft Project (voir mon livre sur le sujet) ou même Microsoft Visio. Mais c'est déjà pas mal pour certain besoin basique.

Passons maintenant au Gantt qui encore une fois est beaucoup moins flexible que Microsoft Project.

Pour cela, nous allons utiliser la fonction **GanttChart( )** du package **PlotPrjNetworks**:



Ce qui donne:

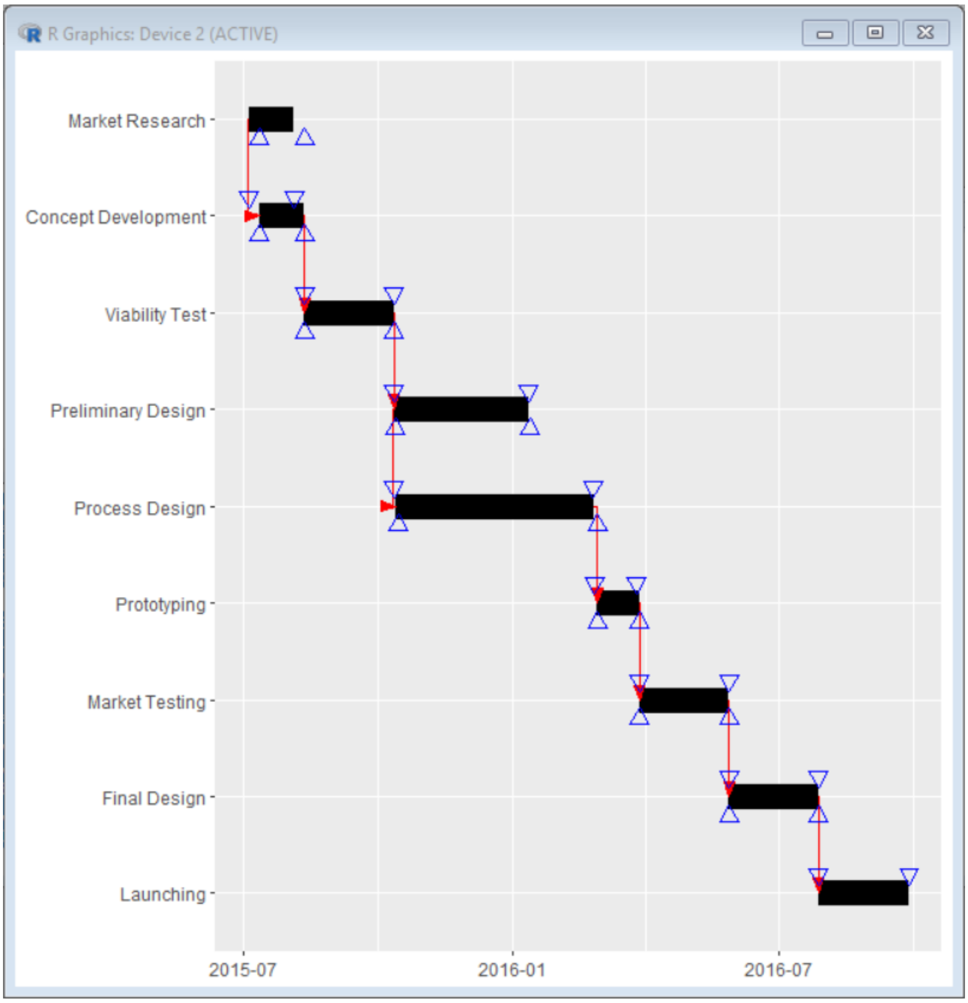


Figure 139 Graph de type Gantt (ggplot)

Exemple 142.: Organigramme

R 3.4.1

Bon là encore on est très loin des possibilités d'un logiciel comme Microsoft Visio mais au cas où voici comment faire un organigramme simple avec R:

```
> library(diagram)
> par(mfrow=c(1,1))
> par(mar=c(0,0,0,0))
> ##initialize new grphics device
> openplotmat()
> ##number of elements per row
> elpos<-coordinates (c(1,1, 2, 2, 2, 3,2 ))
> ##draw arrows from each row to next row
> treearrow(from=elpos[1,],to=elpos[2,],lwd=6)
> treearrow(from=elpos[2,],to=elpos[3:4,],lwd=6)
> treearrow(from=elpos[3,],to=elpos[5,],lwd=6)
> treearrow(from=elpos[4,],to=elpos[6,],lwd=6)
>
> treearrow(from=elpos[5,],to=elpos[7:8,],lwd=6)
>
> treearrow(from=elpos[7,],to=elpos[9:10,],lwd=6)
> treearrow(from=elpos[8,],to=elpos[11,],lwd=6)
>
> treearrow(from=elpos[9,],to=elpos[12,],lwd=6)
> treearrow(from=elpos[10,],to=elpos[13,],lwd=6)
>
> ##create a generic 3-lined label for each textbox
> labels = vector(length=13)
> labels[1] = "Vincent Isoz (CEO)\n Scientific Evolution Sàrl"
> labels[2] = "Daname Kolani\n Quantitative Finance Consultant"
> for(i in 3:14) {
+ labels[i] = "Firstname Lastname\n Function"
+ }
> ##plot text boxes
> for ( i in 1:13) textround (elpos[i,],radx=0.08,rady=0.04,lab=labels[i],cex=0.6)
>
```

Ce qui donne:

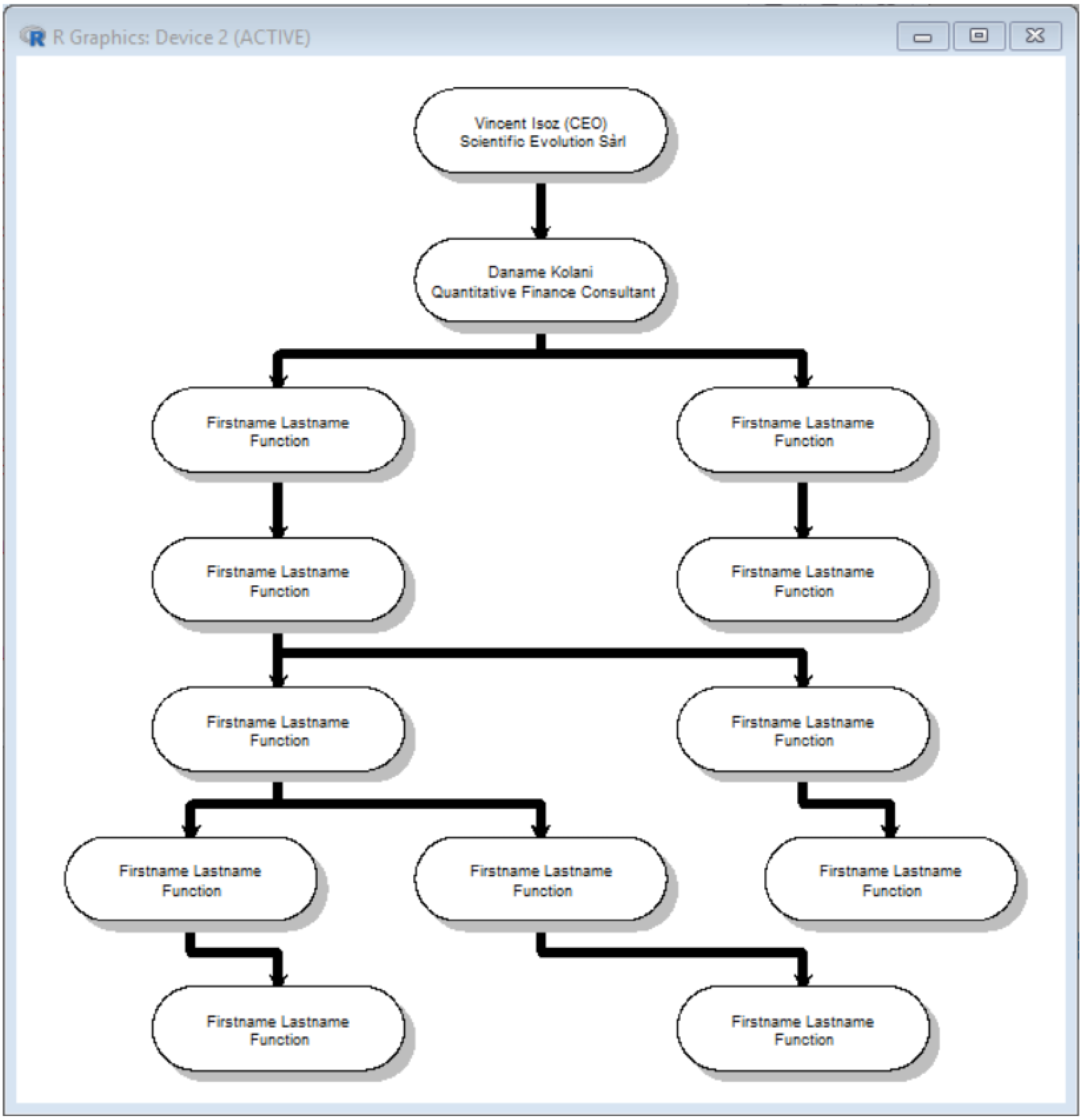


Figure 140 Graph de type Organigramme

Exemple 143.: Gaufres

R 3.4.1

Voyons quelques graphiques faits avec la fonction **waffle( )** du package **waffle**:

```
> library(waffle)
> savings <- c(`Mortgage ($84,911)`=84911, `Auto andntuition loans ($14,414)`=14414,
+              `Home equity loans ($10,062)`=10062, `Credit Cards ($8,565)`=8565)
> waffle(savings/392, rows=7, size=0.5,
+        colors=c("#c7d4b6", "#a3aabd", "#a0d0de", "#97b5cf"),
+        title="Average Household Savings Each Year",
+        xlab="1 square == $392")
> 
```

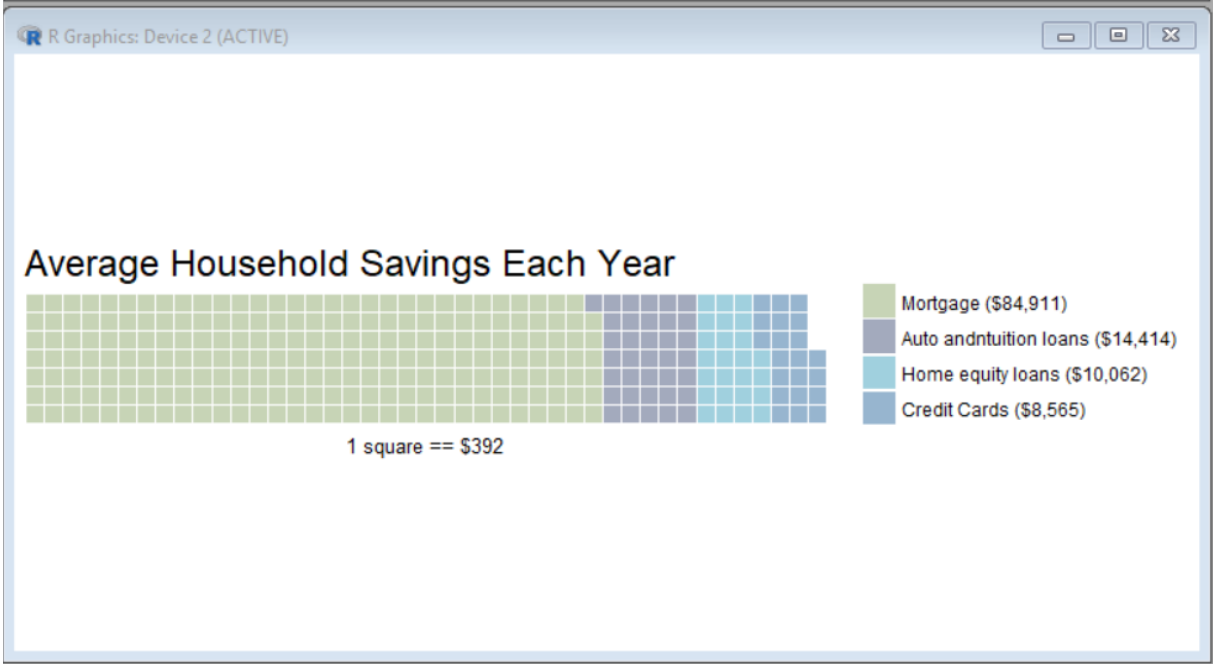
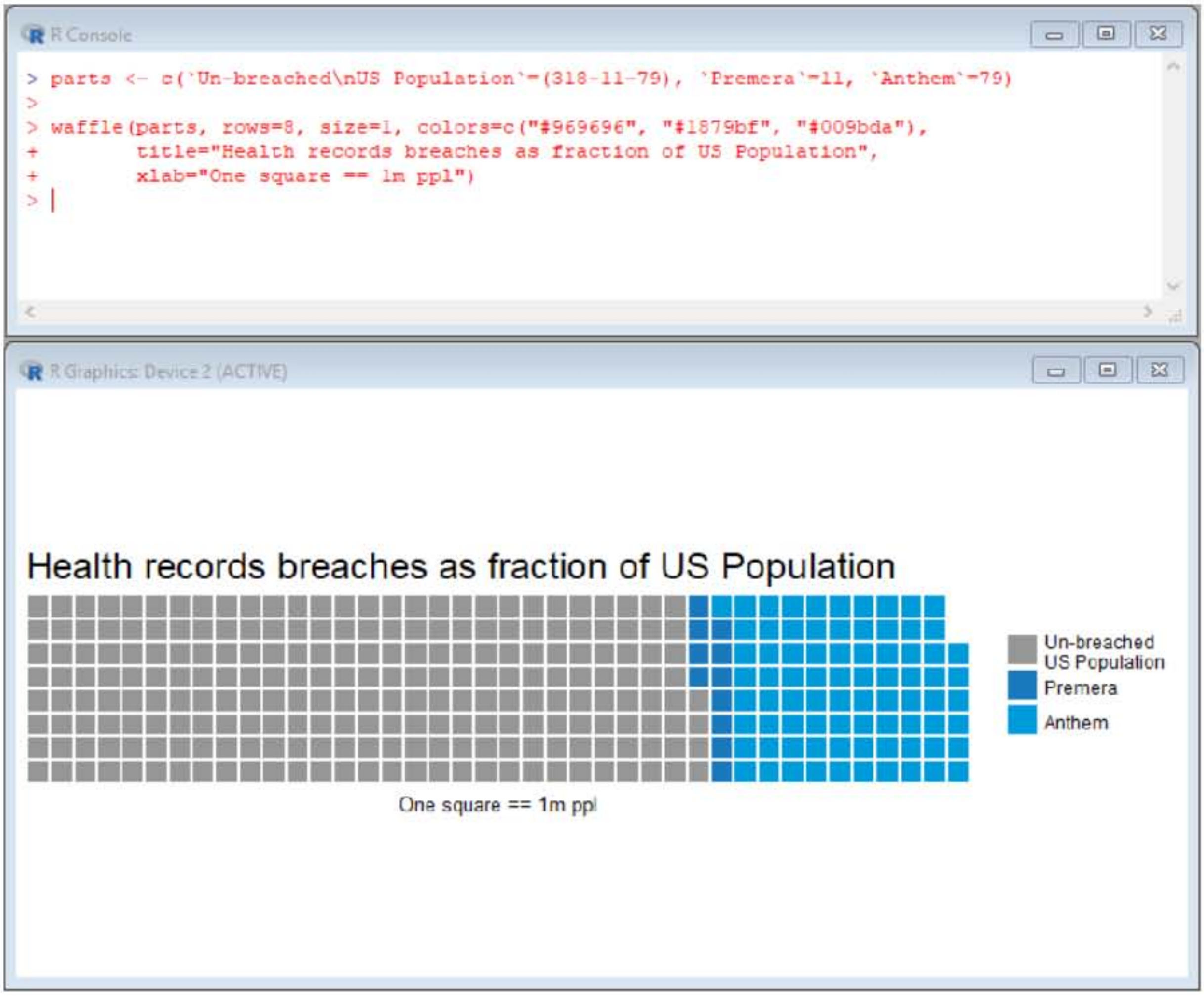


Figure 141 Graph de type Gaufres

ou encore:


R Console
> parts <- c(`Un-breached\nUS Population`=(318-11-79), `Premera`=11, `Anthem`=79)
>
> waffle(parts, rows=8, size=1, colors=c("#969696", "#1879bf", "#009bda"),
+ title="Health records breaches as fraction of US Population",
+ xlab="One square == 1m ppl")
>
R Graphics: Device 2 (ACTIVE)
Health records breaches as fraction of US Population
Un-breached
US Population
Premera
Anthem
One square == 1m ppl

Exemples 144.: Hémicycle

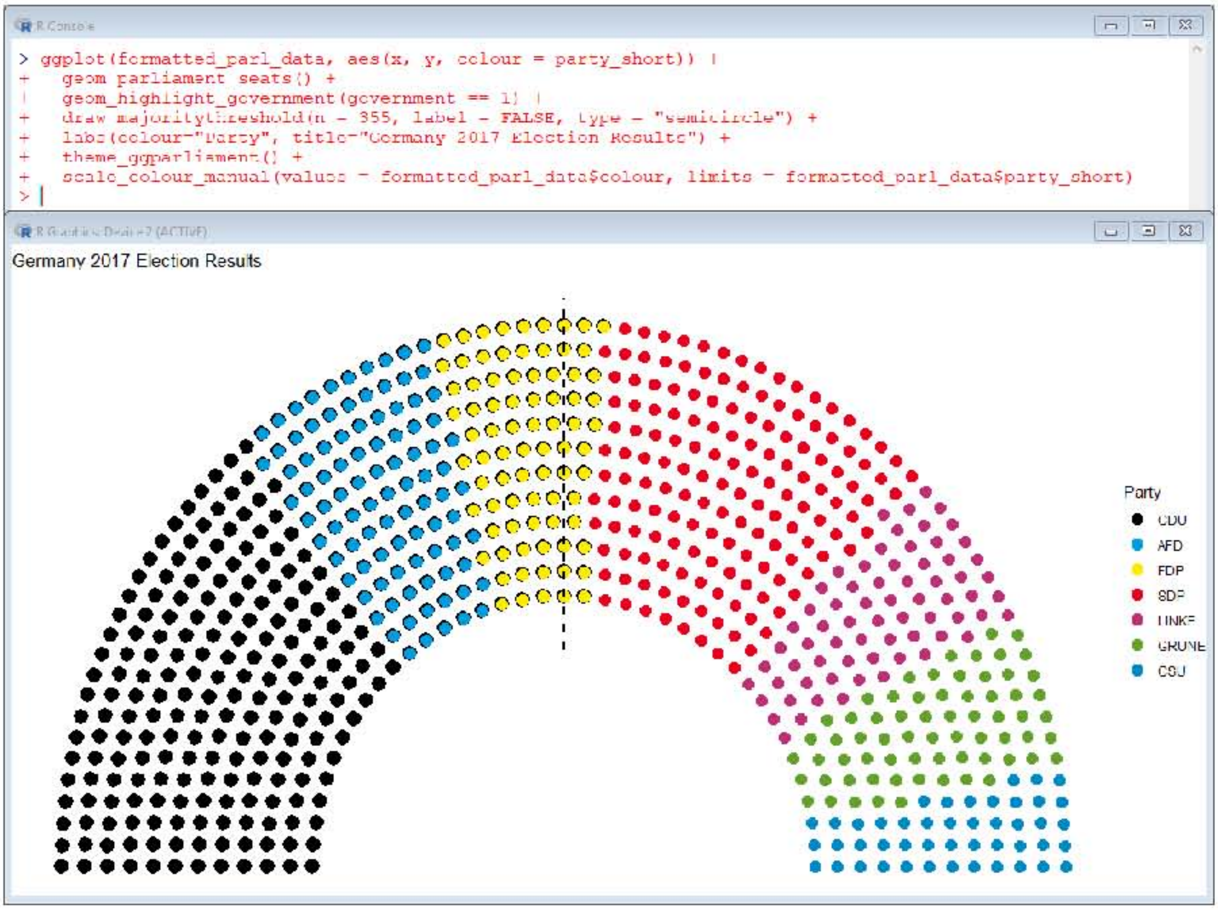
R 3.5.1

Ici rien d'extraordinaire. Juste un copié collé de la vignette du package **ggparliament** et de ses différentes couches (calques) ggplot2 en particulier de **geom_parliament_seats( )**.

Donc d'abord présentons le jeu de données inclus dans le package et formattons-le un peu avec dplyr:

```
> library("ggparliament")
> library("ggplot2")
> library("dplyr")
> unformatted_data <- election_data %>%
+   filter(country == "Germany" & year == "2017") %>%
+   arrange(party_long)
> unformatted_data
  year country     house                       party_long party_short seats government  colour
1 2017 Germany Bundestag           Alliance 90/The Greens       GRUNE    67          0 #64A12D
2 2017 Germany Bundestag           Alternative for Germany         AFD    94          1 #009EE0
3 2017 Germany Bundestag         Christian Democratic Union         CDU   200          1 #000000
4 2017 Germany Bundestag Christian Social Union in Bavaria         CSU    46          0 #008AC5
5 2017 Germany Bundestag             Free Democratic Party         FDP    80          1 #FFED00
6 2017 Germany Bundestag           Social Democratic Party         SDP   153          0 #EB001F
7 2017 Germany Bundestag                          The Left       LINKE    69          0 #BE3075
>
> formatted_data <- unformatted_data %>%
+   arrange(-government, -seats)
> formatted_parl_data <- formatted_data %>%
+   parliament_data(.,parl_rows = 12,party_seats = .$seats,type = "semicircle")
> dim(formatted_parl_data)
[1] 709  12
> head(formatted_parl_data)
    year country     house                 party_long party_short seats government  colour         x            y row    theta
1   2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -2.000000 2.449213e-16  12 3.141593
1.1 2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -1.909091 2.337885e-16  11 3.141593
1.2 2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -1.818182 2.226557e-16  10 3.141593
1.3 2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -1.727273 2.115229e-16   9 3.141593
1.4 2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -1.636364 2.003901e-16   8 3.141593
1.5 2017 Germany Bundestag Christian Democratic Union         CDU   200          1 #000000 -1.545455 1.892573e-16   7 3.141593
> 
```

Enfin, voici la partie pour le graphique:


R Console
> ggplot(formatted_parl_data, aes(x, y, colour = party_short)) +
+ geom_parliament_seats() +
+ geom_highlight_government(government == 1) +
+ draw_majoritythreshold(n = 355, label = FALSE, type = "semicircle") +
+ labs(colour="Party", title="Germany 2017 Election Results") +
+ theme_ggparliament() +
+ scale_colour_manual(values = formatted_parl_data$colour, limits = formatted_parl_data$party_short)
Germany 2017 Election Results
Party
CDU
AFD
FDP
SDP
LINKE
GRUNE
CSU

Exemples 145.: Diagrammes de Taylor

R 4.1.1

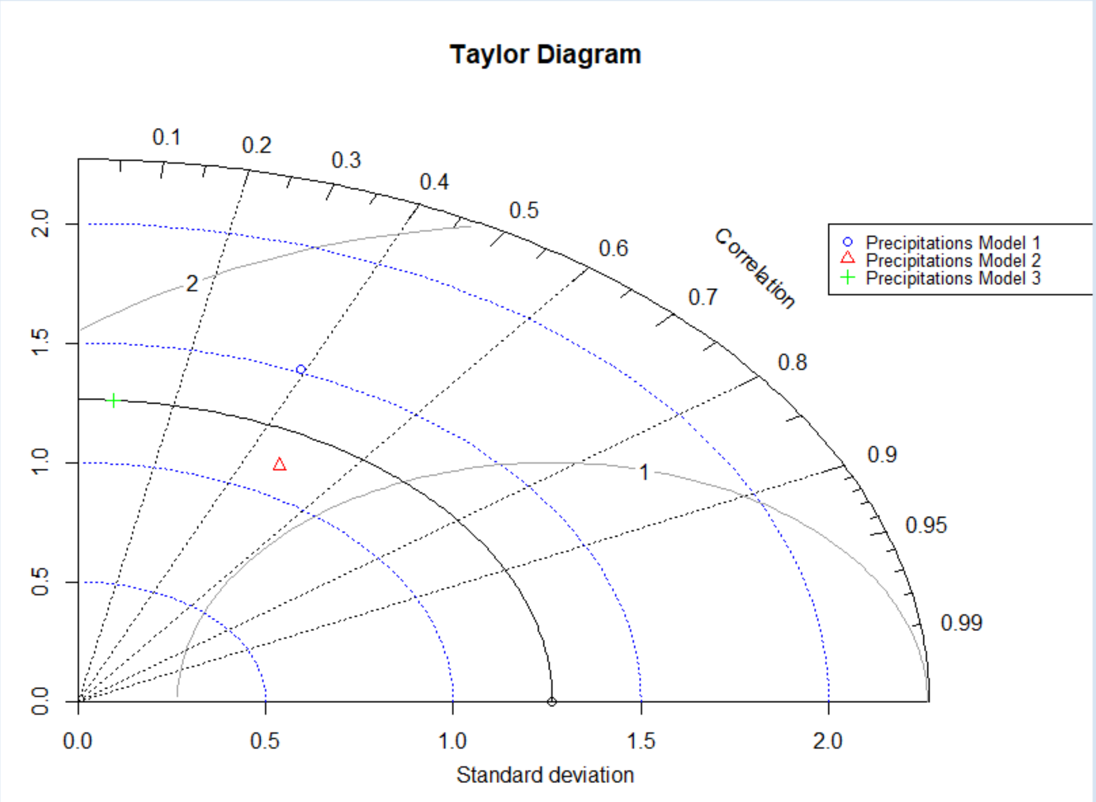
Les diagrammes de Taylor sont des diagrammes mathématiques conçus pour indiquer graphiquement laquelle de plusieurs représentations approximatives (ou modèles) d'un système, d'un processus ou d'un phénomène est la plus réaliste. Ce diagramme, inventé par Karl E. Taylor en 1994 (publié en 2001) facilite l'évaluation comparative de différents modèles. Il est utilisé pour quantifier le degré de correspondance entre le comportement modélisé et observé en termes de trois statistiques : le coefficient de corrélation de Pearson, l'erreur de l'erreur quadratique moyenne (RMSE) et l'écart type.

Nous allons voir que le package plotrix intègre ce diagramme via la fonction **taylor.diagram( )**:

R Console

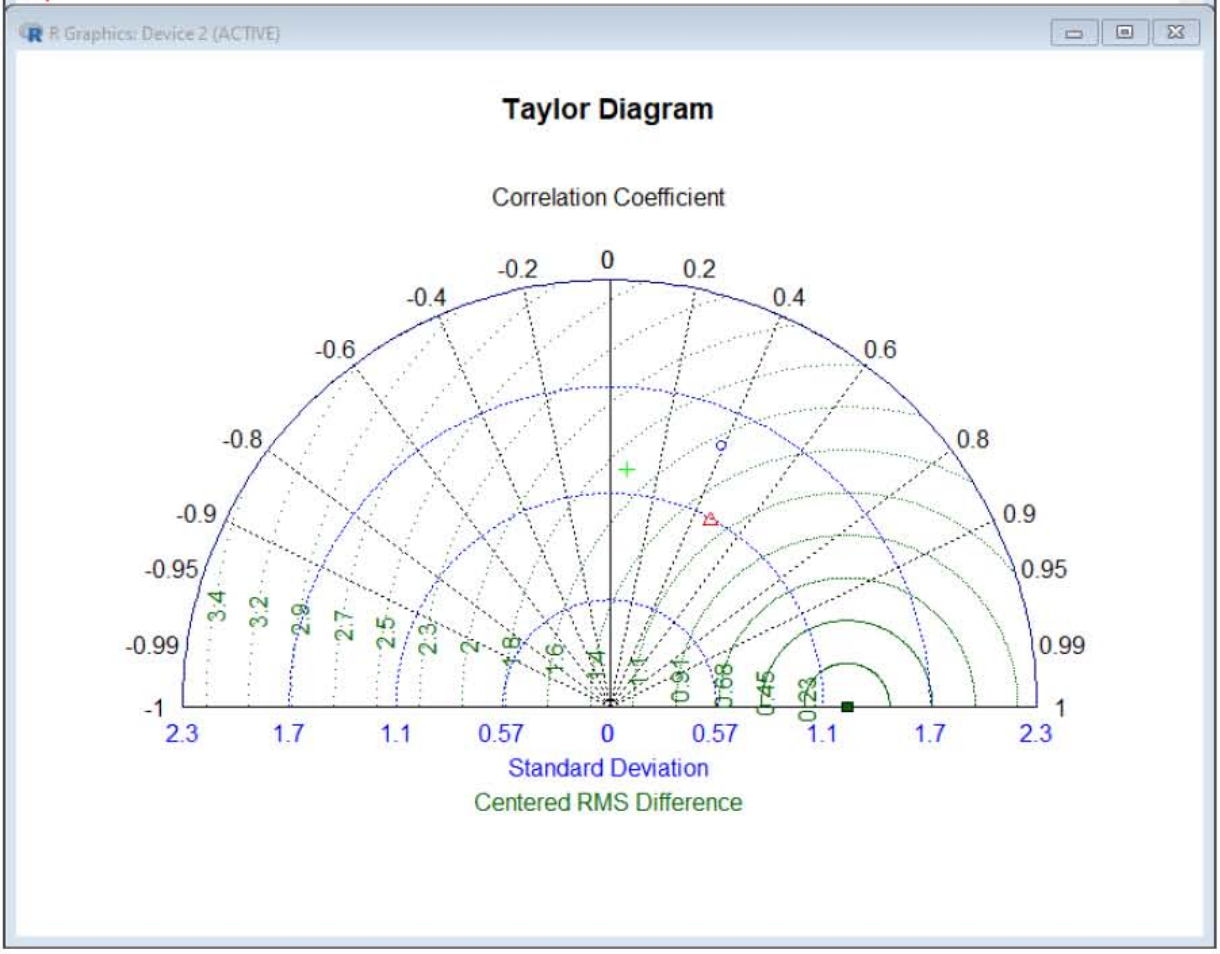
```
> library("plotrix")
>
> str(precipitation<-read.csv("c:/tmp/Precipitations.csv",header=T))
'data.frame':   1000 obs. of  5 variables:
 $ X  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Ref: num  12.1 13.2 12.4 13.6 13.2 ...
 $ M1 : num  8.35 10.57 7.78 9.25 11.57 ...
 $ M2 : num  5.03 5.56 5.62 6.86 5.92 ...
 $ M3 : num  15 13 15.7 14.9 14 ...
>
> taylor.diagram(precipitation$Ref, precipitation$M1, col = "blue", pch = 1,sd.arcs=T, ref.sd=T)
> taylor.diagram(precipitation$Ref, precipitation$M2, col = "red", pch = 2, add = TRUE)
> taylor.diagram(precipitation$Ref, precipitation$M3, col = "green", pch = 3, add = TRUE)
>
> legend(2,2,cex=0.8,pt.cex=1
+ ,legend=c("Precipitations Model 1","Precipitations Model 2","Precipitations Model 3")
+ ,pch=c(1,2,3),col=c("blue","red","green"))
>
```

R Graphics: Device 2 (ACTIVE)



Ou alternativement:

```
> library("plotrix")
>
> str(precipitation<-read.csv("c:/tmp/Precipitations.csv",header=T))
'data.frame':   1000 obs. of  5 variables:
 $ X  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Ref: num  12.1 13.2 12.4 13.6 13.2 ...
 $ M1 : num  8.35 10.57 7.78 9.25 11.57 ...
 $ M2 : num  5.03 5.56 5.62 6.86 5.92 ...
 $ M3 : num  15 13 15.7 14.9 14 ...
>
> taylor.diagram(precipitation$Ref, precipitation$M1, col = "blue", pch = 1,sd.arcs=T, ref.sd=T
+ ,pos.cor=F,xlab="")
> taylor.diagram(precipitation$Ref, precipitation$M2, col = "red", pch = 2, add = TRUE)
> taylor.diagram(precipitation$Ref, precipitation$M3, col = "green", pch = 3, add = TRUE)
>
```



Exemples 146.: Surfaces 3D

R 3.4.1

Une vieille fonction qui nous est bien connue (sinus cardinal) pour introduire la commande **persp( )** (les plot de surface 3D sont très utiles dans l'industrie où l'on mesure des défauts de surfaces à quelques dixièmes de microns ou même moins!):

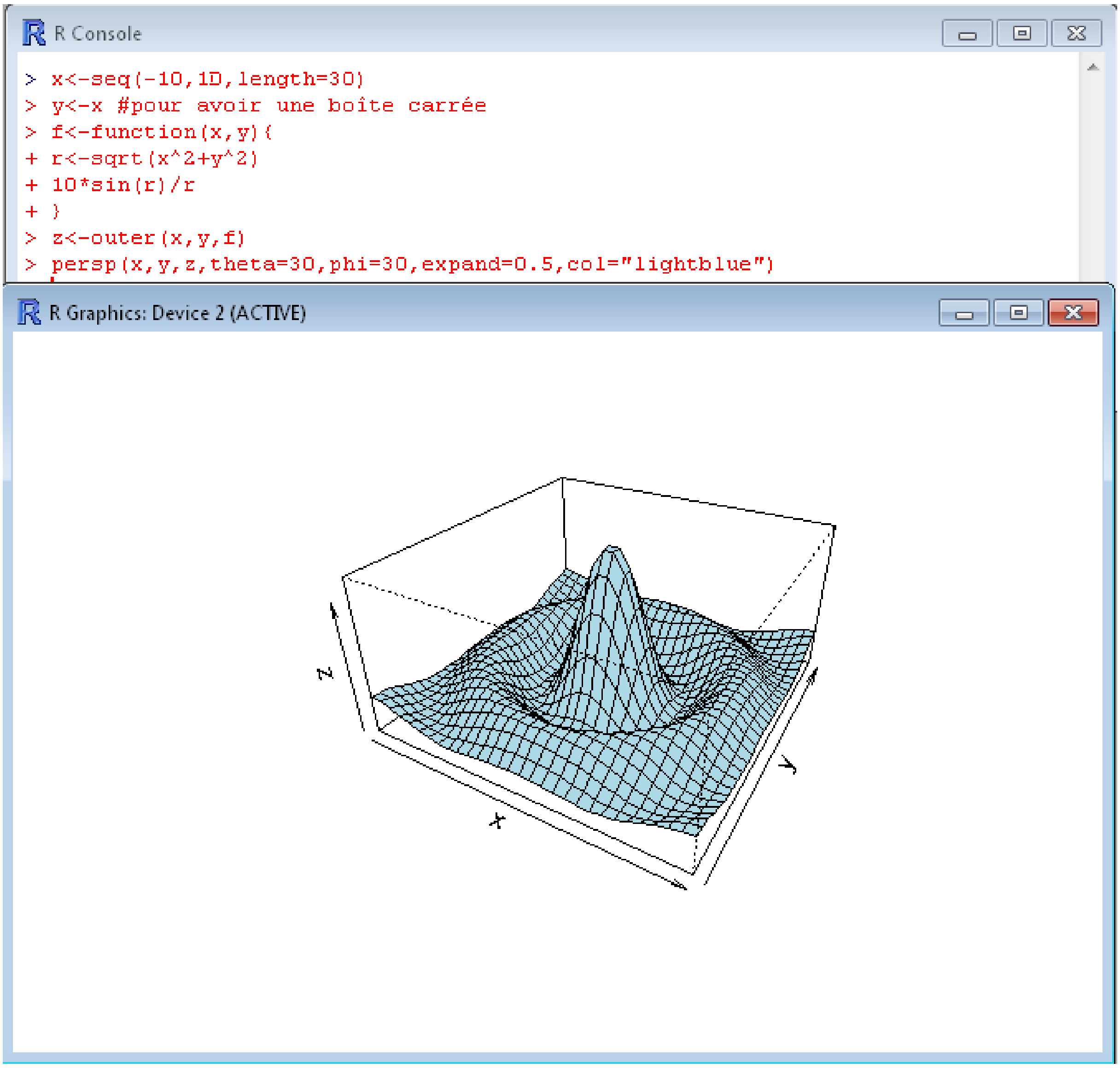
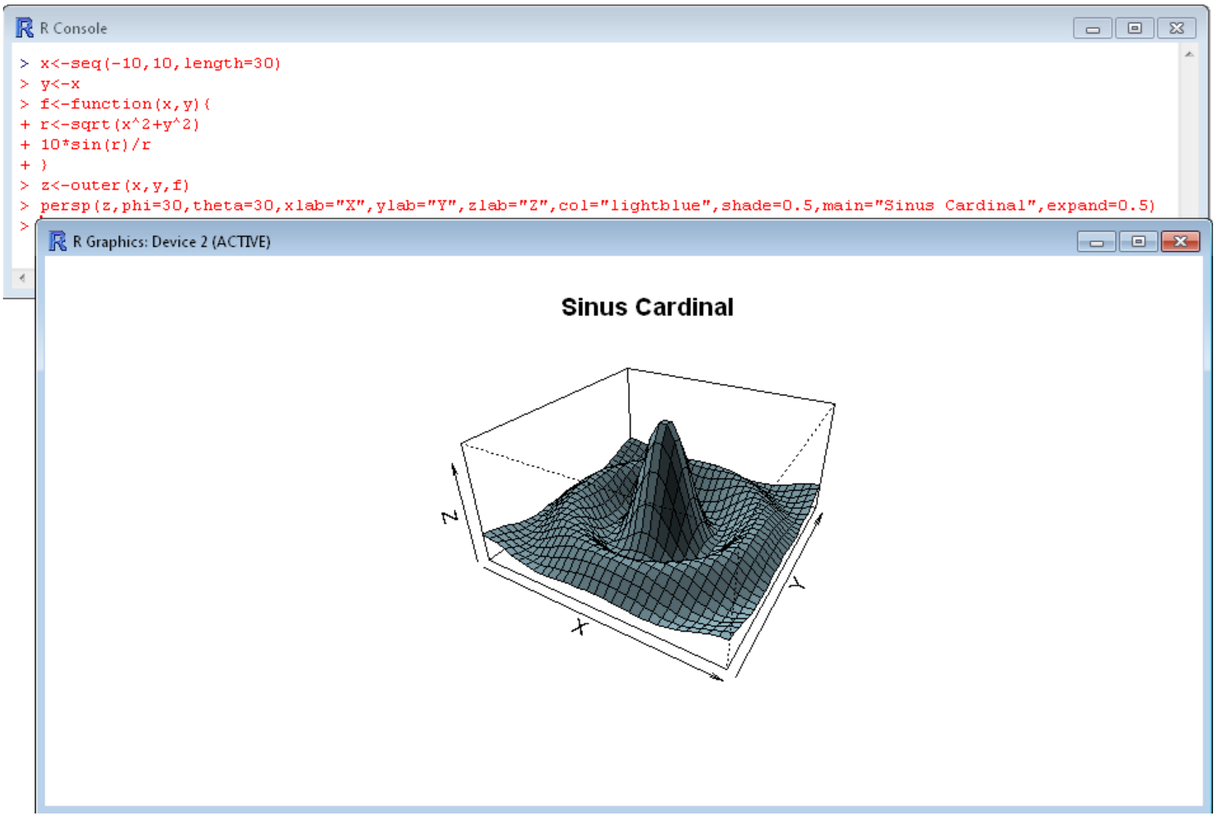
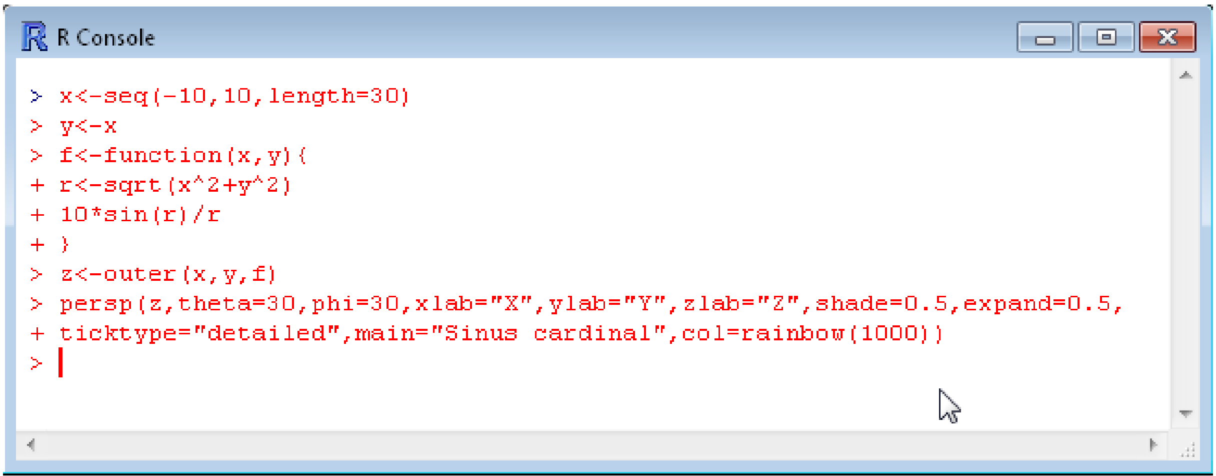


Figure 142 Graph 3D de type surface

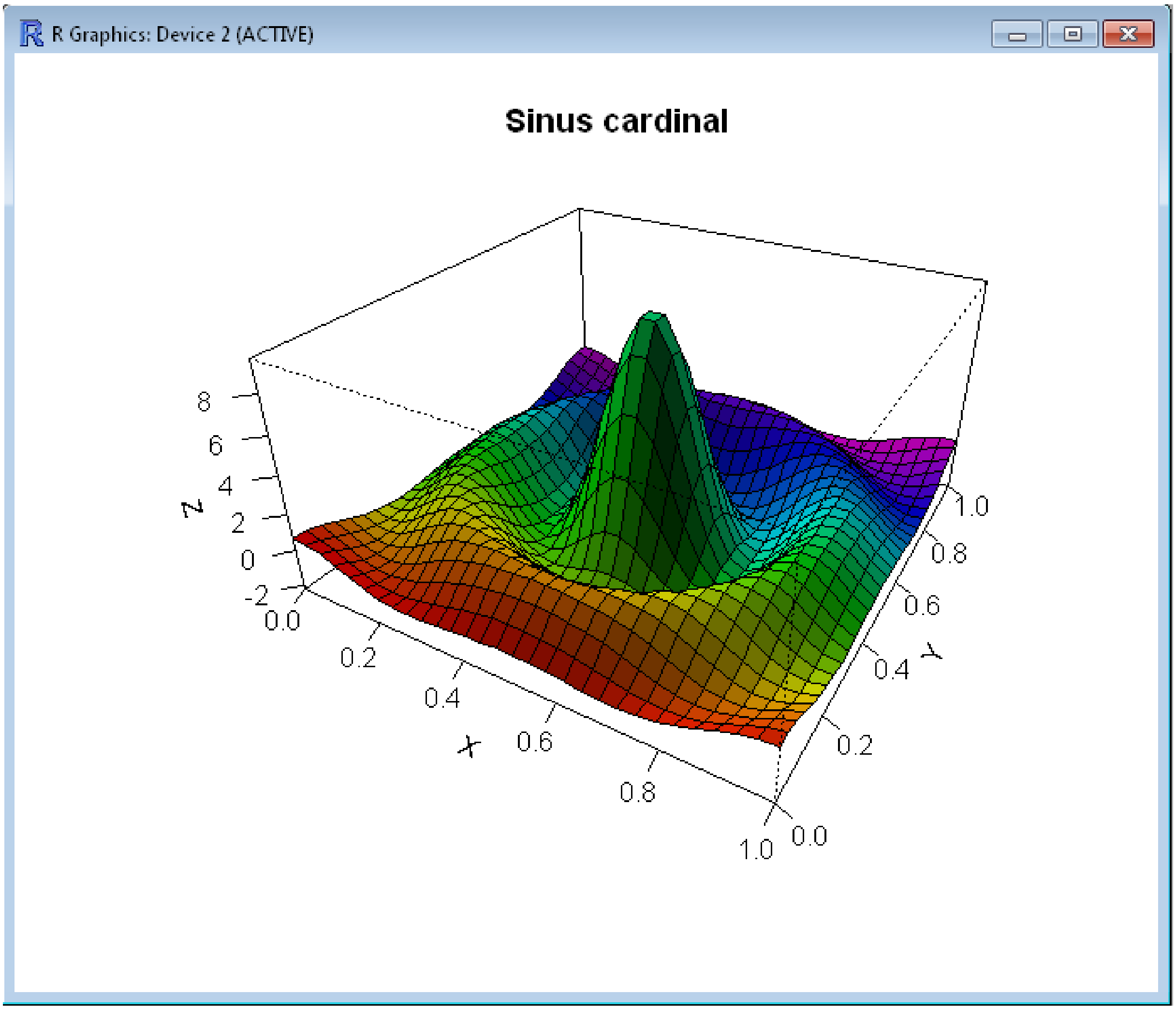
Ou avec un peu d'ombre:



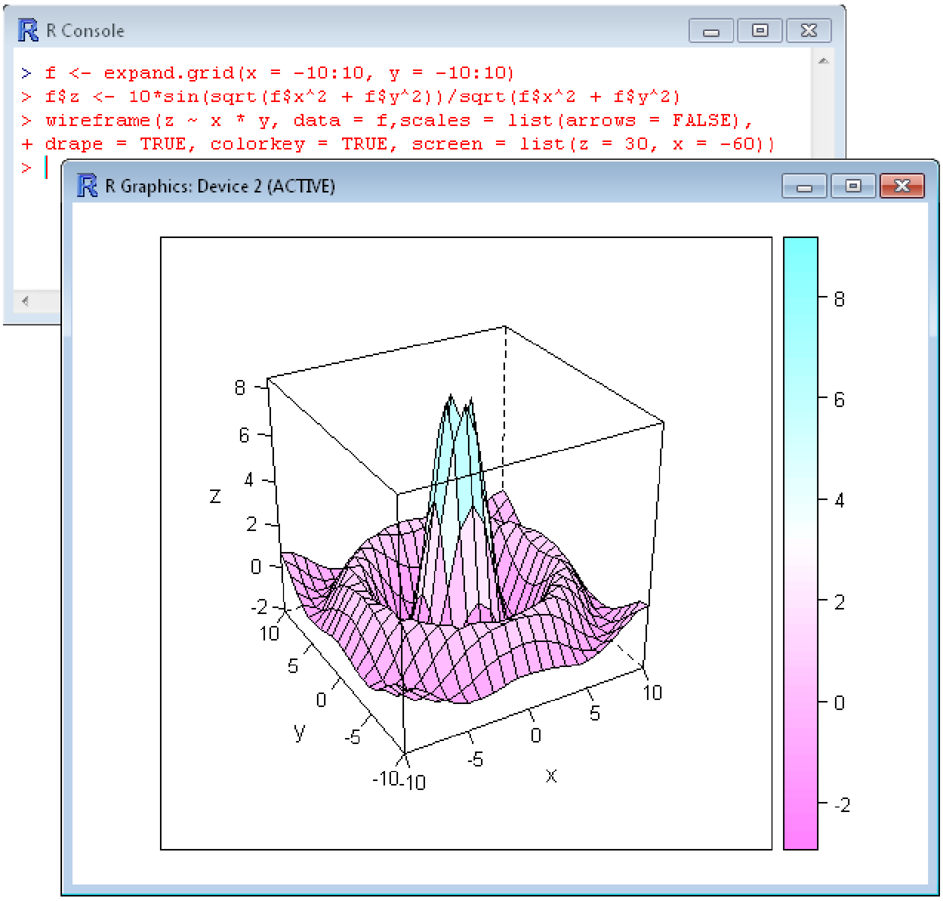
ou dans encore inutilement plus beau:



Ce qui donne (voir page suivante):



Ou encore avec un autre style visuel (voir page suivante):



Ou encore en utilisant le package **fields** pour avoir les dégradés dans le bon sens:

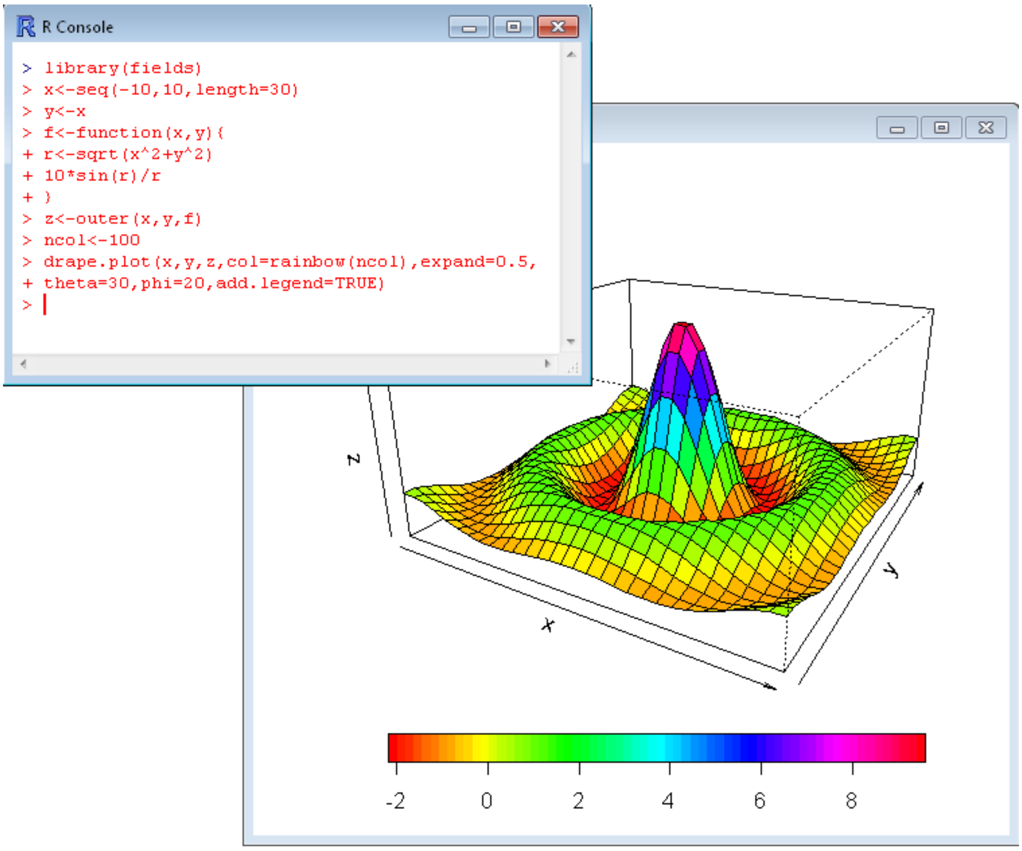


Figure 143 Graph 3D de type surface avec gradient de couleurs

Sinon avec le package **rgl**, nous pouvons animer une surface en utilisant respectivement les commandes **rgl.surface( )** , **plot3d( )** et **spin3d( )** et **rgl.snapshot( )** pour exporter en tant qu'image:

R Console

```
> library(rgl)
> setwd("C:/tmp")
> x<-seq(-10,10,length=30)
> y<-x
> f<-function(x,y){
+ r<-sqrt(x^2+y^2)
+ 10*sin(r)/r
+ }
> z<-outer(x,y,f)
> jet.colors<-colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
+ "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
> colorzjet <- jet.colors(100)
> rgl.surface(x,y,z,alpha=0.75, back="lines",
+ color=colorzjet[ findInterval(z, seq(min(z), max(z), length=100))])
> bg3d("white")
> bbox3d(color=c("white","black"))
> title3d(color="black","","","xlab","zlab","ylab")
> grid3d("y")
> #on exporte en tant qu'image
> rgl.snapshot("3DPlot.png")
> play3d(spin3d(axis=c(0,1,0),rpm=4),duration=20)
```

Ce qui donne:

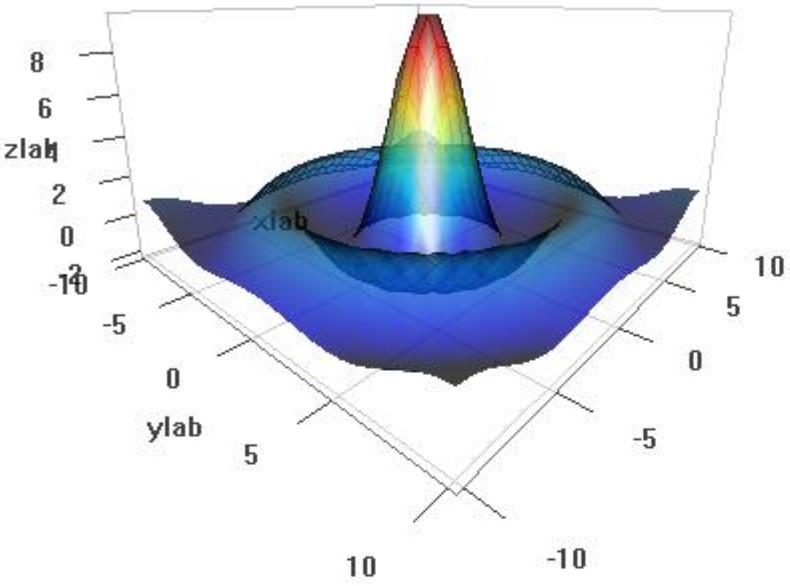


Figure 144 Graph 3D de type surface avec gradient de couleurs et interactif

Remarque: Il vaut mieux faire une capture d'écran que d'exporter en tant qu'image car la qualité de celle-ci est discutable.

Ou sinon si nous revenons à la fonction baleine à bosse que nous avons rencontré lors de l'optimisation non linéaire, nous pouvons tracer les isoclines à l'aide de la commande **contour( )** :

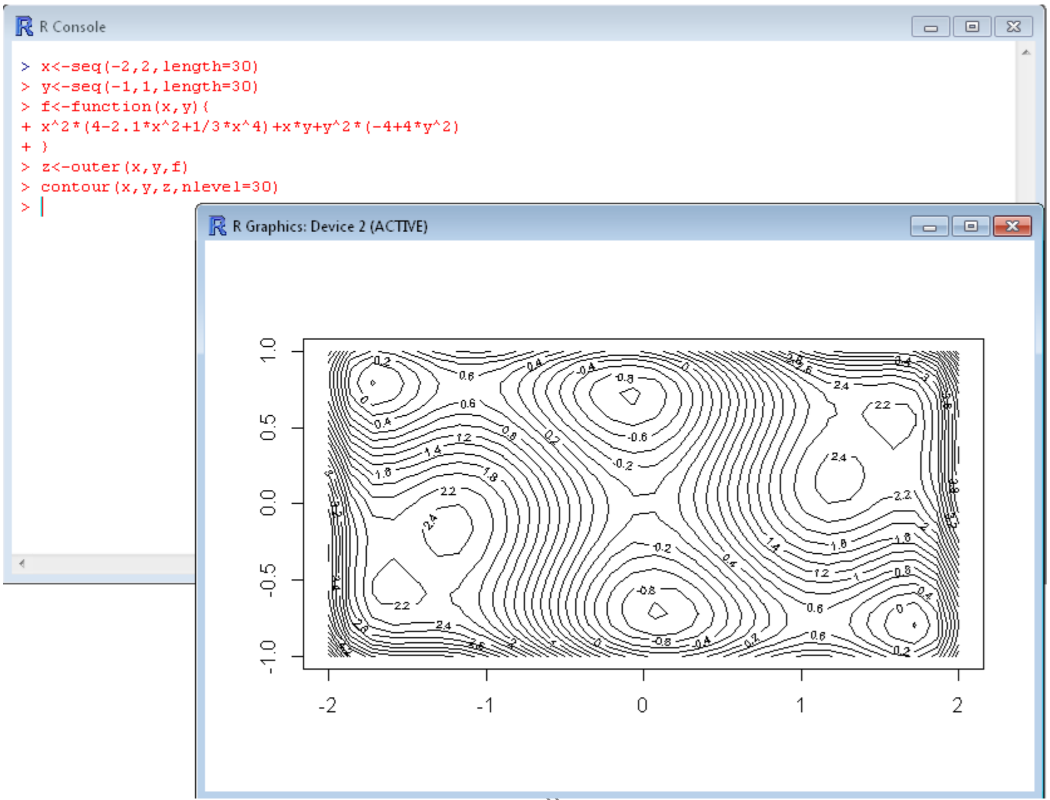


Figure 145 Graph de projection avec isoclines de fonction f(x,y)

ou avec la commande **filled.contour( )** :

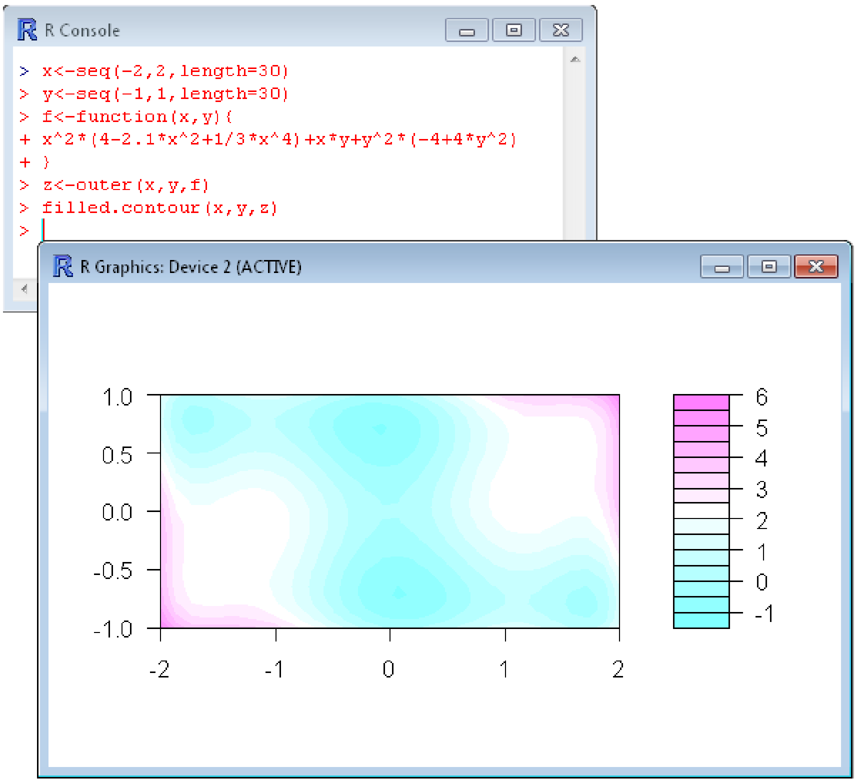
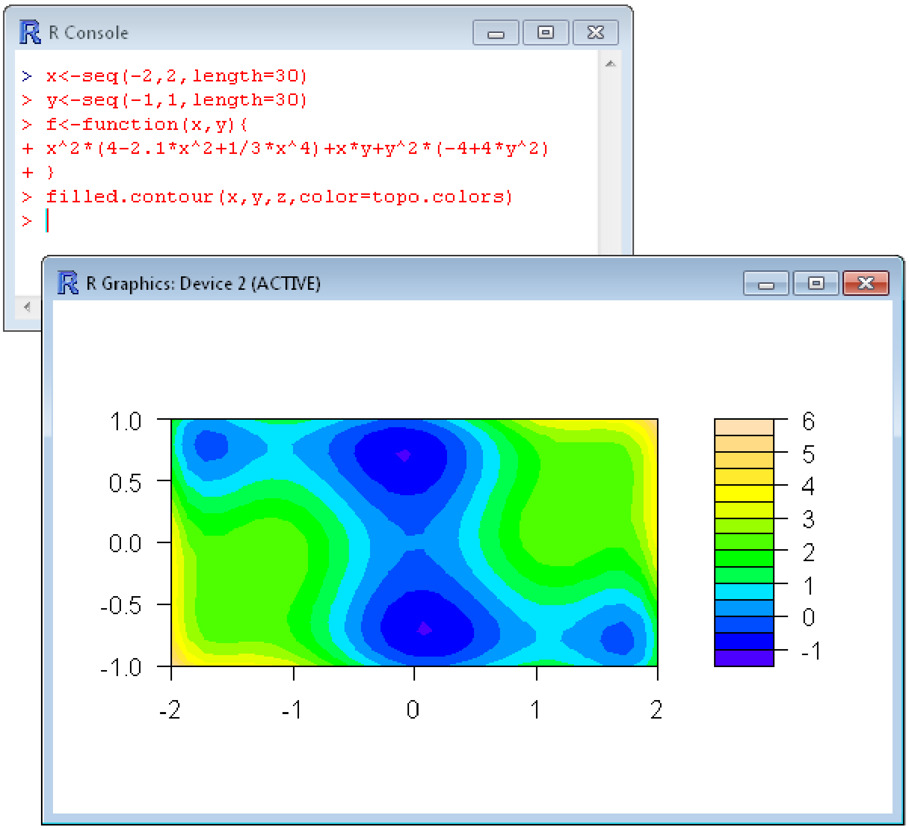
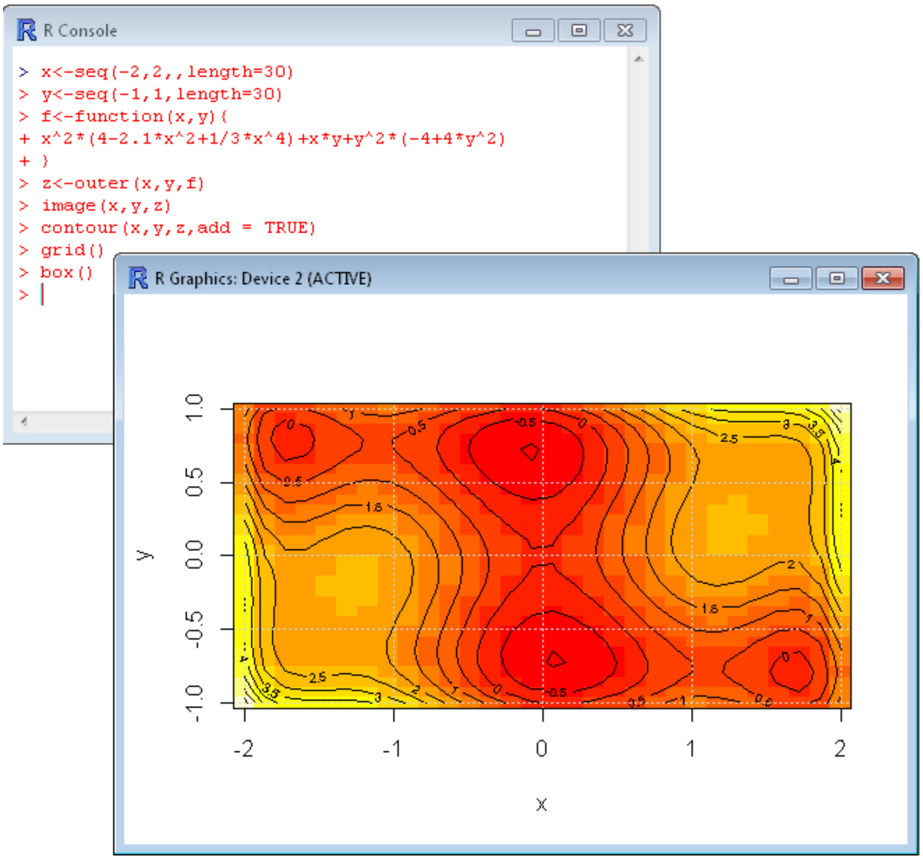


Figure 146 Graph de projection de densité de fonction f(x,y)

Ou en utilisant des couleurs propres au domaine de la topographie:



ou en cumulant ce que nous venons de voir avec la commande **image( )** :



ou avec un style plus propre au domaine de la géomatique avec la commande **image.plot( )** du package **fields**:

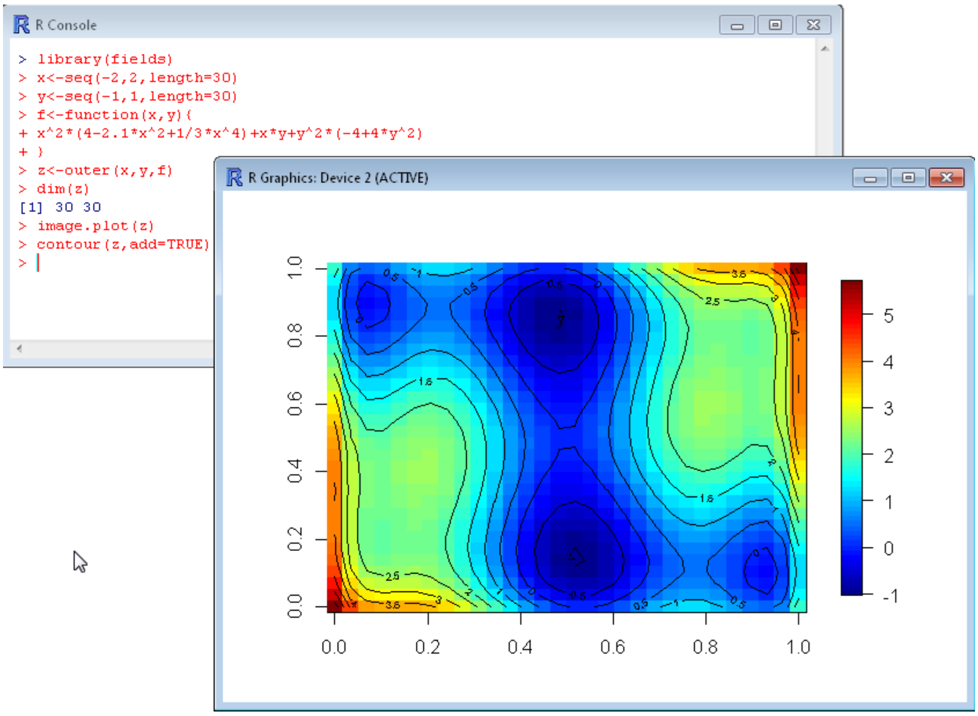
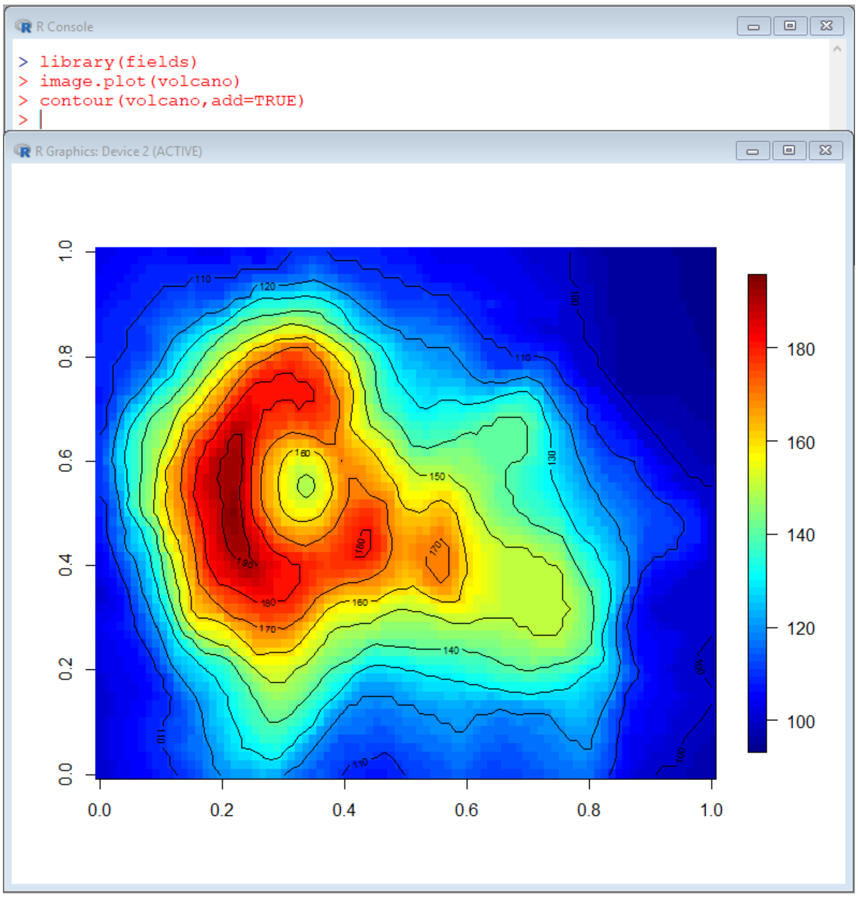
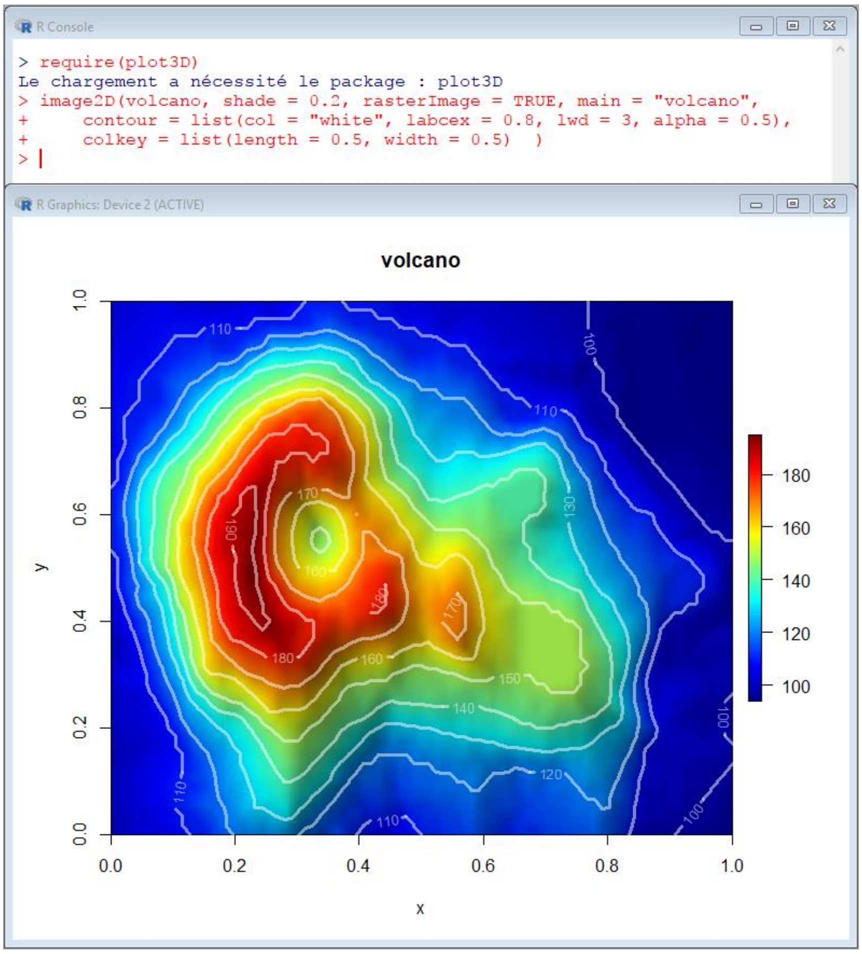


Figure 147 Graph de projection de densité et isoclines de fonction f(x,y)

ou toujours avec la même fonction et le même package, mais sur le jeux de données *volcano* (c'est plus sympa…):



Ou en utilisant le package **plot3D** et sa fonction **image2D( )** :

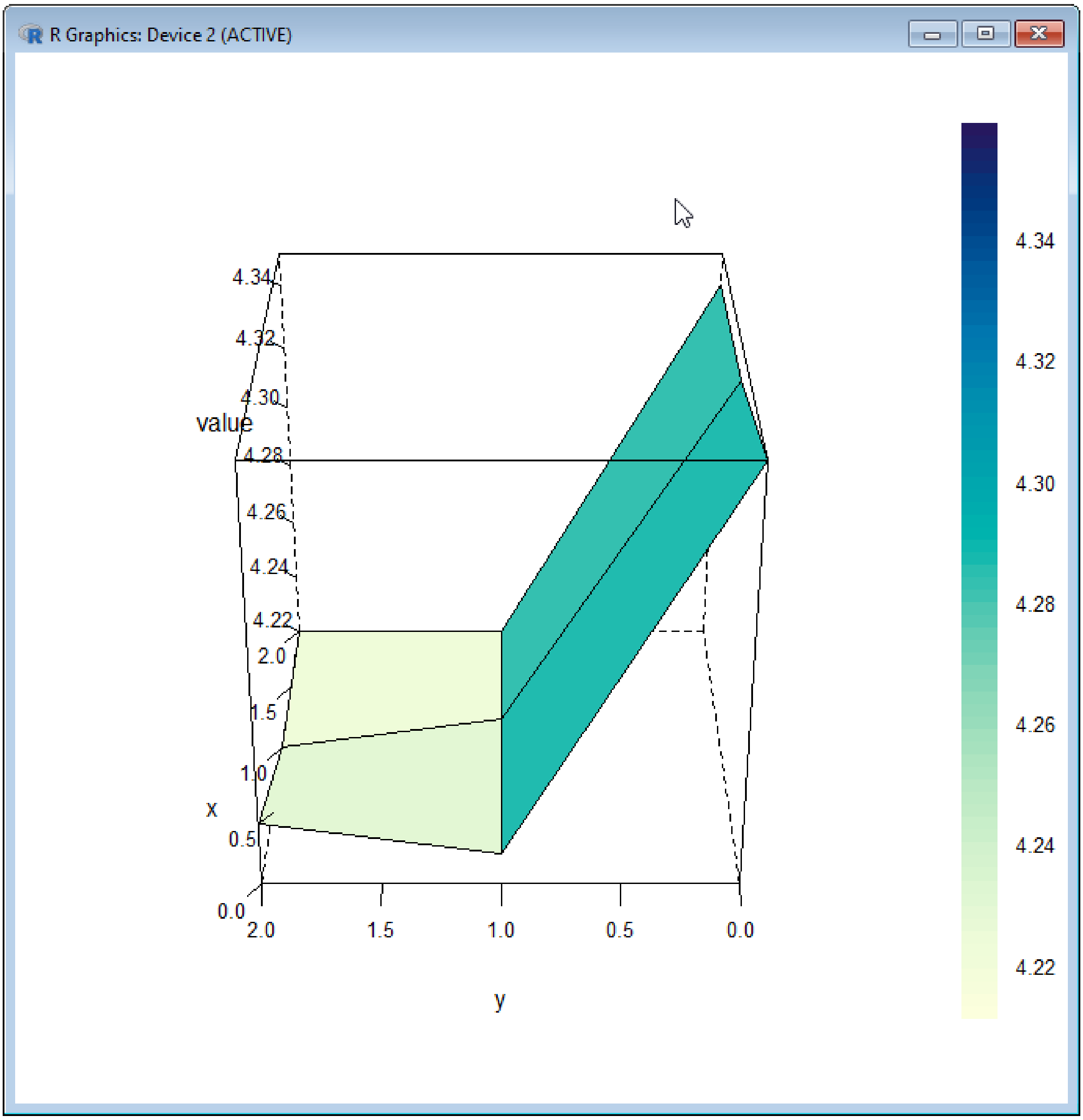


Surface 3D filaire (remplie ou pas)

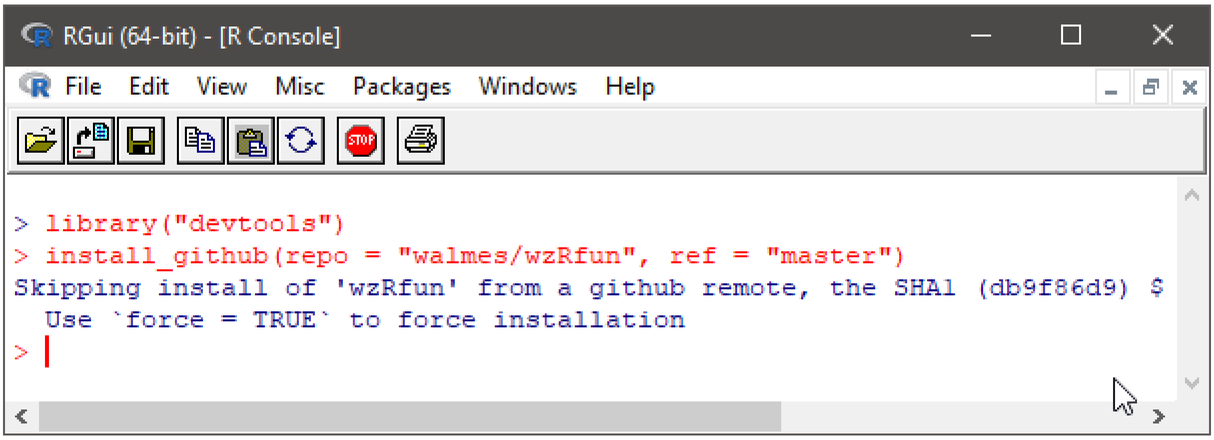
Commençons par le cas simple et moche mais fonctionnel grâce au package **lattice** et à sa fonction **wireframe( )**:

```
> #cela pourrait être un fichier CSV avec trois colonnes
> #ce serait pareil!
> mydata <- read.table(text= "x    y    value
+ 0    0    4.35
+ 0    1    4.23
+ 0    2    4.24
+ 1    0    4.34
+ 1    1    4.23
+ 1    2    4.22
+ 2    0    4.34
+ 2    1    4.22
+ 2    2    4.22", header = TRUE)
>
>
> library("lattice")
> wireframe(value ~ x + y, data = mydata,
+             screen = list(z = 90, x = -60),
+ #on enlève les flèches immondes et on met la graduation
+             scales = list( arrows = FALSE, col="black"),
+             #on remplit les polygones
+ drape = TRUE,
+ #on met la légende de couleurs
+             colorkey = TRUE,
+ #on enlève les bordures horribles
+             par.settings = list(axis.line = list(col = 'transparent'))
+         )
>
```

Ce qui donne:



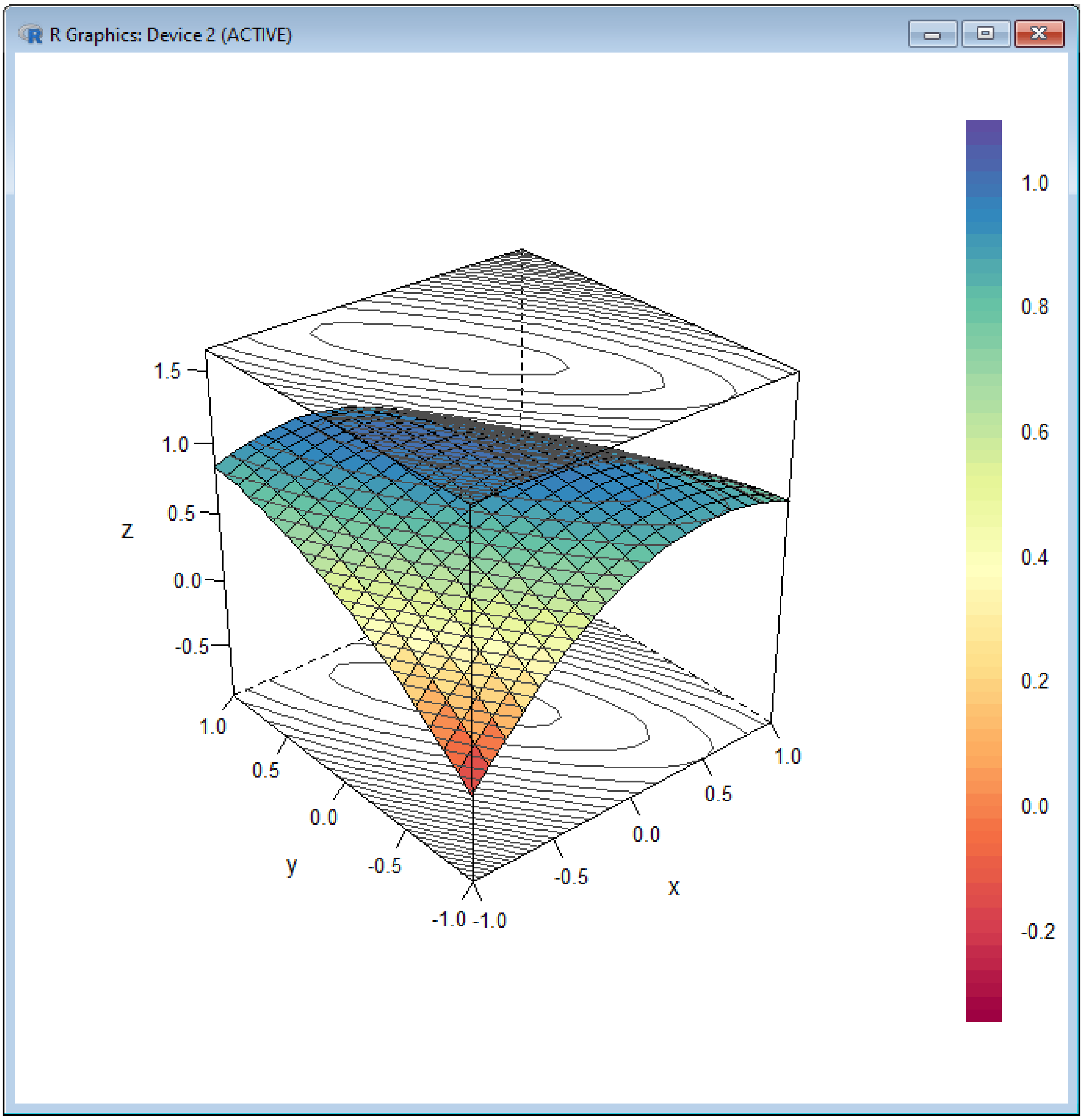
Nous pouvons faire plus élaboré avec un package supplémentaire nommé **wzRfun** se trouvant sur Github et qui permet de dessiner les contours des courbes de niveau:



Ainsi, nous pouvons typiquement avec un script du genre suivant:

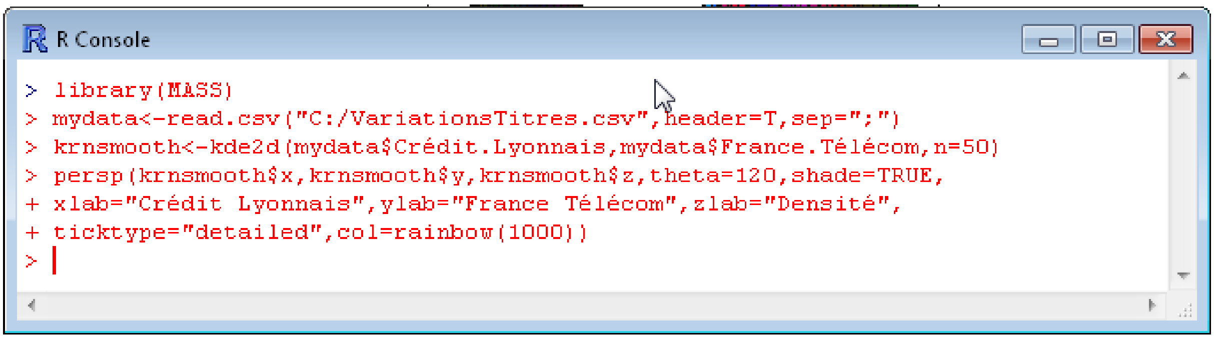
```
> library("lattice")
> library("latticeExtra")
> library("RColorBrewer")
> library("wzRfun")
>
> colr <- brewer.pal(11, "Spectral")
> colr <- colorRampPalette(colr, space = "rgb")
>
> grid <- expand.grid(x = seq(-1, 1, by = 0.1),
+                     y = seq(-1, 1, by = 0.1))
> grid$z <- with(grid, 1 + 0.01 * x + 0.05 * y -
+                      0.5 * x * y - 0.5 * x^2 - 0.2 * y^2)
>
> wireframe(z ~ x + y, data = grid,
+           scales = list( arrows = FALSE, col="black"),
+           zlim = extendrange(grid$z, f = 0.5),
+           panel.3d.wireframe = panel.3d.contour,
+           nlevels = 18, col = "gray30",
+           type = c("on", "top", "bottom"),
+           col.regions = colr(101), drape = TRUE,
+           par.settings = list(axis.line = list(col = 'transparent'))
+           )
>
```

Ce qui permet d'obtenir le résultat suivant:

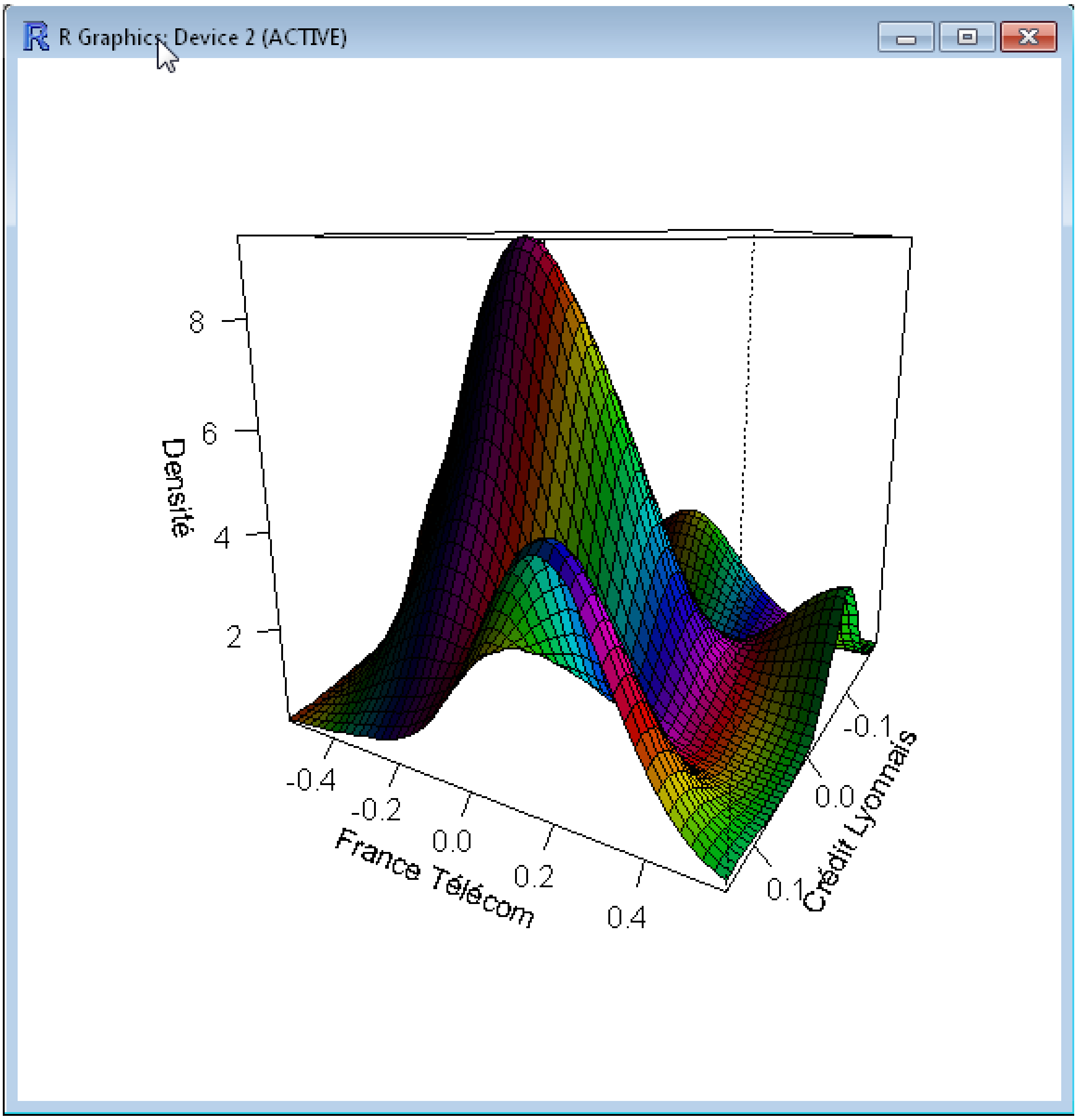

R Graphics: Device 2 (ACTIVE)
1.5
1.0
0.5
0.0
-0.5
z
1.0
0.5
0.0
-0.5
-1.0
y
-1.0
-0.5
0.0
0.5
1.0
x
1.0
0.8
0.6
0.4
0.2
0.0
-0.2

Surface 3D par densité de noyaux 2D

Revenons sur un exemple vu plus haut avec la fonction **kde2d( )** du package MASS et faisons-en une représentation 3D:



Ce qui nous donne:



Surface 3D animée

Nous retrouvons ici encore une fois le package **animation** pour exporter une surface d'une fonction analytique exportée sous forme d'un *.gif animé:

```
R Console
> library(animation)
> x<-seq(-2,2,length=30)
> y<-seq(-1,1,length=30)
> f<-function(x,y){
+ x^2*(4-2.1*x^2+1/3*x^4)+x*y+y^2*(-4+4*y^2)
+ }
> z<-outer(x,y,f)
> saveGIF({
+   for(i in 1:360){
+     persp(x,y,z,theta=i,phi=i,expand=0.4,shade=0.5,expant=0.5,
+     ticktype="detailed",col=rainbow(1000))
+   }
+ }, interval = 0.1, ani.width = 550, ani.height = 550)
Executing:
"C:\Program Files\ImageMagick-6.8.8-Q16\convert.exe" -loop 0 -delay 10
    C:/Users/ISOZVI~1/AppData/Local/Temp/RtmpIvEj2j/Rplot1.png
    C:/Users/ISOZVI~1/AppData/Local/Temp/RtmpIvEj2j/Rplot2.png
```

Après quelques minutes, cela donne une animation de l'image suivante:

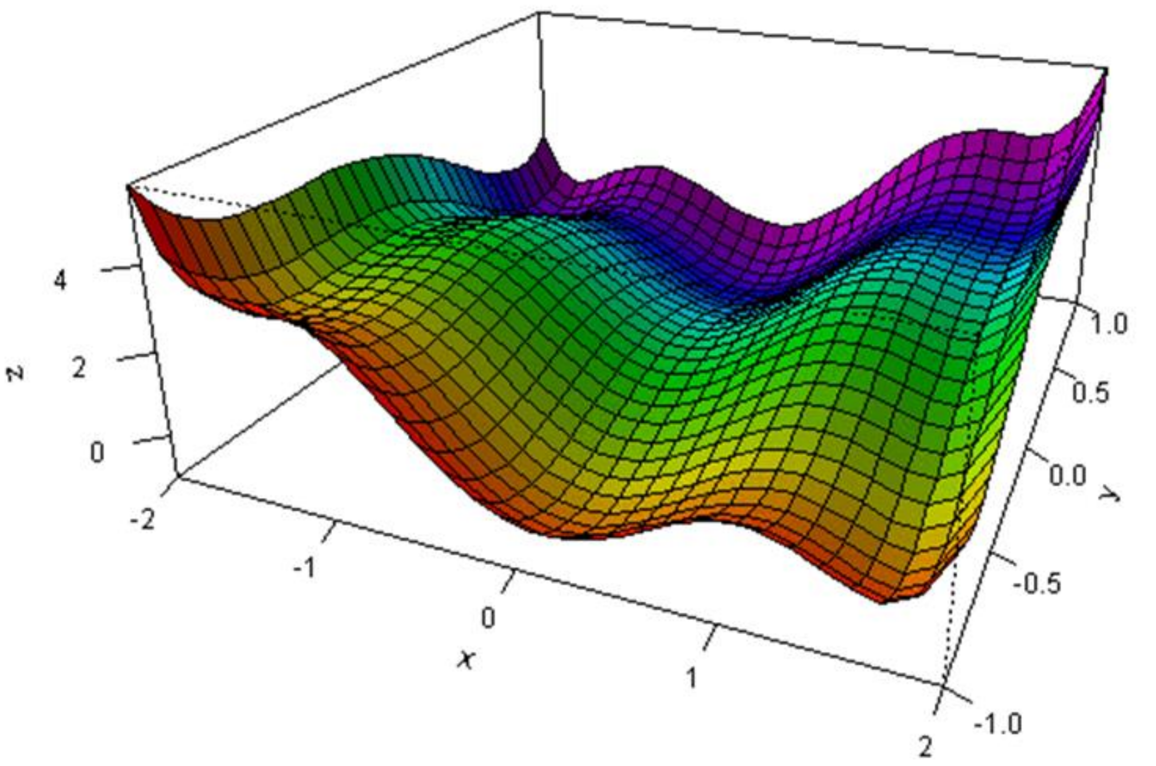


Figure 148 Graph de surface 3D animée

Surface 3D avec points

Il peut arriver que l'on ait des points de mesures d'une surface dont l'expression analytique est connue (ou non). Dès lors, la question est de savoir si nous pouvons mettre des points visibles sur une surface données.

Pour voir cela, nous allons reprendre toujours la fonction sinus cardinal et représenter les points que nous avons calculés.

Nous allons utiliser le package **rgl** avec les commandes **surface3d( )** , **spheres3d( )** , **grid3d( )** et **decorated3d( )** :

R Console

```
> library(rgl)
> #on remet à zéro la fenêtre RGL s'il y en avait d'ouverte avant
> #très utile pour rafraichier un graphe rgl sans avoir à fermer
> #la fenêtre précédente
> clear3d()
> mydata<-read.csv("C:/MesuresSpatiales.csv",header=T,sep=";")
> x<-seq(-10,10,length=30)
> y<-x
> z<-10*sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)
> f<-function(x,y){
+ r<-sqrt(x^2+y^2)
+ 10*sin(r)/r
+ }
> z<-outer(x,y,f)
> x_mesure<-mydata$x;y_mesure<-mydata$y;z_mesure<-mydata$z
> jet.colors<-colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
+ "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
> colorzjet<-jet.colors(100)
>
> surface3d(x,y,z,color=colorzjet[findInterval(z,seq(min(z),max(z),length=100))],alpha=.5)
> spheres3d(x_mesure,y_mesure,z_mesure,col="red",radius=0.5)
> grid3d(side=c("x","y","z"))
> decorate3d()
>
```

Ce qui donne (voir page suivante):

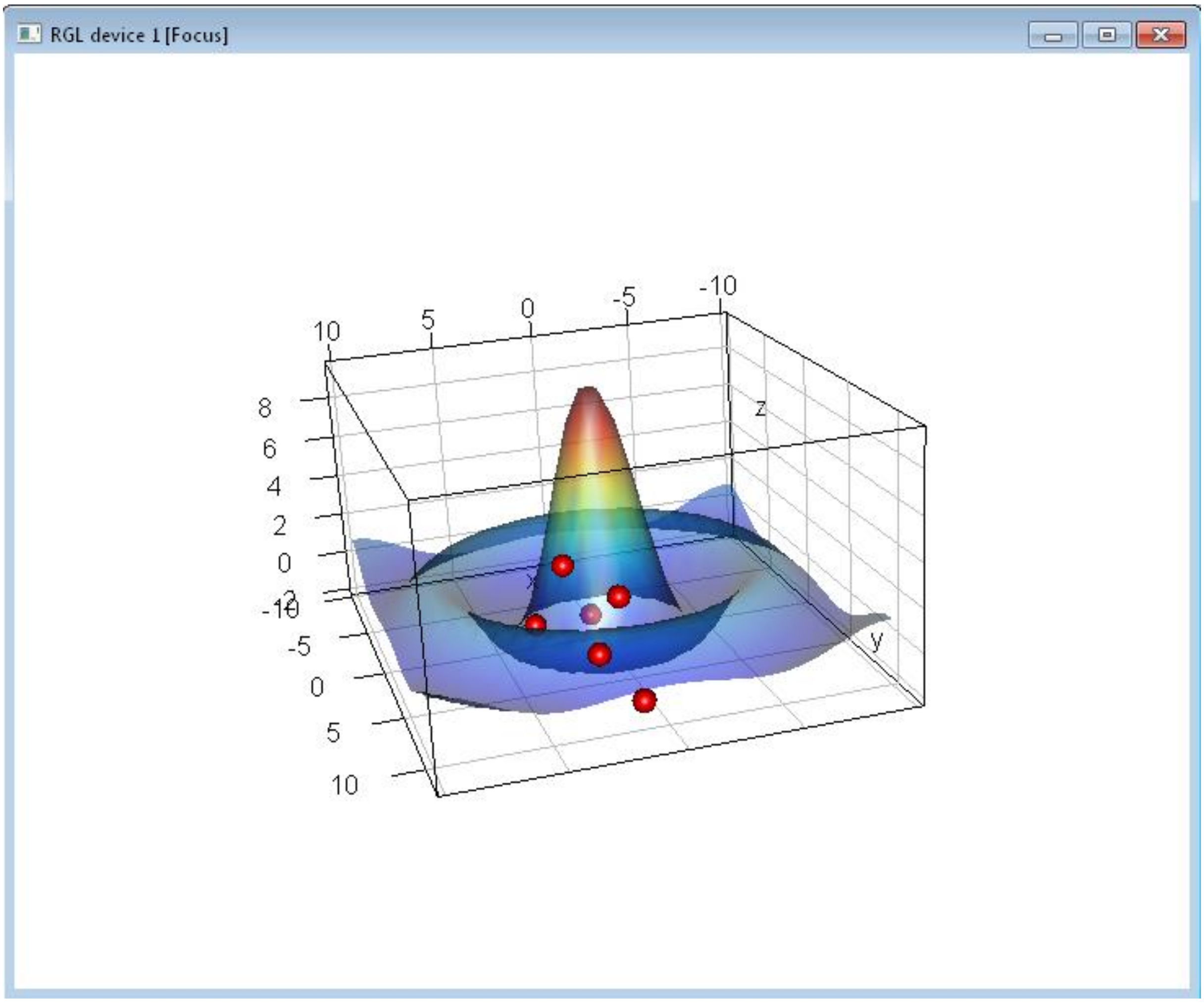


Figure 149 Graph de surface 3D avec points

Surface 3D sphériques (harmoniques sphériques)

Nous poursuivons avec le package **plot3D** et sa fonctions **surf3D( )** mais cette fois pour faire un type de graphique important en cosmologie et physique quantique: les harmoniques sphériques!

Voyons d'abord un cas simple basé sur la fonction **legendre( )** intégrée au package **pracma** utile pour la cosmologie:

```
> library("pracma") #for legendre function
> library("plot3D")
> theta<-seq(from = 0, to = pi, by = pi/320)
> phi<-seq(from = 0, to = 2*pi, by = pi/160)
>
> m<-length(phi); n=length(theta)
> Phi<-matrix(rep(phi,each=n),nrow=n)
> Theta<-matrix(rep(theta,m),nrow=n)
>
> degree <- 6
> order <- 4
> amplitude <- 0.5
> radius <- 5
>
>
> Ymn <- t(legendre(degree,cos(Theta[,1])))
> Ymn = Ymn[,order+1]
> yy = Ymn
>
> for (kk in 2:dim(Theta)[1]){
+   yy<-cbind(yy, Ymn)
+ }
>
> yy<-yy*cos(order*Phi)
>
> order <- max(max(abs(yy)))
> rho = radius + amplitude*yy/order
>
> r = radius*sin(Theta)     # convert to Cartesian coordinates
> x = r*cos(Phi)
> y = r*sin(Phi)
> z = radius*cos(Theta)
>
>
> surf3D(x, y, z, colvar=rho, colkey = FALSE, box = FALSE)
>
```

Ce qui donne:

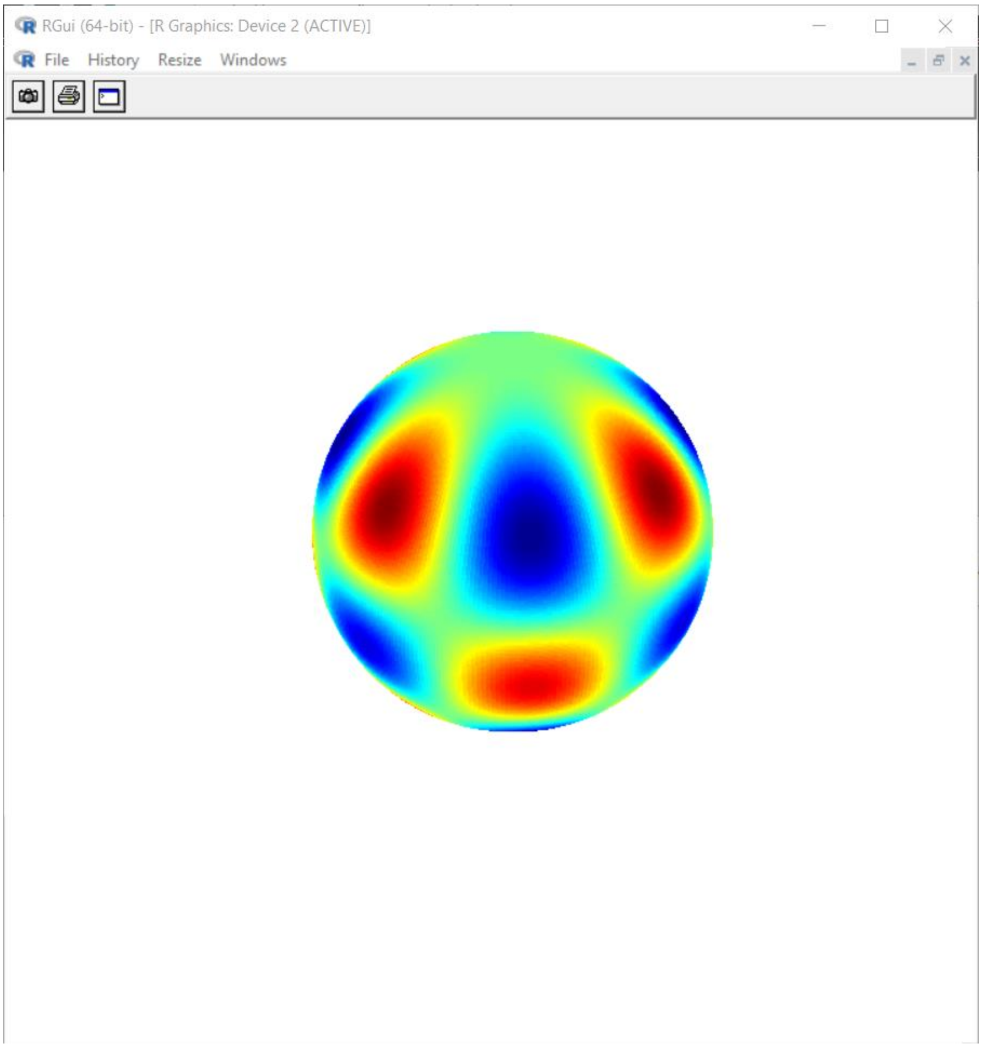
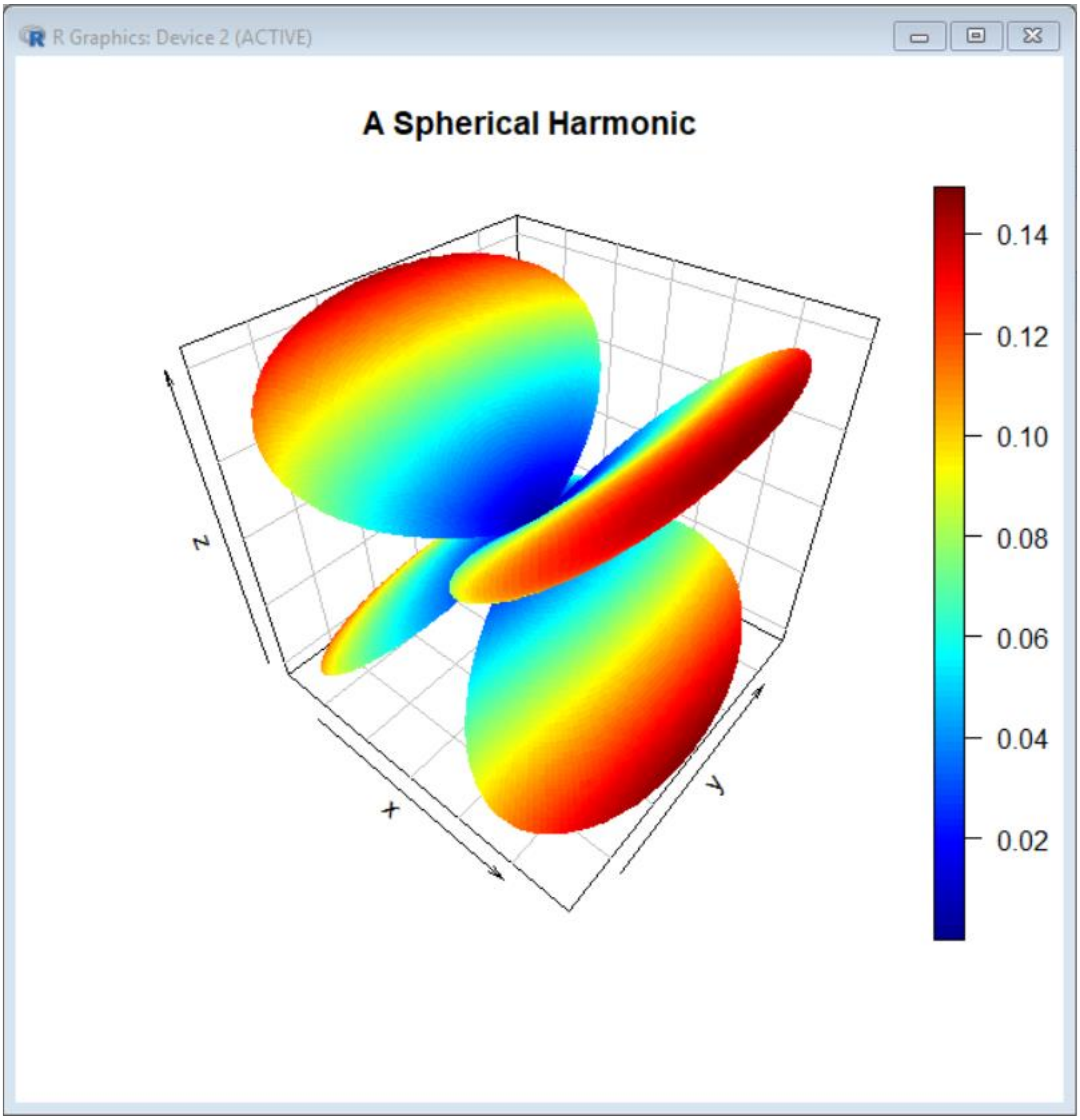


Figure 150 Graph harmoniques sphériques

et un autre exemple utile pour les orbitales atomiques de l'atome hydrogénoïde (rotateur rigide):

```
> library("plot3D")
> theta<-seq(from = 0, to = pi, by = pi/320)
> phi<-seq(from = 0, to = 2*pi, by = pi/160)
>
> m<-length(phi); n=length(theta)
> Phi<-matrix(rep(phi,each=n),nrow=n)
> Theta<-matrix(rep(theta,m),nrow=n)
>
>
> r<-(Re(-1/2*sqrt(15/(2*pi))*exp(-1i*Phi)*sin(theta)*cos(Theta)))^2
> x<-r*cos(Phi)
> y<-r*sin(Phi)
> z<-r*cos(Theta)
>
>
> surf3D(x,y,z,colvar=r,colkey=TRUE,box=TRUE,bty="b2",main="A Spherical Harmonic")
> |
```

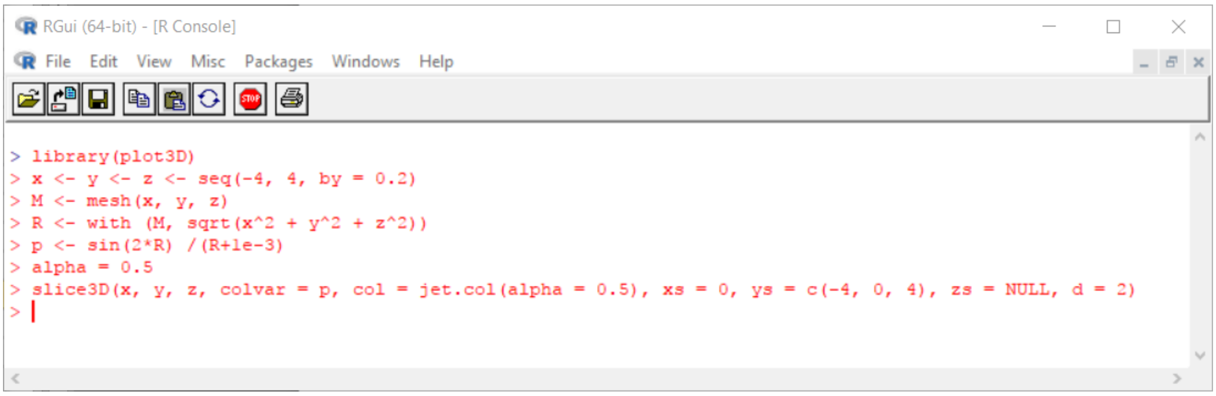
Ce qui donne:


R Graphics: Device 2 (ACTIVE)
A Spherical Harmonic
0.14
0.12
0.10
0.08
0.06
0.04
0.02
z
x
y

Graphiques 4D

Lorsque nous avons une série de capteurs qui mesurent des données dans l'espace, nous avons donc déjà 3 coordonnées pour placer la magnitude de la variable d'intérêt. Mais ceci permet seulement de la placer et non de représenter la magnitude. Nous parlons alors dans ce cas de graphique 4D!

Voici un exemple d'un graphique 4D avec des coupes ("slices" en anglais) utilisant le package **plot3D** et la fonction **slice3D( )** :



Ce qui donne:

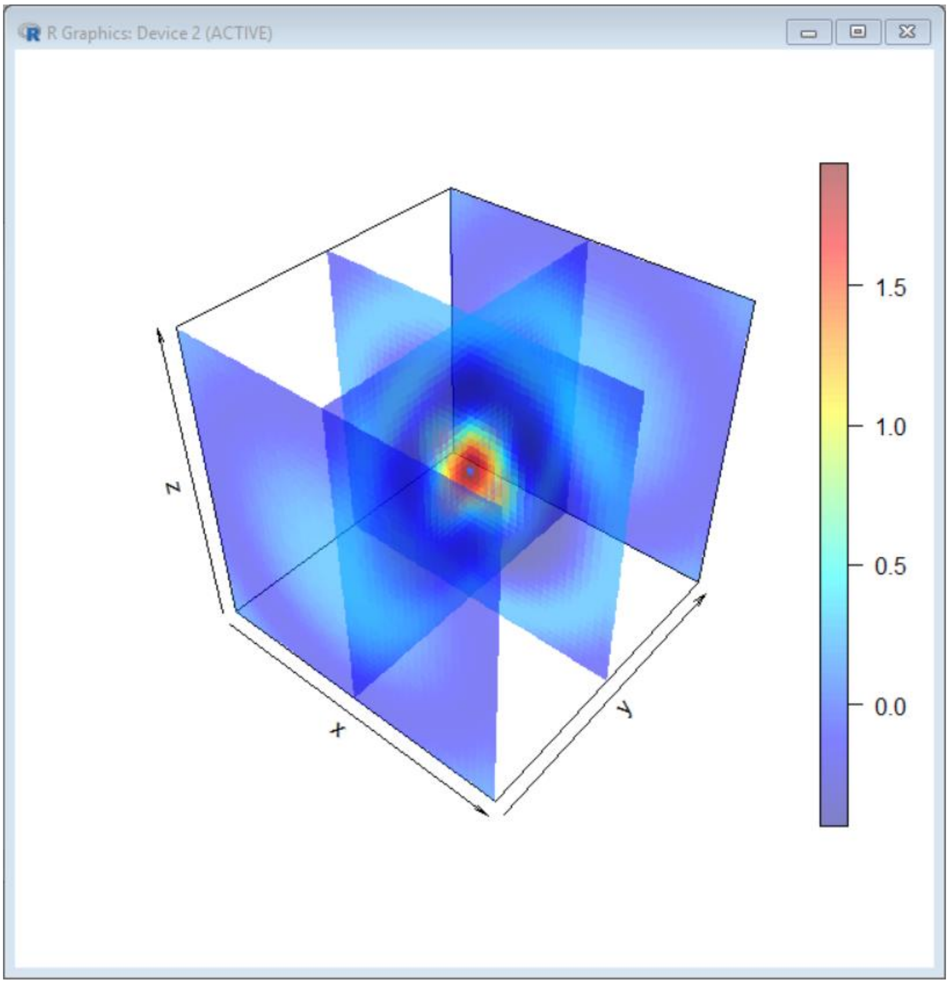


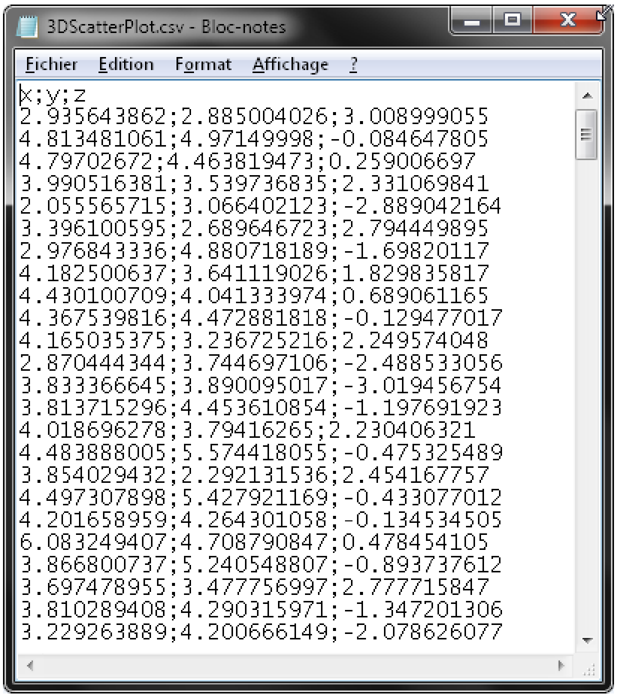
Figure 151 Graph de type 4D avec slices (coupes)

Exemples 147.: Diagramme à points 3D

R 3.0.2

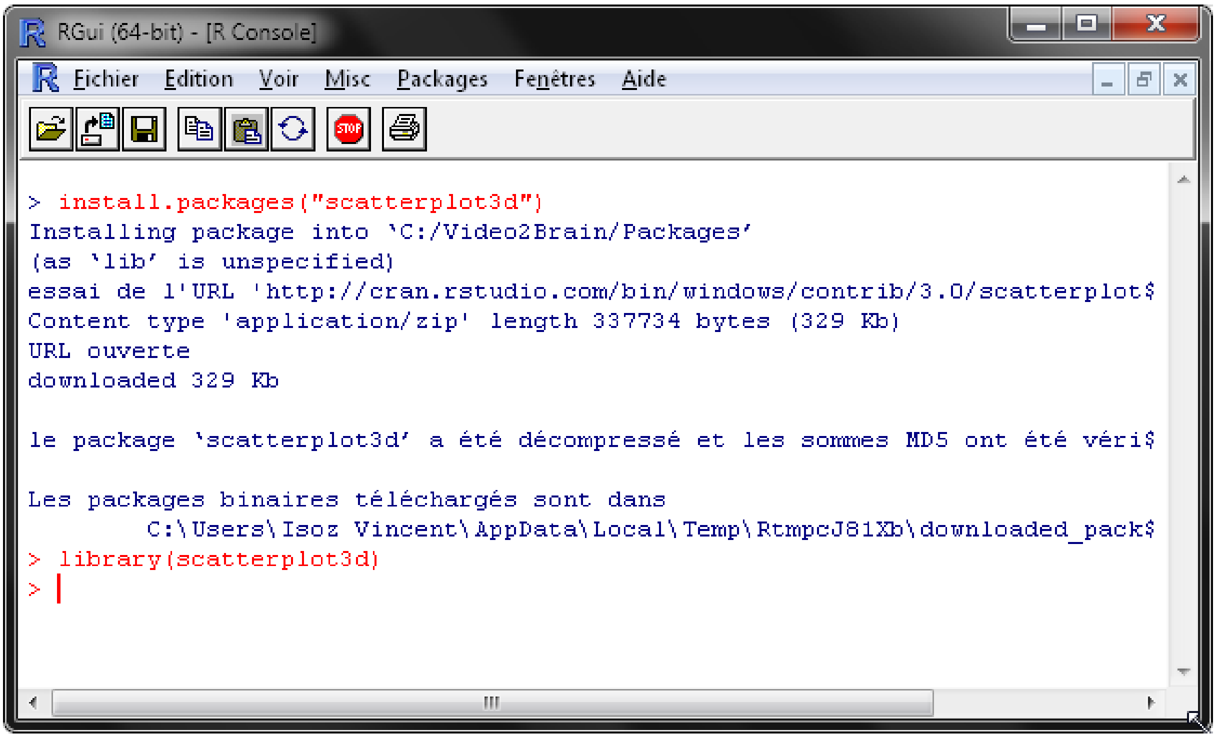
Plot 3D de points simple

Le but ici est de voir que nous pouvons reproduire le graphique à points 3D que nous avions fait avec Microsoft Excel et Minitab. Nous partons donc du fichier suivant:



```
x;y;z
2.935643862;2.885004026;3.008999055
4.813481061;4.97149998;-0.084647805
4.79702672;4.463819473;0.259006697
3.990516381;3.539736835;2.331069841
2.055565715;3.066402123;-2.889042164
3.396100595;2.689646723;2.794449895
2.976843336;4.880718189;-1.69820117
4.182500637;3.641119026;1.829835817
4.430100709;4.041333974;0.689061165
4.367539816;4.472881818;-0.129477017
4.165035375;3.236725216;2.249574048
2.870444344;3.744697106;-2.488533056
3.833366645;3.890095017;-3.019456754
3.813715296;4.453610854;-1.197691923
4.018696278;3.79416265;2.230406321
4.483888005;5.574418055;-0.475325489
3.854029432;2.292131536;2.454167757
4.497307898;5.427921169;-0.433077012
4.201658959;4.264301058;-0.134534505
6.083249407;4.708790847;0.478454105
3.866800737;5.240548807;-0.893737612
3.697478955;3.477756997;2.777715847
3.810289408;4.290315971;-1.347201306
3.229263889;4.200666149;-2.078626077
```

Nous installons le package **scatterplot3d**:



```
> install.packages("scatterplot3d")
Installing package into 'C:/Video2Brain/Packages'
(as 'lib' is unspecified)
essai de l'URL 'http://cran.rstudio.com/bin/windows/contrib/3.0/scatterplot$
Content type 'application/zip' length 337734 bytes (329 Kb)
URL ouverte
downloaded 329 Kb

le package 'scatterplot3d' a été décompressé et les sommes MD5 ont été véri$

Les packages binaires téléchargés sont dans
        C:\Users\Isoz Vincent\AppData\Local\Temp\RtmpcJ81Xb\downloaded_pack$
> library(scatterplot3d)
>
```

et nous y allons en utilisant la commande **scatterplot3d( )** du package:

```
R Console
> setwd("C:/")
> mydata<-read.csv("3DScatterPlot.csv",header=T,sep=";")
> dim(mydata)
[1] 240   3
> names(mydata)
[1] "x" "y" "z"
> scatterplot3d(mydata$x,mydata$y,mydata$z,angle=20,col.axis="blue",
+ col.grid="lightblue",main="3D Scatter Plot",pch=21,box=T,cex.symbols=2)
> 
```

ce qui donne:

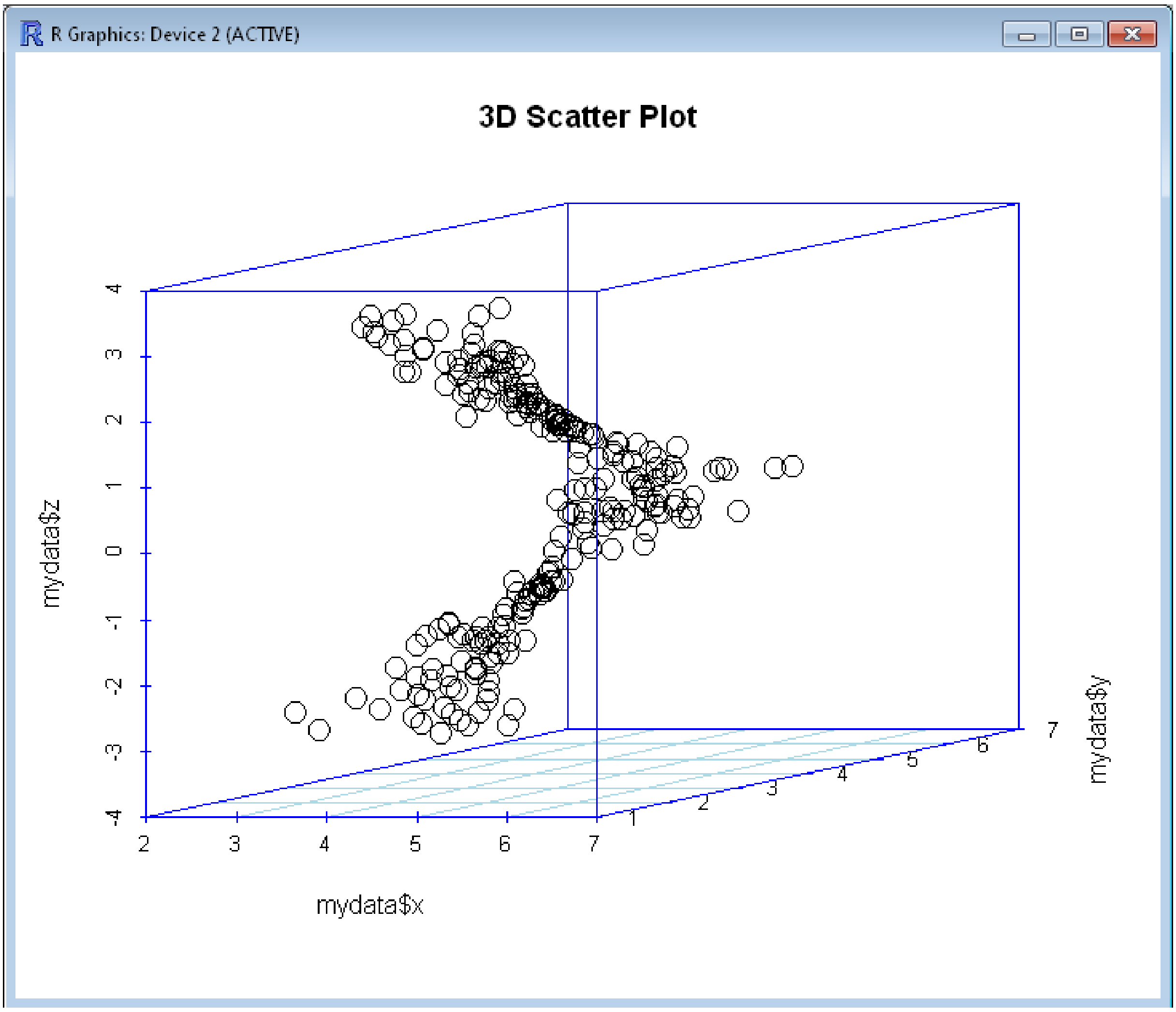
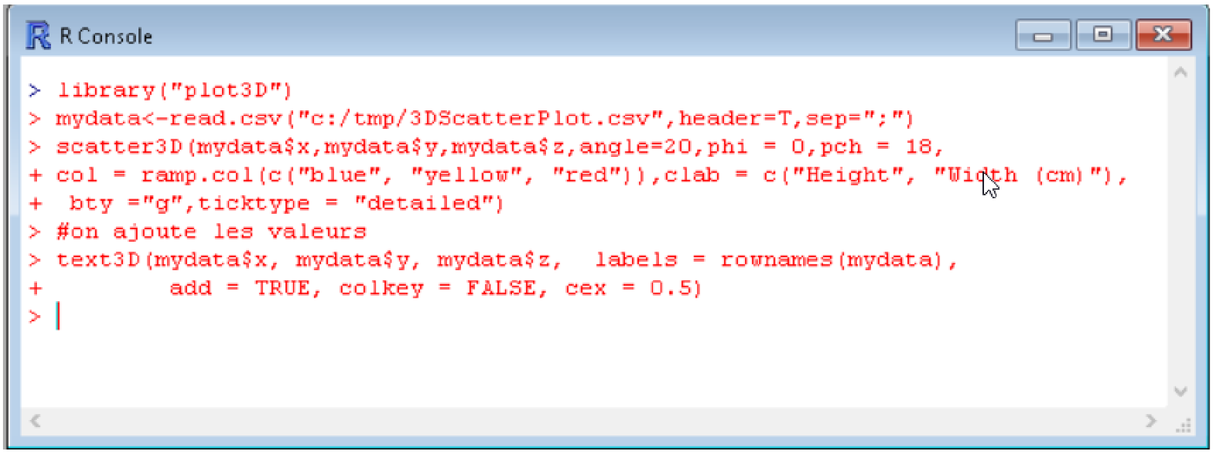


Figure 152 Graph à points 3D

Version un peu améliorée (merci à Alboukadel Kassambara) avec la fonction **scatter3D( )** du package **plot3D**:



Ce qui donne:

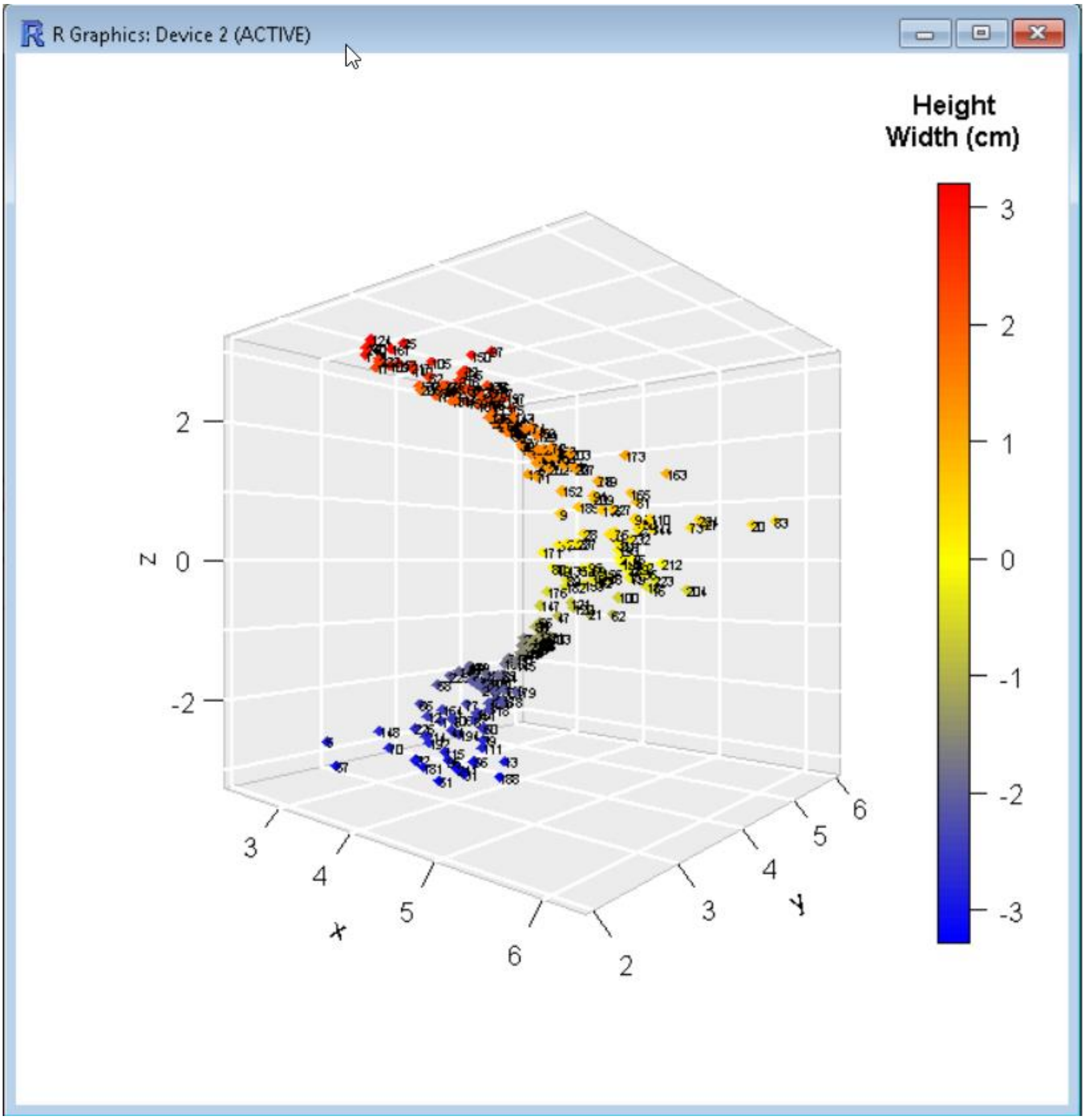
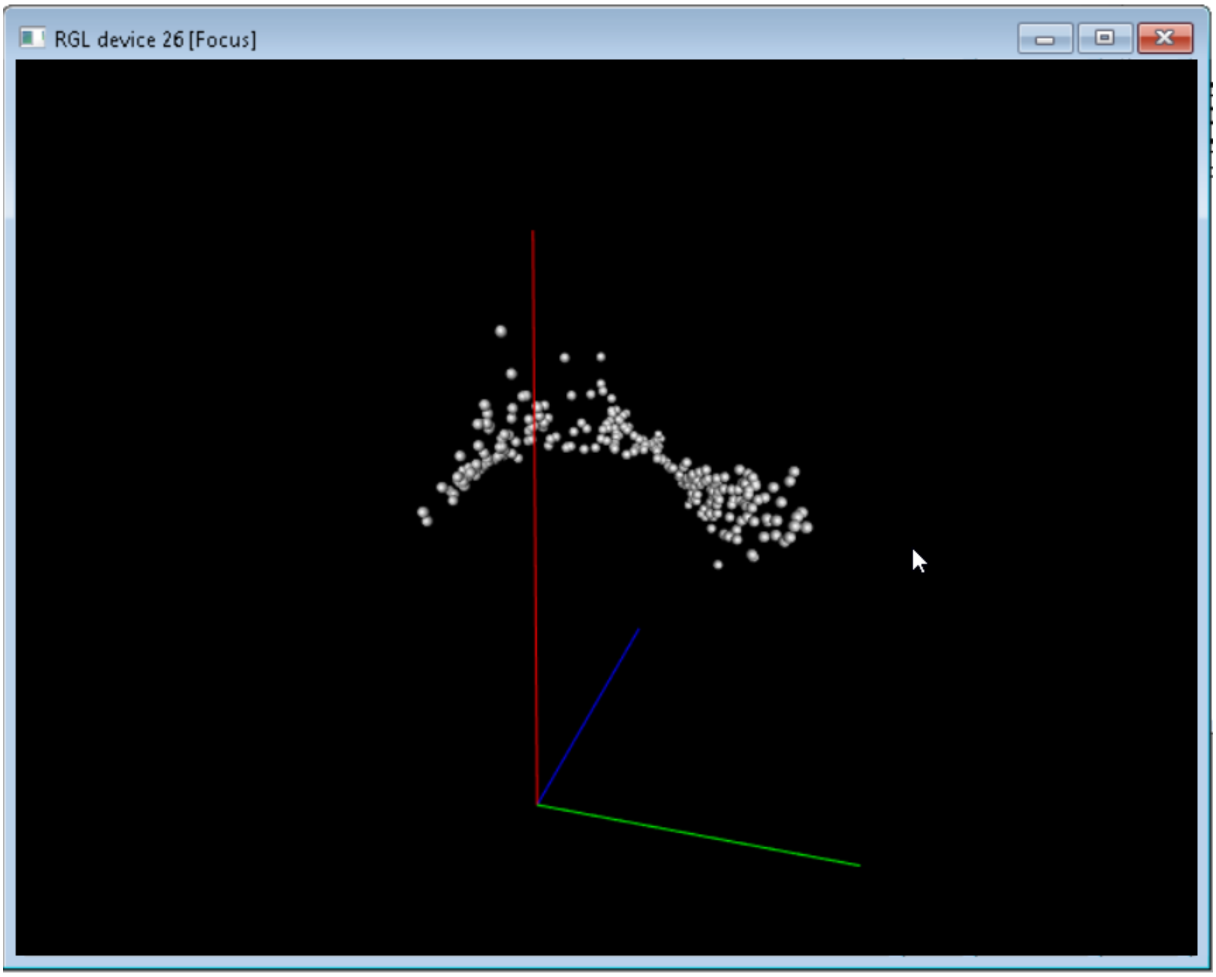


Figure 153 Graph à points 3D avec légendes

Avec le package **rgl** et la fonction **rgl.spheres( )** nous pouvons faire plus simple typiquement lorsque l'on travaille dans le domaine de l'astrophysique ou astronomie:

```
R Console
> library(rgl)
> rgl.open()
> rgl.bg(color = "black")
> rgl.spheres(mydata$x,mydata$y,mydata$z, color ="lightgray", r = 0.1)
> rgl.lines(c(0, max(mydata$x)), c(0, 0), c(0, 0), color = "blue")
> rgl.lines(c(0, 0), c(0,max(mydata$y)), c(0, 0), color = "red")
> rgl.lines(c(0, 0), c(0, 0), c(0,max(mydata$z)), color = "green")
> 
```

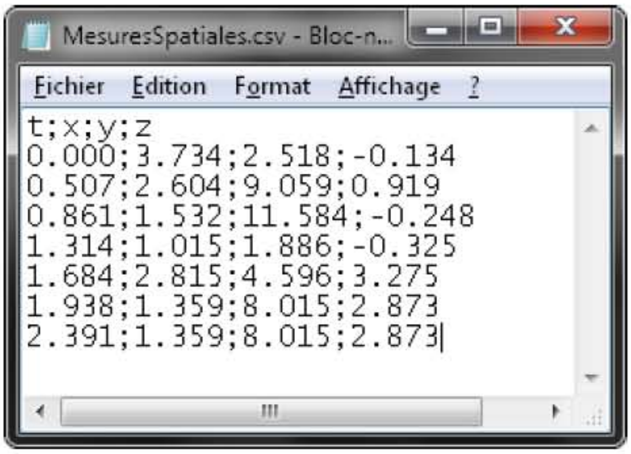
Ce qui donne:



Plot 3D de points mesurés dans le temps (splines 3D)

Lorsque l'on travaille dans l'aéronautique, ou l'aérospatiale ou la mécanique on a souvent besoin de représenter l'évolution d'un point dans l'espace et dans le temps (typiquement aussi la position d'un drone civil en fonction du temps).

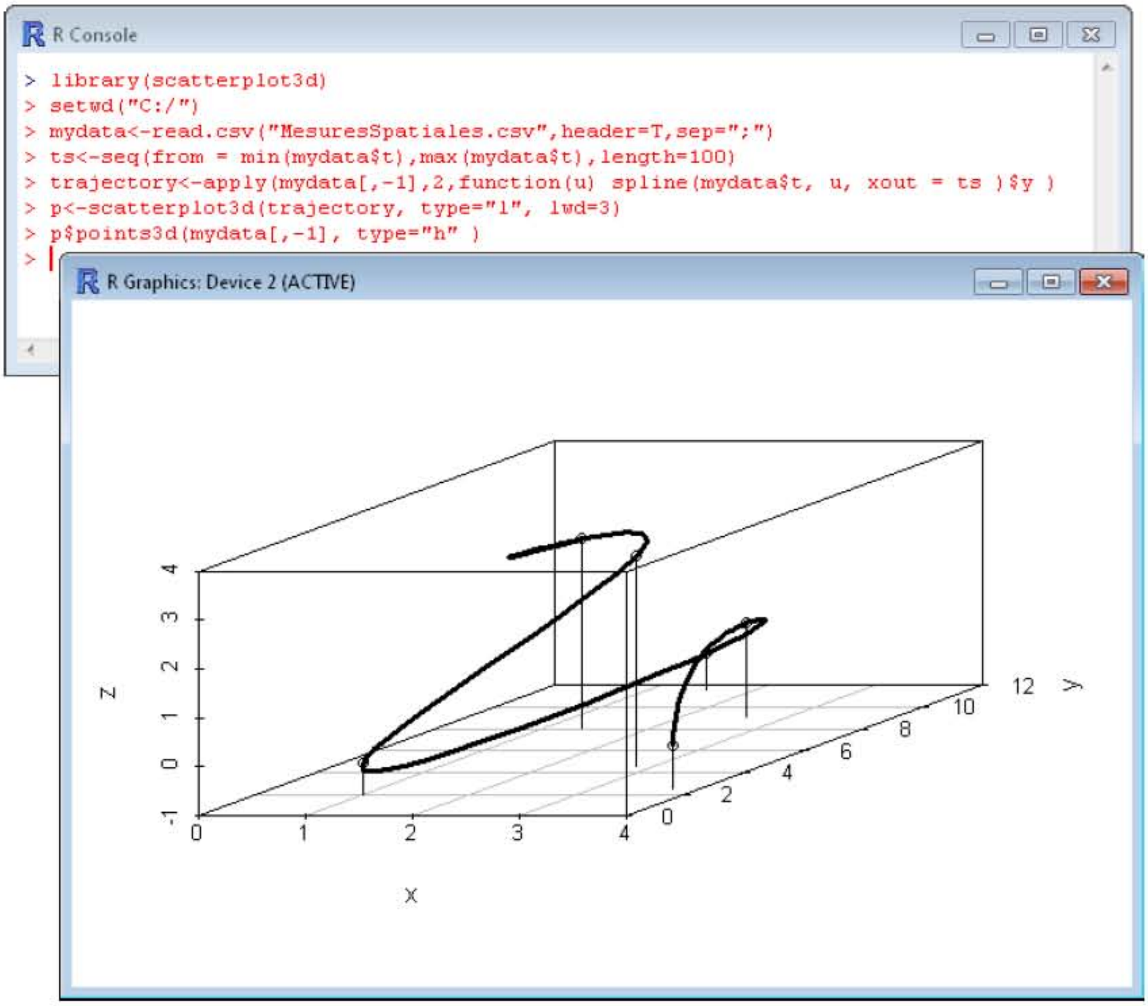
Nous partons des points mesurés suivants:


MesuresSpatiales.csv - Bloc-n...

Fichier Edition Format Affichage ?

```
t;x;y;z
0.000;3.734;2.518;-0.134
0.507;2.604;9.059;0.919
0.861;1.532;11.584;-0.248
1.314;1.015;1.886;-0.325
1.684;2.815;4.596;3.275
1.938;1.359;8.015;2.873
2.391;1.359;8.015;2.873
```

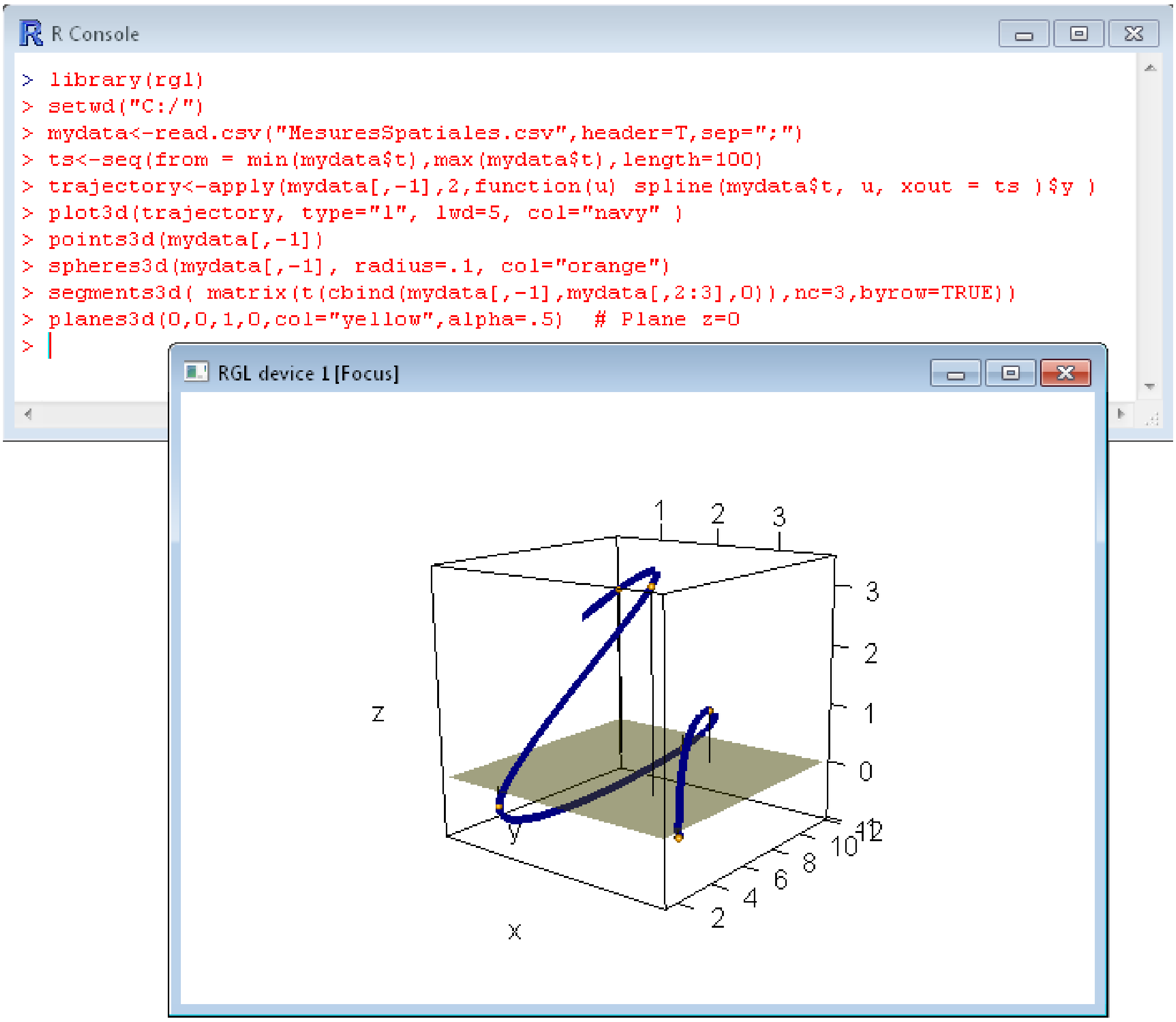
Nous avons alors:


R Console

```
> library(scatterplot3d)
> setwd("C:/")
> mydata<-read.csv("MesuresSpatiales.csv",header=T,sep=";")
> ts<-seq(from = min(mydata$t),max(mydata$t),length=100)
> trajectory<-apply(mydata[,-1],2,function(u) spline(mydata$t, u, xout = ts )$y )
> p<-scatterplot3d(trajectory, type="l", lwd=3)
> p$points3d(mydata[,-1], type="h" )
>
```


Figure 154 Graph 3D à points avec splines de trajectoires

Ou encore pour rendre la rotation du diagramme interactive, nous pouvons utiliser le package **rgl** avec les commandes **plot3d( )** , **points3d( )** , **spheres3d( )** et **segments3d( )** et enfin **planes3d( )** :



Plot 3D de vecteurs

Nous allons voir ici comment représenter des vecteurs dans l'espace en utilisant le package **plot3D** et la fonction **arrows3D( )** :

```
> library("plot3D")
> #points de départs des 4 vecteurs
> x0 <- c(0, 0, 0, 0)
> y0 <- c(0, 0, 0, 0)
> z0 <- c(0, 0, 0, 0)
> #points d'arrivées des 4 vecteus
> x1 <- c(0.89, -0.46, 0.99, 0.96)
> y1 <- c(0.36,  0.88, 0.02, 0.06)
> z1 <- c(-0.28, 0.09, 0.05, 0.24)
> cols <- c("#1B9E77", "#D95F02", "#7570B3", "#E7298A")
>
> arrows3D(x0, y0, z0, x1, y1, z1, col = cols,
+          lwd = 2, d = 1, main = "Arrows 3D", bty ="g",
+          ticktype = "detailed")
> # Ajouter un point en (0,0,0)
> points3D(x0, y0, z0, add = TRUE, col="darkred",
+           colkey = FALSE, pch = 19, cex = 1)
> # Ajout de légendes aux vecteurs
> text3D(x1, y1, z1, c("Sepal.L", "Sepal.W", "Petal.L", "Petal.W"),
+         colvar = x1^2, col = cols, add=TRUE, colkey = FALSE)
>
```

Ce qui donne:

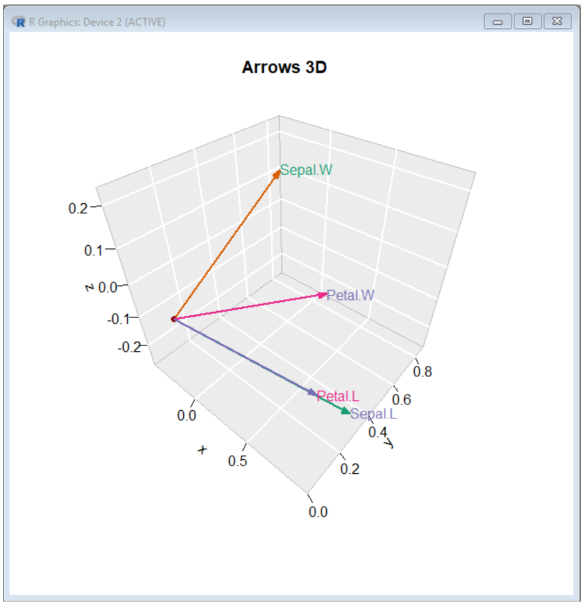


Figure 155 Graph 3D avec vecteurs

En ajoutant les deux commandes suivantes, on peut rendre le graphique interactif!:

library("plot3Drgl")
plotrgl()

Exemple 148.: Diagramme à bulles

R 3.0.2

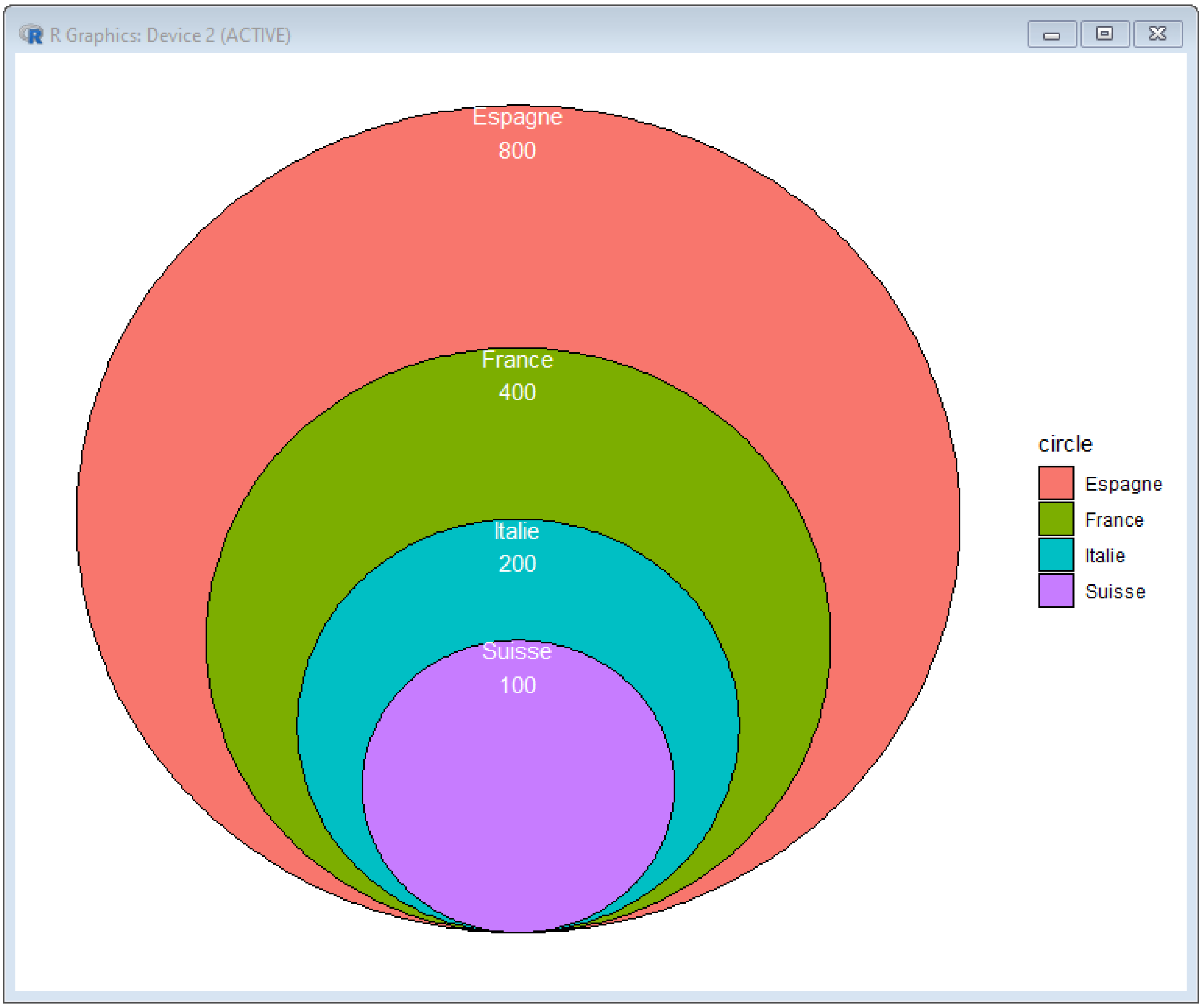
Diagrammes à bulles imbriquées

Je n'ai pas trouvé de package permettant de faire ce type de graphiques. J'ai donc du le coder complétement de A à Z.

Donc voici comment faire un tel graphique:

```
> library("ggforce")
> #avoir la première colonne dans l'ordre alphabétique croissant!
> (d <- read.table(text = "circle;x
+ Espagne;800
+ France;400
+ Italie;200
+ Suisse;100", header = TRUE, sep = ";"))
   circle   x
1 Espagne 800
2  France 400
3  Italie 200
4  Suisse 100
>
> #convert area to R
> d$r <- sqrt(d$x / pi)
> d$x0 <- max(d$r) / 2
> d$y0 <- d$r
> d$texty<-2*d$r-1
> d$label<-paste(d$circle,d$x,sep="\n")
>
> d
   circle   x         r       x0        y0    texty        label
1 Espagne 800 15.957691 7.978846 15.957691 30.91538 Espagne\n800
2  France 400 11.283792 7.978846 11.283792 21.56758  France\n400
3  Italie 200  7.978846 7.978846  7.978846 14.95769  Italie\n200
4  Suisse 100  5.641896 7.978846  5.641896 10.28379  Suisse\n100
>
>
> ggplot(data=d, aes(x0 = x0, y0 = y0, r = r, fill = circle)) +
+   geom_circle() +
+    geom_text(data=d,aes(x=x0,y=texty,label=label),color="white")+
+   theme(axis.text=element_blank(),
+         axis.title.x = element_blank(),
+         axis.title.y = element_blank(),
+   axis.ticks=element_blank(),
+         panel.grid=element_blank(),panel.background = element_rect(fill = 'white'))
>
```

Ce qui donne:



Diagrammes à bulles 2D

Là aussi rien de bien technique… Le but étant de voir si nous arrivons à faire un diagramme à bulle comme dans n'importe quel tableur (évidemment ce sera encore une fois beaucoup moins esthétique que le résultat obtenu dans le cours Microsoft Excel).

En reprenant les données du cours Microsoft Excel directement dans la console, nous écrivons:

```
> plot(x,y,type="n",asp=1)
> y<-c(0.2,0.15,-0.05,0.18,0.25,0,3)
> x<-c(0,0.15,0.05,0.2,0.45,0.4,0.25)
> size<-c(10000,20000,50000,5000,6000,80000,45000)
> size<-size/max(size)
> radius<-sqrt(size/pi)
> name<-c("Produit A", "Produit B","Produit C","Produit D","Produit E","Produit F","Produit G")
> symbols(x,y,circles=radius,fg="white",bg=col,ylim=c(-0.1,0.3),xlim=c(-0.1,0.6)
+ ,xlab="Parts de marché",ylab="Croissance",main="Diagramme à bulle")
> grid()
> text(x,y,name,cex=1.5,font=4)
> 
```

Ce qui donne:

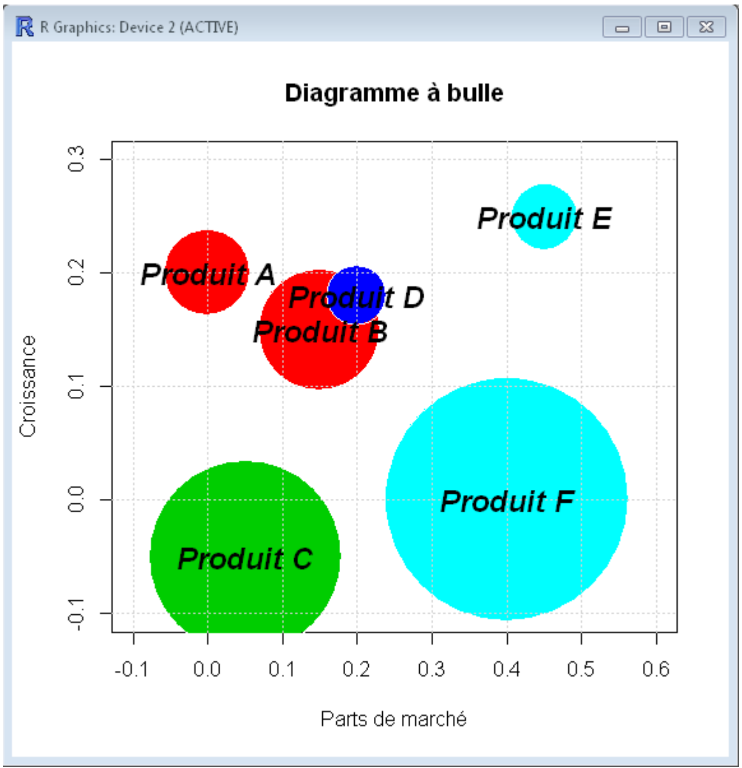


Figure 156 Graph à bulles 2D

ou avec le package **ggplot2** avec les mêmes données:

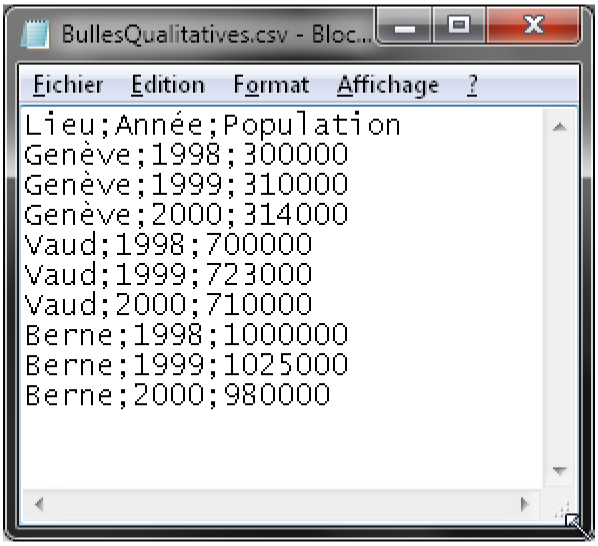
```
R Console
> library(ggplot2)
> Croissance<-c(0.2,0.15,-0.05,0.18,0.25,0,0.3)
> PartsMarche<-c(0,0.15,0.05,0.2,0.45,0.4,0.25)
> Volume<-c(10000,20000,50000,5000,6000,80000,45000)
> Names<-c("Produit A","Produit B","Produit C","Produit D","Produit E",
+ "Produit F","Produit G")
>
> mydata<-as.data.frame.matrix(cbind(Names,Croissance,PartsMarche,Volume))
> mydata$Croissance<-as.numeric(as.character(mydata$Croissance))
> mydata$PartsMarche<-as.numeric(as.character(mydata$PartsMarche))
> mydata$Volume<-as.numeric(as.character(mydata$Volume))
>
> ggplot(mydata, aes(x=PartsMarche,y=Croissance,size=Volume,label=Names,color=Names))+
+ geom_point(shape=21,coulour="black",fill="cornsilk")+
+ geom_text(size=3)+ylab("Croissance")+xlab("Parts de marché")+
+ scale_size_area(max_size=20)+scale_x_continuous(limits=c(-0.05,0.5))
> 
```

Ce qui nous donne:



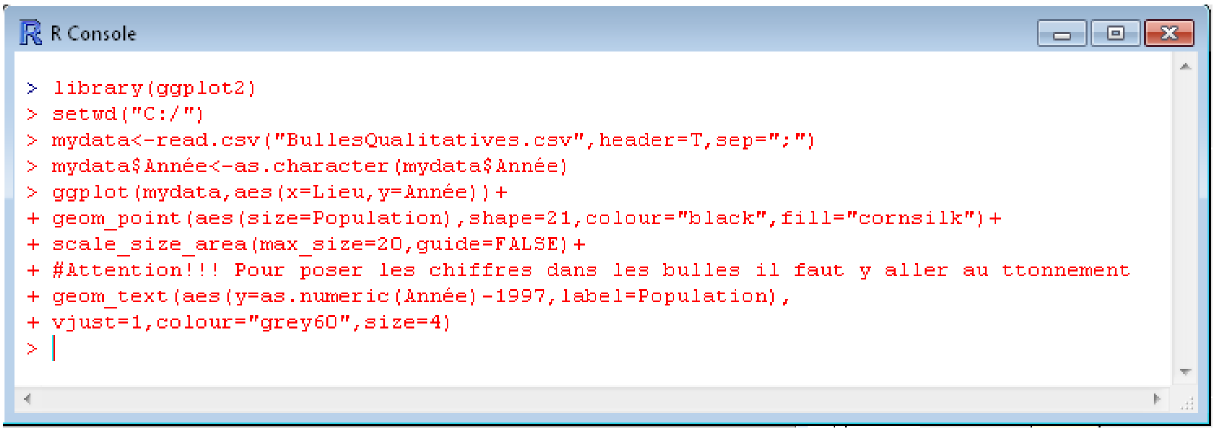
Figure 157 Graph à bulles 2D (ggplot)

Par contre avec R il est beaucoup plus facile et rapide de faire le graphique à bulle suivant que nous avions aussi construit dans Microsoft Excel avec les données suivantes:



```
Lieu;Année;Population
Genève;1998;300000
Genève;1999;310000
Genève;2000;314000
Vaud;1998;700000
Vaud;1999;723000
Vaud;2000;710000
Berne;1998;1000000
Berne;1999;1025000
Berne;2000;980000
```

Donc maintenant en utilisant toujours le package **ggplot2**:



```
> library(ggplot2)
> setwd("C:/")
> mydata<-read.csv("BullesQualitatives.csv",header=T,sep=";")
> mydata$Année<-as.character(mydata$Année)
> ggplot(mydata,aes(x=Lieu,y=Année))+
+ geom_point(aes(size=Population),shape=21,colour="black",fill="cornsilk")+
+ scale_size_area(max_size=20,guide=FALSE)+
+ #Attention!!! Pour poser les chiffres dans les bulles il faut y aller au ttonnement
+ geom_text(aes(y=as.numeric(Année)-1997,label=Population),
+ vjust=1,colour="grey60",size=4)
>
```

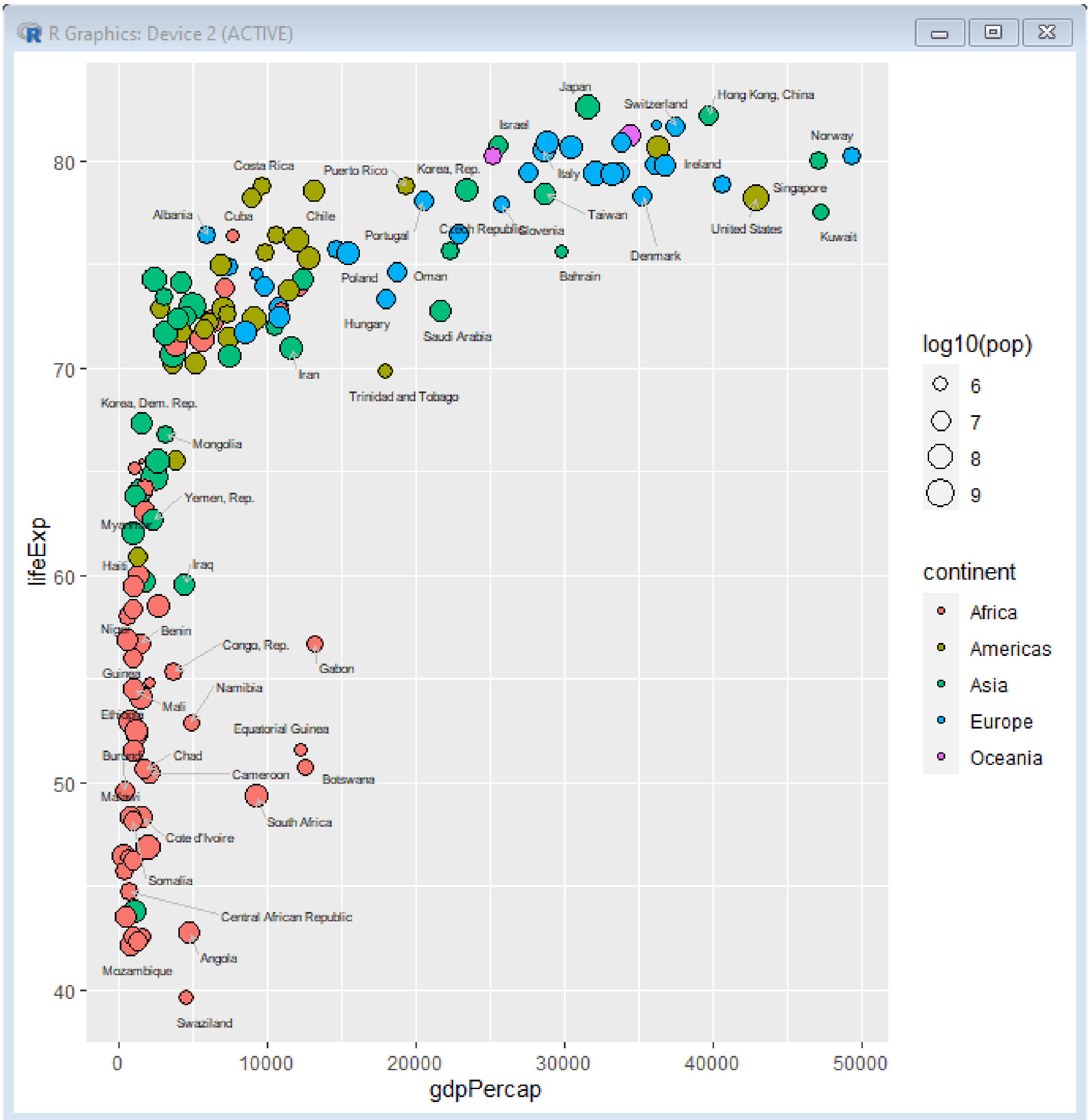
Cela nous donne:



Sinon globalement on peut faire évidemment beaucoup mieux comme l'illustre l'exemple suivant:

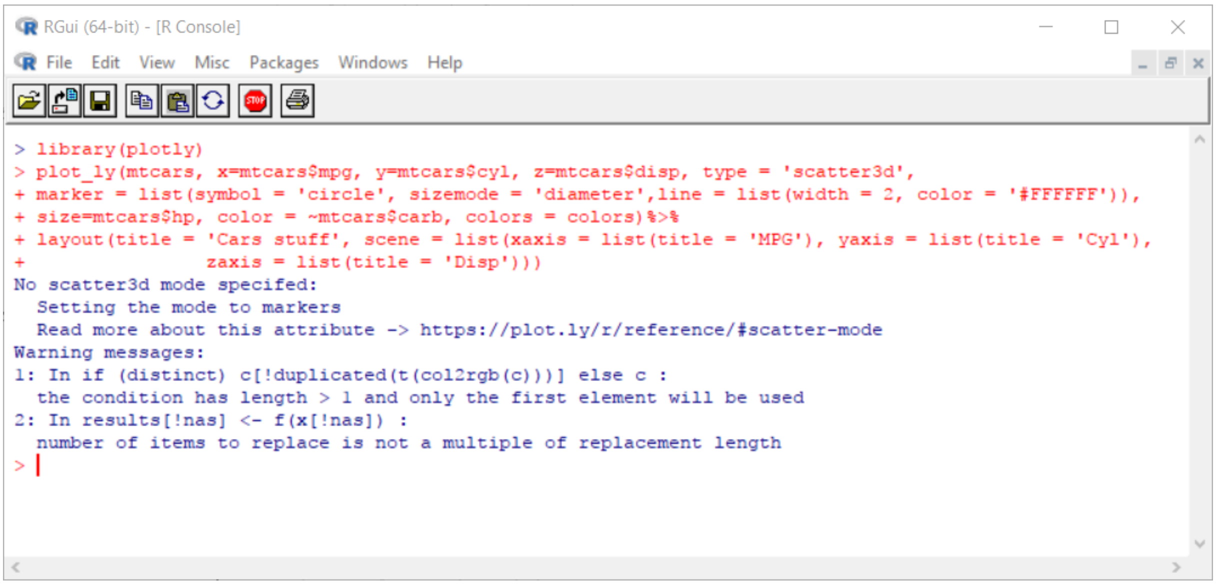
```
> library("tibble")
> library("ggplot2")
> library ("dplyr")
> library ("ggrepel")
> load ("gapdata.RData")
>
> df <-  filter(gapdata,year == 2007)
> colfill <- c('#7F3B08','#A50026',
+                         '#40004B','#276419','#313695')
> ggplot(df) + aes(x = gdpPercap,y=lifeExp,label=country)+
+   geom_point(aes(fill= continent,size=log10(pop)),shape = 21)+
+   geom_text_repel(size=2,colour = "black",
+     arrow = arrow(length = unit(0.01, 'npc')),
+     box.padding = unit(0.5, 'lines'),
+     force = 1, max.iter = 3e3,
+     segment.color = 'gray')
Warning message:
ggrepel: 87 unlabeled data points (too many overlaps). Consider increasing max.overlaps
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
lifeExp
gdpPercap
log10(pop)
continent
Africa
Americas
Asia
Europe
Oceania
Japan
Switzerland
Hong Kong, China
Norway
Israel
Costa Rica
Puerto Rico
Korea, Rep.
Italy
Ireland
Singapore
Albania
Cuba
Chile
Portugal
Czech Republic
Slovenia
Taiwan
United States
Kuwait
Denmark
Poland
Oman
Bahrain
Hungary
Saudi Arabia
Iran
Trinidad and Tobago
Korea, Dem. Rep.
Mongolia
Yemen, Rep.
Iraq
Benin
Congo, Rep.
Gabon
Namibia
Mali
Equatorial Guinea
Chad
Cameroon
Botswana
South Africa
Cote d'Ivoire
Somalia
Central African Republic
Angola
Mozambique
Swaziland

Diagrammes à bulles 3D

Pour obtenir un diagramme à bulle 3D, nous pouvons utiliser le package **plotly** avec la fonction **plot_ly( )** :



Ce qui nous donne:

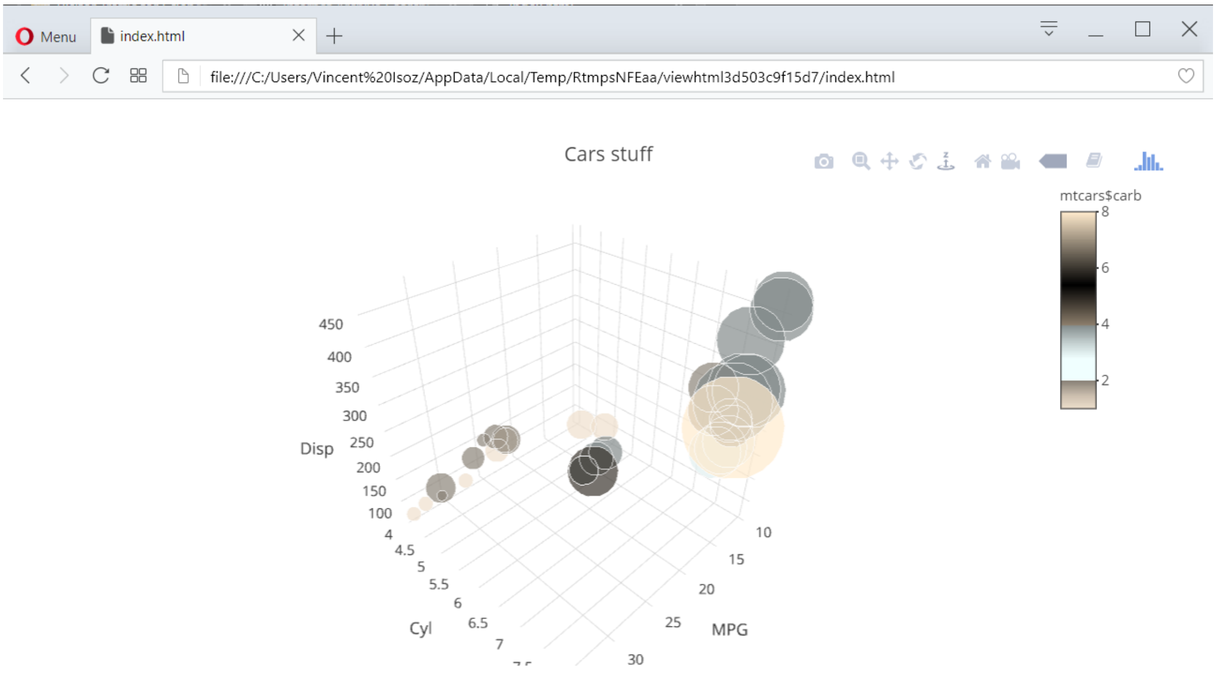


Figure 158 Graph 3D à bulles

Le résultat peut certainement être amélioré mais je ne suis pas un grand fan des graphiques donc n'hésitez pas à partager vos créations!

Diagramme de Rosling (bulles animées)

R 3.2.1

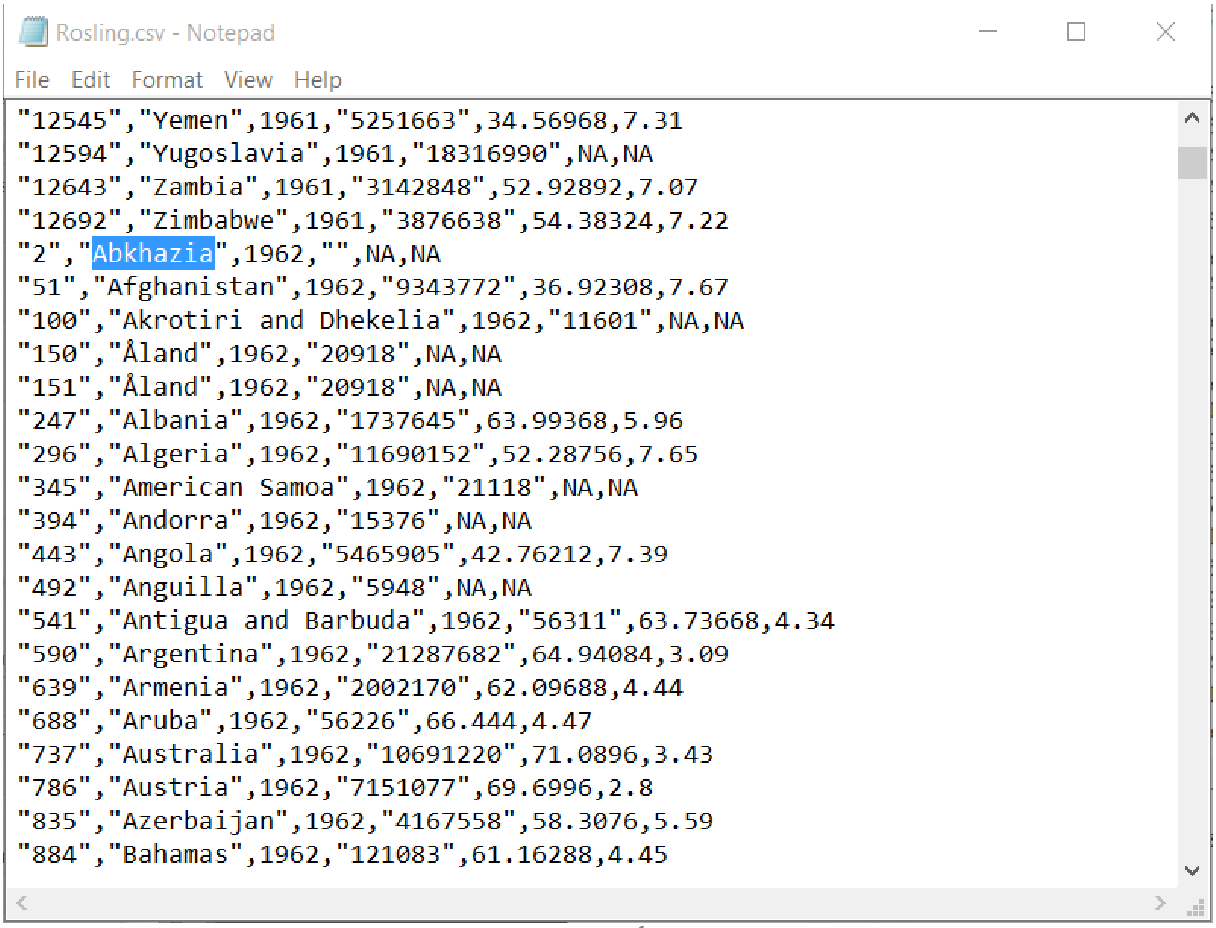
Suite au Buzz effectué par la présentation de Rosling sur Internet je pense qu'il est inutile de préciser de quoi il s'agit… Donc pour voir comment reproduire cela, considérons le jeu de données suivant:

Rosling.csv - Notepad

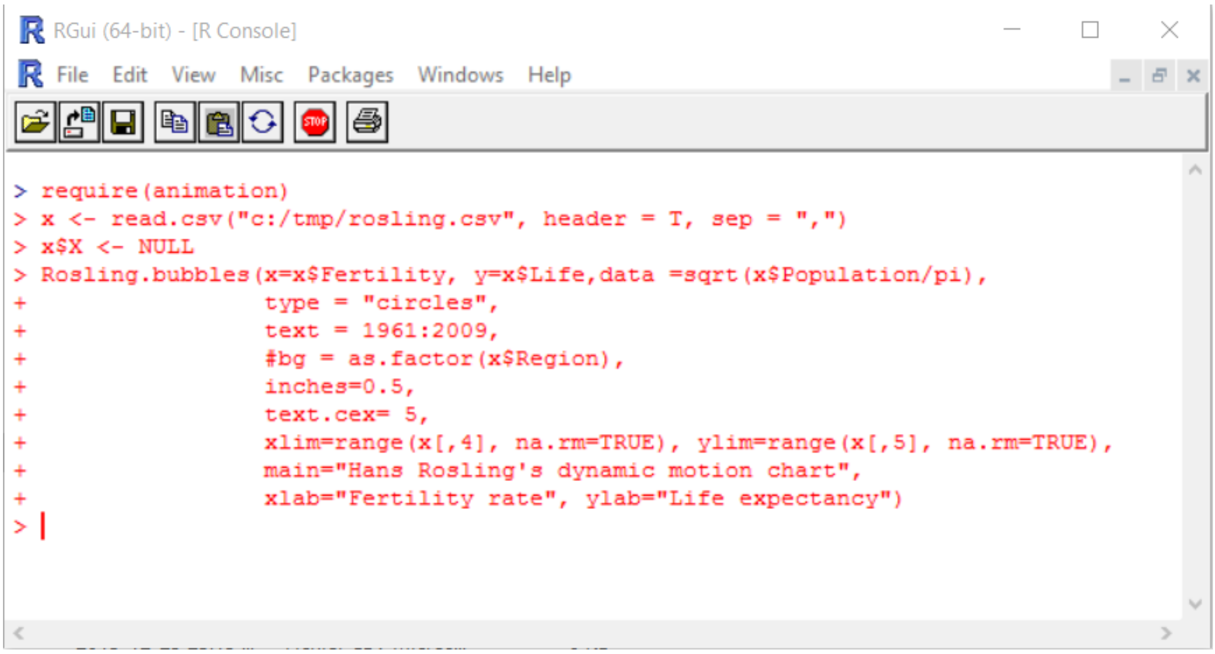
File Edit Format View Help

```
"","country","year","population","lifeexpectancy","birthrate"
"1","Abkhazia",1961,"",NA,NA
"50","Afghanistan",1961,"9164945",36.30334,7.67
"99","Akrotiri and Dhekelia",1961,"11519",NA,NA
"148","Åland",1961,"20949",NA,NA
"149","Åland",1961,"20949",NA,NA
"246","Albania",1961,"1685901",63.10964,6.08
"295","Algeria",1961,"11404859",51.70588,7.65
"344","American Samoa",1961,"20478",NA,NA
"393","Andorra",1961,"14376",NA,NA
"442","Angola",1961,"5367287",42.14676,7.35
"491","Anguilla",1961,"5901",NA,NA
"540","Antigua and Barbuda",1961,"55403",63.29464,4.39
"589","Argentina",1961,"20953079",64.85582,3.1
"638","Armenia",1961,"1934239",61.77374,4.51
"687","Aruba",1961,"55435",66.074,4.66
"736","Australia",1961,"10494911",71.3158,3.55
"785","Austria",1961,"7105654",69.9058,2.79
"834","Azerbaijan",1961,"4030130",57.9743,5.6
"883","Bahamas",1961,"115108",60.88874,4.5
"932","Bahrain",1961,"167924",54.503,7.13
"981","Bangladesh",1961,"49593610",45.2564,6.76
"1030","Barbados",1961,"231674",64.47752,4.31
```

où chaque pays se répète plusieurs fois comme le montre la capture suivante:



Avec la fonction **Rosling.bubbles( )** du package animation nous arrivons à obtenir quelque chose proche du de la présentation de Rosling à part que les couleurs changent à chaque itération et il n'y a pas de légendes sur les bullets donc pour l'instant c'est moins bien que Microsoft Excel:



Ce qui donne une animation année par année:

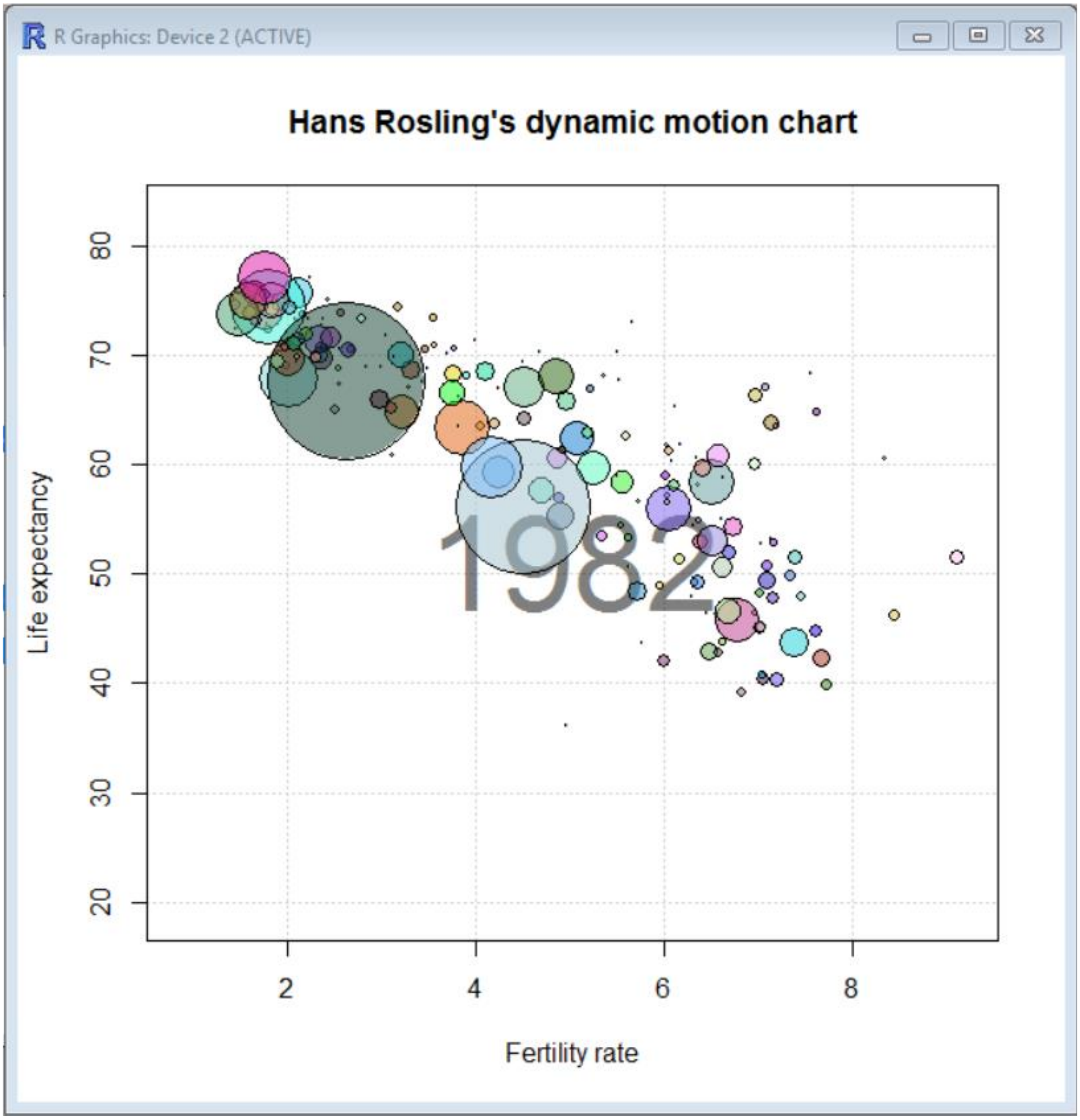


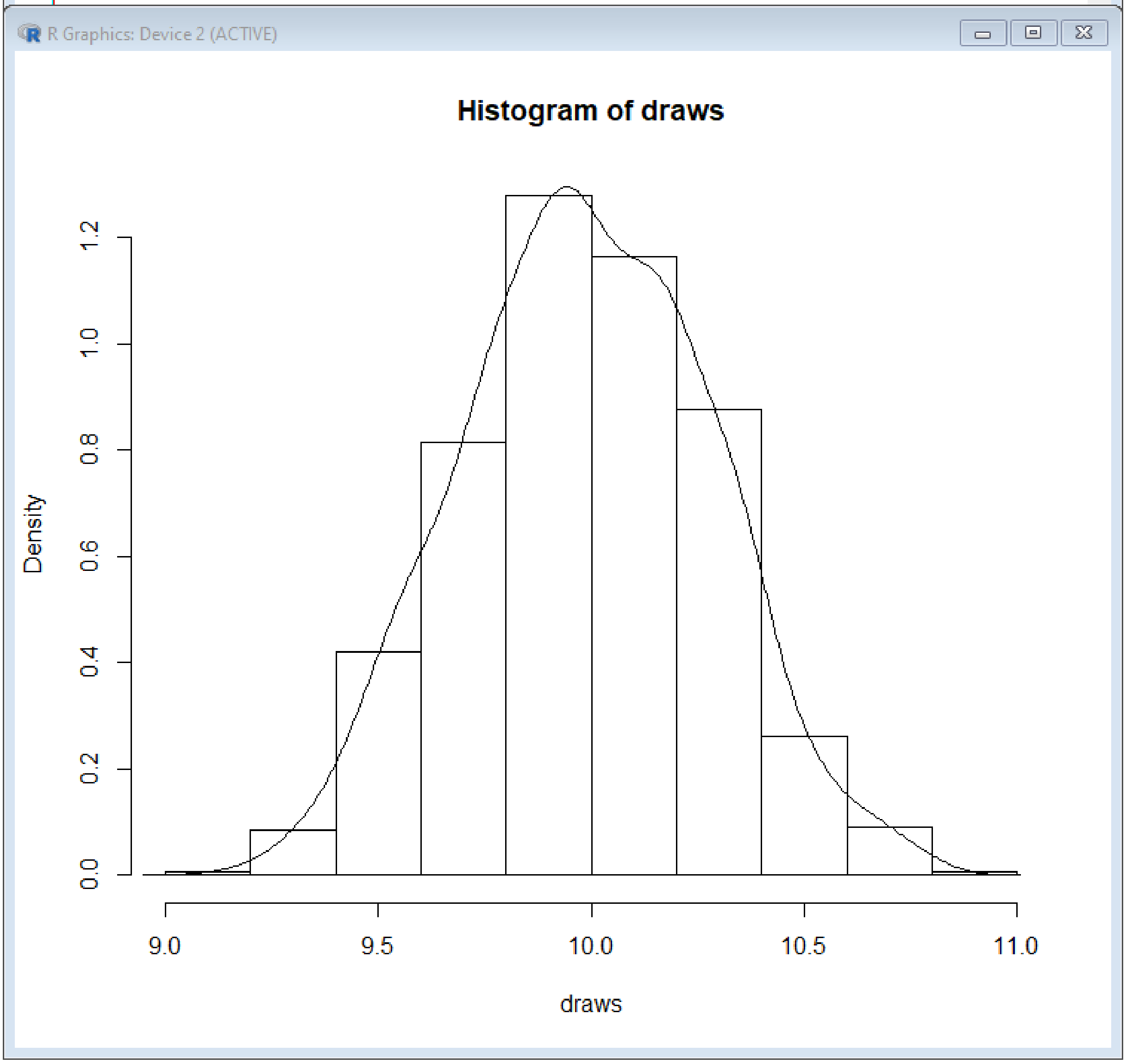
Figure 159 Graph à bulles de Rosling (animé)

Exemple 149.: Intervalles de distributions par densité de noyaux (kernel smoothing)

R 3.0.2

Voyons comment faire un simple kernel smoothing pour commencer avec la fonction **density( )**:

```
> draws<-rnorm(1000,10,0.3)
> hist(draws,prob=TRUE)
> lines(density(draws))
> 
```



Pour ce qui est des plots des surfaces de probabilité cumulée utilisant les méthodes de densité par noyaux (nous avons vu lors de notre découverte des graphiques comment faire le même travail pour l'expression exacte de distributions lorsque nous n'utilisons pas la méthode de densité par noyaux) et les quantiles:

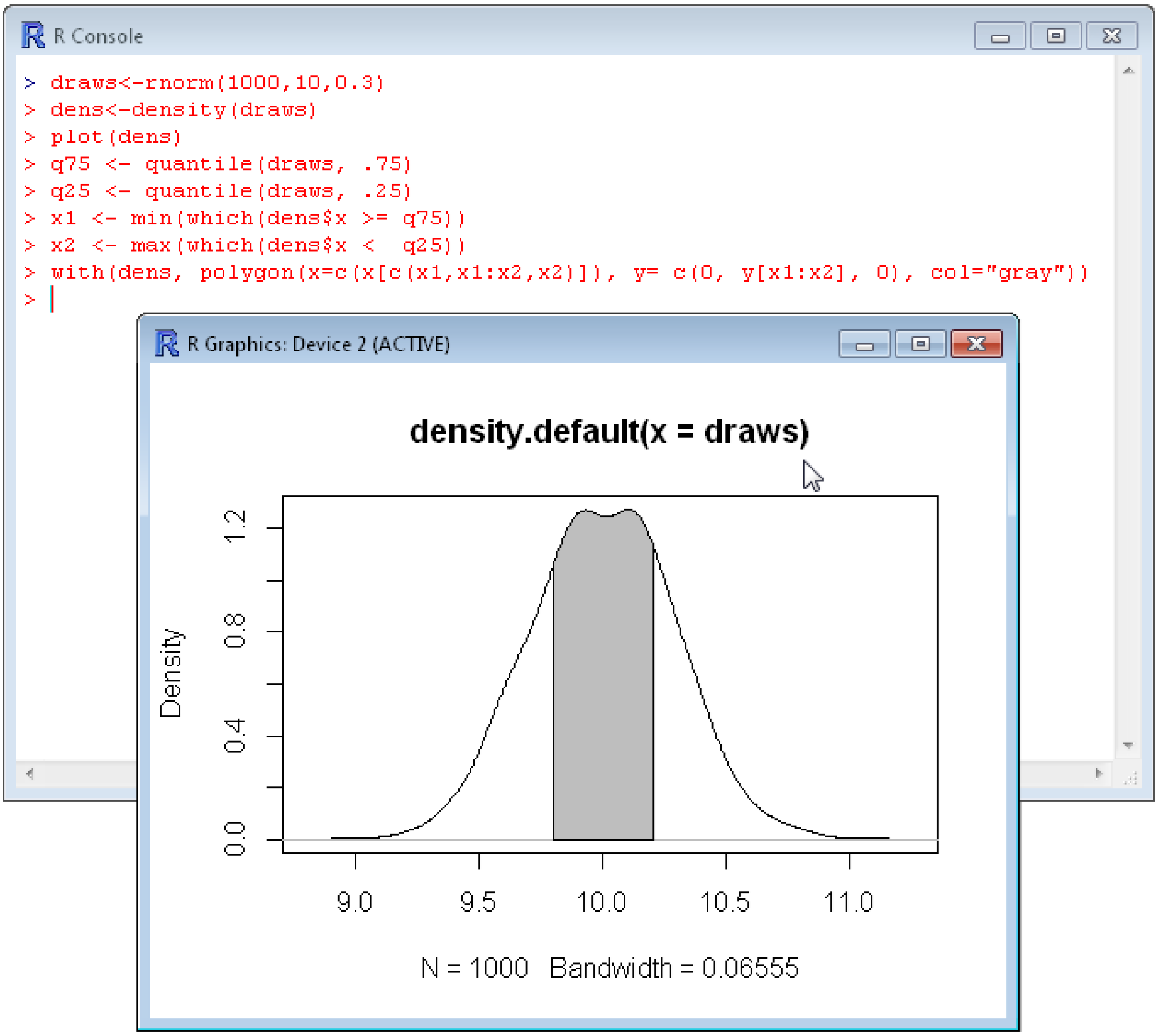
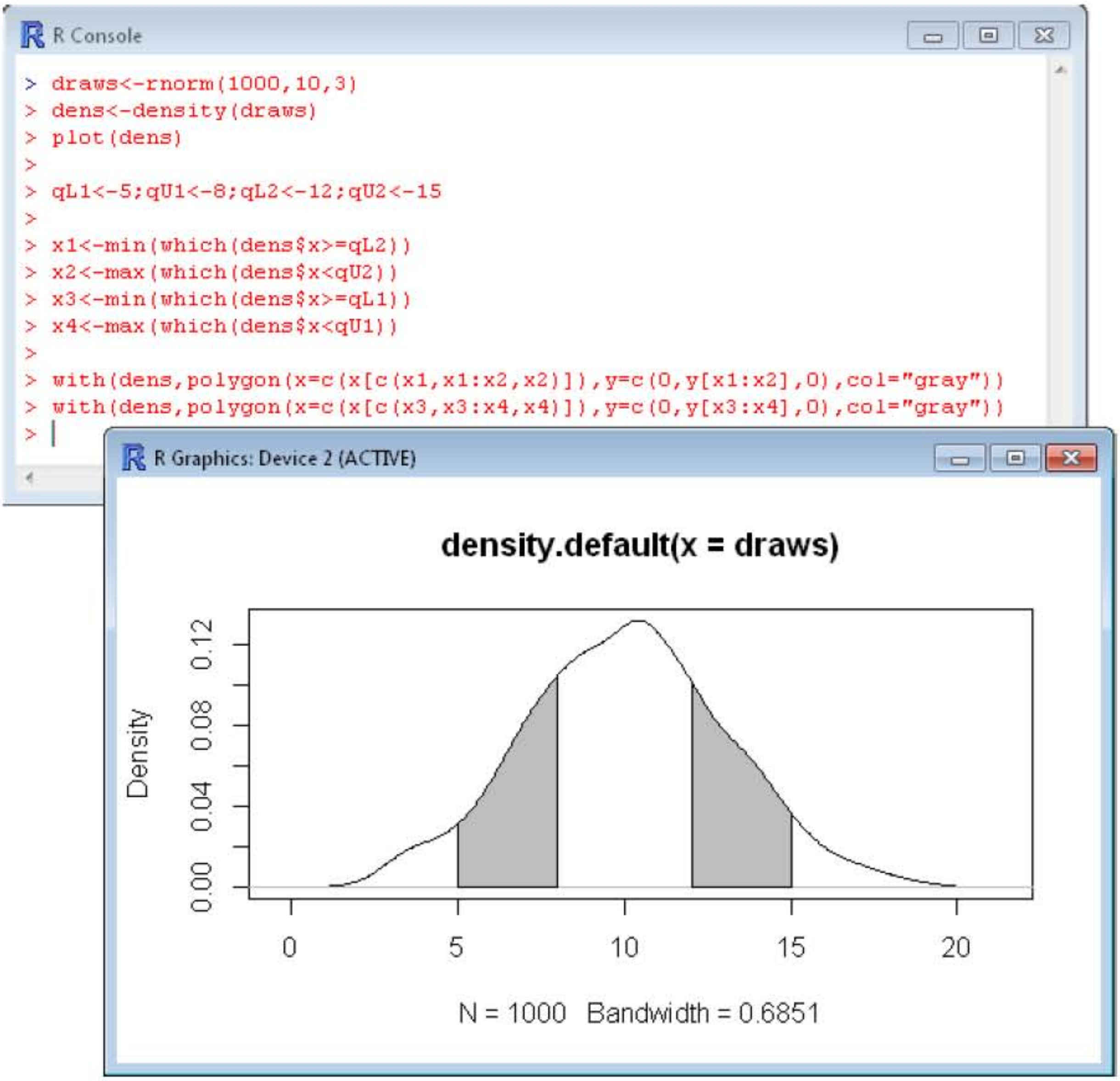


Figure 160 Graph d'intervalle de fonction de distribution

S'il s'agit non pas des quantiles mais du support et que nous ne voulons pas 1 seul intervalle mais plusieurs, voici la méthode:


R Console
> draws<-rnorm(1000,10,3)
> dens<-density(draws)
> plot(dens)
>
> qL1<-5;qU1<-8;qL2<-12;qU2<-15
>
> x1<-min(which(dens$x>=qL2))
> x2<-max(which(dens$x<qU2))
> x3<-min(which(dens$x>=qL1))
> x4<-max(which(dens$x<qU1))
>
> with(dens,polygon(x=c(x[c(x1,x1:x2,x2)]),y=c(0,y[x1:x2],0),col="gray"))
> with(dens,polygon(x=c(x[c(x3,x3:x4,x4)]),y=c(0,y[x3:x4],0),col="gray"))
R Graphics: Device 2 (ACTIVE)
density.default(x = draws)
Density
0.00
0.04
0.08
0.12
0
5
10
15
20
N = 1000 Bandwidth = 0.6851

Nous pouvons plotter l'intervalle des quelques quantiles pour la simulation de la fonction de densité:

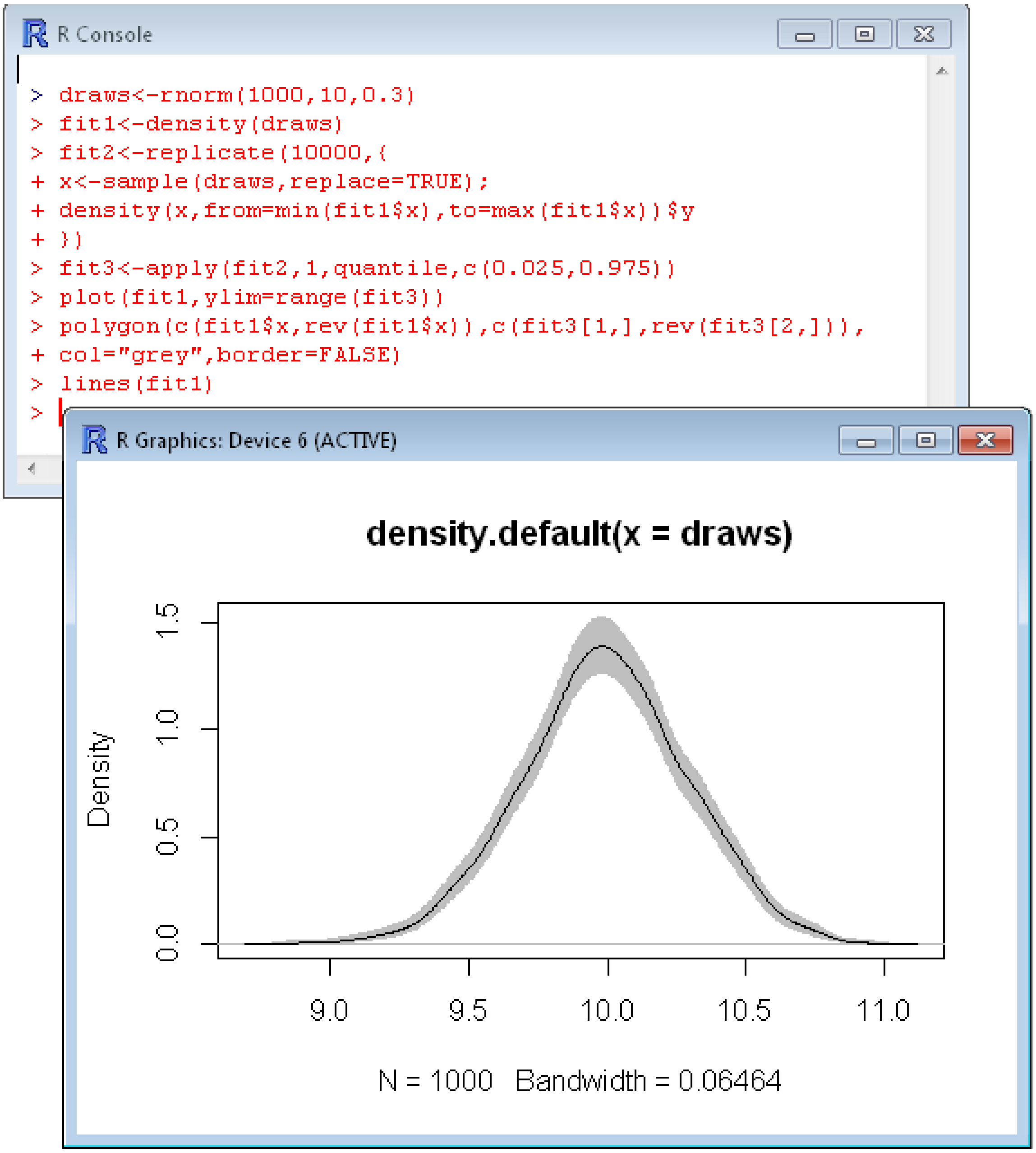


Figure 161 Graph polygone avec densité d'erreurs

Probabilité cumulée d'un intervalle d'une distribution estimée par noyaux

Voilà un exemple très important dans la pratique industrielle qui utilise la commande **KernelSmoothing.cdf( )** du package **DiagTest3Grp**:

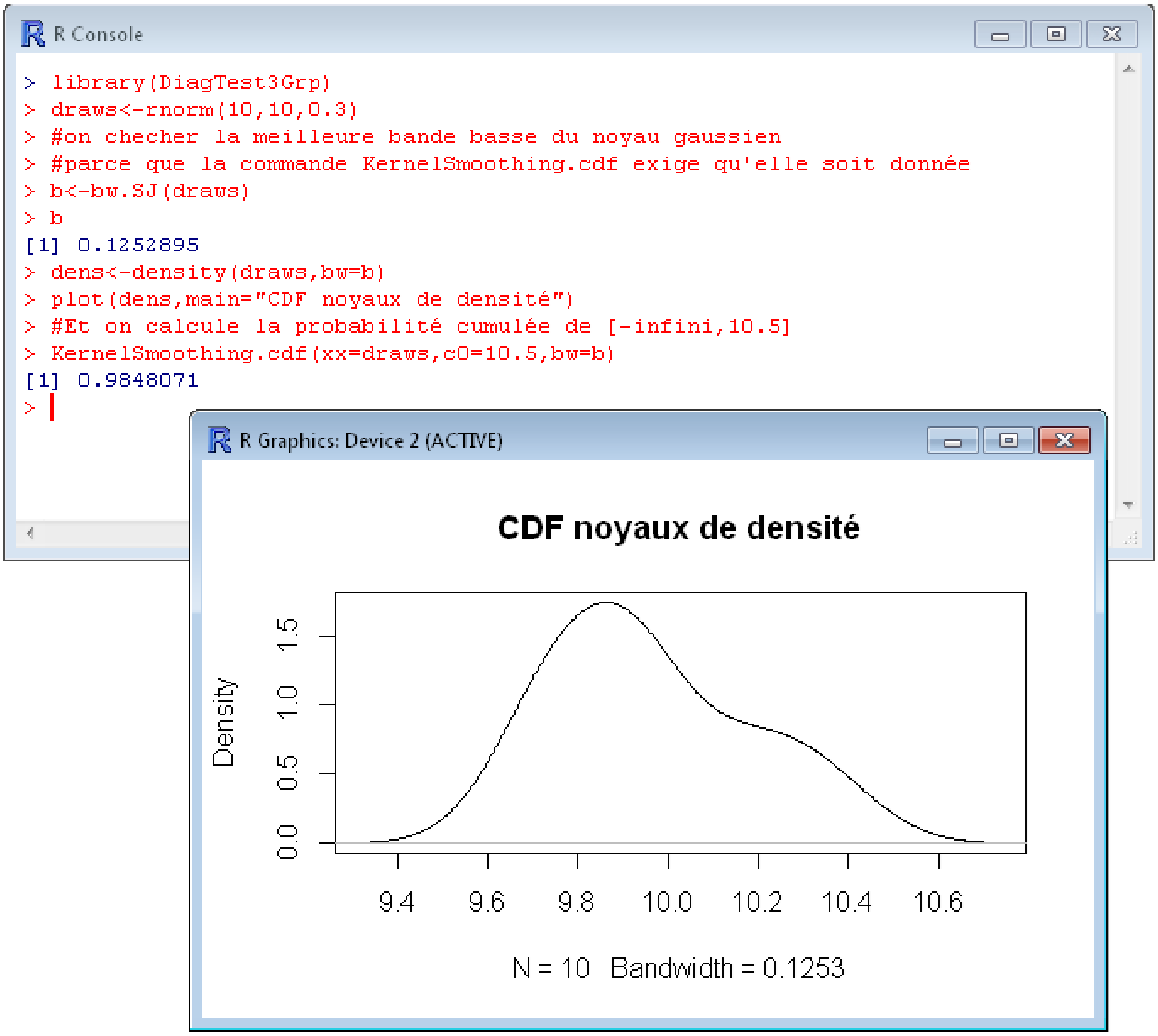


Figure 162 Graph de fonction de densité par la méthode des noyaux

Exemple 150.: Diagrammes combinés à lattes (facettes)

R 3.0.2

Nous allons ici utiliser un package relativement courant qui est la package **lattice**

Voyons quelques exemples:

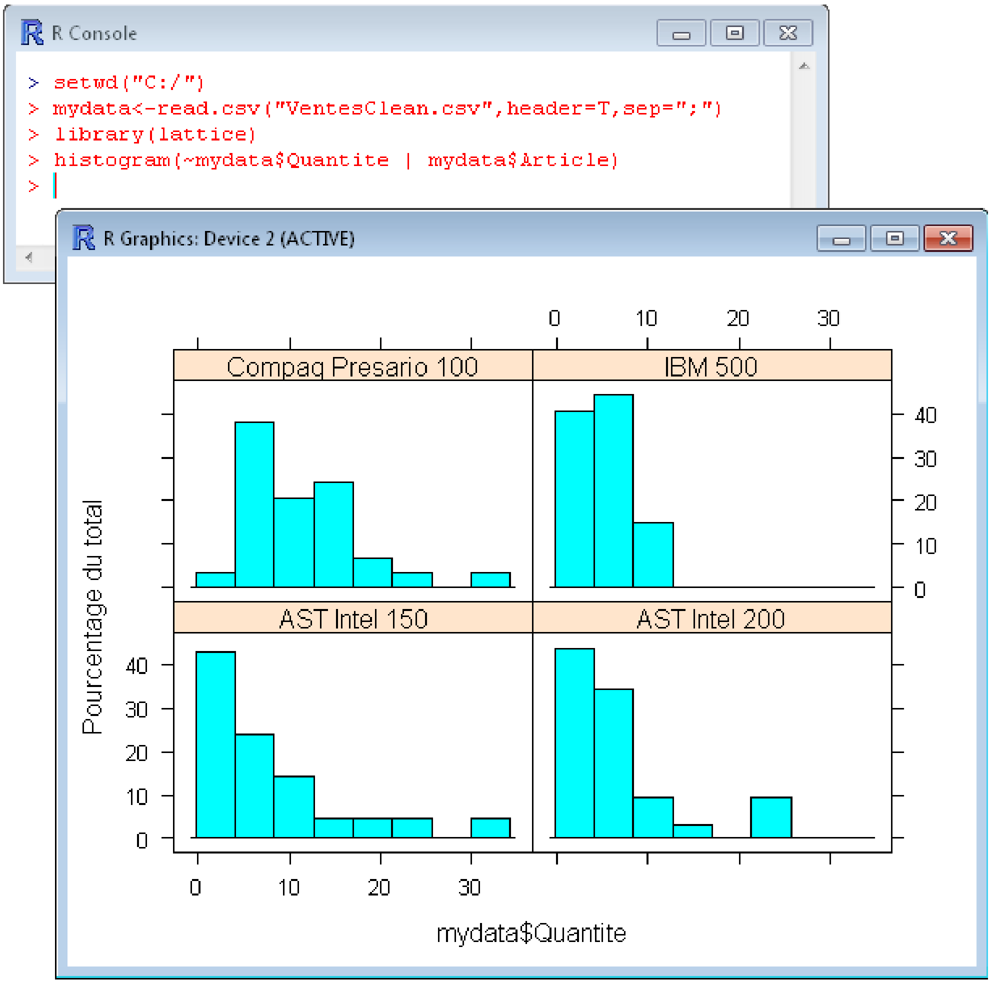


Figure 163 Graph à lattes (avec histogrammes)

ou:

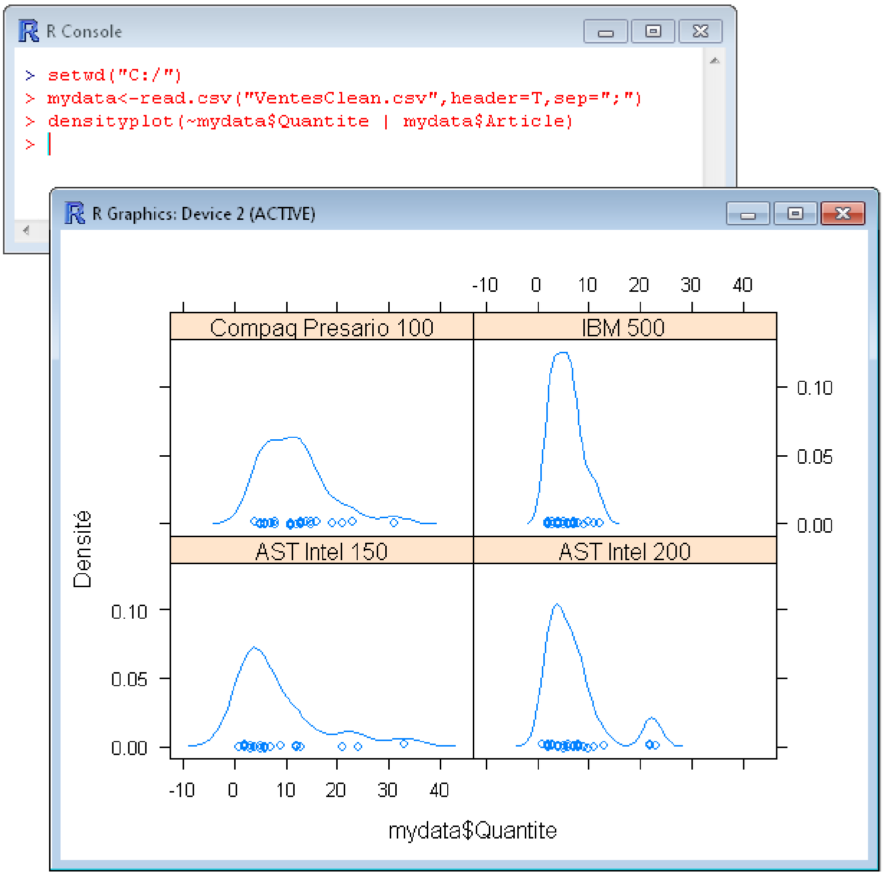


Figure 164 Graph à lattes avec fonctions de densité

ou encore:

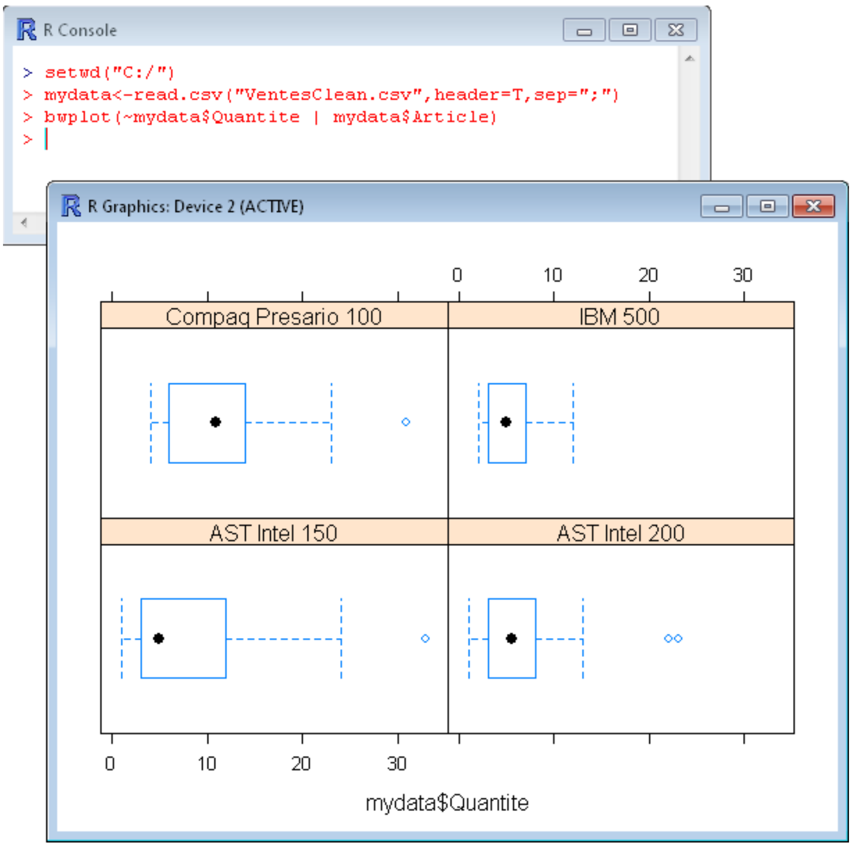
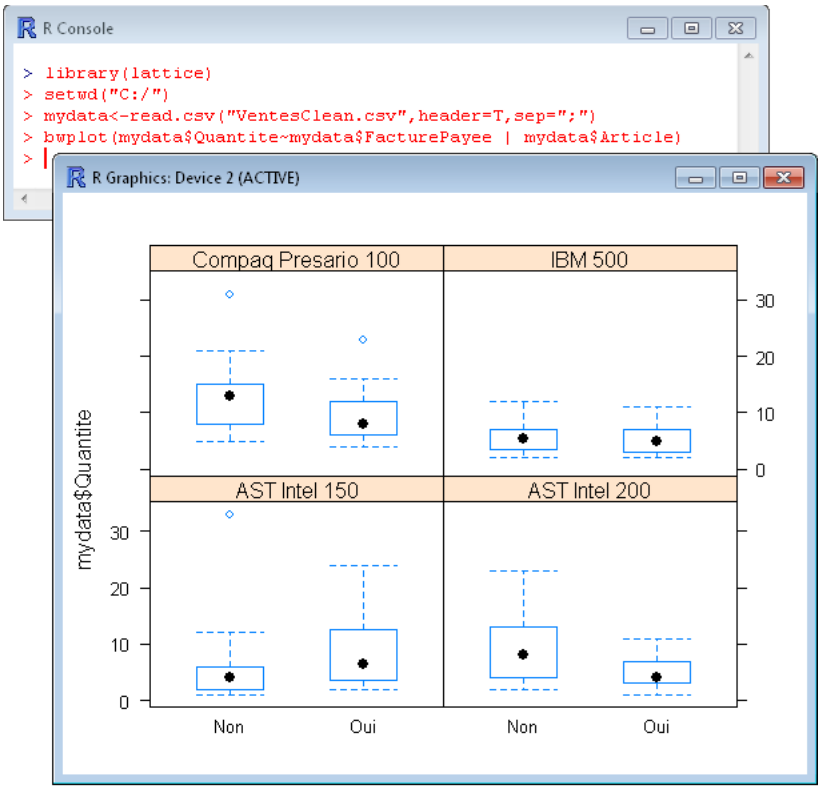
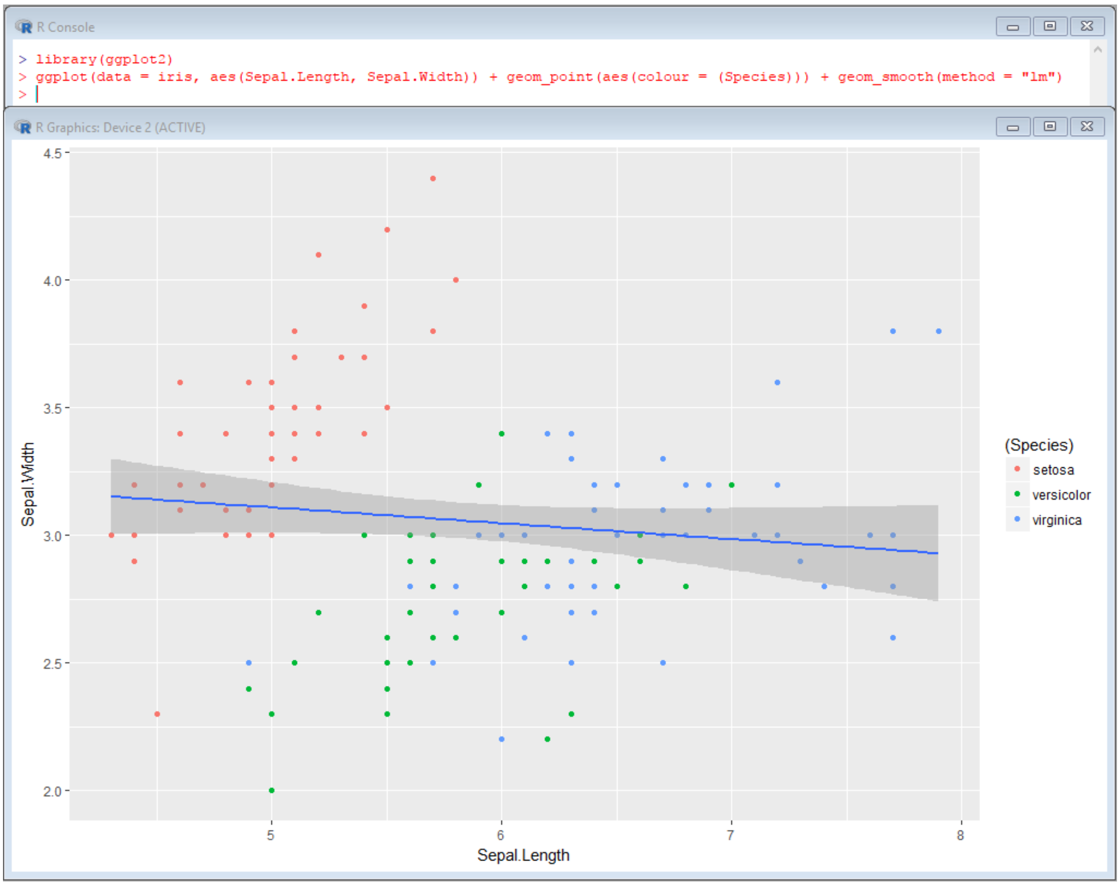


Figure 165 Graph à lattes avec Box-Whisker

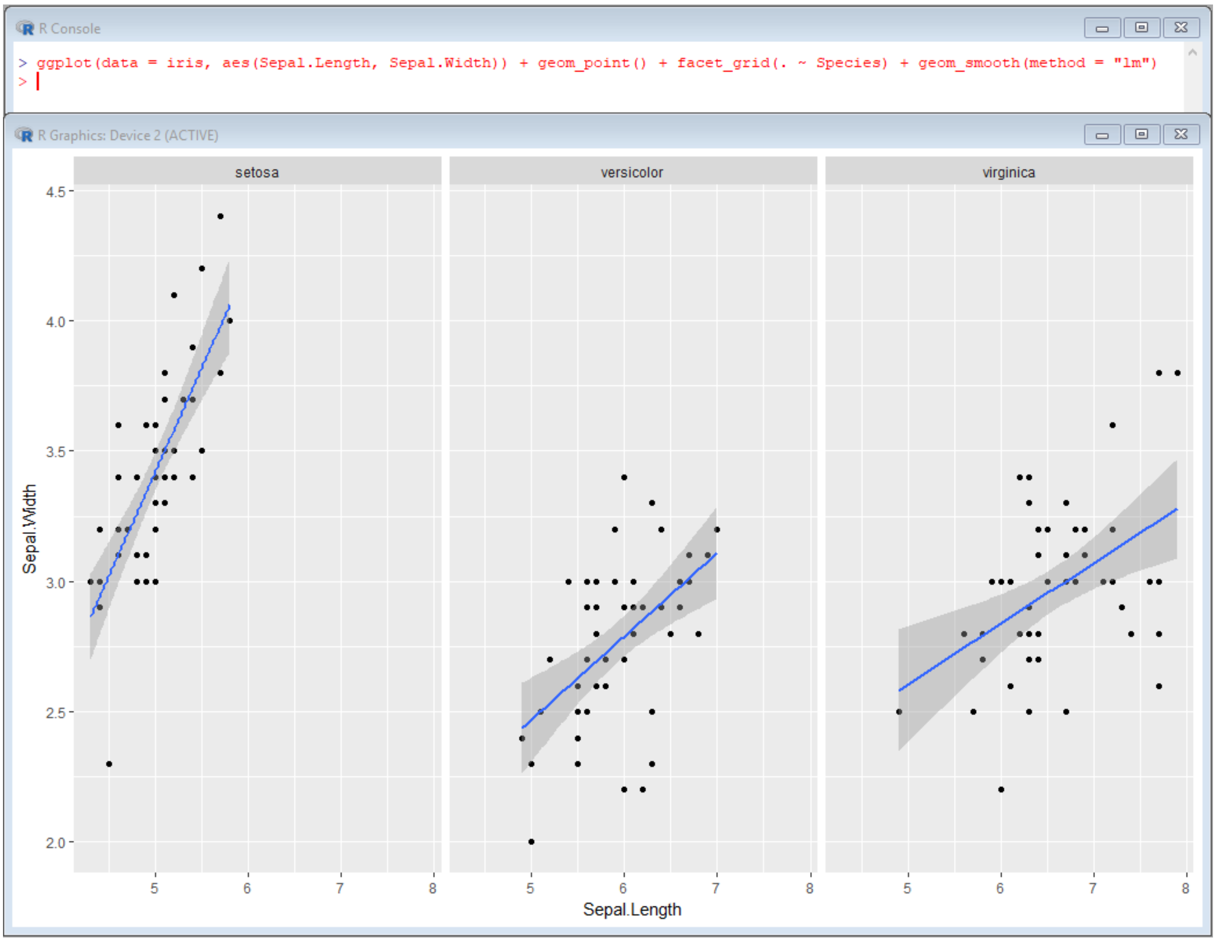
ou encore dans un autre style:



Un cas magnifique que l'on peut faire (avec ggplot) c'est la découverte du paradoxe de Simpson avec ggplot2. Donc d'abord on effectue une simple analyse de la régression:

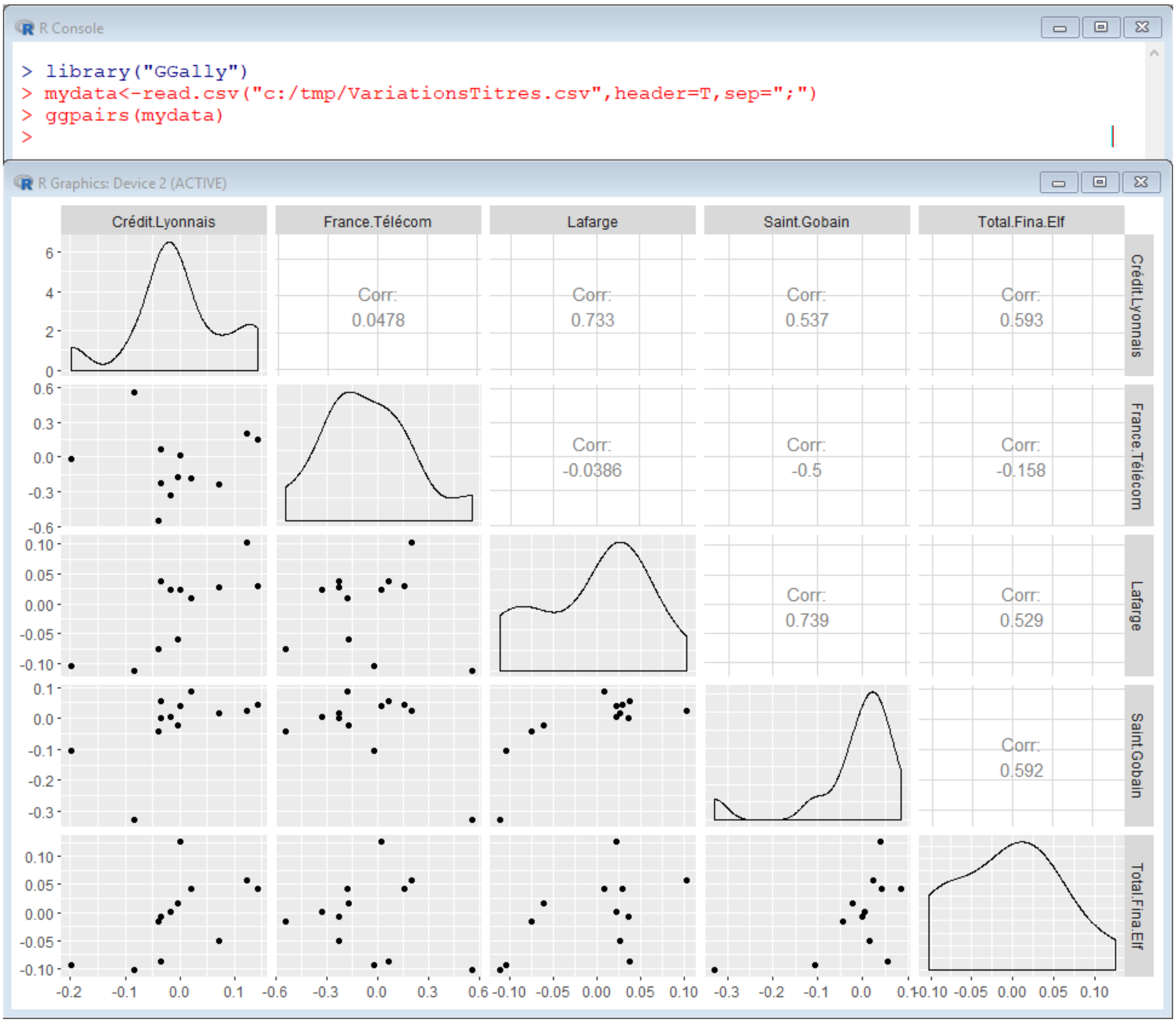


Et ensuite on clusterise l'analyse:

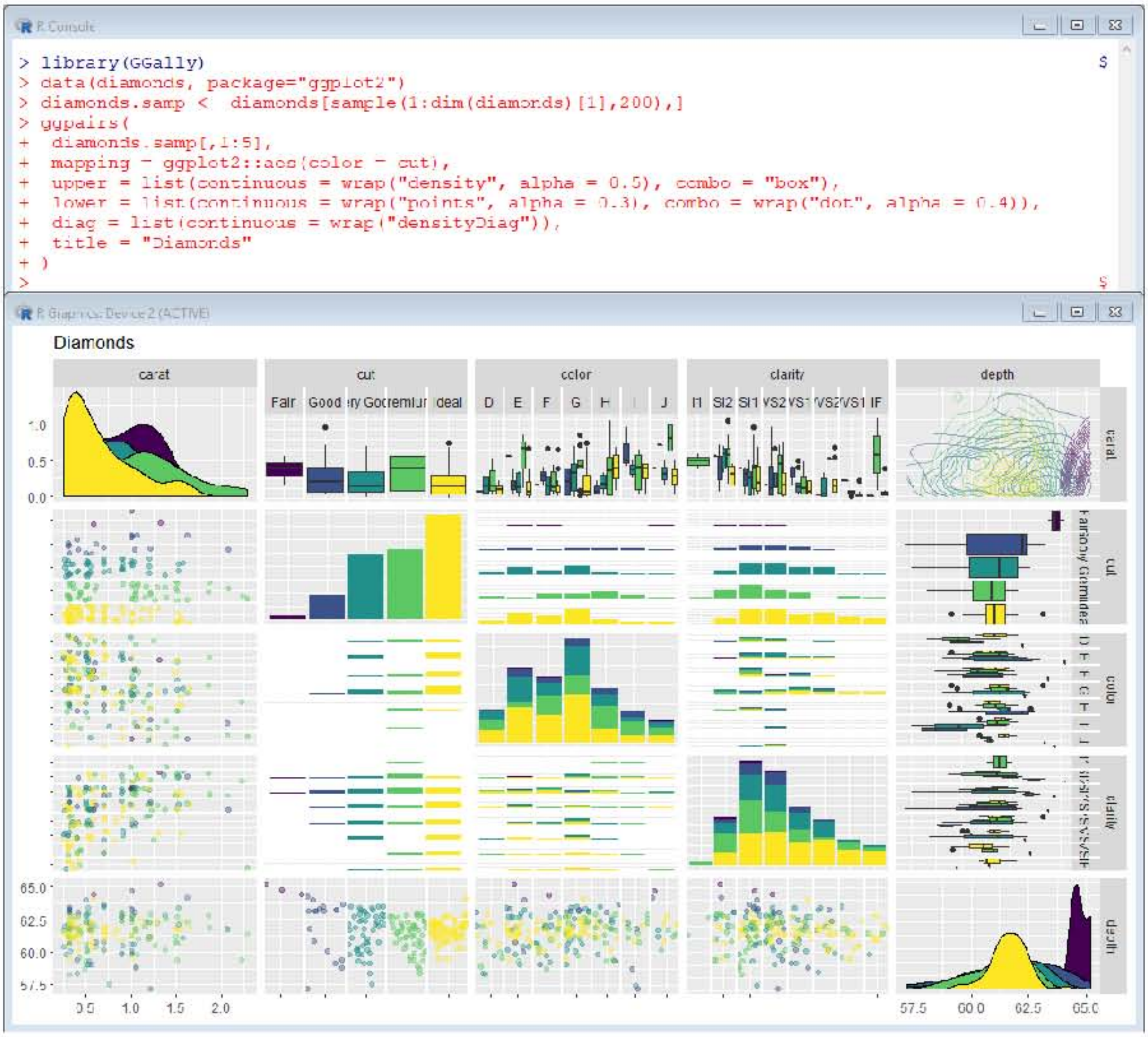


et voilà donc un magnifique exemple de paradoxe de Simpson!

Sinon encore une fois, toujours avec le package **GGally** et la fonction **ggpairs( )** déjà vue plus haut, nous avons:



Ou sinon un exemple impressionnant basé sur le jeu de données diamonds:


R Console
> library(GGally)
> data(diamonds, package="ggplot2")
> diamonds.samp <- diamonds[sample(1:dim(diamonds)[1],200),]
> ggpairs(
+ diamonds.samp[,1:5],
+ mapping = ggplot2::aes(color = cut),
+ upper = list(continuous = wrap("density", alpha = 0.5), combo = "box"),
+ lower = list(continuous = wrap("points", alpha = 0.3), combo = wrap("dot", alpha = 0.4)),
+ diag = list(continuous = wrap("densityDiag")),
+ title = "Diamonds"
+)
>
R Graphics: Device 2 (ACTIVE)
Diamonds
carat
cut
color
clarity
depth
Fair
Good
Ideal
D
E
F
G
H
J
I1
SI2
SI1
VS2
VS1
IF
1.0
0.5
0.0
65.0
62.5
60.0
57.5
0.5
1.0
1.5
2.0
57.5
60.0
62.5
65.0

Exemples: Diagrammes de flux

R 3.0.2

Flux parallèles (diagrammes de Sankey/Alluvial)

Bon ce n'est pas élégant et il n'existe pas de package à ce jour (2016) permettant d'automatiser la procédure à notre connaissance, mais voici déjà le script:

```
parallelset <- function(..., freq, col="gray", border=0, layer,
                             alpha=0.5, gap.width=0.05) {
  p <- data.frame(..., freq, col, border, alpha, stringsAsFactors=FALSE)
  n <- nrow(p)
  if(missing(layer)) { layer <- 1:n }
  p$layer <- layer
  np <- ncol(p) - 5
  d <- p[ , 1:np, drop=FALSE]
  p <- p[ , -c(1:np), drop=FALSE]
  p$freq <- with(p, freq/sum(freq))
  col <- col2rgb(p$col, alpha=TRUE)
  if(!identical(alpha, FALSE)) { col["alpha", ] <- p$alpha*256 }
  p$col <- apply(col, 2, function(x) do.call(rgb, c(as.list(x),
maxColorValue = 256)))
  getp <- function(i, d, f, w=gap.width) {
    a <- c(i, (1:ncol(d))[-i])
    o <- do.call(order, d[a])
    x <- c(0, cumsum(f[o])) * (1-w)
    x <- cbind(x[-length(x)], x[-1])
    gap <- cumsum( c(0L, diff(as.numeric(d[o,i])) != 0) )
    gap <- gap / max(gap) * w
    (x + gap)[order(o),]
  }
  dd <- lapply(seq_along(d), getp, d=d, f=p$freq)
  par(mar = c(0, 0, 2, 0) + 0.1, xpd=TRUE )
  plot(NULL, type="n",xlim=c(0, 1), ylim=c(np, 1),
       xaxt="n", yaxt="n", xaxs="i", yaxs="i", xlab='', ylab='',
frame=FALSE)
  for(i in rev(order(p$layer)) ) {
     for(j in 1:(np-1) )
     polygon(c(dd[[j]][i,], rev(dd[[j+1]][i,])), c(j, j, j+1, j+1),
             col=p$col[i], border=p$border[i])
   }
   text(0, seq_along(dd), labels=names(d), adj=c(0,-2), font=2)
   for(j in seq_along(dd)) {
     ax <- lapply(split(dd[[j]], d[,j]), range)
     for(k in seq_along(ax)) {
       lines(ax[[k]], c(j, j))
       text(ax[[k]][1], j, labels=names(ax)[k], adj=c(0, -0.25))
     }
   }
}

data(Titanic)
myt <- subset(as.data.frame(Titanic), Age=="Adult",
              select=c("Survived","Sex","Class","Freq"))
myt <- within(myt, {
  Survived <- factor(Survived, levels=c("Yes","No"))
  levels(Class) <- c(paste(c("First", "Second", "Third"), "Class"), "Crew")
  color <- ifelse(Survived=="Yes","#008888","#330066")
```

```
})

with(myt, parallelset(Survived, Sex, Class, freq=Freq, col=color,
alpha=0.2))
```

Ce qui donnera:

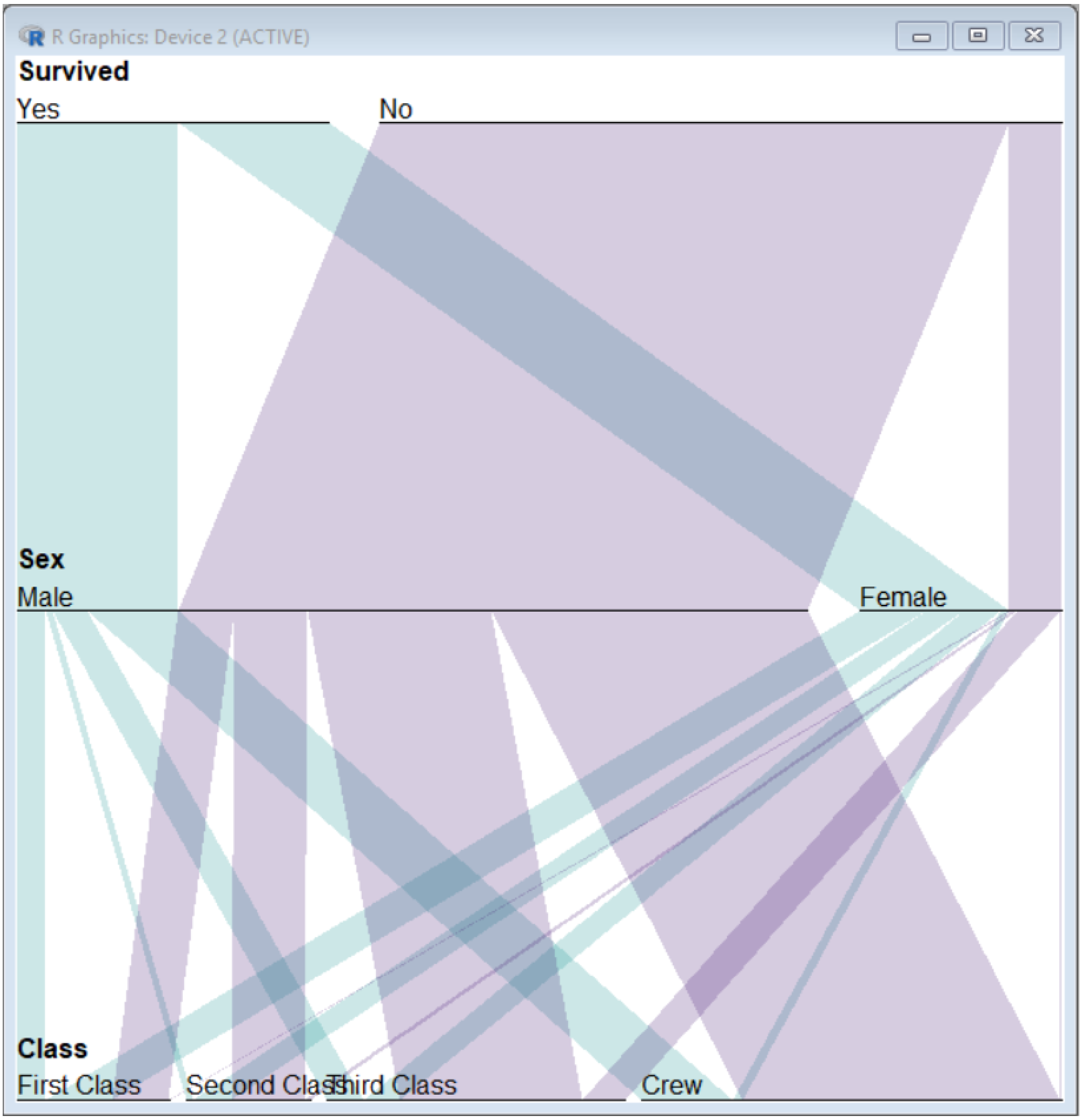
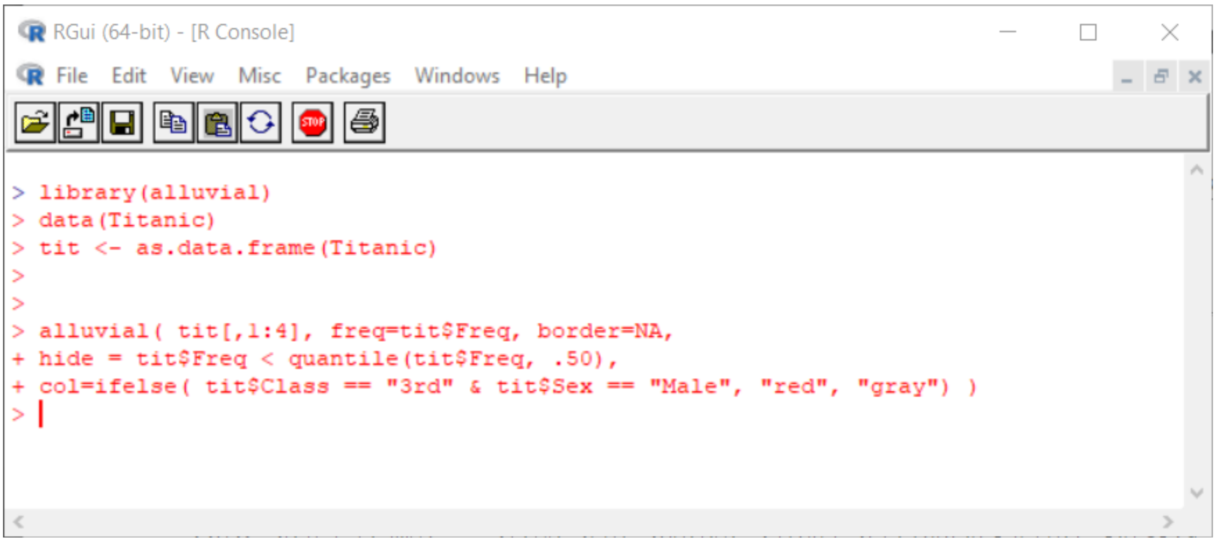


Figure 166 Graph de Sankey vertical

Ou encore plus simple, avec le package **alluvial** et la fonction **alluvial( )** :



Ce qui donne:

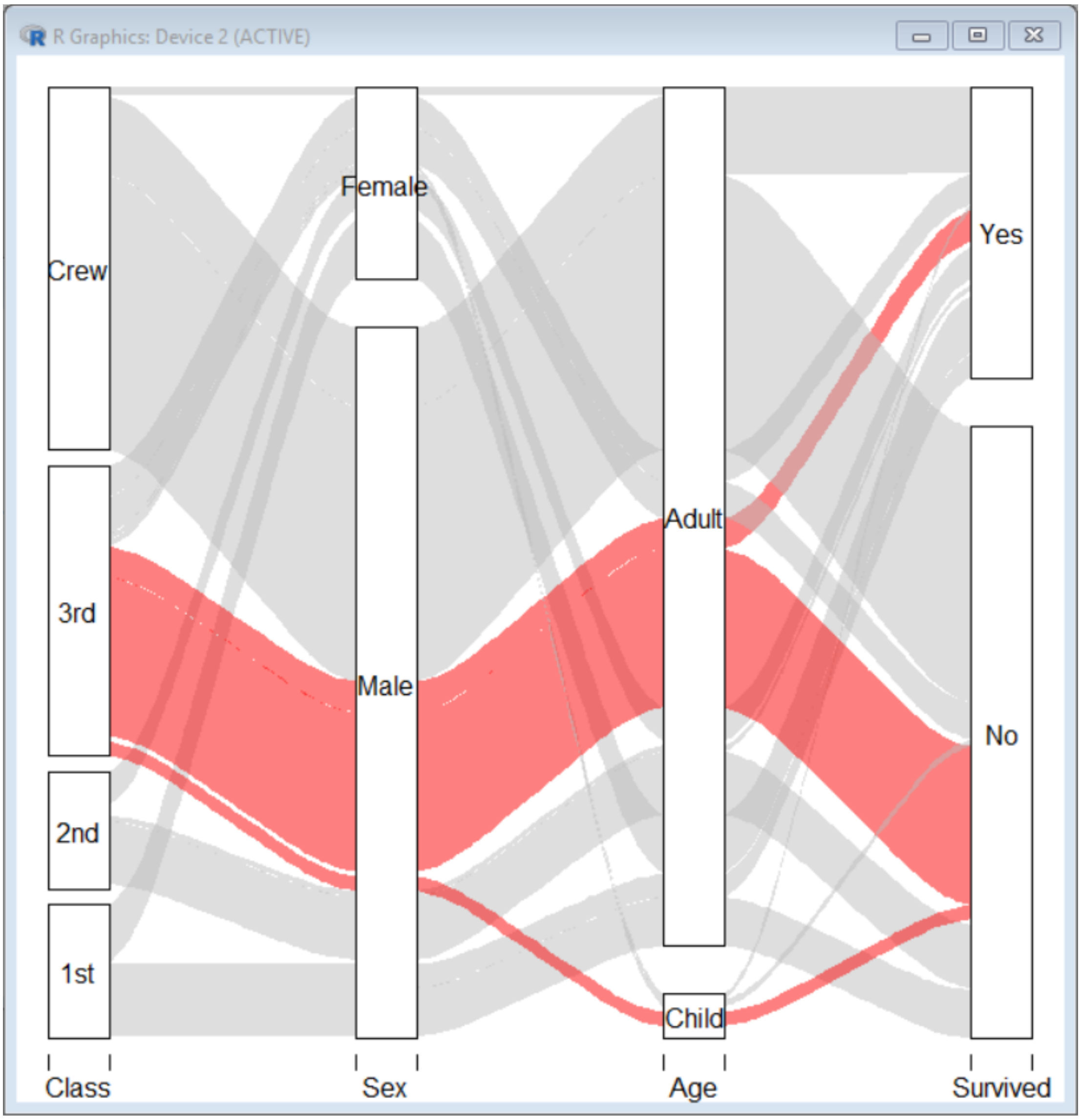


Figure 167 Graph de Sankay horizontal de type alluvial

Pour le domaine médical on peut faire une version plus élaborée. Considérons alors le jeu de donnés suivant:

allu_data.csv - Notepad

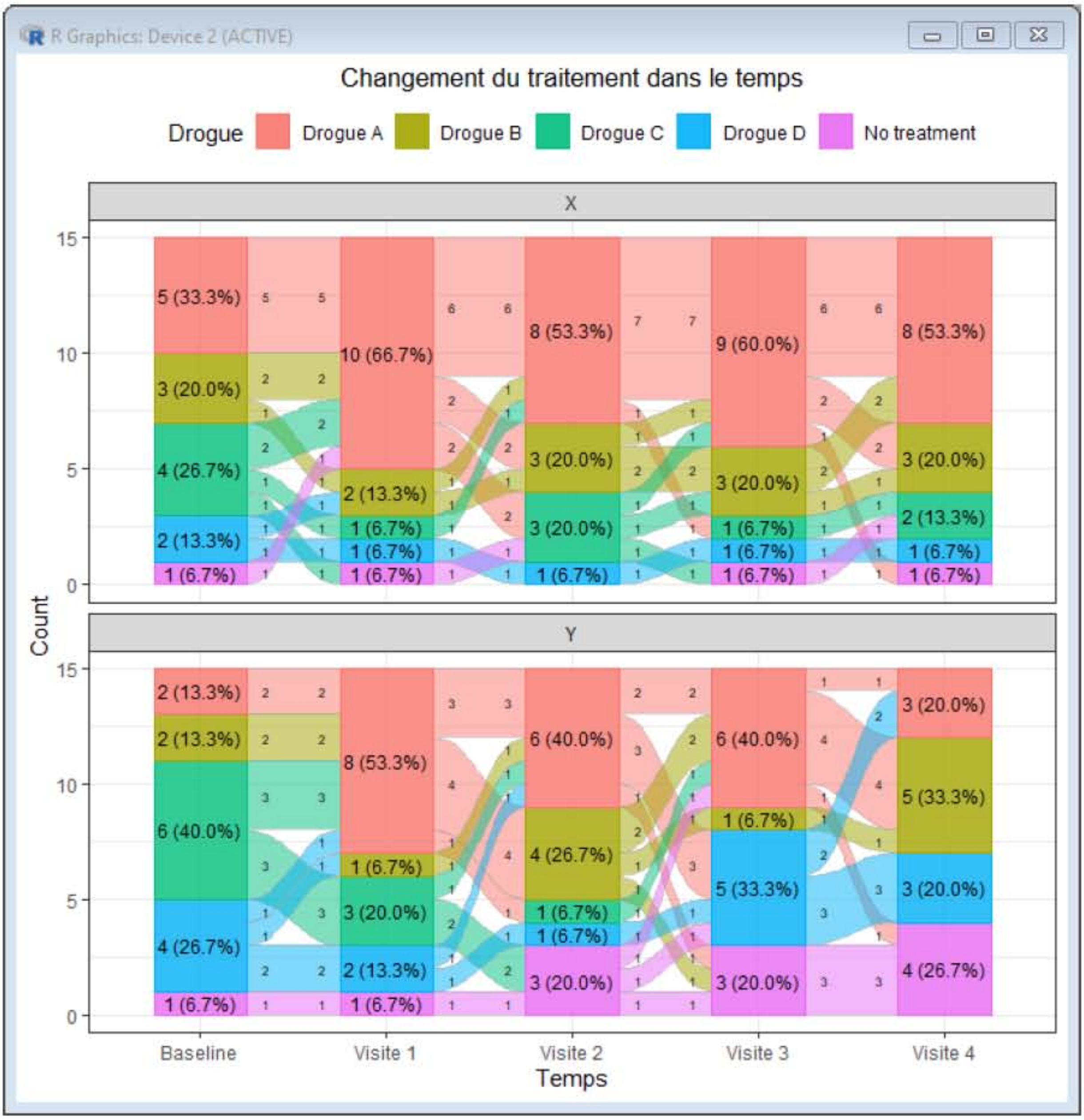
```
Id_Patient,Temps,Drogue,Groupe
1,Baseline,Drogue A,X
1,Visite 1,Drogue A,X
1,Visite 2,Drogue A,X
1,Visite 3,Drogue A,X
1,Visite 4,Drogue A,X
2,Baseline,Drogue C,X
2,Visite 1,Drogue A,X
2,Visite 2,Drogue B,X
2,Visite 3,Drogue B,X
2,Visite 4,Drogue B,X
3,Baseline,Drogue B,X
3,Visite 1,Drogue A,X
3,Visite 2,Drogue B,X
3,Visite 3,Drogue A,X
3,Visite 4,Drogue B,X
4,Baseline,Drogue C,X
4,Visite 1,Drogue C,X
4,Visite 2,Drogue A,X
4,Visite 3,Drogue A,X
4,Visite 4,NA,X
5,Baseline,Drogue A,X
5,Visite 1,Drogue A,X
5,Visite 2,Drogue A,X
5,Visite 3,Drogue A,X
5,Visite 4,Drogue A,X
6,Baseline,Drogue B,X
6,Visite 1,Drogue B,X
6,Visite 2,Drogue B,X
6,Visite 3,Drogue B,X
6,Visite 4,Drogue A,X
7,Baseline,Drogue B,X
7,Visite 1,Drogue A,X
7,Visite 2,Drogue A,X
7,Visite 3,Drogue A,X
7,Visite 4,Drogue A,X
8,Baseline,Drogue D,X
8,Visite 1,Drogue B,X
8,Visite 2,Drogue A,X
8,Visite 3,Drogue C,X
8,Visite 4,Drogue C,X
9,Baseline,Drogue C,X
9,Visite 1,NA,X
9,Visite 2,Drogue C,X
9,Visite 3,NA,X
9,Visite 4,Drogue C,X
10,Baseline,Drogue A,X
10,Visite 1,Drogue A,X
10,Visite 2,Drogue C,X
10,Visite 3,Drogue B,X
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

Écrivons le script suivant:

```
> library("ggplot2")
> library("dplyr")
> library("ggalluvial")
> allu_data <- read.csv("c:/tmp/allu_data.csv",header=T,sep=",")
>
>
> h_just    <- 0.35
> font_size <- 2.2
>
> ggplot(data = allu_data %>% mutate(Drogue = ifelse(is.na(Drogue), "No treatment", as.character(Drogue))),
+        aes(x = Temps, stratum=Drogue, alluvium = Id_Patient)) +
+   geom_flow(aes(fill = Drogue),
+             stat = "flow",
+             width = .5,
+             aes.flow = "forward",
+             color="darkgrey") +
+   geom_stratum(aes(fill = Drogue, color=Drogue), alpha = .8, width = .5) +
+   theme_bw() +
+   theme(legend.position="top") +
+   labs(title = "Changement du traitement dans le temps") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   geom_text(aes(label = after_stat(sprintf("%d (%.1f%%)", n, 100*prop))),
+             stat = "stratum",
+             size=3.1) +
+   geom_text(aes(label = after_stat(ifelse(flow ==   "from", n, ""))),
+             nudge_x = h_just,
+             stat = "flow",
+             size = font_size,
+             col = "black",
+             show.legend = FALSE) +
+   geom_text(aes(label = after_stat(ifelse(flow ==   "to", n, ""))),
+             nudge_x = -1*h_just,
+             stat = "flow",
+             size = font_size,
+             col = "black",
+             show.legend = FALSE) +
+   ylab("Count") +
+   facet_wrap(~Groupe, nrow = 2)
>
```

Ce qui donne:

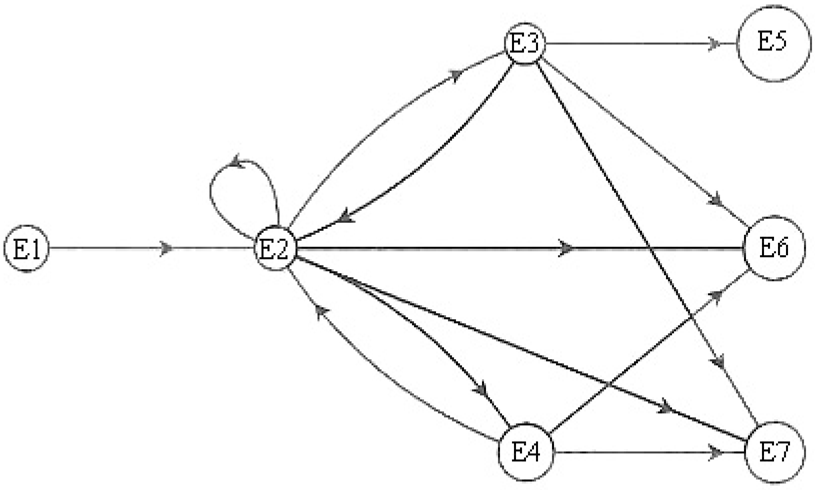

R Graphics: Device 2 (ACTIVE)
Changement du traitement dans le temps
Drogue
Drogue A
Drogue B
Drogue C
Drogue D
No treatment
X
Y
Count
Baseline
Visite 1
Visite 2
Visite 3
Visite 4
Temps

Théorie des graphes

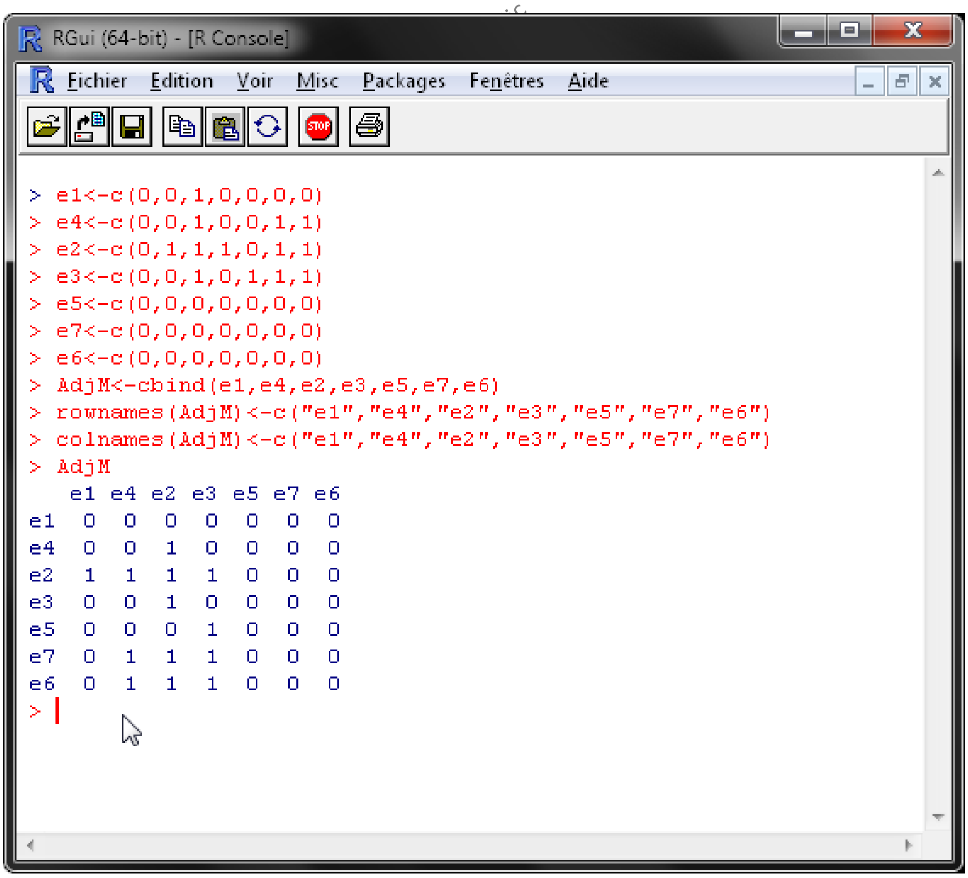
Exemple 1.: Graphe de matrice d'adjacence

R 3.0.2

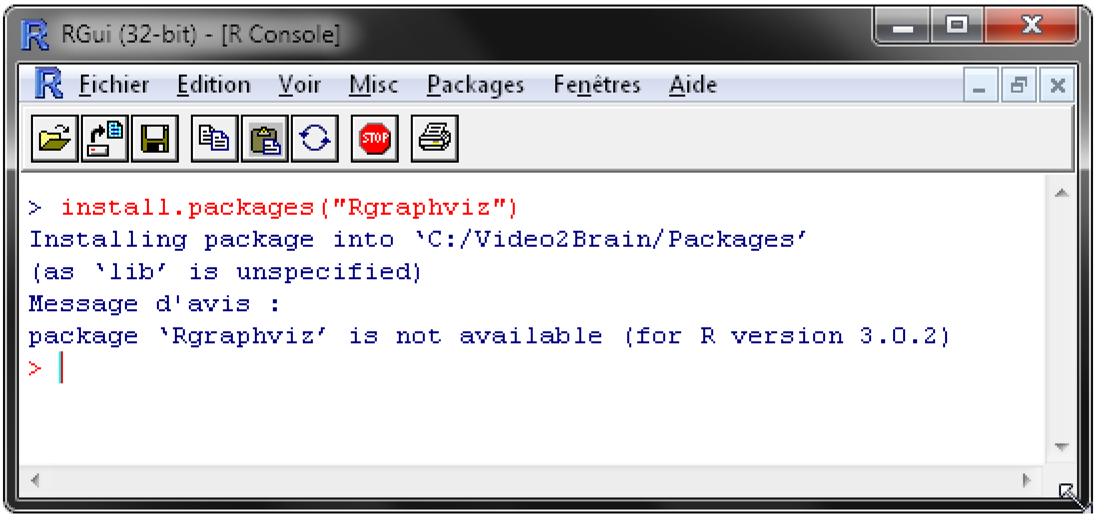
Nous souhaitons voir ici si nous pouvons reproduire avec R le graphe suivant utilisé dans le cours théorique de théorie des graphes à partir de sa matrice d'adjacence:



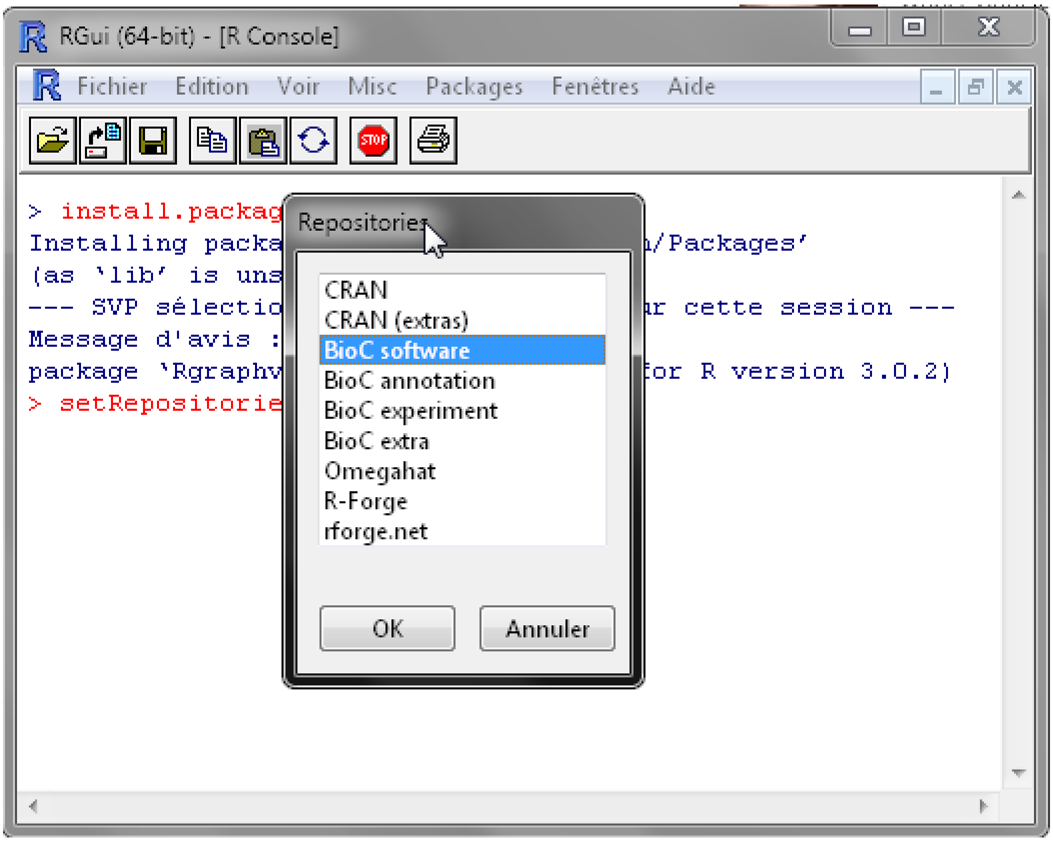
Donc d'abord nous créons la matrice d'adjacence correspondante dans R:



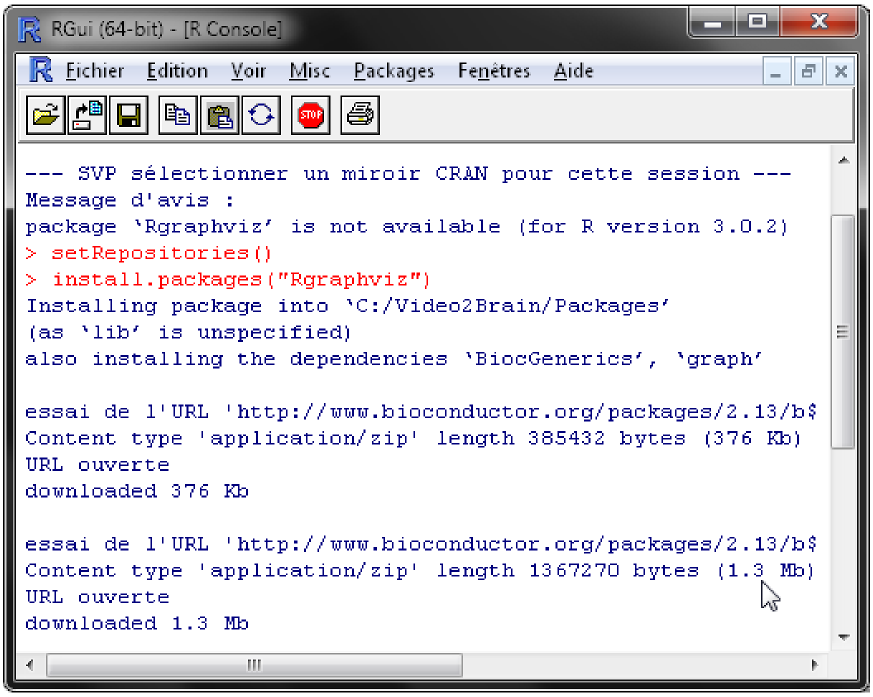
Nous installons ensuite le package **Rgraphviz**:



Mais nous avons un message d'erreur. Pas de soucis! L'astuce va consister à définir la source nous même avec la commande **setRepositories( )** et de prendre *BioC Software:*



Alors vous pouvez installer le package:



Maintenant nous chargeons le package et faisons un plot de notre graph en utilisant la commande **am.graph( )** :

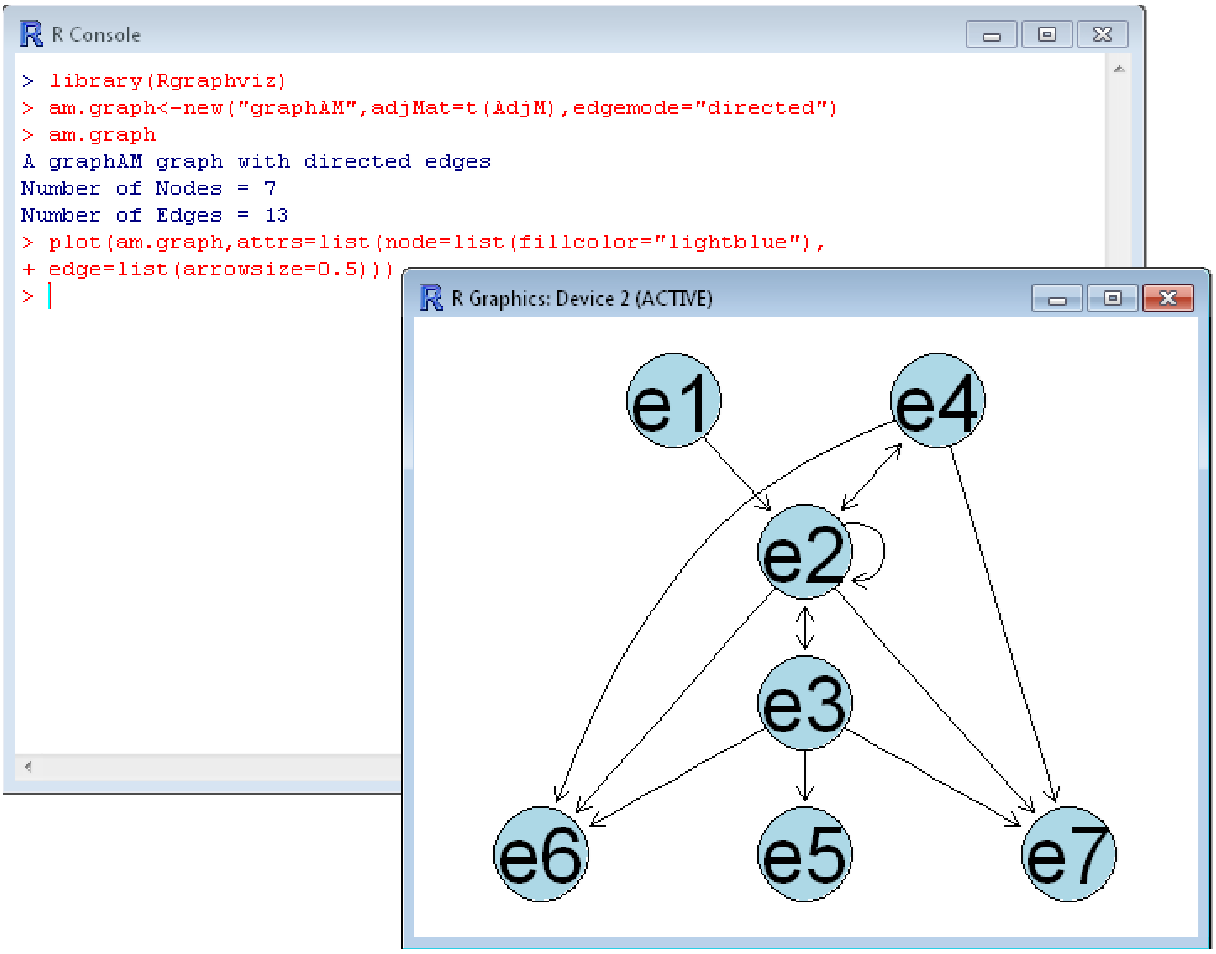


Figure 168 Graph de matrice d'adjacence

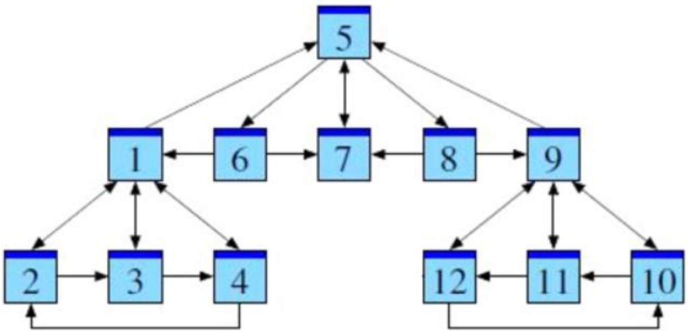
et bingo!!!

Exemple 2.: Graphe de matrice d'adjacence et poids des sommets selon algorithme du Google Page Rank

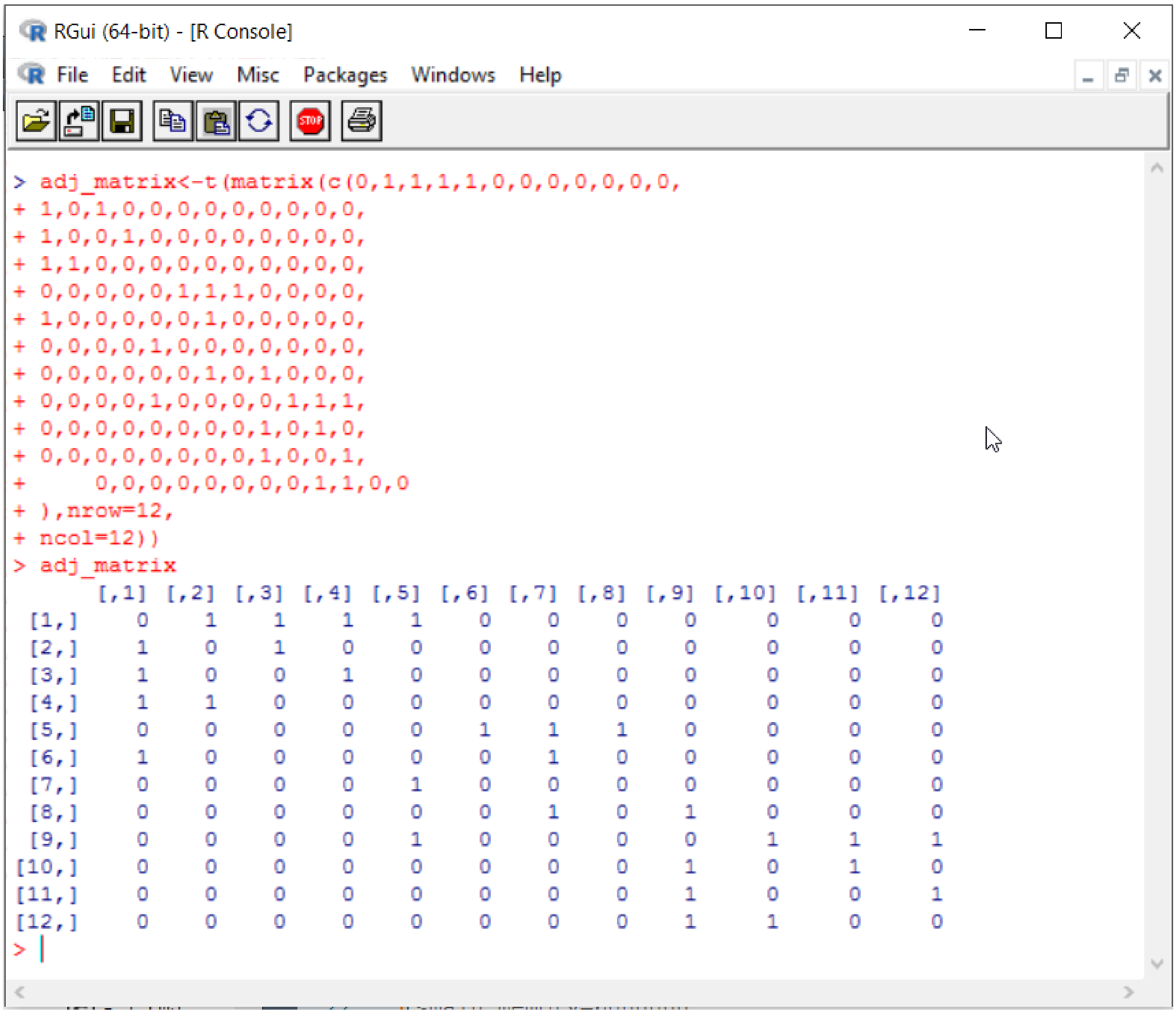
R 3.3.2

Nous allons voir ici si nous retombons avec R sur la mesure invariante d'un réseau tel que déterminé dans le cours théorique.

Rappelons que nous partons de:

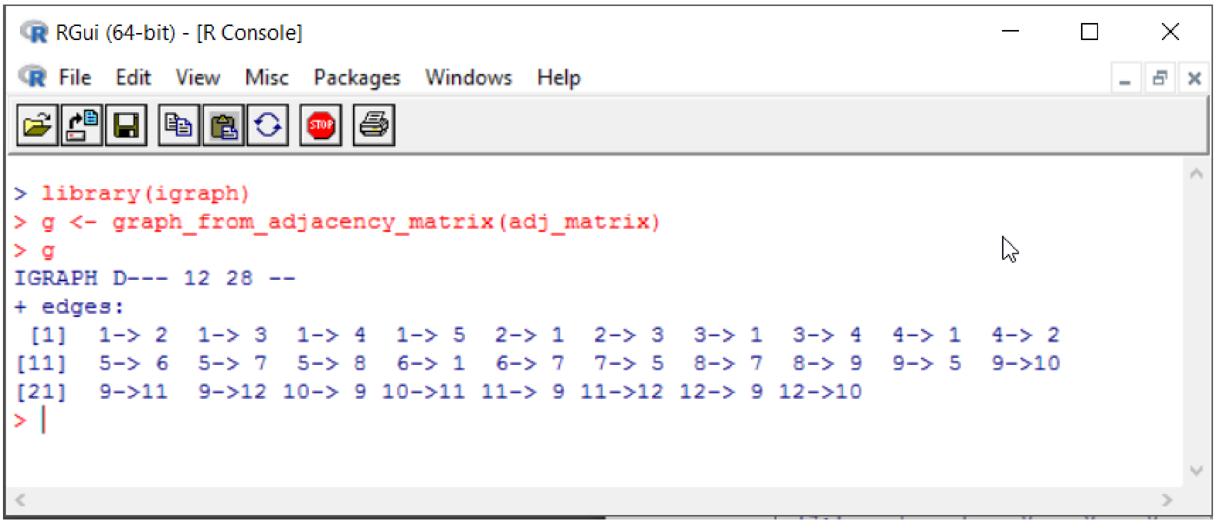


Ce qui donne en matrice d'adjacence dans R:



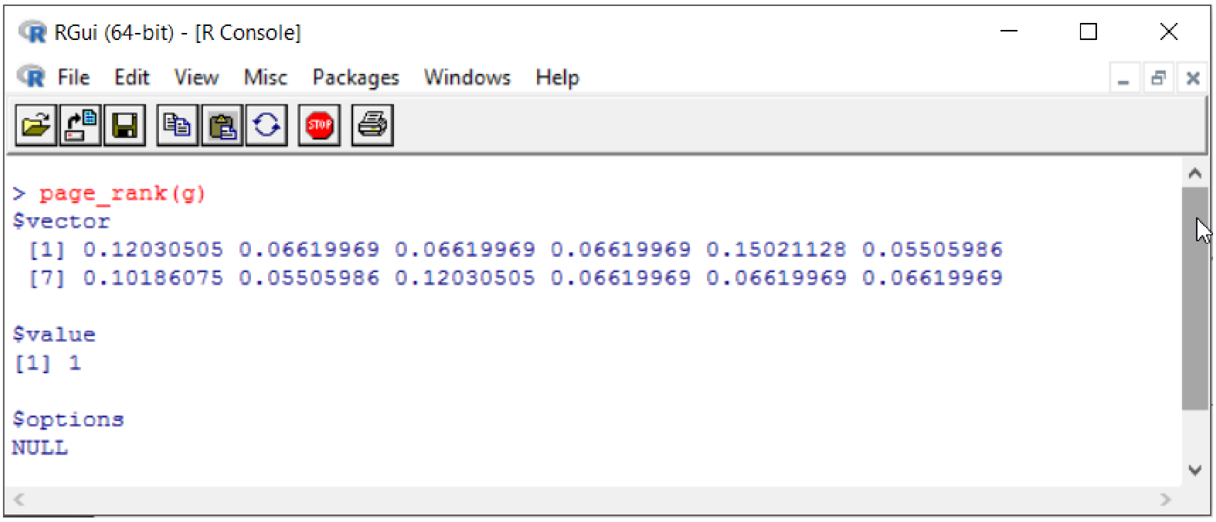

```
> adj_matrix<-t(matrix(c(0,1,1,1,1,0,0,0,0,0,0,0,
+ 1,0,1,0,0,0,0,0,0,0,0,0,
+ 1,0,0,1,0,0,0,0,0,0,0,0,
+ 1,1,0,0,0,0,0,0,0,0,0,0,
+ 0,0,0,0,0,1,1,1,0,0,0,0,
+ 1,0,0,0,0,0,1,0,0,0,0,0,
+ 0,0,0,0,1,0,0,0,0,0,0,0,
+ 0,0,0,0,0,0,1,0,1,0,0,0,
+ 0,0,0,0,1,0,0,0,0,1,1,1,
+ 0,0,0,0,0,0,0,0,1,0,1,0,
+ 0,0,0,0,0,0,0,0,1,0,0,1,
+     0,0,0,0,0,0,0,0,1,1,0,0
+ ),nrow=12,
+ ncol=12))
> adj_matrix
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
 [1,]    0    1    1    1    1    0    0    0    0     0     0     0
 [2,]    1    0    1    0    0    0    0    0    0     0     0     0
 [3,]    1    0    0    1    0    0    0    0    0     0     0     0
 [4,]    1    1    0    0    0    0    0    0    0     0     0     0
 [5,]    0    0    0    0    0    1    1    1    0     0     0     0
 [6,]    1    0    0    0    0    0    1    0    0     0     0     0
 [7,]    0    0    0    0    1    0    0    0    0     0     0     0
 [8,]    0    0    0    0    0    0    1    0    1     0     0     0
 [9,]    0    0    0    0    1    0    0    0    0     1     1     1
[10,]    0    0    0    0    0    0    0    0    1     0     1     0
[11,]    0    0    0    0    0    0    0    0    1     0     0     1
[12,]    0    0    0    0    0    0    0    0    1     1     0     0
>
```

Et maintenant nous utilisons la fonction **graph_from_adjacency_matrix( )** du package **igraph** pour obtenir un objet graph:




```
> library(igraph)
> g <- graph_from_adjacency_matrix(adj_matrix)
> g
IGRAPH D--- 12 28 --
+ edges:
 [1]  1-> 2  1-> 3  1-> 4  1-> 5  2-> 1  2-> 3  3-> 1  3-> 4  4-> 1  4-> 2
[11]  5-> 6  5-> 7  5-> 8  6-> 1  6-> 7  7-> 5  8-> 7  8-> 9  9-> 5  9->10
[21]  9->11  9->12 10-> 9 10->11 11-> 9 11->12 12-> 9 12->10
>
```

Ensuite, nous calculons enfin la mesure invariante avec la fonction **page_rank( )** toujours du package **igraph( ):**



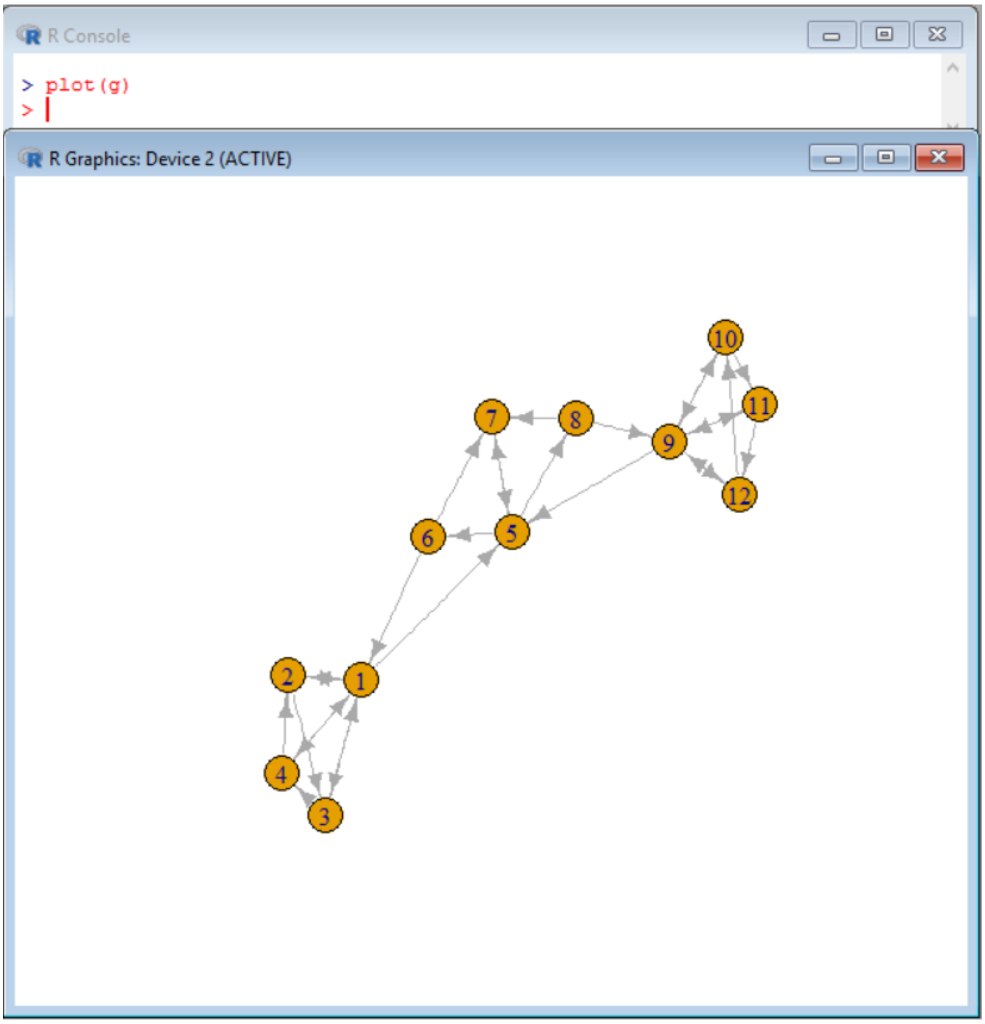

```
> page_rank(g)
$vector
 [1] 0.12030505 0.06619969 0.06619969 0.06619969 0.15021128 0.05505986
 [7] 0.10186075 0.05505986 0.12030505 0.06619969 0.06619969 0.06619969

$value
[1] 1

$options
NULL
```

Nous ne sommes donc pas trop loin des 34 premières itérations données dans Microsoft Excel!

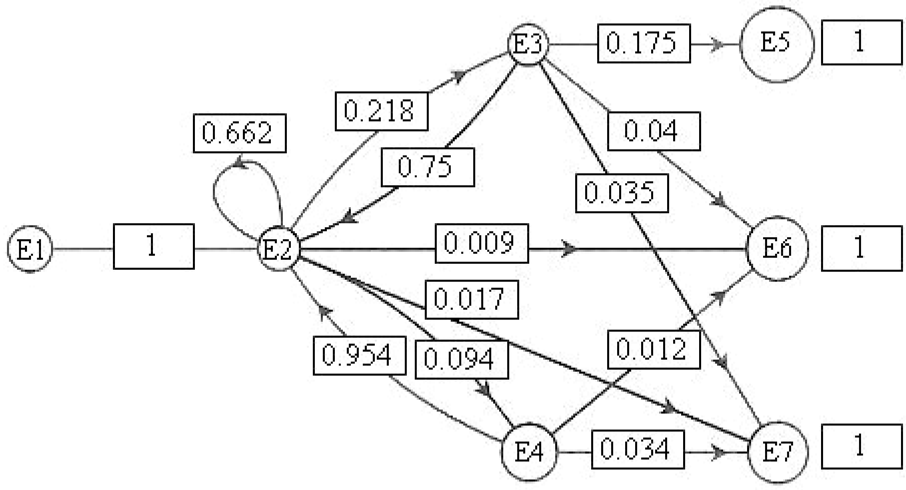
Et finalement au besoin voici le graph:


R Console
> plot(g)
>
R Graphics: Device 2 (ACTIVE)
10
11
7
8
9
12
6
5
2
1
4
3

Exemple 3.: Graphe de migration

R 3.0.2

Nous allons maintenant utiliser la matrice de transition de probabilités définie dans le cours théorique sur la théorie des graphes:



et correspondant à la matrice d'adjacence du graphe précédemment construit pour en faire une représentation graphique sous la forme d'un graphe de migration.

D'abord nous construisons la matrice de transition correspondante dans R:

```
R Console
> mat=t(matrix(c(0,0,0,0,0,0,0,
+ 0,0,0.094,0,0,0,0,
+ 1,0.954,0.662,0.75,0,0,0,
+ 0,0,0.218,0,0,0,0,
+ 0,0,0,0.175,1,0,0,
+ 0,0.034,0.017,0.035,0,1,0,
+ 0,0.012,0.009,0.04,0,0,1),nrow=7))
> noms<-c("E1","E2","E3","E4","E5","E6","E7")
> rownames(mat)<-noms
> colnames(mat)<-noms
> mat
   E1    E2    E3    E4 E5 E6 E7
E1  0 0.000 0.000 0.000  0  0  0
E2  0 0.000 0.094 0.000  0  0  0
E3  1 0.954 0.662 0.750  0  0  0
E4  0 0.000 0.218 0.000  0  0  0
E5  0 0.000 0.000 0.175  1  0  0
E6  0 0.034 0.017 0.035  0  1  0
E7  0 0.012 0.009 0.040  0  0  1
> N=dim(mat)[1]
> 
```

Une fois ceci fait, nous chargeons les packages **plyr**, **migest** et **circlize** et utilisons les **circos.par( )** , **chordDiagram( )** et **circos.trackPlotRegion( )** comme suite (ce n'est de loin pas évident à deviner):

```
R Console
> library(plyr)
> library(migest)
> library(circlize)
>
> circos.par(gap.degree = 5) #espace vide entre les arcs extérieurs
> chordDiagram(mat, grid.col = 1:N, directional = TRUE, annotationTrack = "grid",
+   preAllocateTracks = list(list(track.height = 0.05),
+                            list(track.height = 0.05)))
> circos.trackPlotRegion(track.index = 1, panel.fun = function(x, y) {
+   xlim = get.cell.meta.data("xlim")
+   ylim = get.cell.meta.data("ylim")
+   sector.index = get.cell.meta.data("sector.index")
+   circos.text(mean(xlim), mean(ylim), sector.index, facing = "inside", niceFacing = TRUE)
+ }, bg.border = NA)
> circos.trackPlotRegion(track.index =2, panel.fun = function(x, y) {
+   circos.axis("bottom", major.tick.percentage = 0.2, labels.cex = 0.4)
+ }, bg.border = NA)
> circos.clear()
>
```

Ce qui donne le graphe suivant qui après quelques secondes d'analyse est facilement à interpréter:

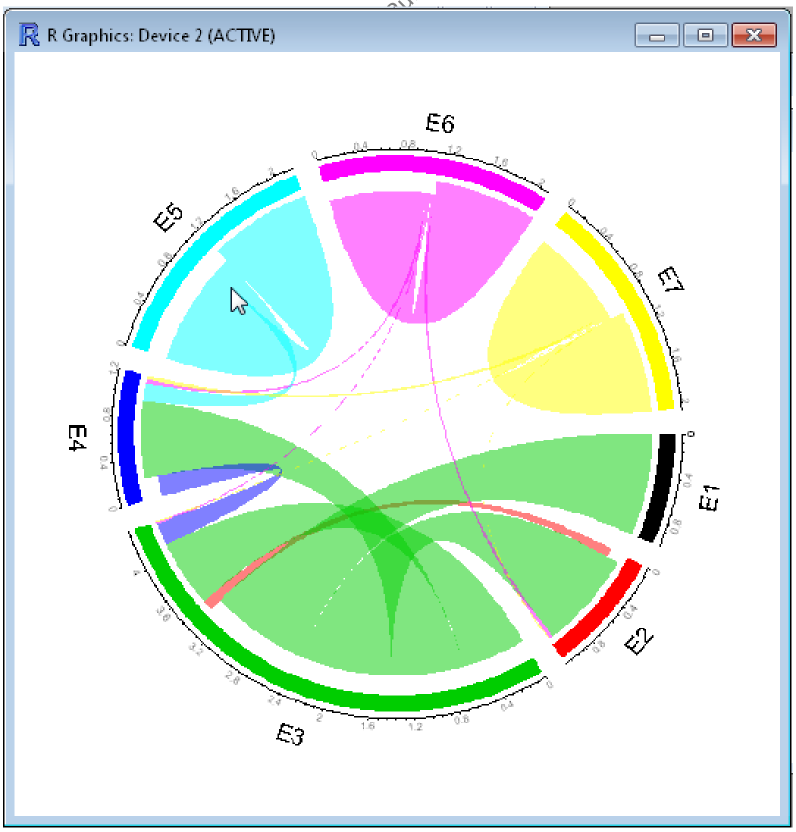
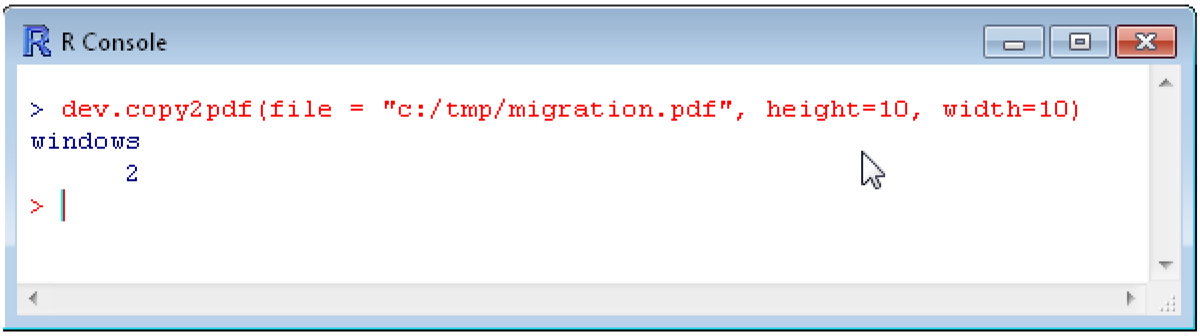
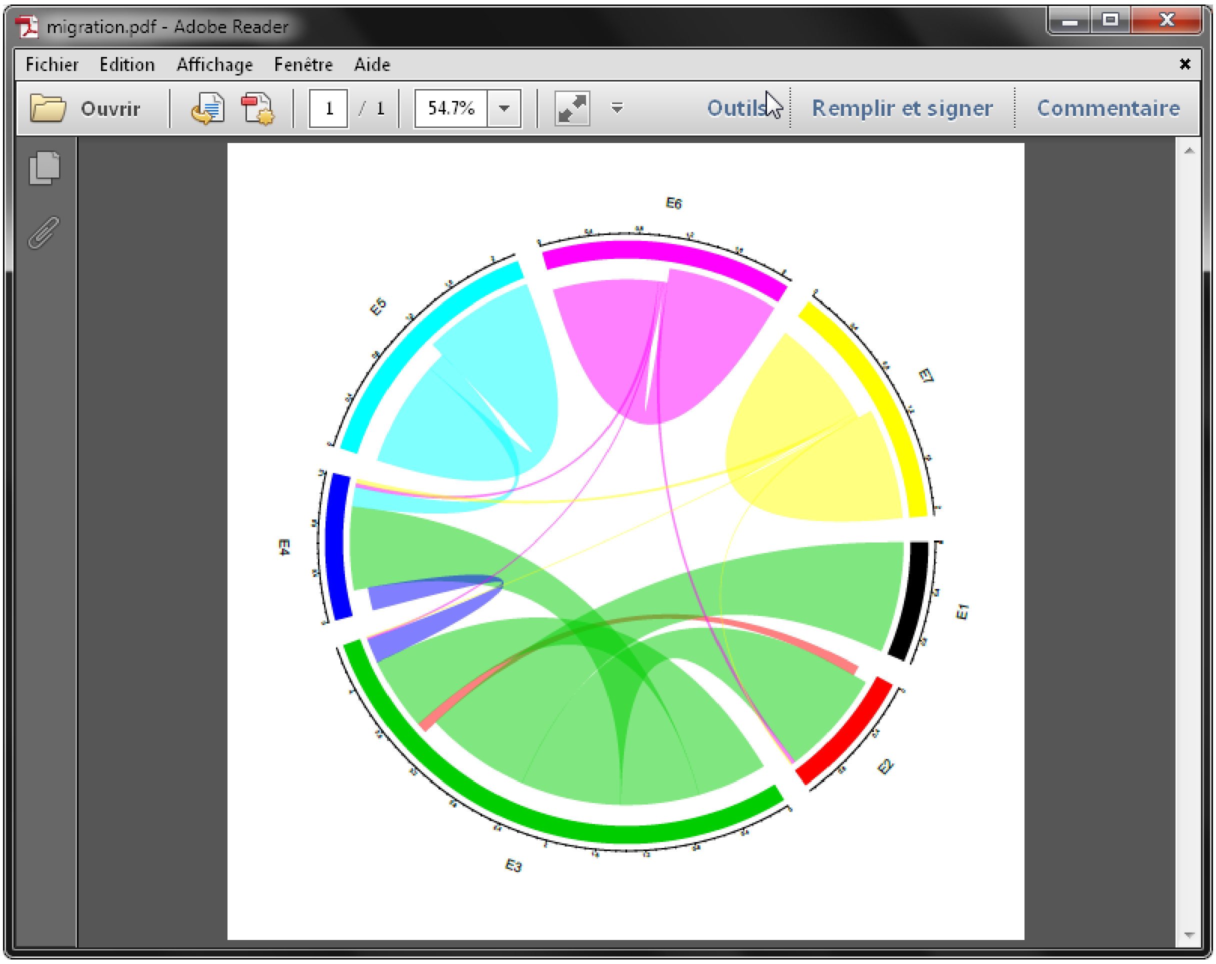


Figure 169 Graph de migration

La qualtié est mauvais il vaut donc mieux faire un export pour avoir un résultat HD:



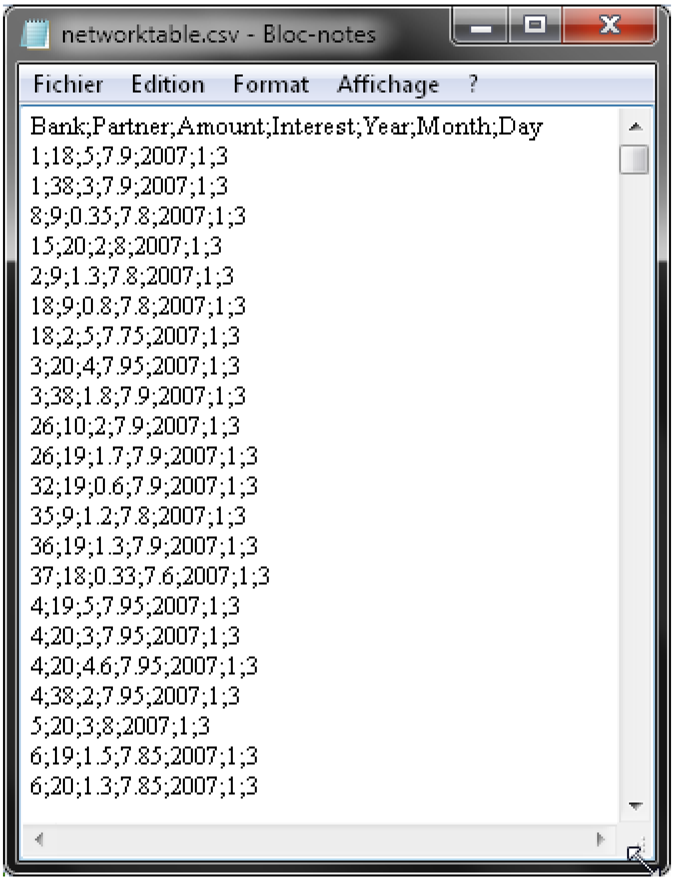
Ce qui donne:



Exemple 4.: Réseaux de relations

R 3.0.2

Considérons le fichier suivant:

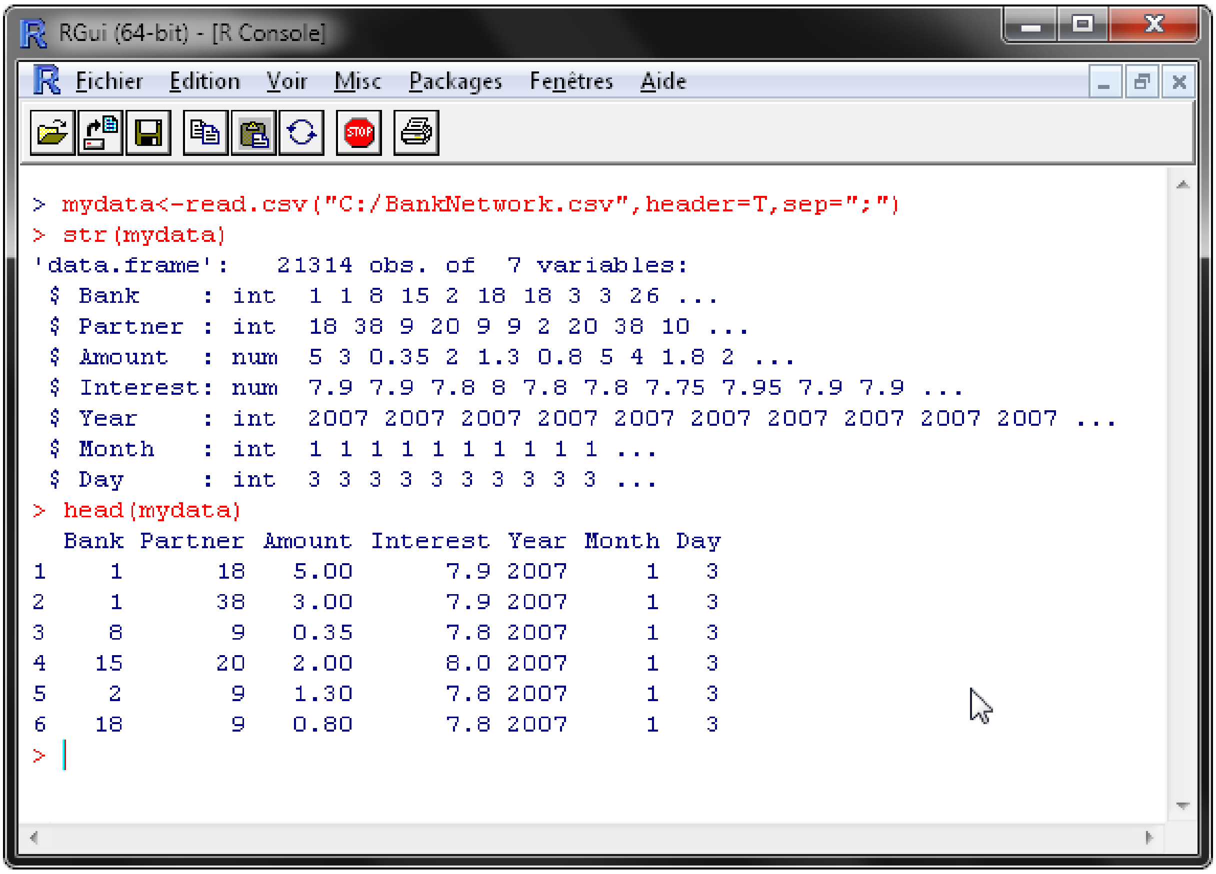


networktable.csv - Bloc-notes

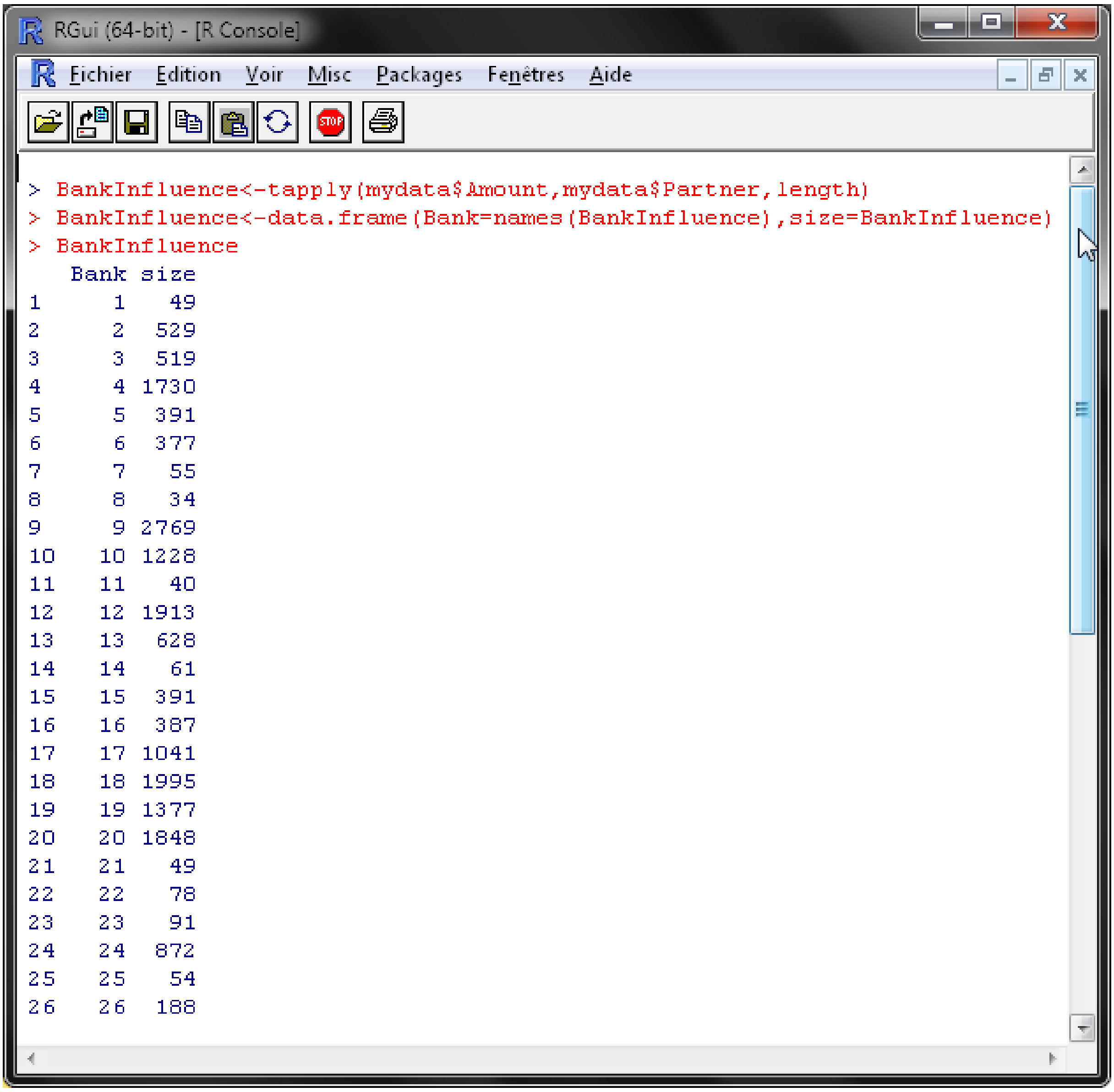
Fichier Edition Format Affichage ?

```
Bank;Partner;Amount;Interest;Year;Month;Day
1;18;5;7.9;2007;1;3
1;38;3;7.9;2007;1;3
8;9;0.35;7.8;2007;1;3
15;20;2;8;2007;1;3
2;9;1.3;7.8;2007;1;3
18;9;0.8;7.8;2007;1;3
18;2;5;7.75;2007;1;3
3;20;4;7.95;2007;1;3
3;38;1.8;7.9;2007;1;3
26;10;2;7.9;2007;1;3
26;19;1.7;7.9;2007;1;3
32;19;0.6;7.9;2007;1;3
35;9;1.2;7.8;2007;1;3
36;19;1.3;7.9;2007;1;3
37;18;0.33;7.6;2007;1;3
4;19;5;7.95;2007;1;3
4;20;3;7.95;2007;1;3
4;20;4.6;7.95;2007;1;3
4;38;2;7.95;2007;1;3
5;20;3;8;2007;1;3
6;19;1.5;7.85;2007;1;3
6;20;1.3;7.85;2007;1;3
```

De relations interbancaires. Nous le chargeons dans R:

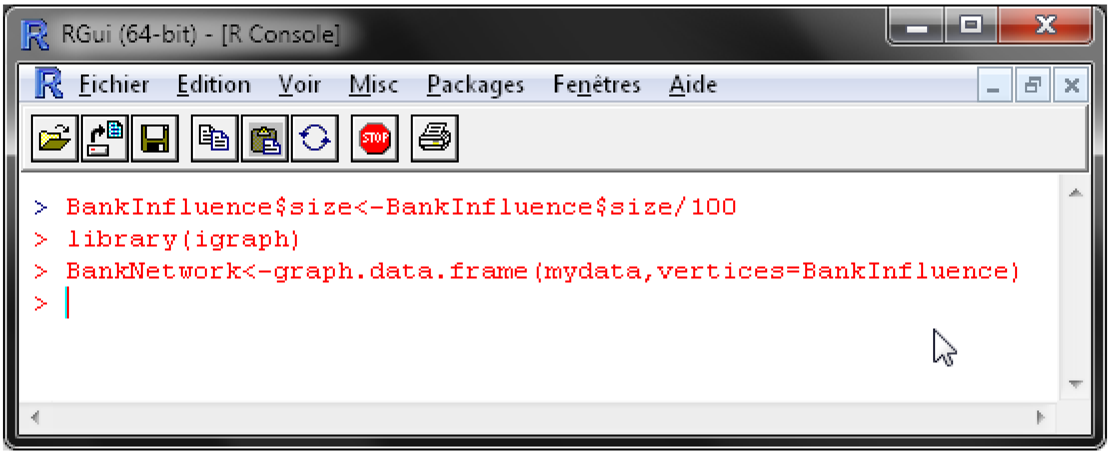


Nous faisons la synthèse sur les partenaires car toutes les banques sont listées dans les partenaires et nous comptons combien de fois chacun apparaît en tant que partenaire ainsi les plus importantes seront les plus représentatives sur le graphe final:



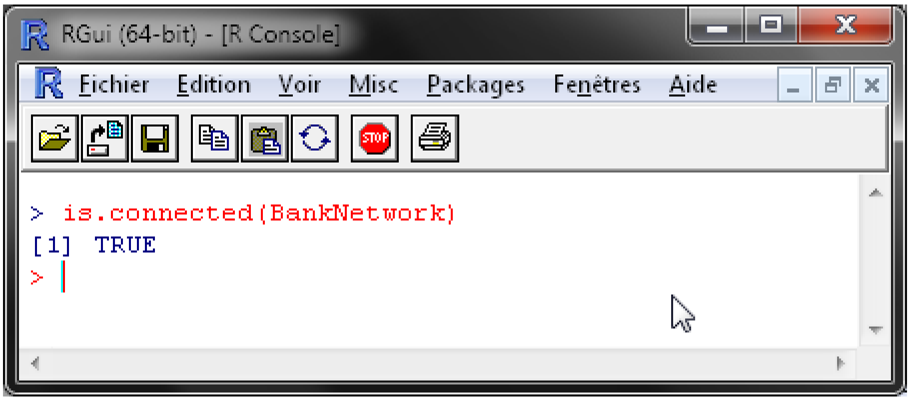
```
> BankInfluence<-tapply(mydata$Amount,mydata$Partner,length)
> BankInfluence<-data.frame(Bank=names(BankInfluence),size=BankInfluence)
> BankInfluence
   Bank size
1     1   49
2     2  529
3     3  519
4     4 1730
5     5  391
6     6  377
7     7   55
8     8   34
9     9 2769
10   10 1228
11   11   40
12   12 1913
13   13  628
14   14   61
15   15  391
16   16  387
17   17 1041
18   18 1995
19   19 1377
20   20 1848
21   21   49
22   22   78
23   23   91
24   24  872
25   25   54
26   26  188
```

Nous normalisons cela de façon empirique afin d'avoi une taille convenable sur le graph final (normalisation faite par essais successifs) et nous construisons à l'aide de la commande **graph.data.frame( )** du package **igraph** l'objet réseau qui sera utilisé un peu plus loin pour le graph:



```
> BankInfluence$size<-BankInfluence$size/100
> library(igraph)
> BankNetwork<-graph.data.frame(mydata,vertices=BankInfluence)
>
```

Nous pouvons ensuite vérifier si par hasard le graphe est totalement connexe (si chaque banque est liée à toutes les autres banques, ou autrement dit si chaque banque à fait affaire avec toutes les autres banques au moins une fois):



Ensuite nous faisons le plot:

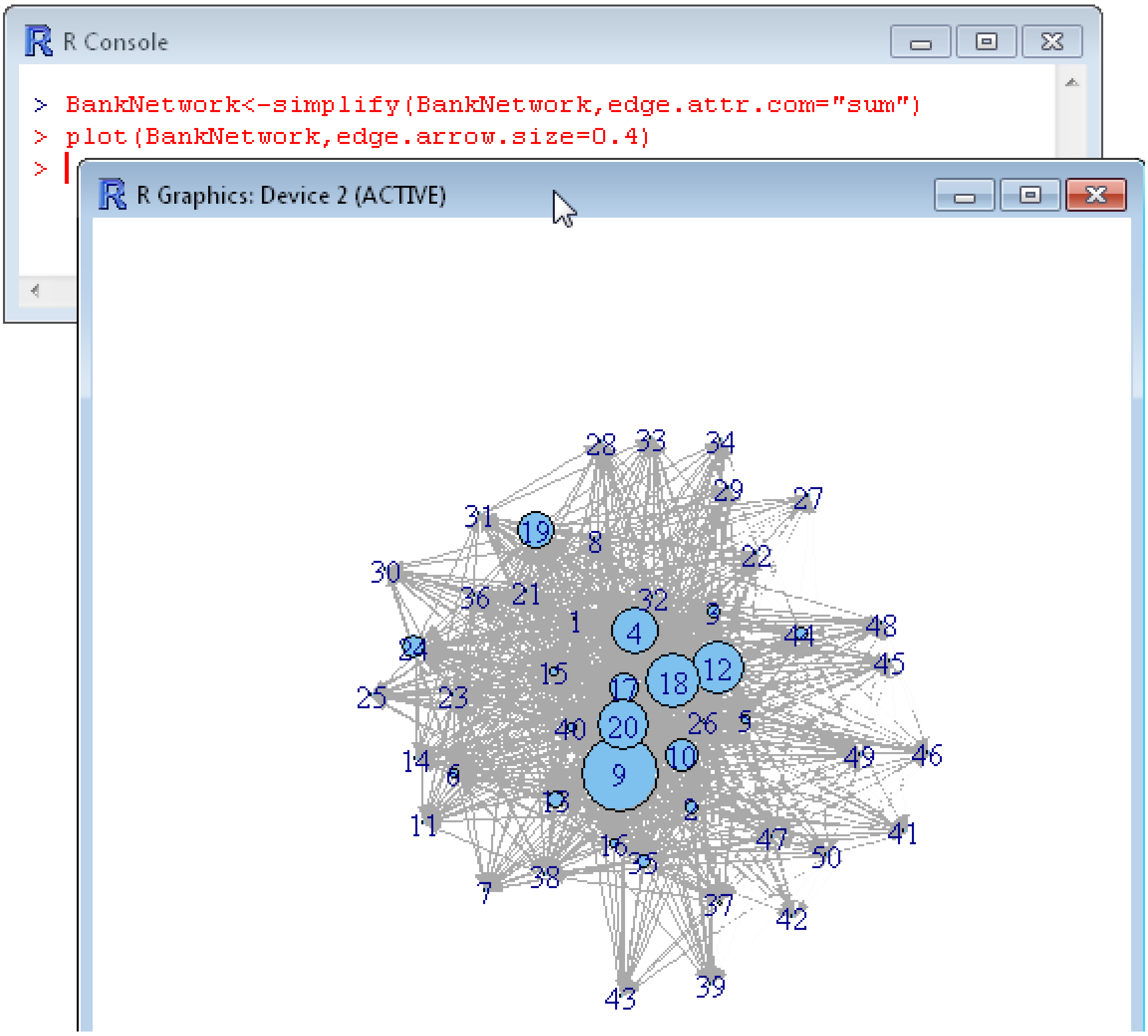
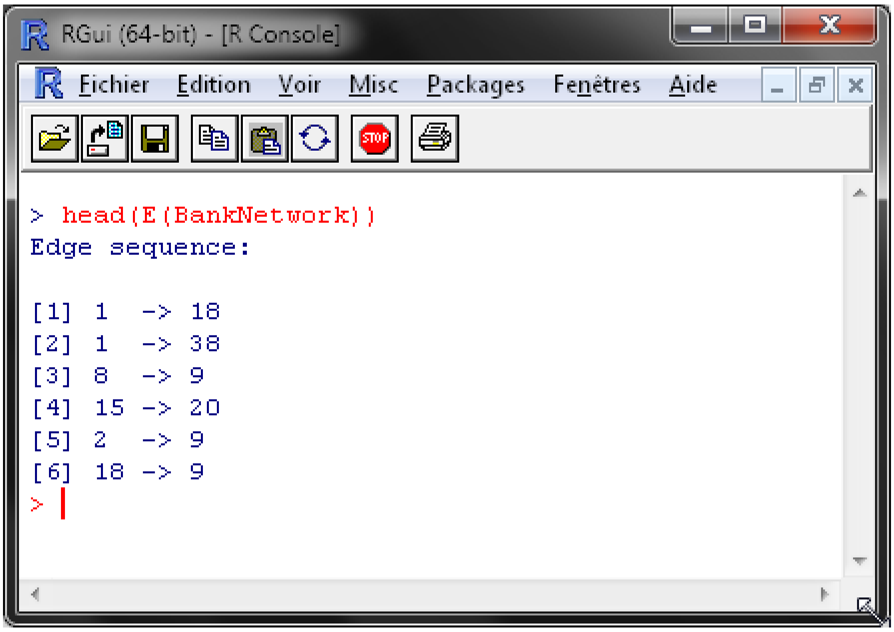
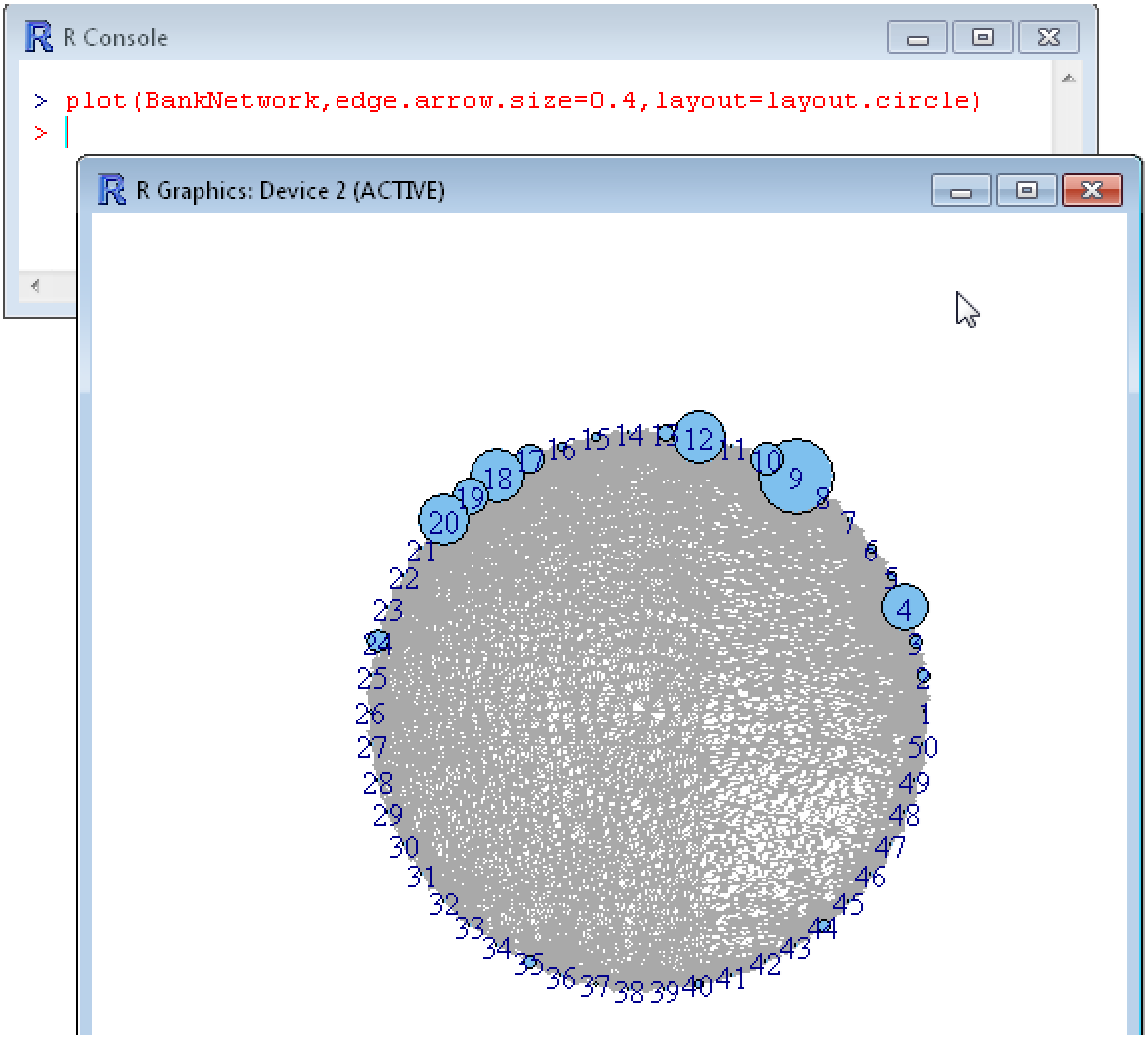


Figure 170 Graph de Réseau (relations)

Nous pouvons obtenir la liste des relations avec la commande **E( )** (pour "Edges"):



Nous pouvons faire de même en cercle mais comme il y a des milliers de relations dans le cas présent ce n'est évidemment guère lisible...:



Nous pouvons également regrouper certains éléments par couleurs mais lorsqu'il y a beaucoup de liaisons ces dernières sont difficilement visibles:

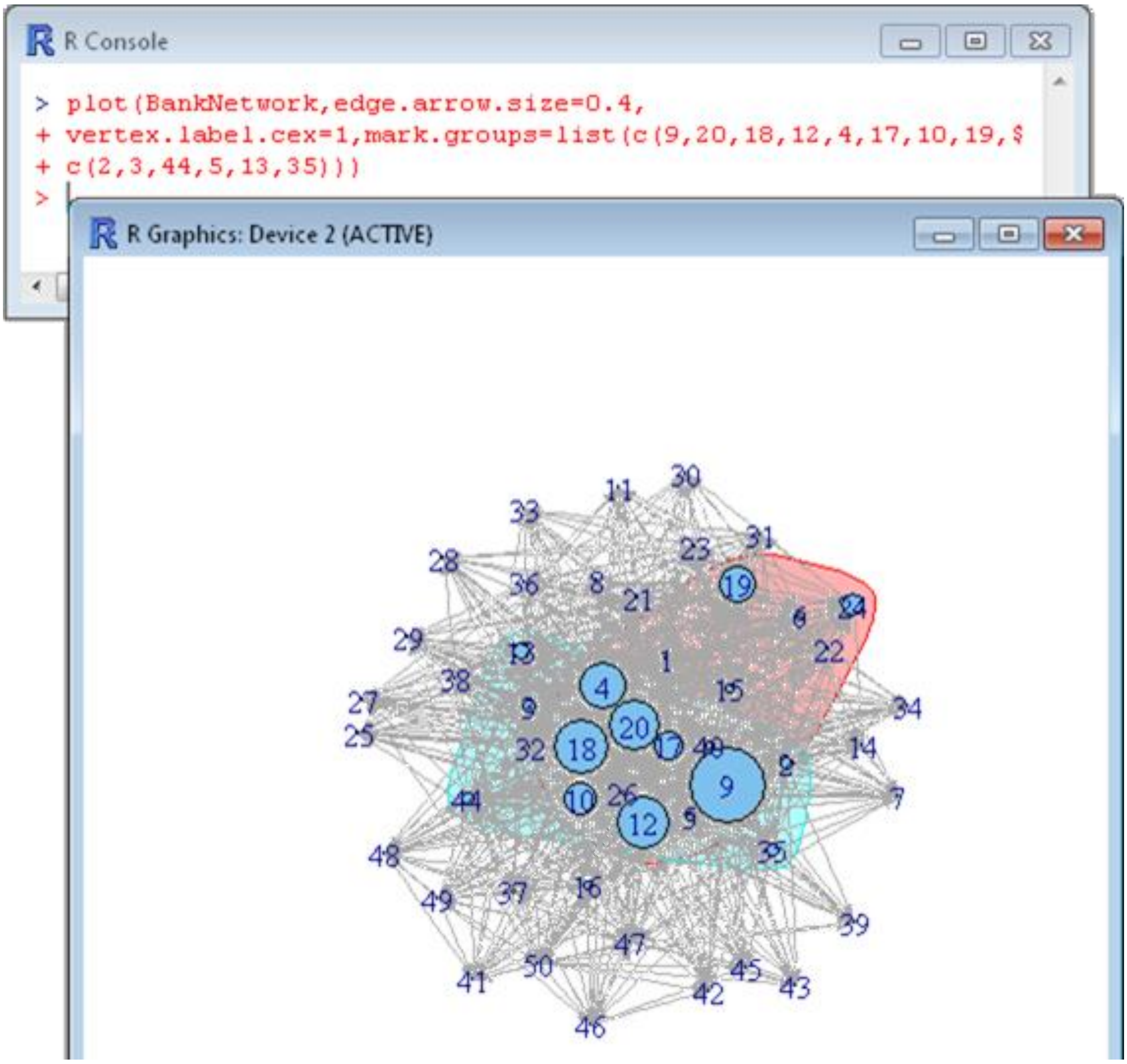
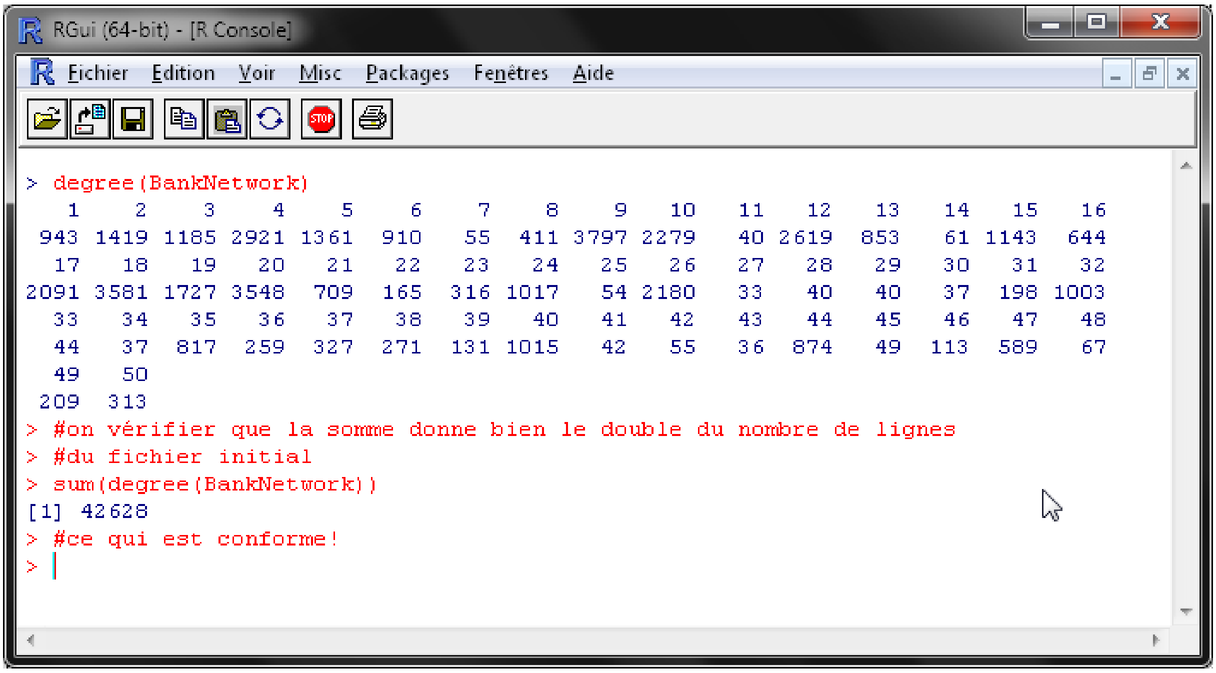
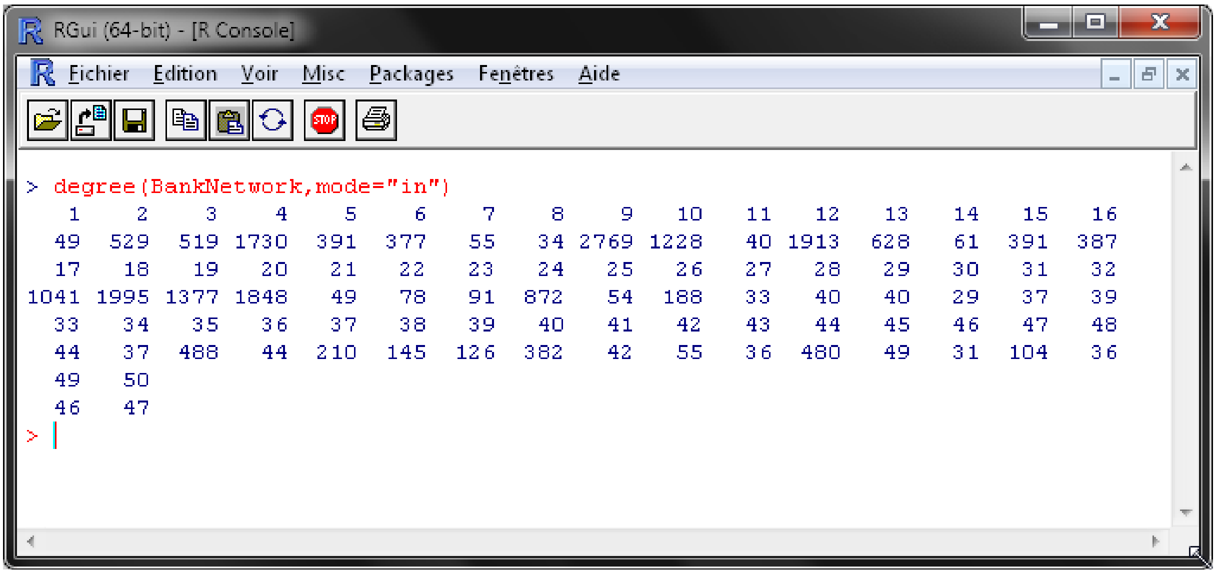


Figure 171 Graph de réseau avec clustering

Nous pouvons ensuite faire quelques petits calculs élémentaires. Par exemple pour chaque banque, connaître le nombre de liens entrants+sortants avec la commande **degree( )** :



On peut avoir que les liens entrants (et respectivement sortant en mettant *mode="out"*):



Exemple 5.: Réseaux de relations dynamique

R 3.0.2

En matière d'exploration, ce que nous avons vu précédemment n'est guère convivial et dynamique, et ce particulièrement pour les gens travaillant dans le marketing, les relations politiques internationales, la crimonologie ou le data journalisme.

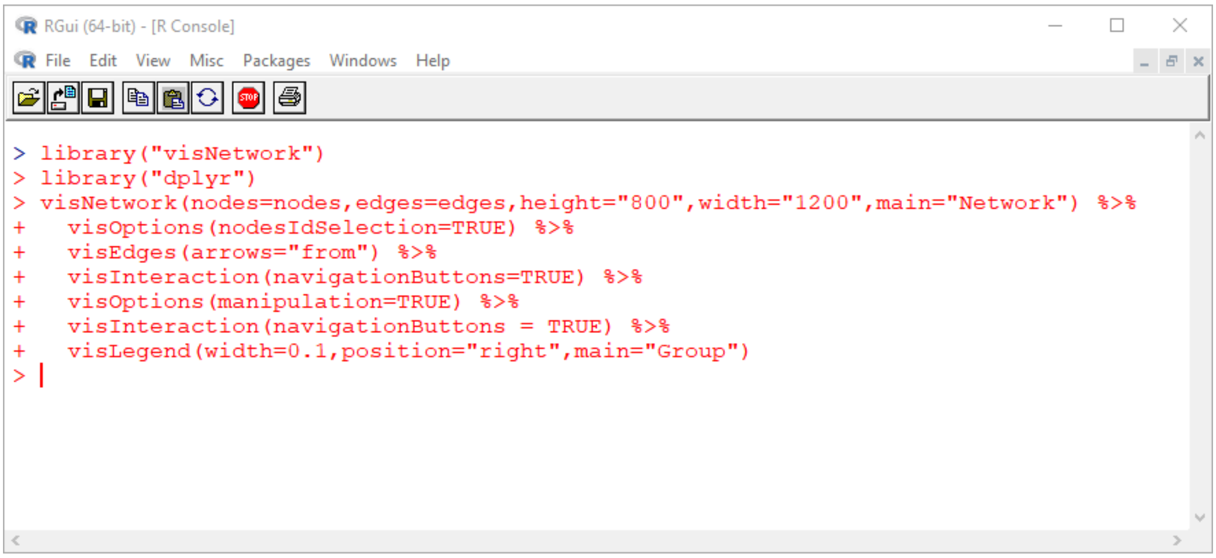
D'abord voyons un échantillon de la structure des deux jeux de données qui vont être utilisés dans cet exemple! Il y a d'abord:

```
> load(file="DynamicNetwork.RData")
> edges
            from            to width
1         Myriel      Napoleon     1
2         Myriel MlleBaptistine    8
3         Myriel   MmeMagloire    10
4 MlleBaptistine   MmeMagloire     6
5         Myriel  CountessDeLo     1
6         Myriel      Geborand     1
7         Myriel  Champtercier     1
8         Myriel      Cravatte     1
9         Myriel         Count     2
10        Myriel        OldMan     1
11       Labarre       Valjean     1
12   MmeMagloire       Valjean     3
13 MlleBaptistine      Valjean     3
14        Myriel       Valjean     5
15       Valjean    Marguerite     1
16       Valjean        MmeDeR     1
17       Valjean       Isabeau     1
18       Valjean       Gervais     1
19     Tholomyes     Listolier     4
20     Tholomyes       Fameuil     4
21     Listolier       Fameuil     4
22     Tholomyes   Blacheville     4
23     Listolier   Blacheville     4
24       Fameuil   Blacheville     4
25     Tholomyes     Favourite     3
26     Listolier     Favourite     3
27       Fameuil     Favourite     3
28   Blacheville     Favourite     4
29     Tholomyes        Dahlia     3
30     Listolier        Dahlia     3
31       Fameuil        Dahlia     3
32   Blacheville        Dahlia     3
33     Favourite        Dahlia     5
34     Tholomyes       Zephine     3
35     Listolier       Zephine     3
36       Fameuil       Zephine     3
37   Blacheville       Zephine     3
38     Favourite       Zephine     4
39        Dahlia       Zephine     4
40     Tholomyes       Fantine     3
41     Listolier       Fantine     3
42       Fameuil       Fantine     3
43   Blacheville       Fantine     3
44     Favourite       Fantine     4
45        Dahlia       Fantine     4
46       Zephine       Fantine     4
47    Marguerite       Fantine     2
48       Valjean       Fantine     9
```

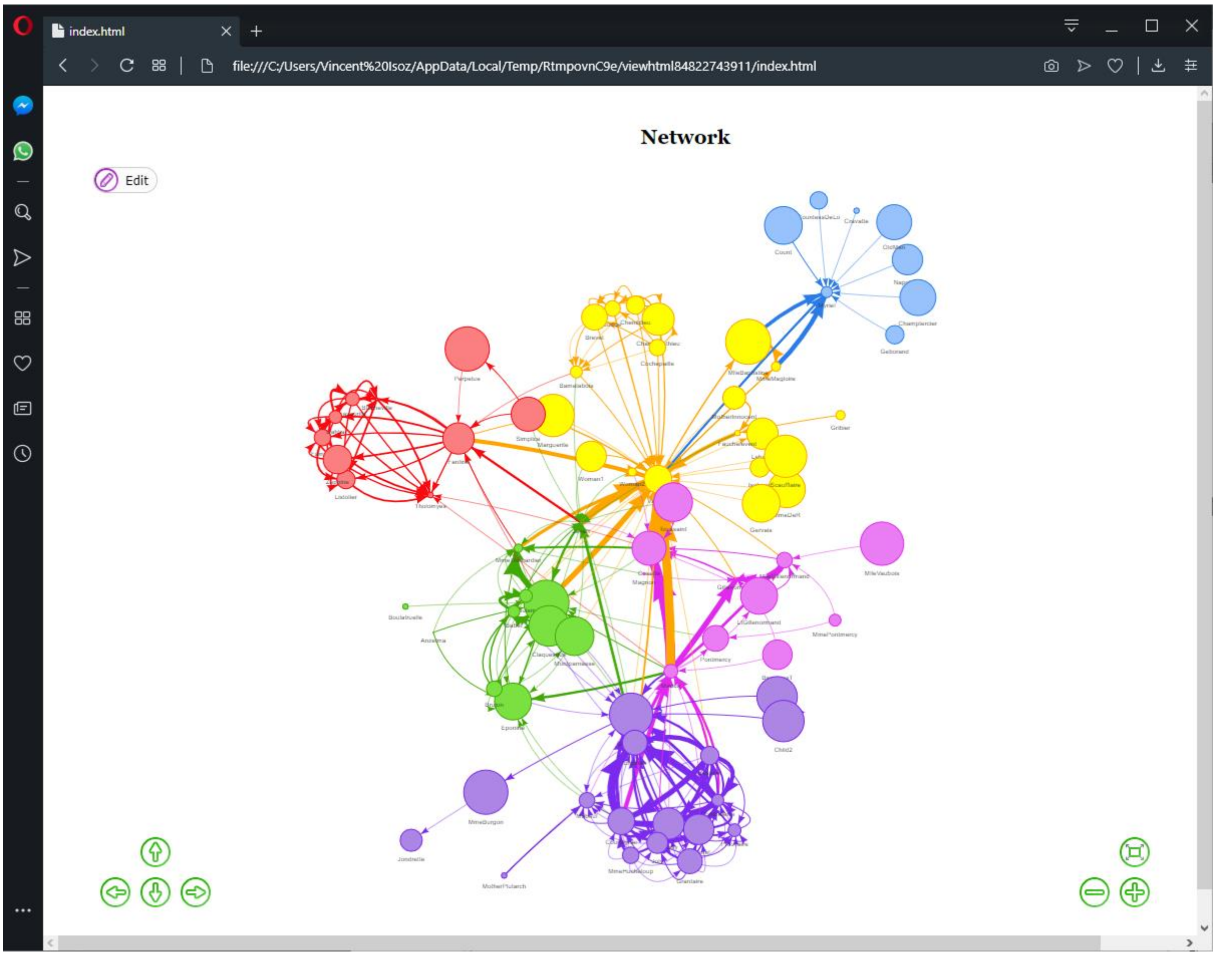
Et:

```
> nodes
               id            label group size
1          Myriel           Myriel     2   12
2        Napoleon         Napoleon     2   33
3  MlleBaptistine   MlleBaptistine     1   49
4     MmeMagloire      MmeMagloire     1   10
5    CountessDeLo     CountessDeLo     2   19
6        Geborand         Geborand     2   20
7    Champtercier     Champtercier     2   39
8        Cravatte         Cravatte     2    6
9           Count            Count     2   41
10         OldMan           OldMan     2   38
11        Labarre          Labarre     1   34
12        Valjean          Valjean     1   30
13     Marguerite       Marguerite     1   46
14         MmeDeR           MmeDeR     1   40
15        Isabeau          Isabeau     1   21
16        Gervais          Gervais     1   41
17      Tholomyes        Tholomyes     3    6
18      Listolier        Listolier     3   20
19        Fameuil          Fameuil     3   18
20    Blacheville      Blacheville     3    2
21      Favourite        Favourite     3   15
22         Dahlia           Dahlia     3   14
23        Zephine          Zephine     3   31
24        Fantine          Fantine     3   34
25  MmeThenardier    MmeThenardier     6    9
26     Thenardier       Thenardier     6   49
27        Cosette          Cosette     4   37
28         Javert           Javert     6    2
29   Fauchelevent     Fauchelevent     1    6
30     Bamatabois       Bamatabois     1   13
31       Perpetue         Perpetue     3   48
32       Simplice         Simplice     3   37
33    Scaufflaire      Scaufflaire     1   46
34         Woman1           Woman1     1   33
35          Judge            Judge     1   17
36   Champmathieu     Champmathieu     1   35
37         Brevet           Brevet     1   28
38     Chenildieu       Chenildieu     1   20
39    Cochepaille      Cochepaille     1   18
40      Pontmercy        Pontmercy     4   28
41   Boulatruelle     Boulatruelle     6    6
42        Eponine          Eponine     6   40
43        Anzelma          Anzelma     6    1
44         Woman2           Woman2     1    9
45 MotherInnocent   MotherInnocent     1   25
46        Gribier          Gribier     1   10
47      Jondrette        Jondrette     5   24
48      MmeBurgon        MmeBurgon     5   48
49       Gavroche         Gavroche     5   47
```

Maintenant, voyons donc ce qu'est capable de faire le package **visNetwork** et sa fonction du même nom **visNetwork( )**:



Ce qui ouvrira dans le navigateur:



Exemple 151.: Carte de chaleur (heatmap)

R 3.0.2

Nous allons voir ici si avec R nous retrouvons le même dendrogramme que dans le cours Minitab en utilisant le même jeu de données (rappelez-vous qu'avec Minitab nous avions obtenu avec la distance euclidienne les mêmes résultats que ceux calculés à la main dans le cours théorique):

```
Dendrogramme.csv - Bloc-notes
Fichier  Edition  Format  Affichage  ?
Observation;Revenus;Surface;Propriétaire
1;60;18.4;1
2;85.5;16.8;1
3;64.8;21.6;1
4;61.5;20.8;1
5;87;23.6;1
6;110.1;19.2;1
7;108;17.6;1
8;82.8;22.4;1
9;69;20;1
10;93;20.8;1
11;51;22;1
12;81;20;1
13;75;19.6;2
14;52.8;20.8;2
15;64.8;17.2;2
16;43.2;20.4;2
17;84;17.6;2
18;49.2;17.6;2
19;59.4;16;2
20;66;18.4;2
21;47.4;16.4;2
22;33;18.8;2
23;51;14;2
24;63;14.8;2
```

La commande **heatmap( )** et le résultat associé sont donnés par (voir page suivante):

```
R Console
> mydata<-read.csv("T:/R_Charts/Graphs/Dendrogramme.csv",header=T,sep=";")
> heatmap(as.matrix(mydata[,2:3]),distfun=function(x) dist(x,method='euclidean'),
+ Colv=NA,col=cm.colors(256),scale="column",xlab="Specification variables",
+ ylab="Observation number",main="Heatmap",cexCol=1)
> 
```

Ce qui donne (voir page suivante):

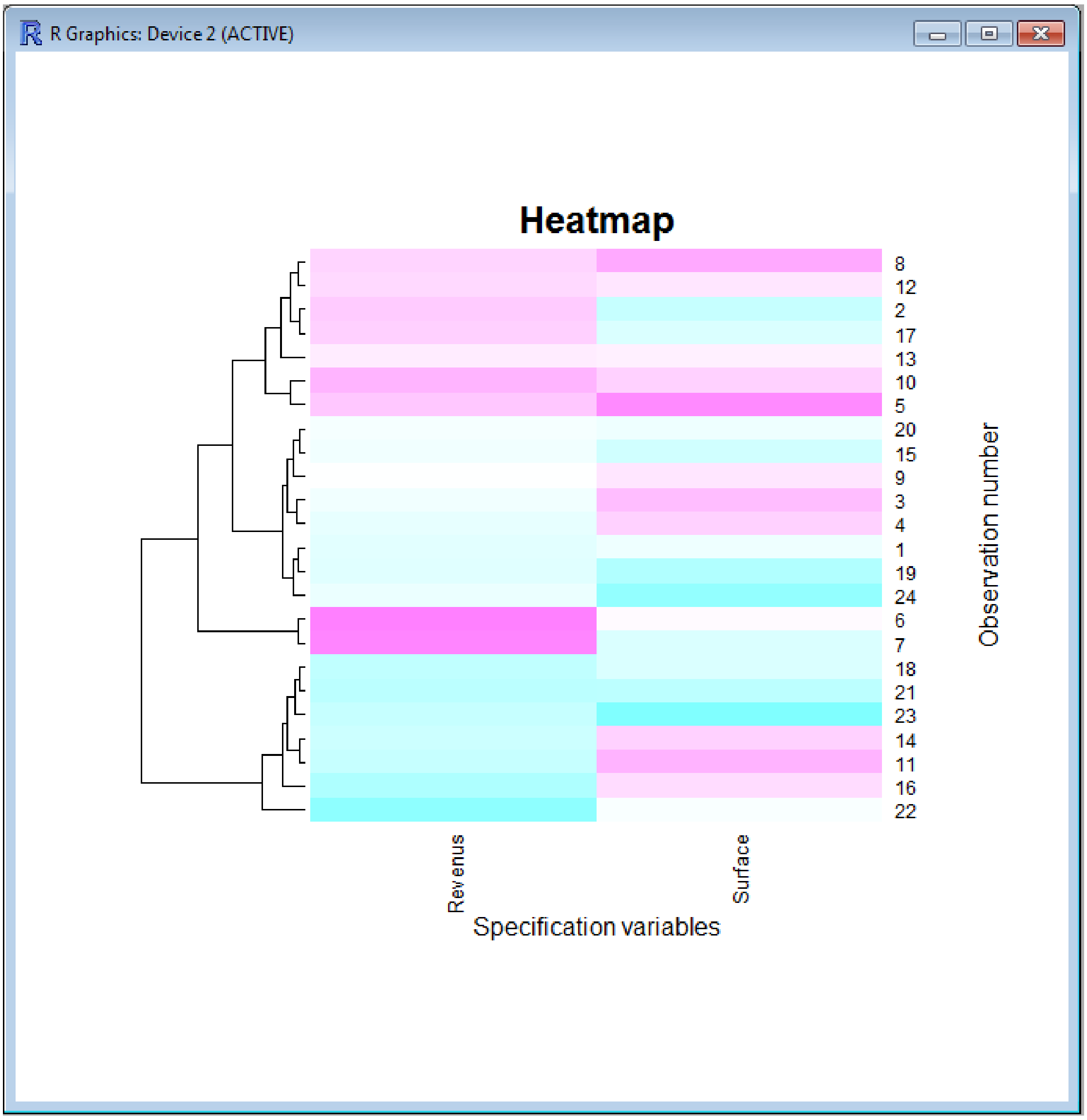
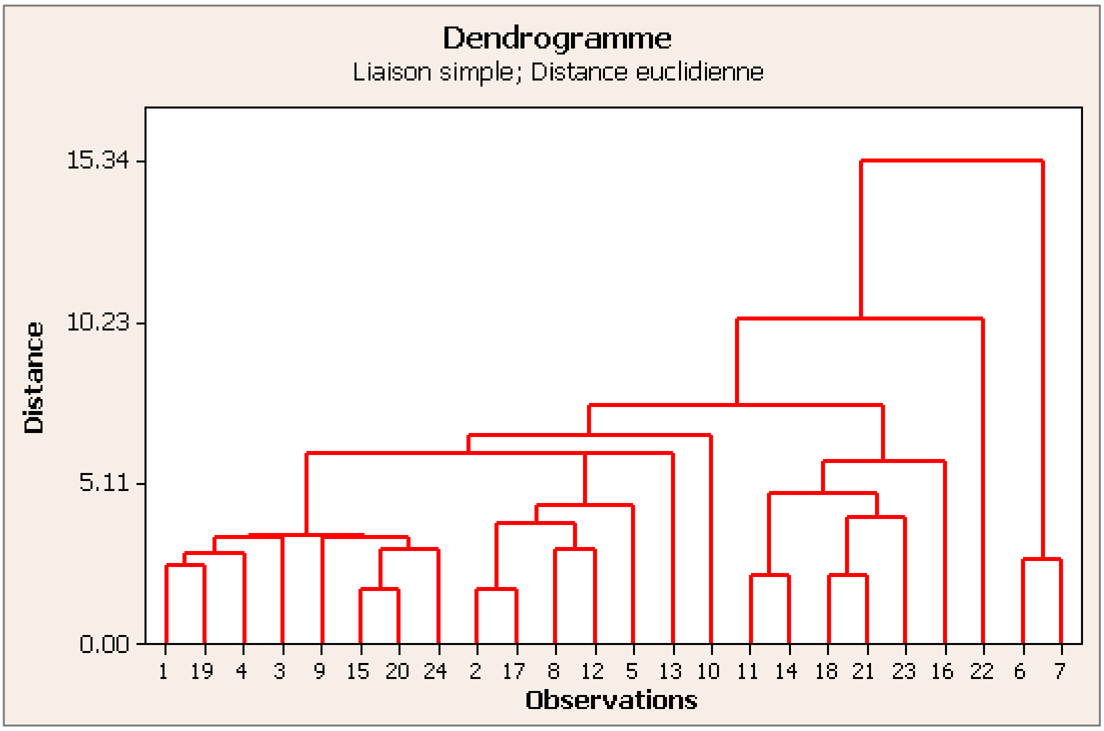


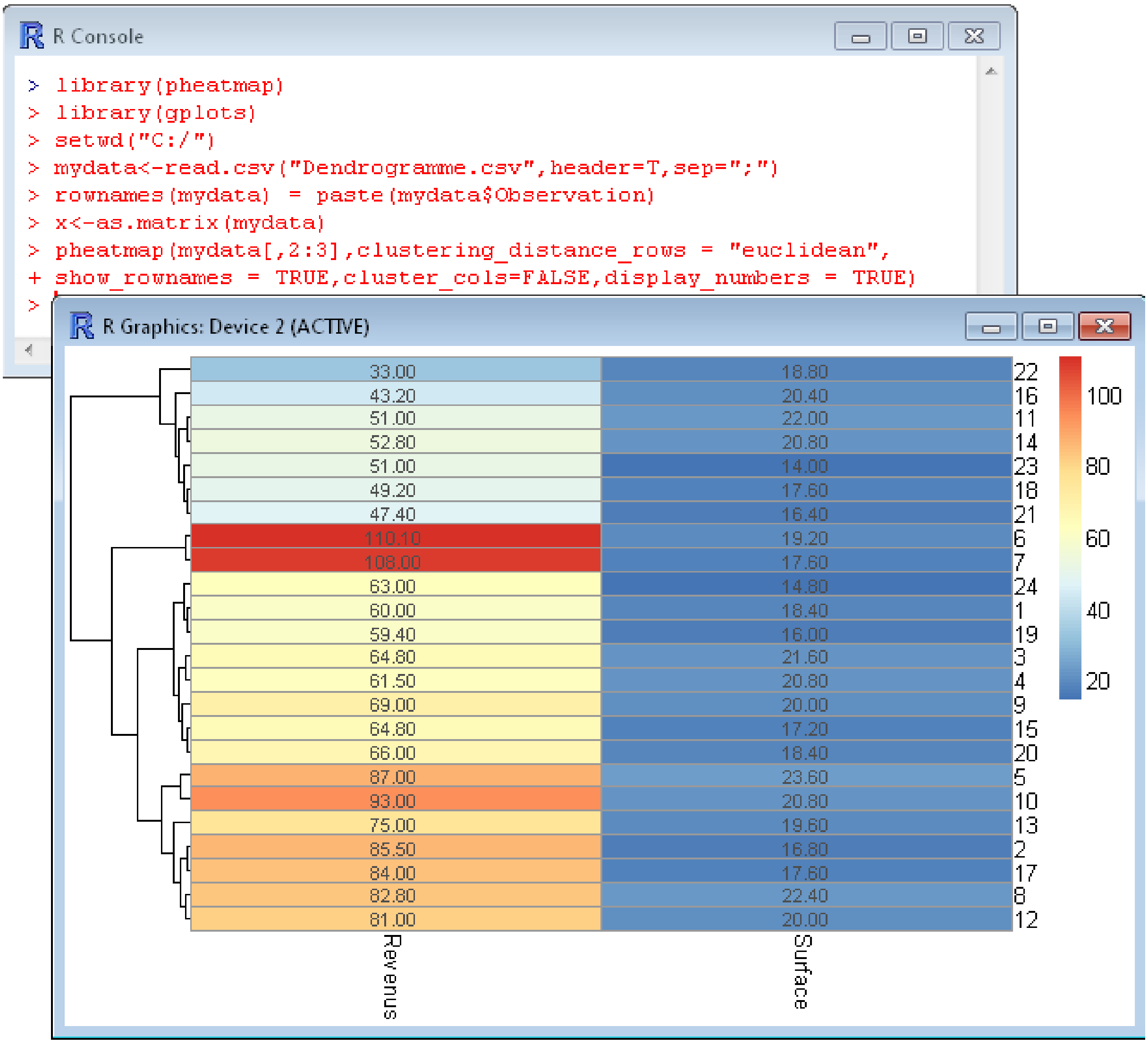
Figure 172 Graph dendrogramme avec carte de chaleur

Le résultat est esthétiquement discutable… et si nous comparons à Minitab:



Nous n'obtenons pas la même chose! Nous verrons cependant dans le chapitre data mining lors de l'exemple du CAH quels sont les paramètres à mettre dans R pour retomber sur ce que nous avons démontré dans le cours théorique.

Cependant on peut faire mieux que l'horrible commande **heatmap( )** qui consiste à utiliser la commande **pheatmap( )** en association avec les packages **pheatmap** et **gplots**:



ou dans un style beaucoup plus sobre et simple:

Sinon nous pouvons revenir à du plus classique avec les packages **ggplot2** et **ggdendro** et en utilisant les commandes **hclust( )** et **ggdendrogram( )** :

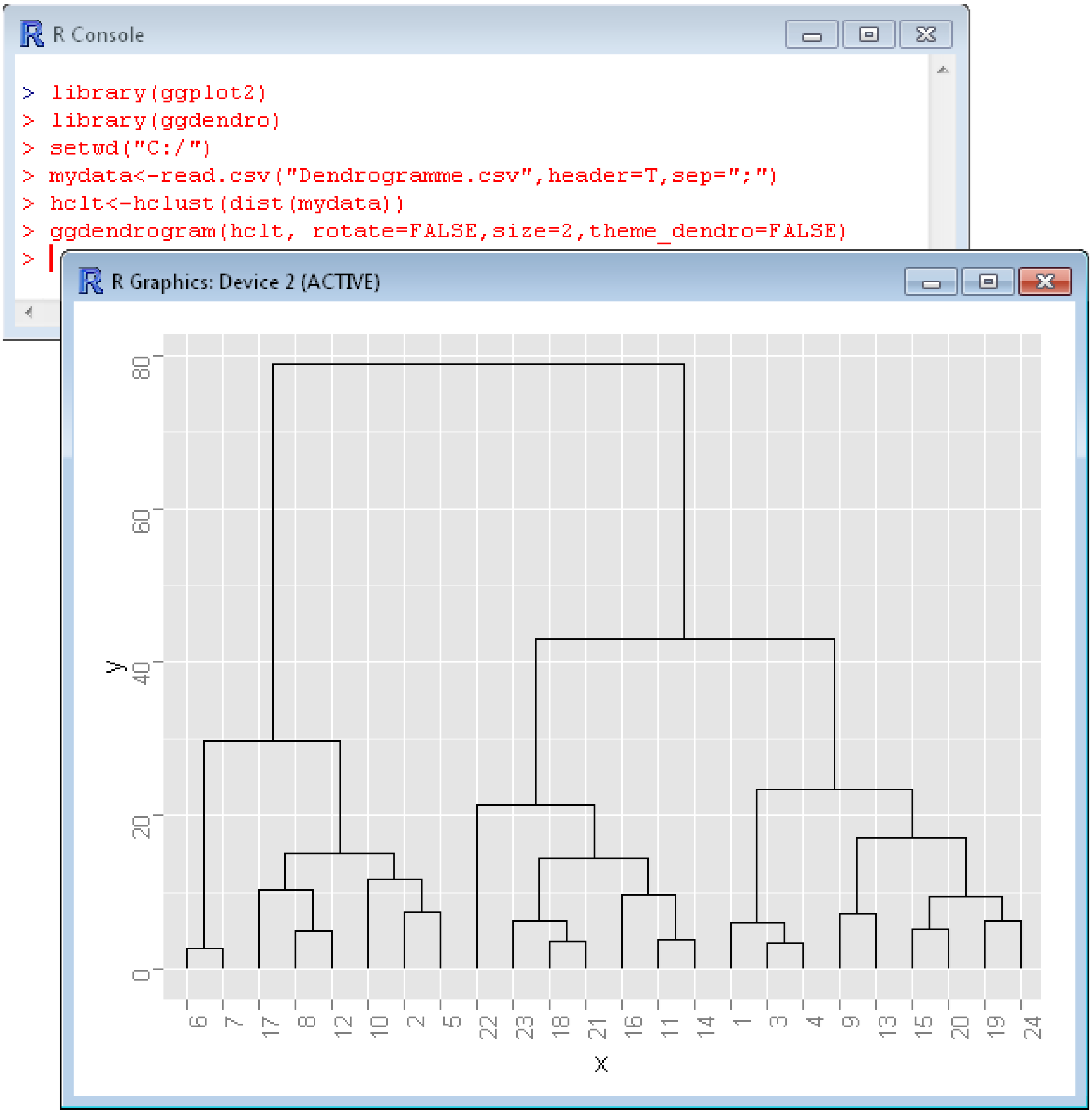
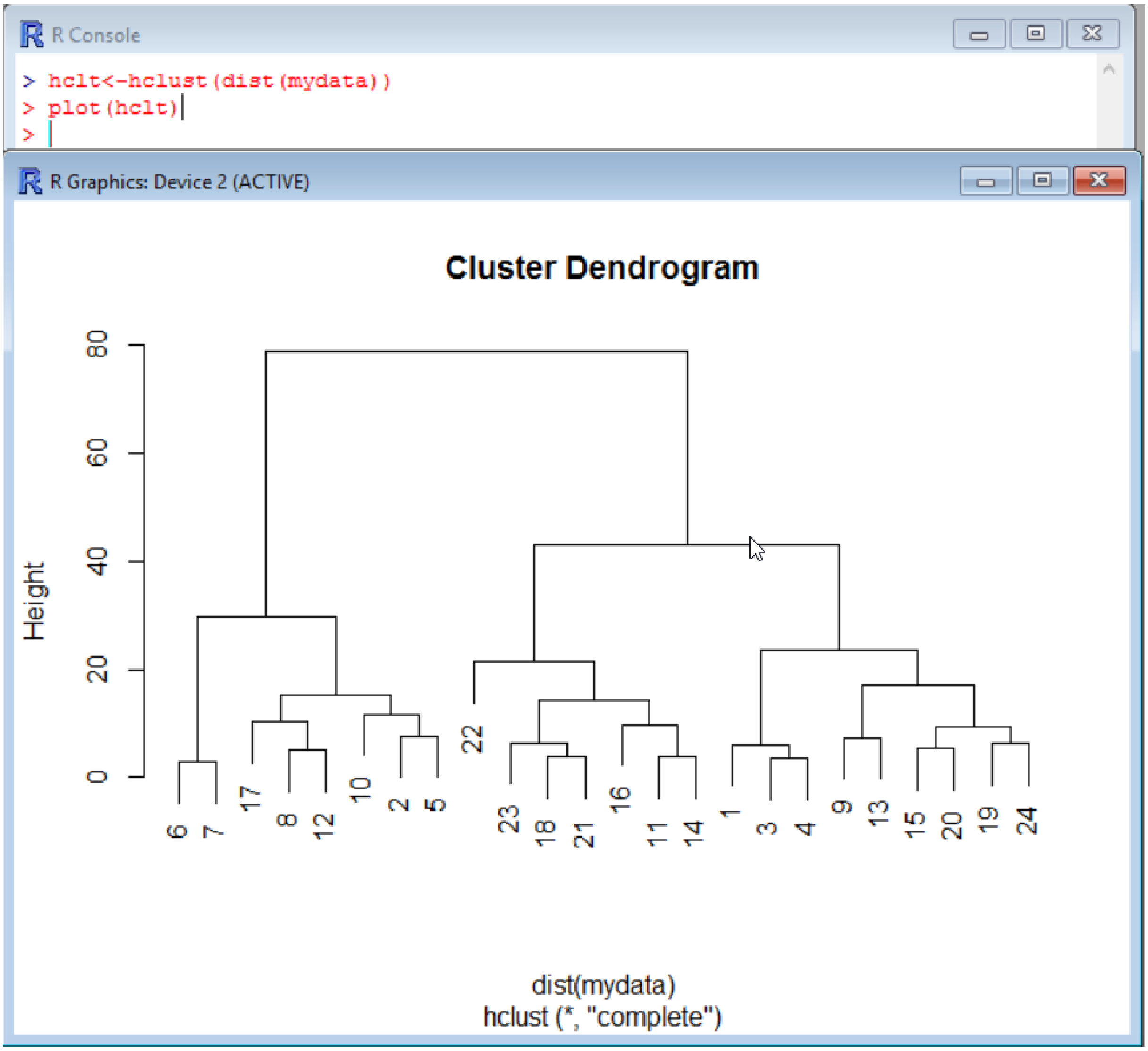
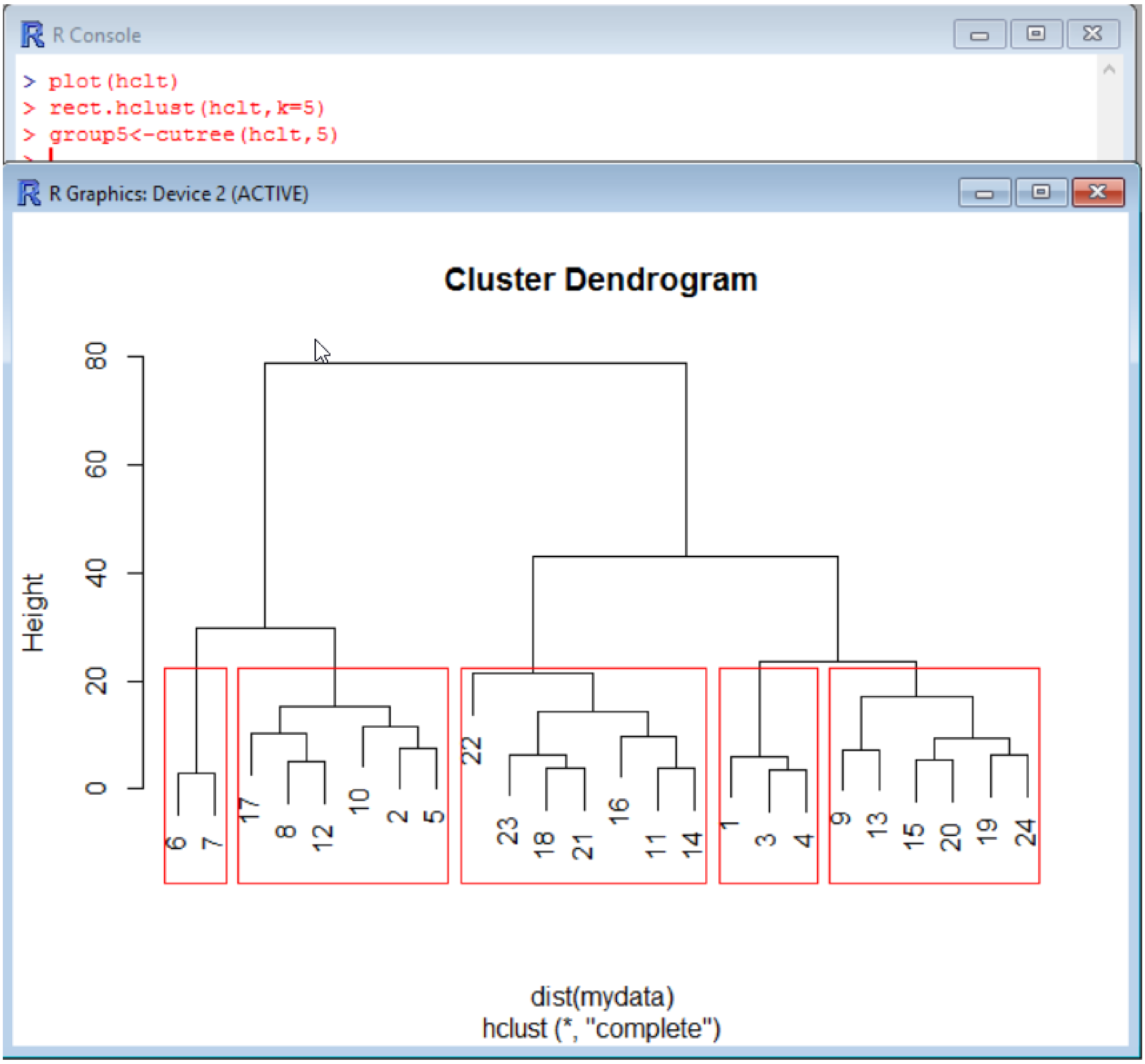


Figure 173 Graph de type dendrogramme (simple)

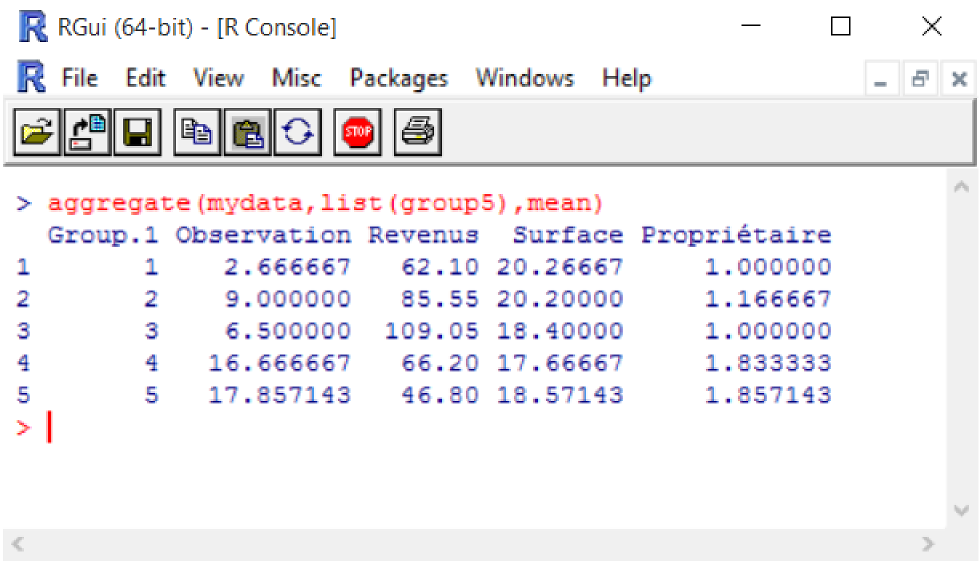
ou encore sans packages:



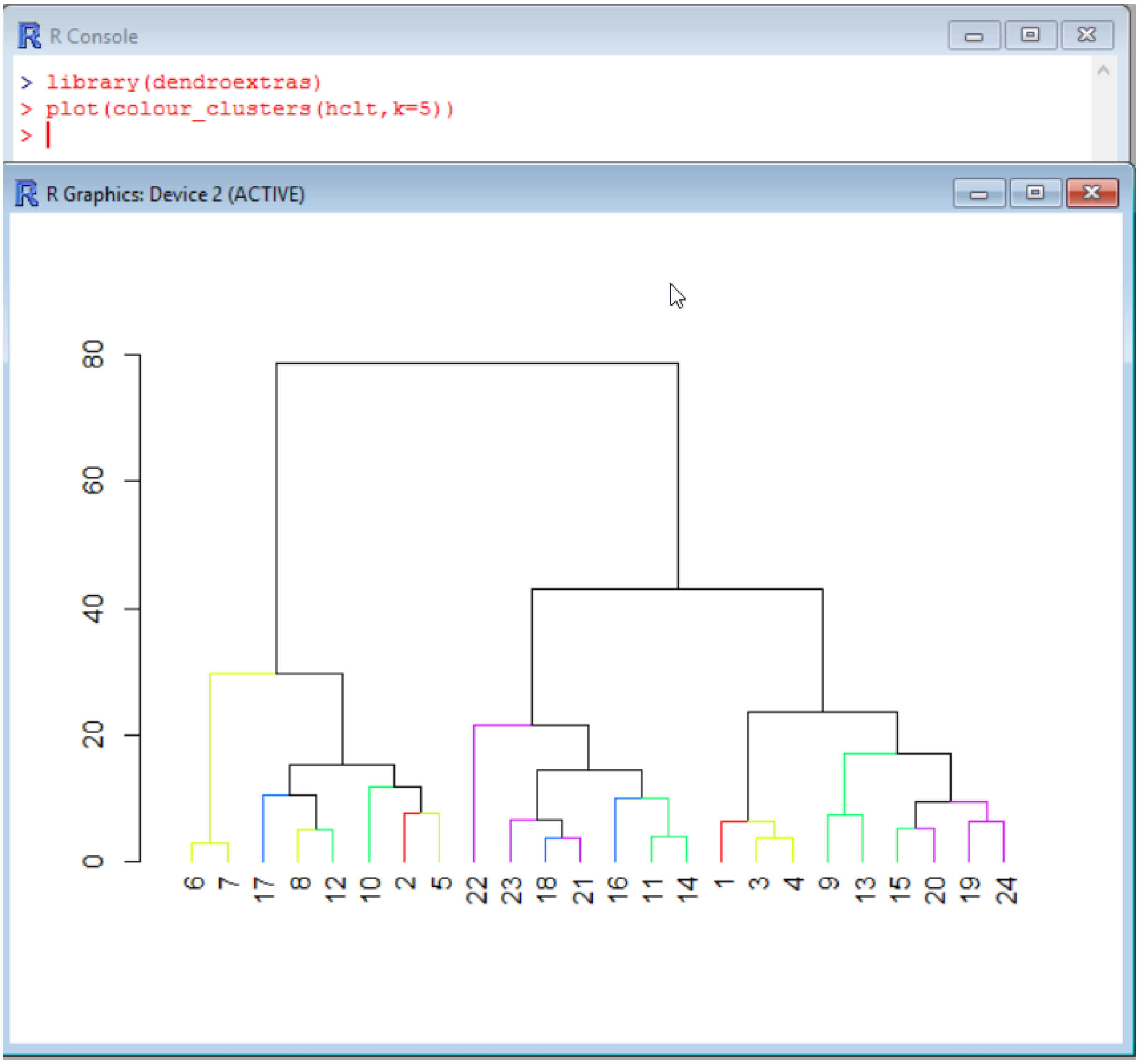
et toujours sans package additionnels:



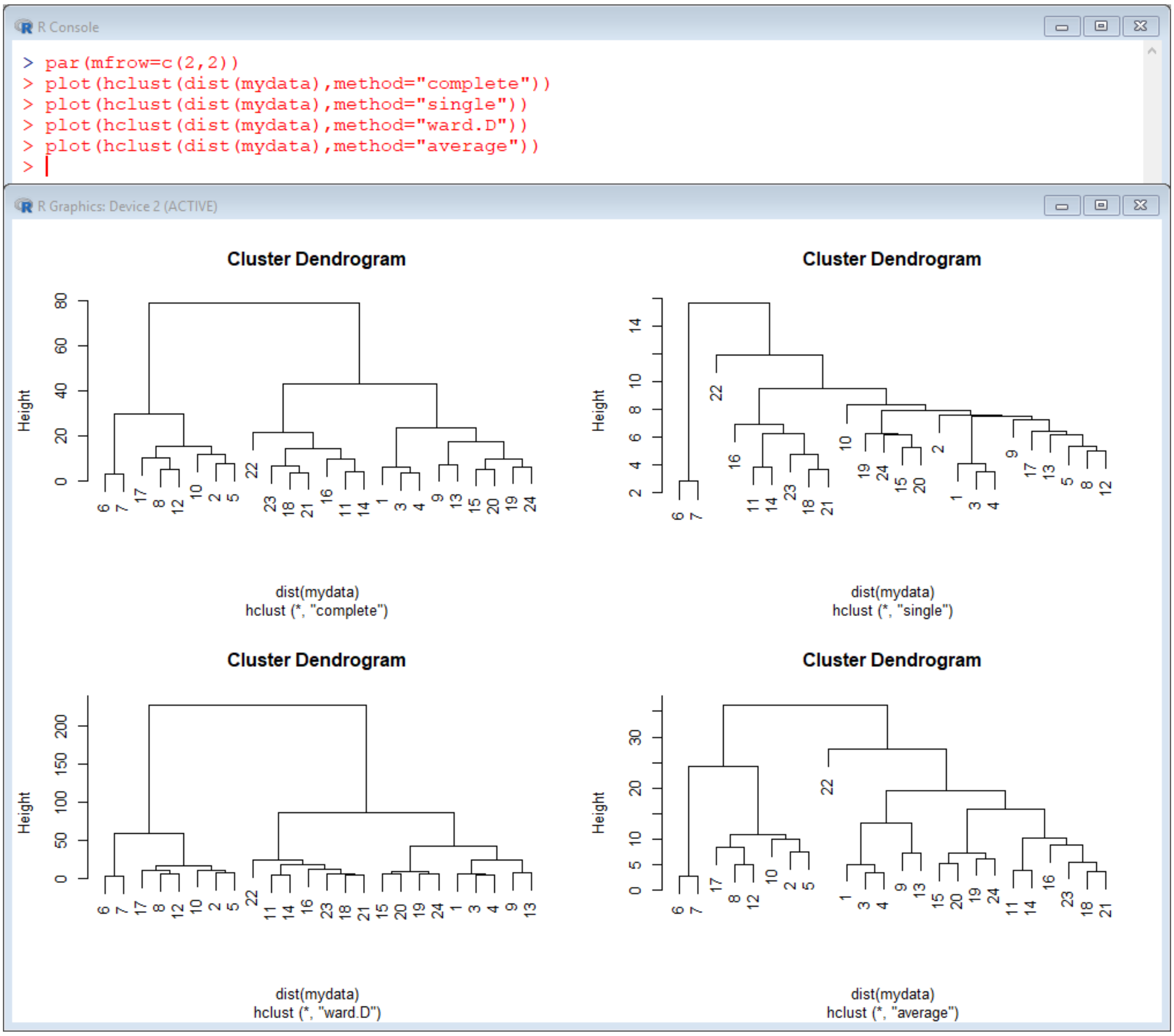
ou une interprétation sous forme tabulaire:



```
> aggregate(mydata,list(group5),mean)
  Group.1 Observation Revenus  Surface Propriétaire
1       1    2.666667   62.10 20.26667     1.000000
2       2    9.000000   85.55 20.20000     1.166667
3       3    6.500000  109.05 18.40000     1.000000
4       4   16.666667   66.20 17.66667     1.833333
5       5   17.857143   46.80 18.57143     1.857143
>
```

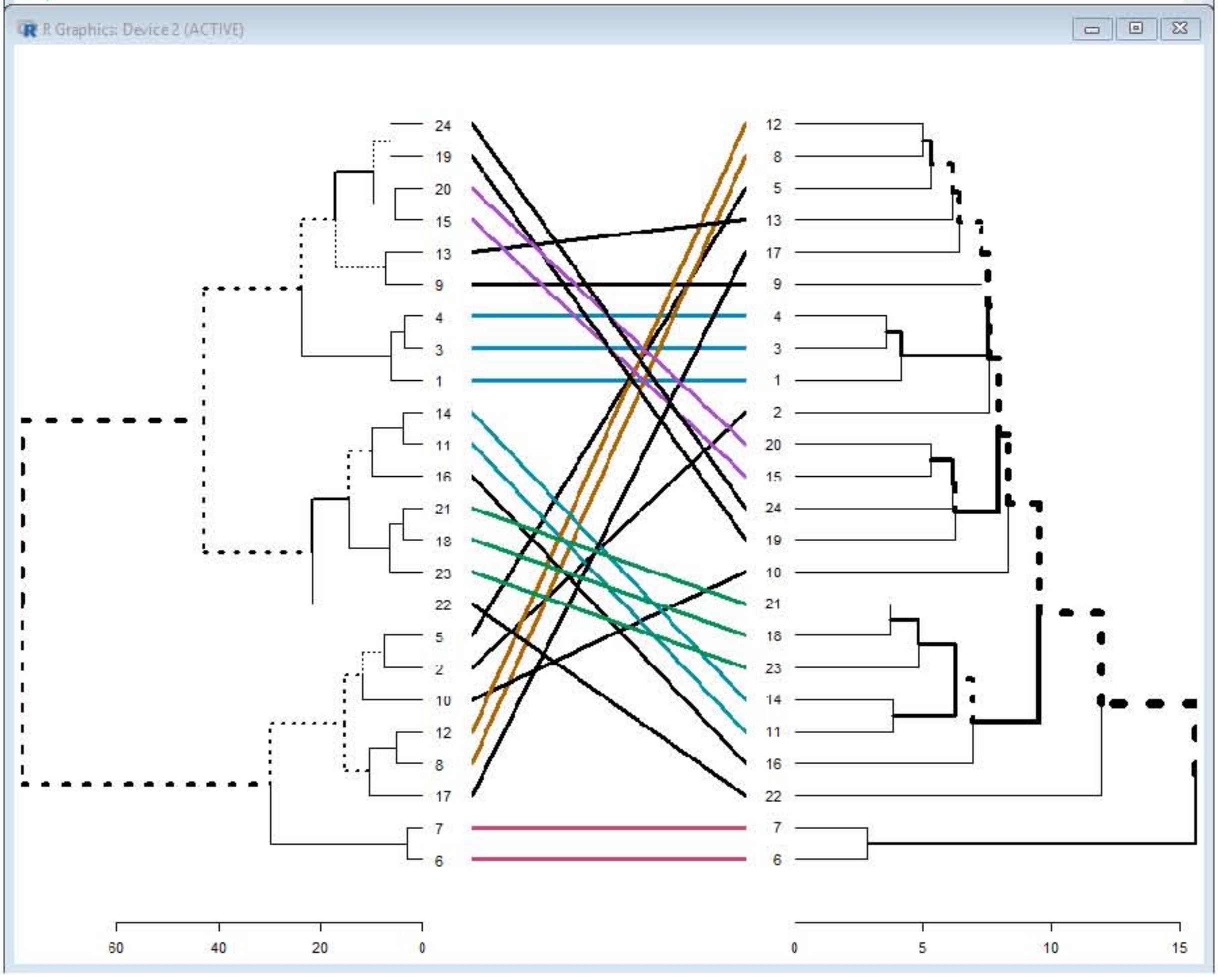


Sinon on peut bien évidemment comparer avec plusieurs fonctions de lien:



On peut aussi comparer deux dendrogrammes de manière visuelle avec la fonction **tanglegram( )** du package **dendextend**:

```
> library("dendextend")
> dend1 <- as.dendrogram(hclust(dist(mydata),method="complete"))
> dend2 <- as.dendrogram(hclust(dist(mydata),method="single"))
> tanglegram(dend1, dend2)
>
```



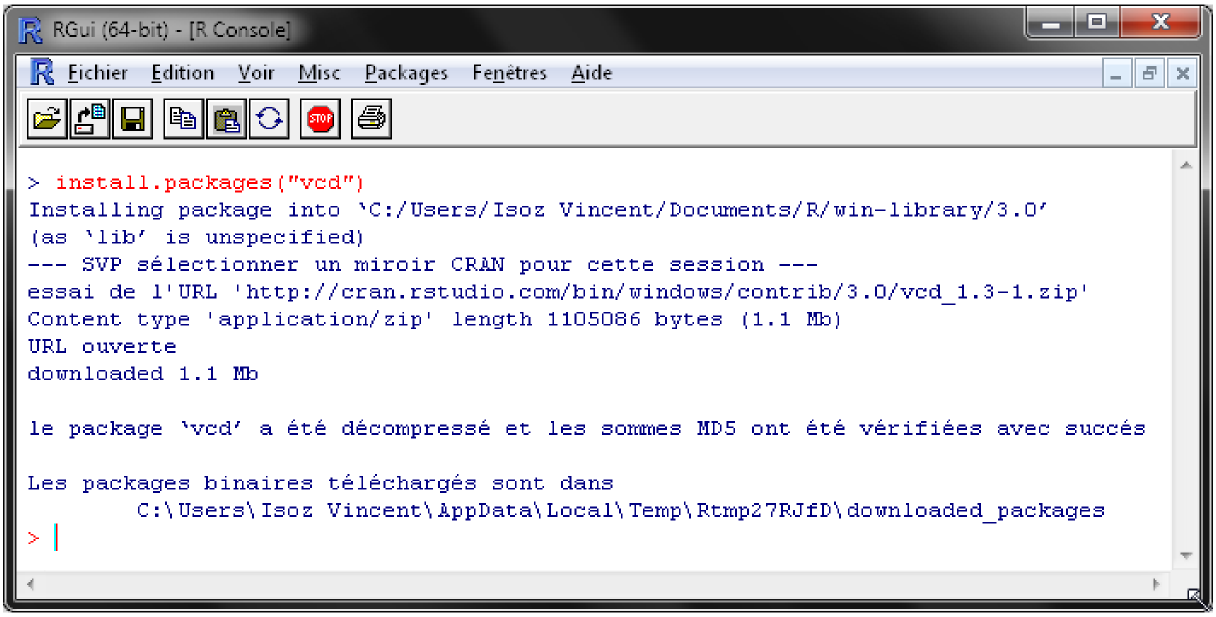
Exemple 152.: Diagramme ternaire (plans de mélange)

R 3.0.2

Dans le cours d'ingénierie nous avons étudié en détails la théorie mathématique des plans de mélange avec mesures aux sommets (plans de mélanges centrés) très utilisés en chimie.

Nous allons voir ici que nous pouvons reproduire graphiquement le mélange qui nous avait servi d'exemple comme cas pratique de la théorie.

Nous installons d'abord le package **vcd**:



```
> install.packages("vcd")
Installing package into 'C:/Users/Isoz Vincent/Documents/R/win-library/3.0'
(as 'lib' is unspecified)
--- SVP sélectionner un miroir CRAN pour cette session ---
essai de l'URL 'http://cran.rstudio.com/bin/windows/contrib/3.0/vcd_1.3-1.zip'
Content type 'application/zip' length 1105086 bytes (1.1 Mb)
URL ouverte
downloaded 1.1 Mb

le package 'vcd' a été décompressé et les sommes MD5 ont été vérifiées avec succés

Les packages binaires téléchargés sont dans
        C:\Users\Isoz Vincent\AppData\Local\Temp\Rtmp27RJfD\downloaded_packages
>
```

Et ensuite nous saisissons les données dans la joie en utilisant pour finir la commande **ternaryplot( )** :

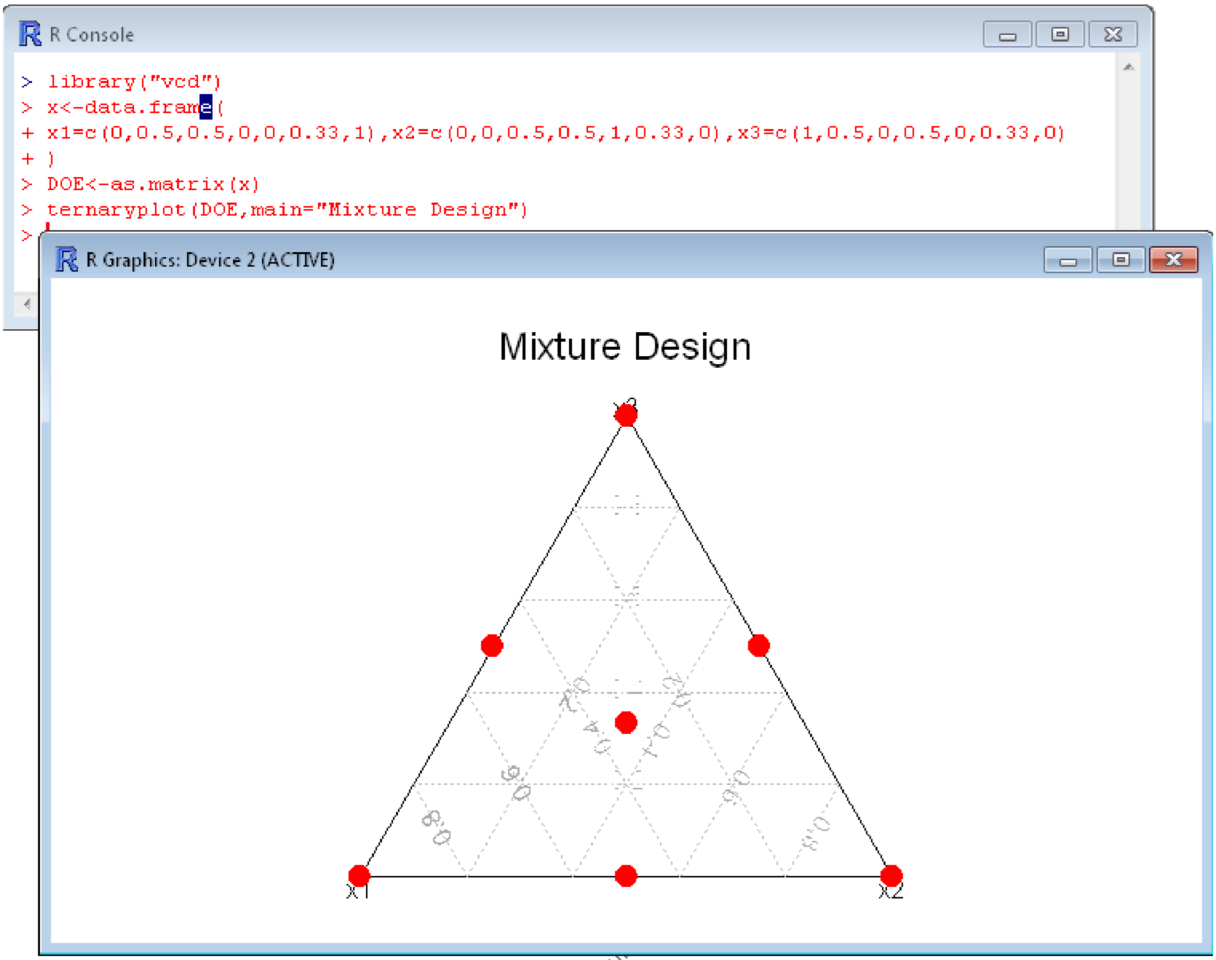


Figure 174 Graph ternaire (plans de mélanges)

Exemples 153.: Fractales

R 3.0.2

Fractale de Mandelbrot

Dans le cours de mathématique théorique nous avons étudiés la théorie des fractales pendant un temps significatif en utilisant Maple. Nous allons voir ci-dessous qu'il est tout à fait possible d'obtenir la même chose avec R.

C'est par ailleurs un excellent exercice pour les apprenants à tout langage concernant la mise en application des boucles:

```
R Console
> Limits=c(-2,0.8)
> MaxIter=25
> cl=colours()
> Step=seq(Limits[1],Limits[2],by=0.005)
> S=floor(length(cl)/MaxIter)
> Dist=0
> PointsMatrix=array(0,dim=c(length(Step)*length(Step),3))
> t=0
>
>
> for(a in Step)
+ {
+ for(b in Step+0.6)
+ {
+ x=0;y=0;n=0;Dist=0
+ while(n<MaxIter & Dist<4)
+ {
+ n=n+1
+ newx=a+x^2-y^2
+ newy=b+2*x*y
+ Dist=newx^2+newy^2
+ x=newx;y=newy
+ }
+ if(Dist<4) colour=24 # black colour
+ else colour=n*S
+ t=t+1
+ PointsMatrix[t,]=c(a,b,colour)
+ }
+ }
>
> X11()
>
> plot(PointsMatrix[,1], PointsMatrix[,2], xlim=Limits, ylim=Limits+0.6, col=cl[PointsMatrix[,3]], pch=".")
>
```

Ce qui donne:

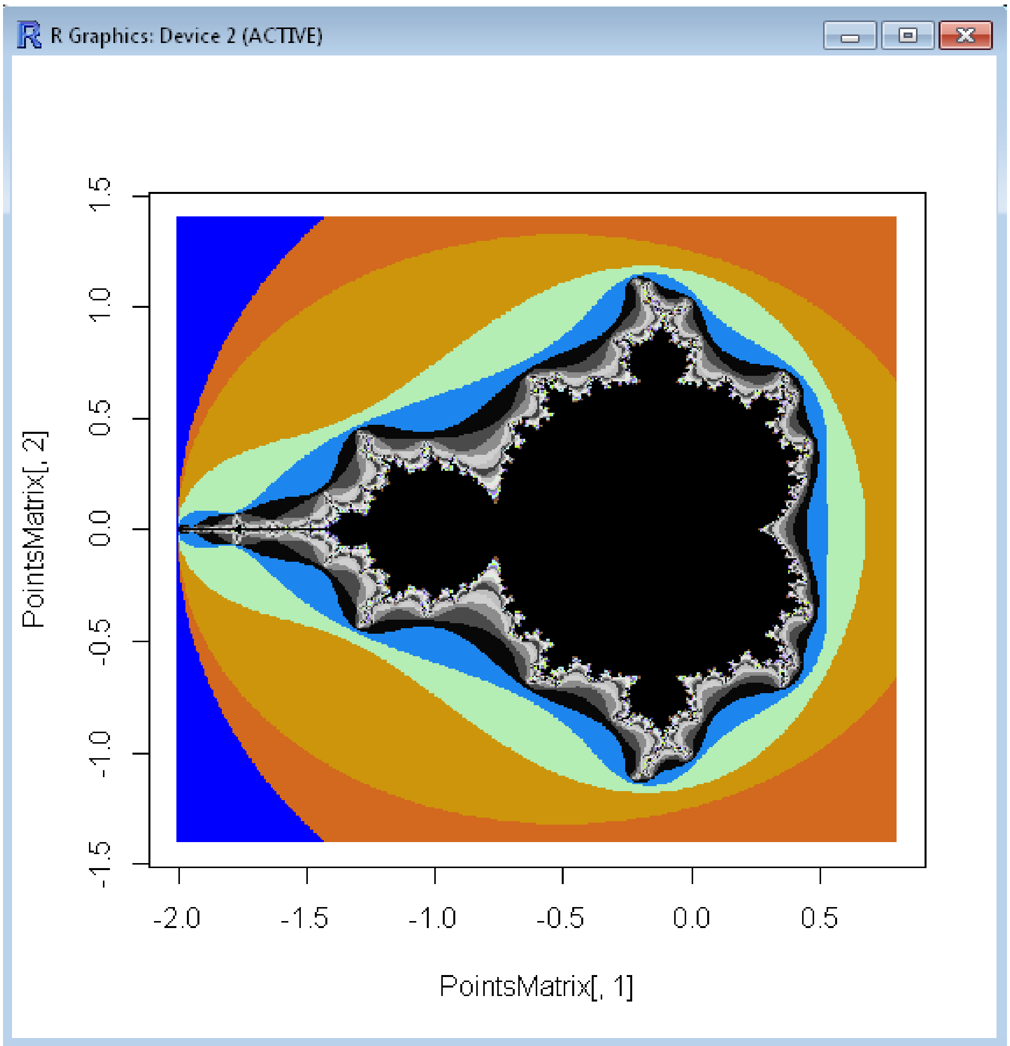
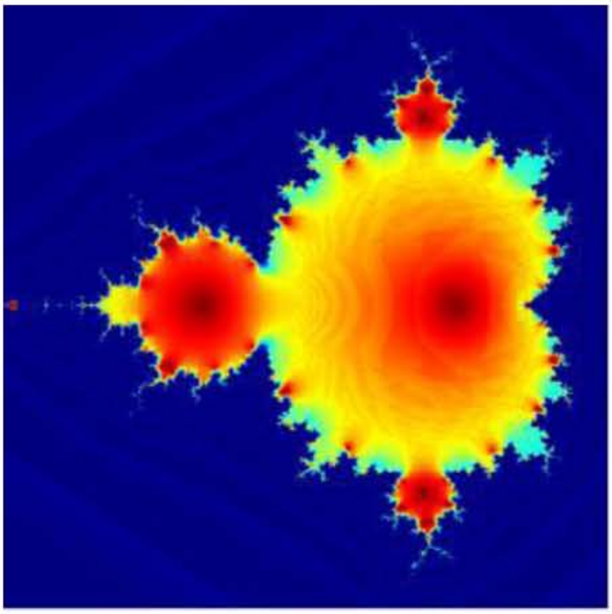


Figure 175 Graph fractale

ou encore plus fort en en animé mais en utilisant le package **catools**:

```
RGui (64-bit) - [R Console]
> setwd("C:/tmp")
> library(caTools) #ext pkg providing write.gif function
> jet.colors<-colorRampPalette(c("#00007F","blue","#007FFF","cyan","#7FFF7F",
+ "yellow","#FF7F00","red","#7F0000"))
> m<-1200 #define size
> C<-complex(real=rep(seq(-1.8,0.6,length.out=m),each=m),
+ imag=rep(seq(-1.2,1.2, length.out=m),m))
> C<-matrix(C,m,m) #reshape as square matrix of complex numbers
> Z<-0 #initialize Z to zero
> X<-array(0, c(m,m,20)) #initialize output 3D array
> for(k in 1:20) {
+ Z<-Z^2+C
+ X[,,k] <- exp(-abs(Z))
+ write.gif(X, "Mandelbrot.gif",col=jet.colors,delay=100)
+ }
>
```

Ce qui donnera une animation *.gif de 20 images avec comme image finale:



Fractale de Feigenbaum

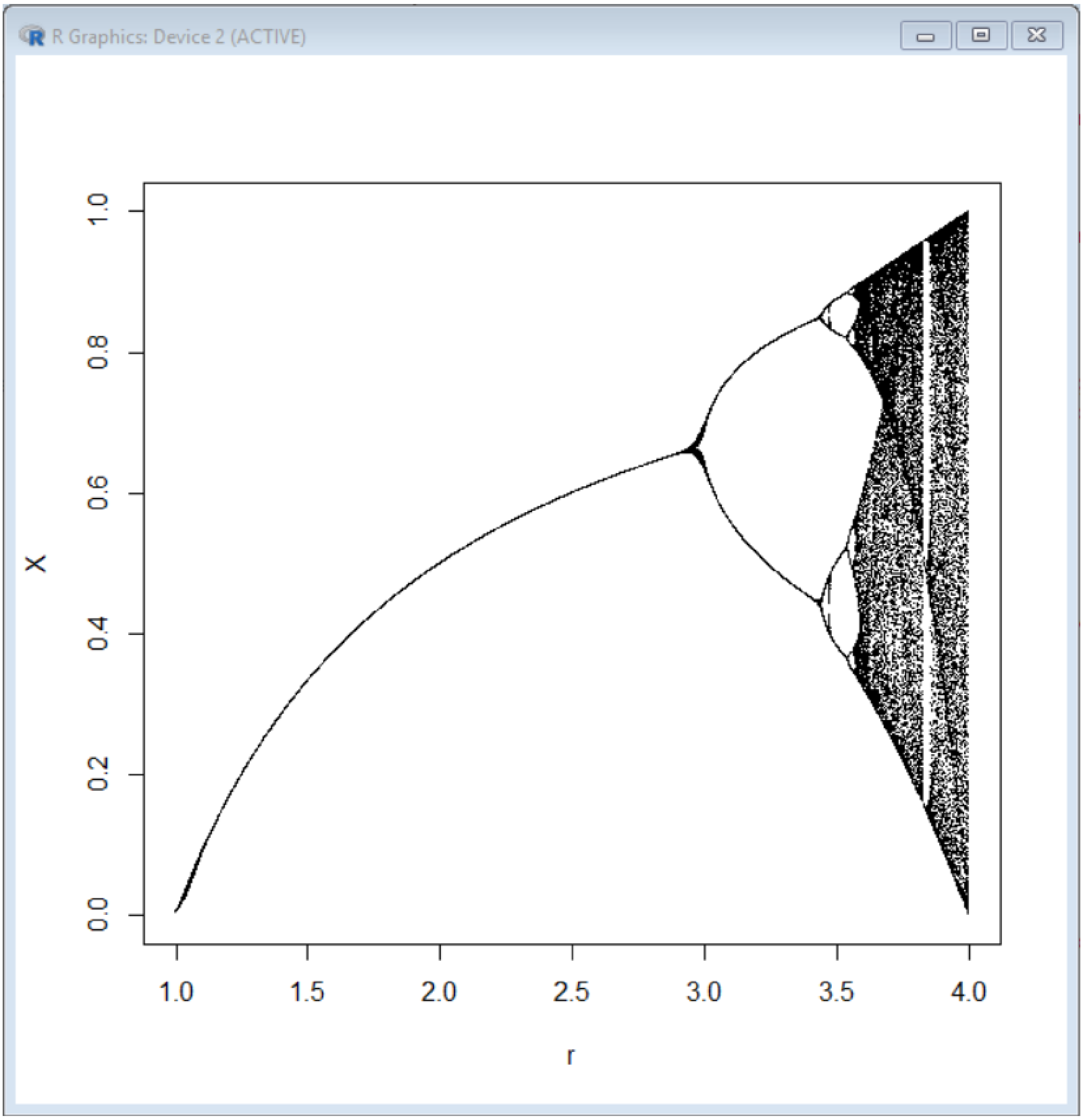
Dans le cours de mathématique théorique nous avons étudiés la théorie des fractales pendant un temps significatif en utilisant Maple comme nous l'avons déjà mentionné juste précédemment.

Nous allons voir ci-dessous qu'il est tout à fait possible d'obtenir la même chose avec R relativement à la fractale de Feigenbaum (merci à Mario dos Reis pour avoir écrit ce script en Octobre 2003):

```
> # non linear, chaotic function
> # r is growth rate and x is population size (from 0 to 1)
> lg = function(x, r) r * x * (1 - x)
>
>
> gen = 50        # number of initial generations
> r.min = 1
> r.max = 4
> p = '.'
> cl = 'black'    # plotting color
> div = 4000      # plot resolution along the x axis
> k = 64          # generations to be plotted, increase to 64 to get a high
>                 # resolution plot
>
> # full plot
> plot(x=0, y=0, type = 'n', xlim = c(r.min, r.max), ylim = c(0, 1),
+      xlab = 'r', ylab = 'X')
>
> for(r in seq(r.min, r.max, by = r.max/div)) {
+   x.init = .01
+
+   for(i in seq(0, gen)) {
+     x.next = lg(x.init, r)
+     x.init = x.next
+    }
+
+   for(i in seq(0, k)) {
+     x.next = lg(x.init, r)
+     x.init = x.next
+     points(r, x.next, pch = p, col = cl)
+   }
+ }
>
```

Nous voyons déjà que le diagramme de Feigenbaum est encore un meilleur exercice que la fractale de Mandelbrot pour maîtriser les boucles.

Le résultat de l'exécution du script ci-dessous donnera:



Et ce script permet aussi de zoomer simplement sur certaines zones pour visualiser l'aspect fractale de la figure!

Exemples 154.: Géostatistiques & Topographie (sciences de l'information géographique)

R 3.0.2

Encore une fois, même au niveau de géostatistiques, R ne vaut pas ce que l'on peut faire avec le tableur Microsoft Excel en termes de rapidités et d'esthétique dans les cas les plus classiques depuis la version 2010 du tableur en utilisant conjointement Power Pivot et Power Map.

Topographie

Commençons par une trivialité en utilisant la matrice *volcano* inclue nativement avec R:

```
R Console
> library(rgl);
> data(volcano);
> z<-3*volcano;
> x<-10*(1:nrow(z));
> y<-10*(1:ncol(z));
> zlim<-range(z);
> zlen<-zlim[2]-zlim[1]+1;
> colorlut<-terrain.colors(zlen,alpha=0);
> col<-colorlut[z-zlim[1]+1];
> open3d();
wgl
 29
> par3d("windowRect"= c(0,0,800,500))
> rgl.surface(x,y,z,color=col,back="lines");
> bbox3d(color=c("white","black"))
> grid3d("y")
> rgl.viewpoint(30,20,zoom=.8)
>
```

Ce qui donnera:

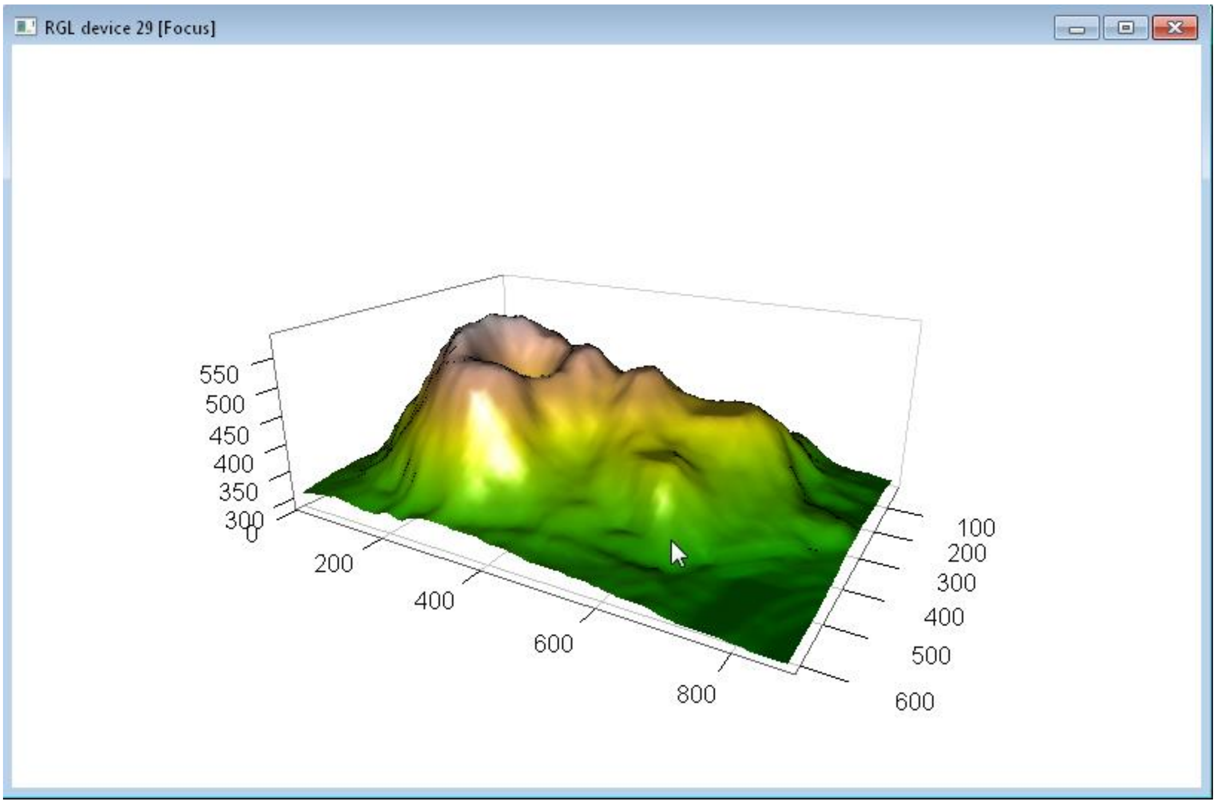
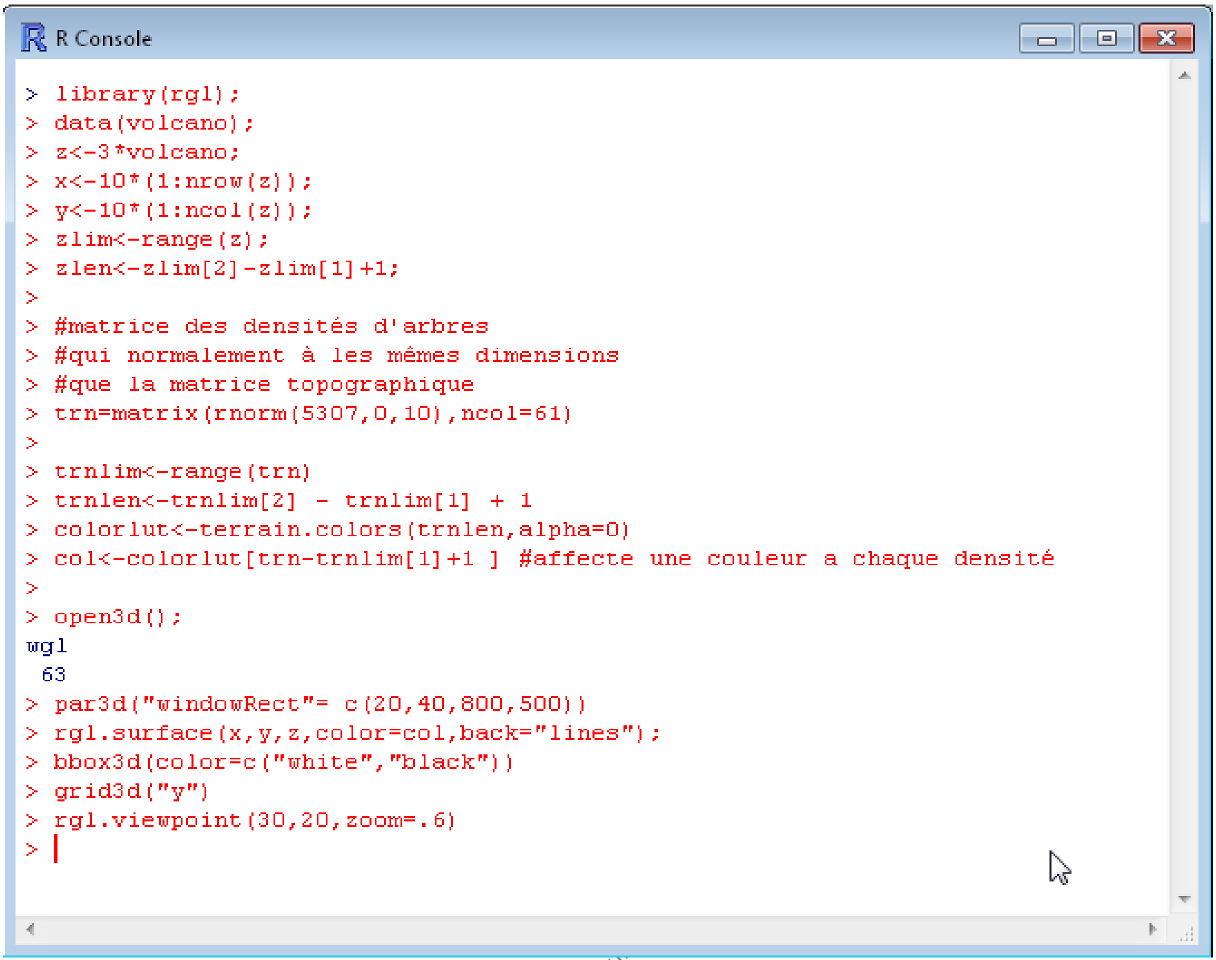


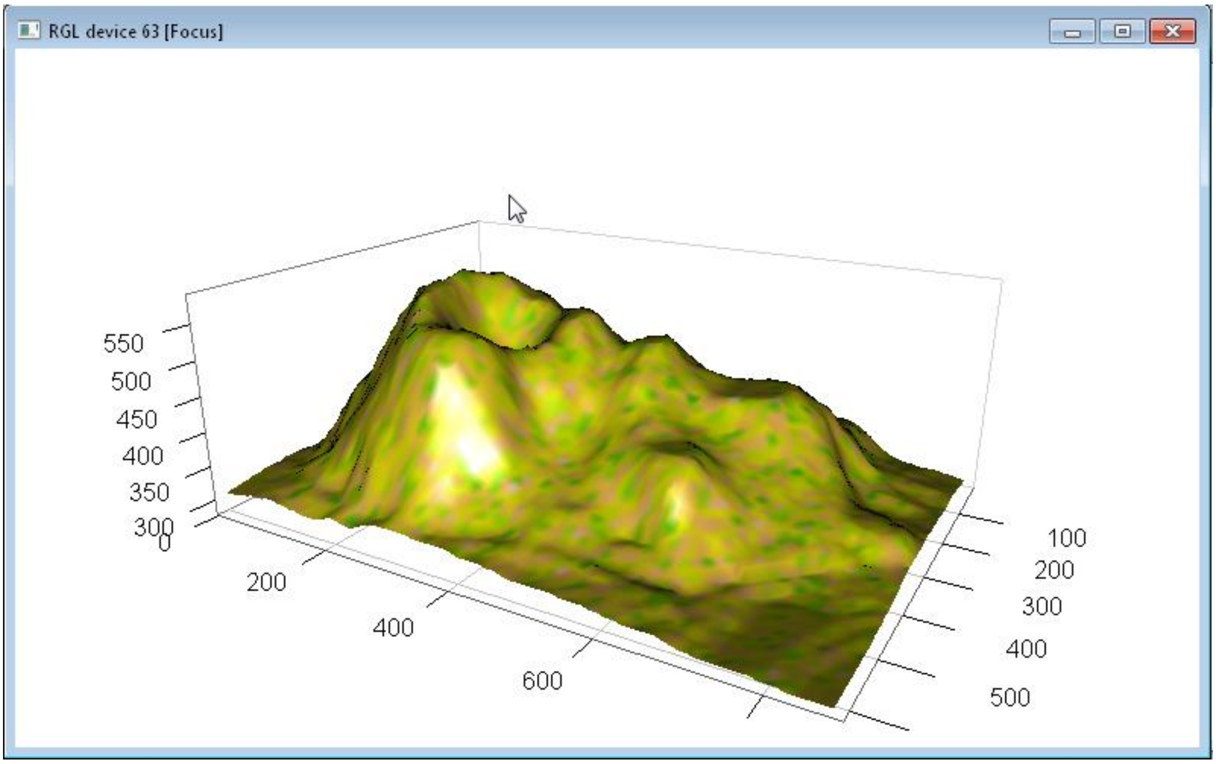
Figure 176 Graph de topographie avec gradient d'altitude

Imaginons maintenant que nous avons une matrice avec le type de terrain indiquant la densité de verdure. Nous pouvons alors utiliser le script suivant:



```
R Console
> library(rgl);
> data(volcano);
> z<-3*volcano;
> x<-10*(1:nrow(z));
> y<-10*(1:ncol(z));
> zlim<-range(z);
> zlen<-zlim[2]-zlim[1]+1;
>
> #matrice des densités d'arbres
> #qui normalement à les mêmes dimensions
> #que la matrice topographique
> trn=matrix(rnorm(5307,0,10),ncol=61)
>
> trnlim<-range(trn)
> trnlen<-trnlim[2] - trnlim[1] + 1
> colorlut<-terrain.colors(trnlen,alpha=0)
> col<-colorlut[trn-trnlim[1]+1 ] #affecte une couleur a chaque densité
>
> open3d();
wgl
 63
> par3d("windowRect"= c(20,40,800,500))
> rgl.surface(x,y,z,color=col,back="lines");
> bbox3d(color=c("white","black"))
> grid3d("y")
> rgl.viewpoint(30,20,zoom=.6)
>
```

Ce qui donne:



Et avec les courbes de niveau planes:

```
> filled.contour(volcano, color.palette = terrain.colors)
>
> #try
> cont <- contourLines(volcano)
> fun <- function(x) x$level
> LEVS <- sort(unique(unlist(lapply(cont, fun))))
> COLS <- terrain.colors(length(LEVS))
> contour(volcano)
> for(i in seq(cont)){
+     COLNUM <- match(cont[[i]]$level, LEVS)
+     polygon(cont[[i]], col=COLS[COLNUM], border="NA")
+ }
> contour(volcano, add=TRUE)
>
```

Ce qui donne:

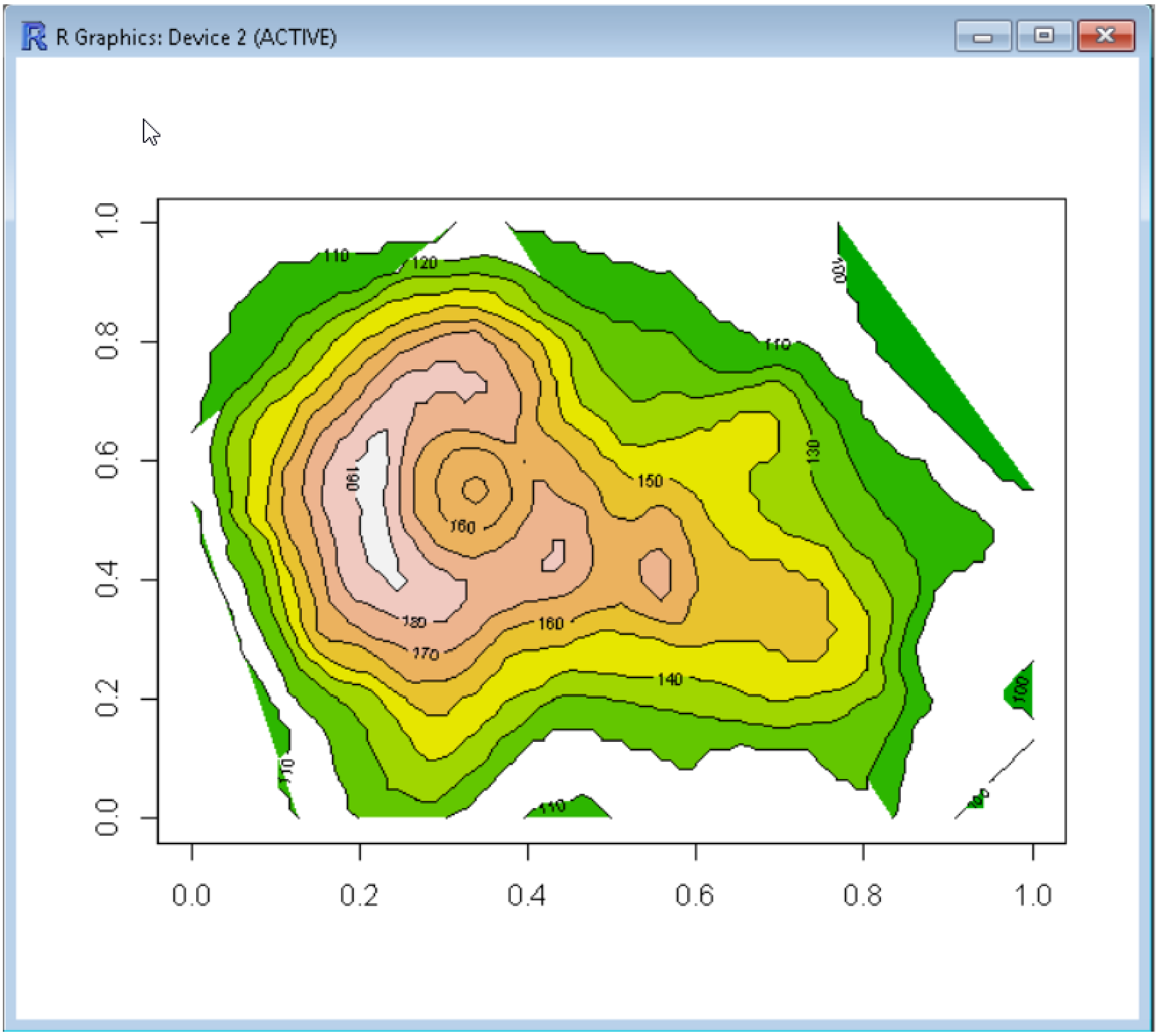
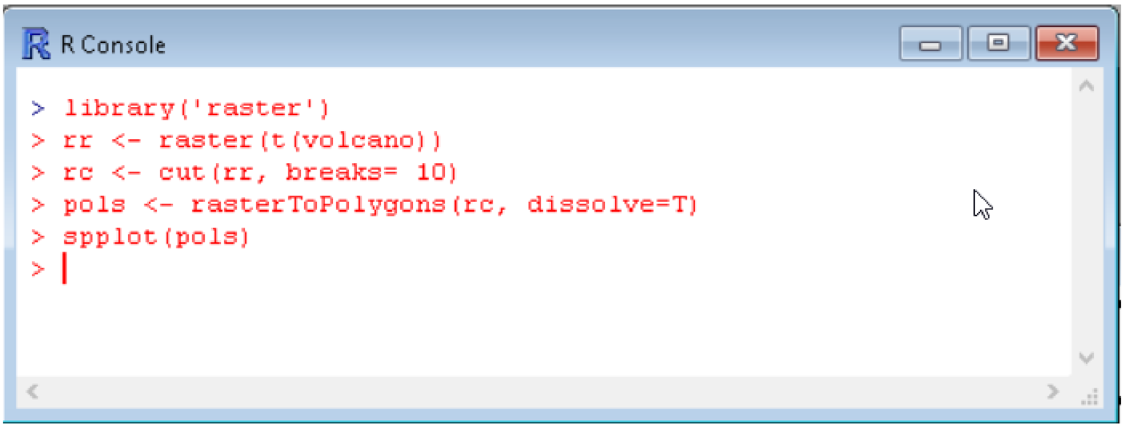
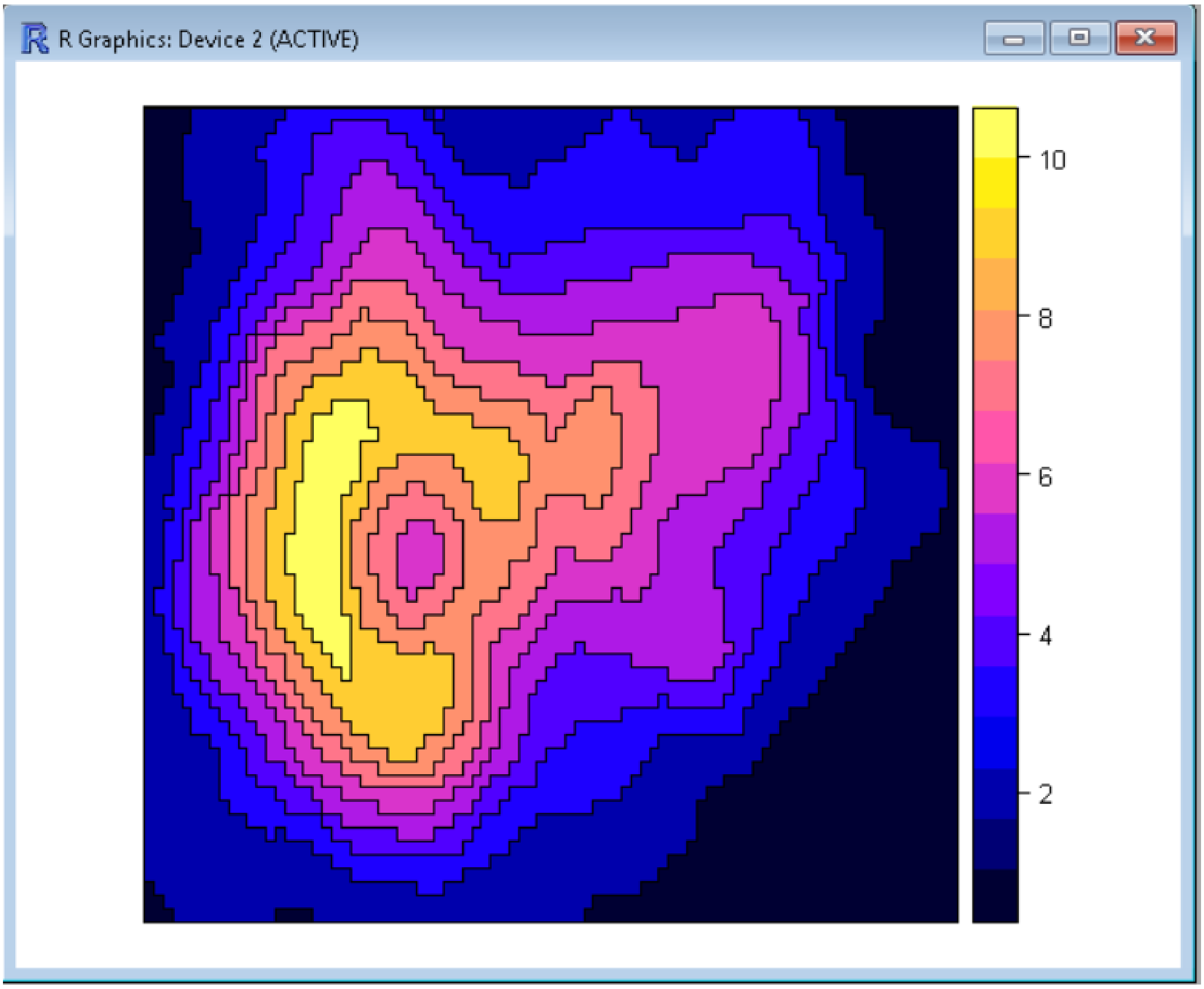


Figure 177 Graph topographique avec projection isoclines

Ou dans un autre genre:



Ce qui donne:



Ou encore dans d'autres genres que nous avons déjà vu précédemment:

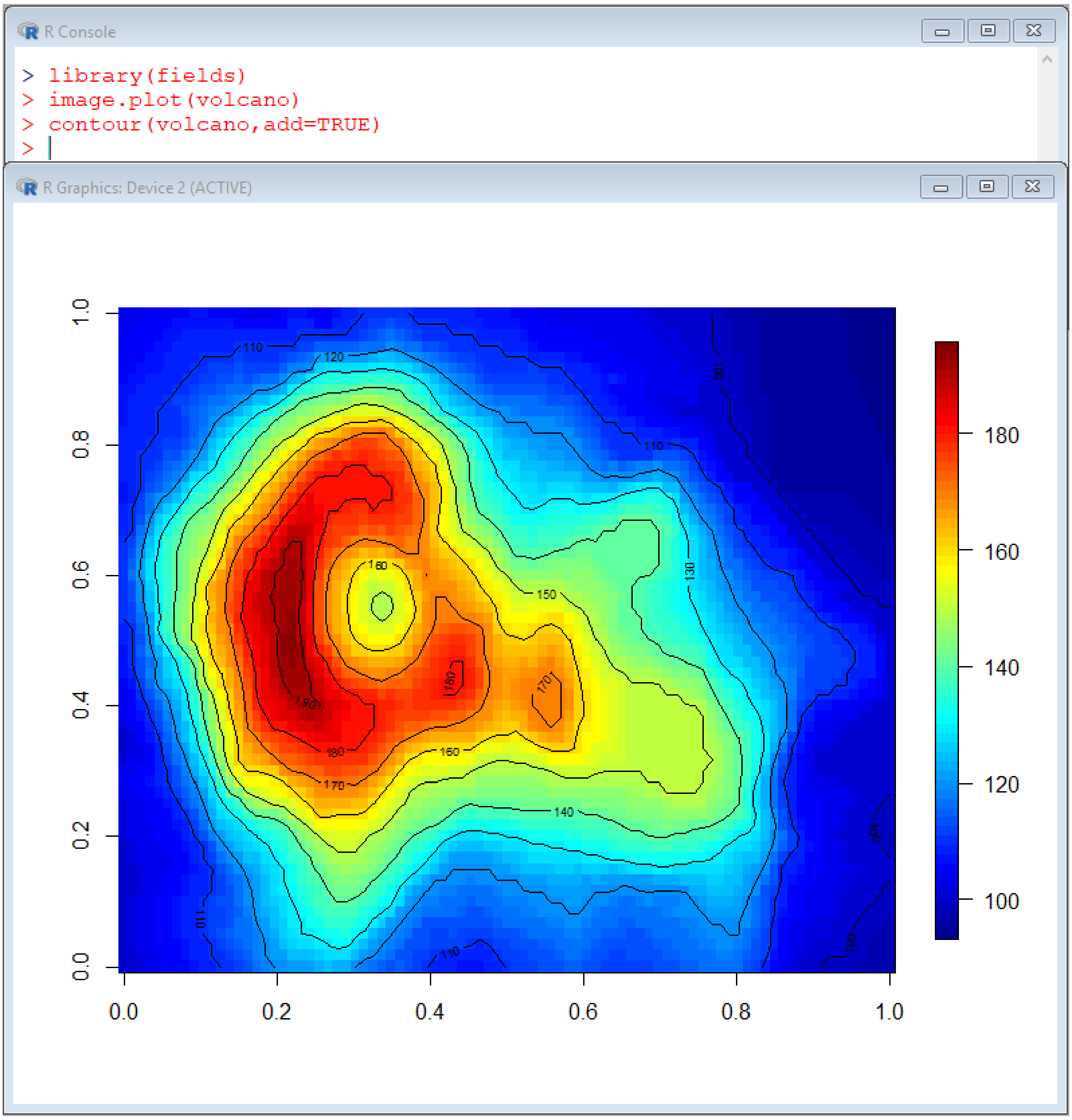
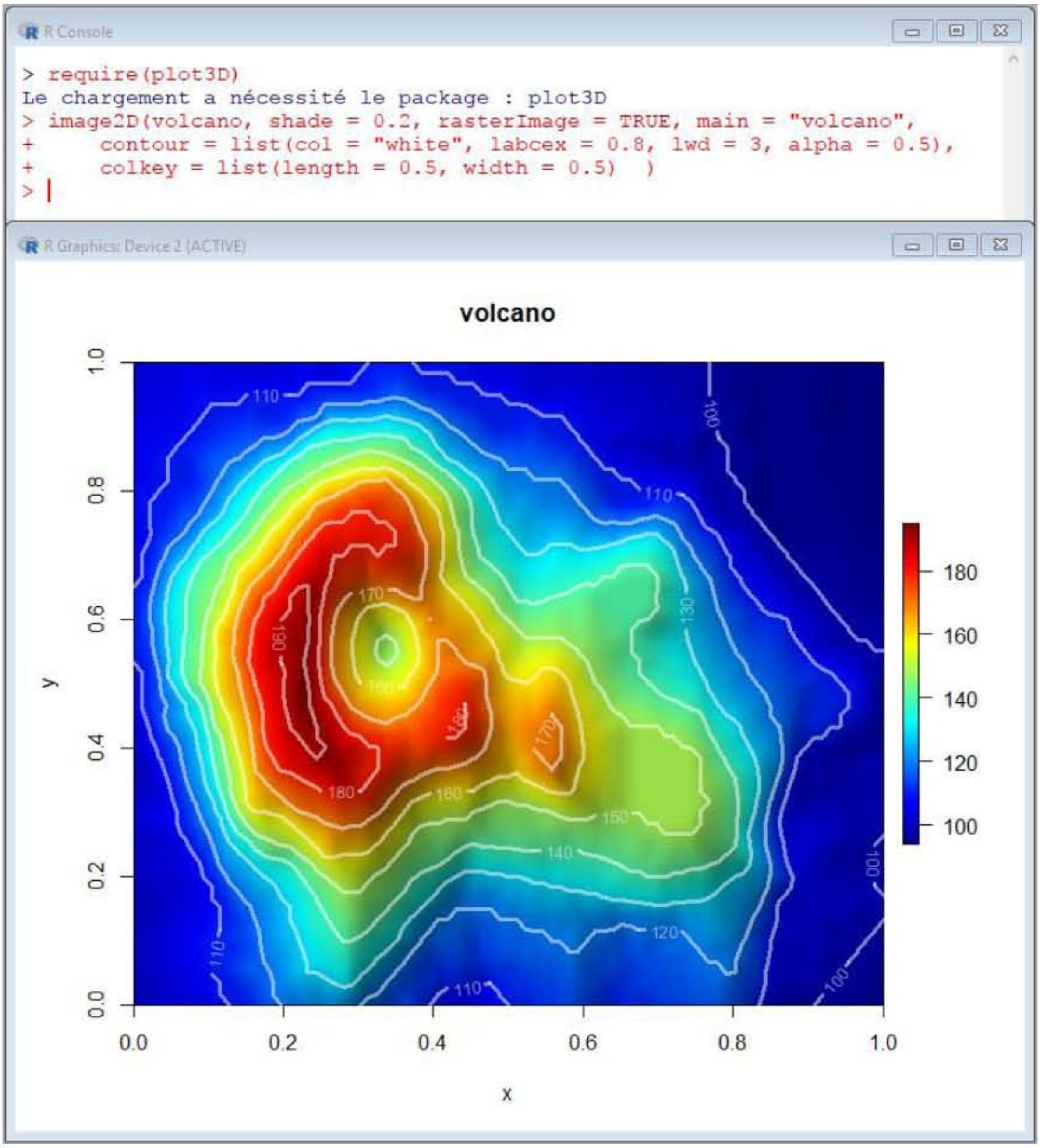


Figure 178 Graph topographique avec projection isoclines et gradients de couleurs

Ou en utilisant le package **plot3D** et sa fonction **image2D( )** :


R Console
> require(plot3D)
Le chargement a nécessité le package : plot3D
> image2D(volcano, shade = 0.2, rasterImage = TRUE, main = "volcano",
+ contour = list(col = "white", labcex = 0.8, lwd = 3, alpha = 0.5),
+ colkey = list(length = 0.5, width = 0.5))
>
R Graphics: Device 2 (ACTIVE)
volcano
y
x
0.0
0.2
0.4
0.6
0.8
1.0
180
160
140
120
100

Dans un autre style, avec le package **rayshader** on et de simples matrices on peut représenter des topographies de manière "hyperréaliste". Pour cela considérons le jeu de données *montereybay* du package:

```
> library(rayshader)
>
> #Affichage partiel de la matrice 'montereybay' du package
>
> class(montereybay)
[1] "matrix"
> dim(montereybay)
[1] 401 401
> montereybay[1:10,1:10]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
 [1,] -72.08319 -73.01987 -73.43484 -74.38454 -75.39100 -75.94428
 [2,] -72.45919 -72.44397 -72.76334 -73.39445 -73.78801 -75.38335
 [3,] -71.72905 -70.94937 -71.01565 -72.54224 -73.74262 -74.48636
 [4,] -70.73199 -70.49197 -70.19824 -72.36424 -74.44852 -73.55296
 [5,] -69.66419 -69.50726 -70.46214 -72.18534 -72.94451 -71.28796
 [6,] -68.65099 -68.75919 -70.11314 -71.47185 -72.78450 -69.25088
 [7,] -67.42379 -67.77757 -68.15453 -69.55094 -70.75121 -70.54028
 [8,] -66.42179 -67.13607 -67.86475 -68.48322 -69.52381 -70.50438
 [9,] -65.32139 -66.30607 -67.10574 -67.70616 -68.72651 -69.77498
[10,] -64.37639 -65.31107 -66.31574 -67.13544 -67.92520 -68.90496
           [,7]      [,8]      [,9]     [,10]
 [1,] -76.47496 -76.99753 -77.68941 -78.29829
 [2,] -76.24136 -77.06083 -77.67200 -78.27857
 [3,] -75.48029 -75.98043 -77.12461 -78.04729
 [4,] -74.78137 -75.34313 -76.41372 -77.26330
 [5,] -73.30176 -74.39844 -75.48483 -76.49879
 [6,] -71.33166 -73.48690 -74.64693 -75.87568
 [7,] -71.31096 -72.61573 -73.75421 -74.80500
 [8,] -71.01875 -71.89953 -72.97741 -74.05820
 [9,] -70.54476 -71.20833 -72.21089 -73.41259
[10,] -69.88470 -70.58443 -71.43471 -72.52389
> |
```

Et commençons avec les fonctions **sphere_shade( )** et **plot_3d( )** du package:

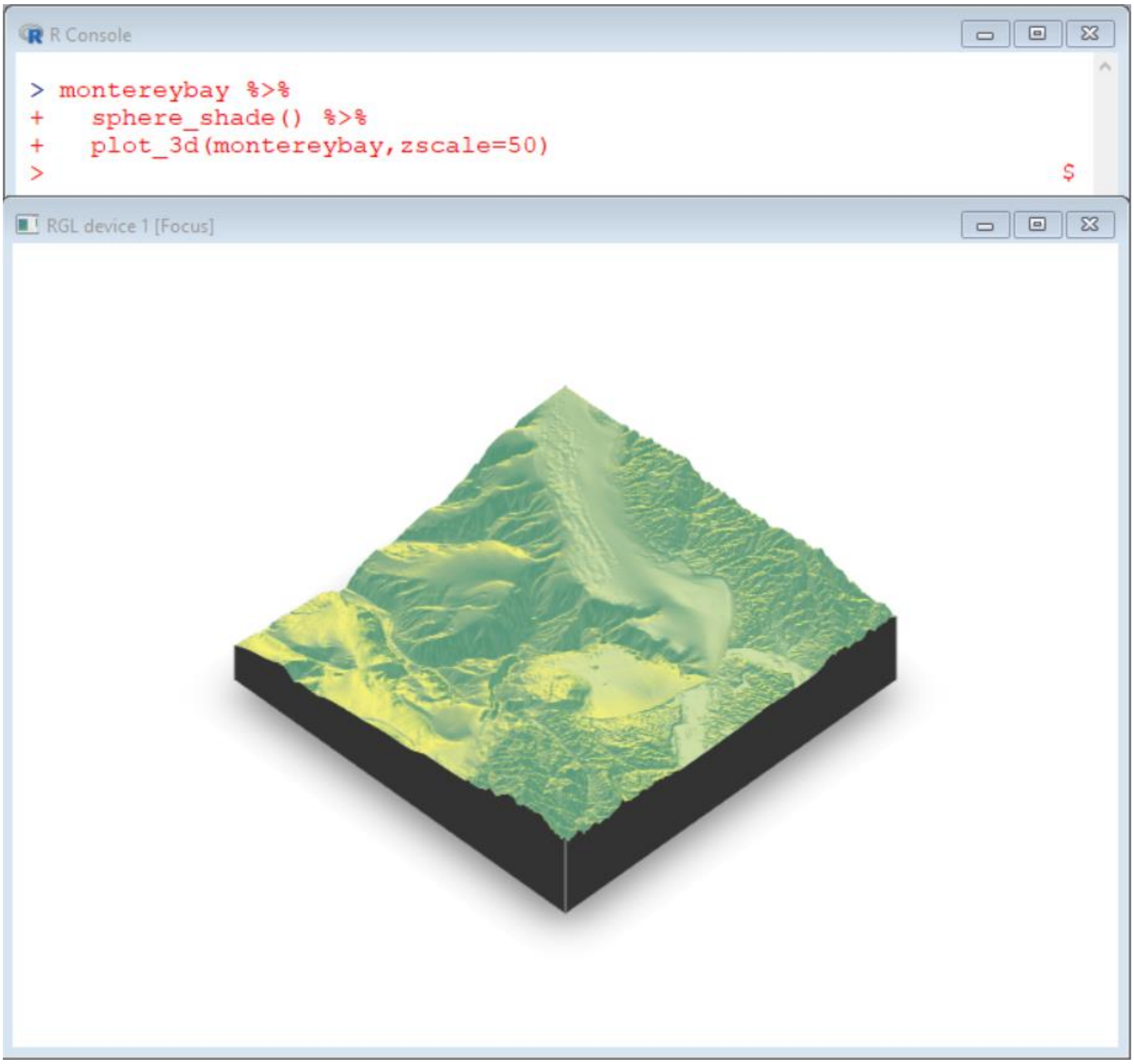
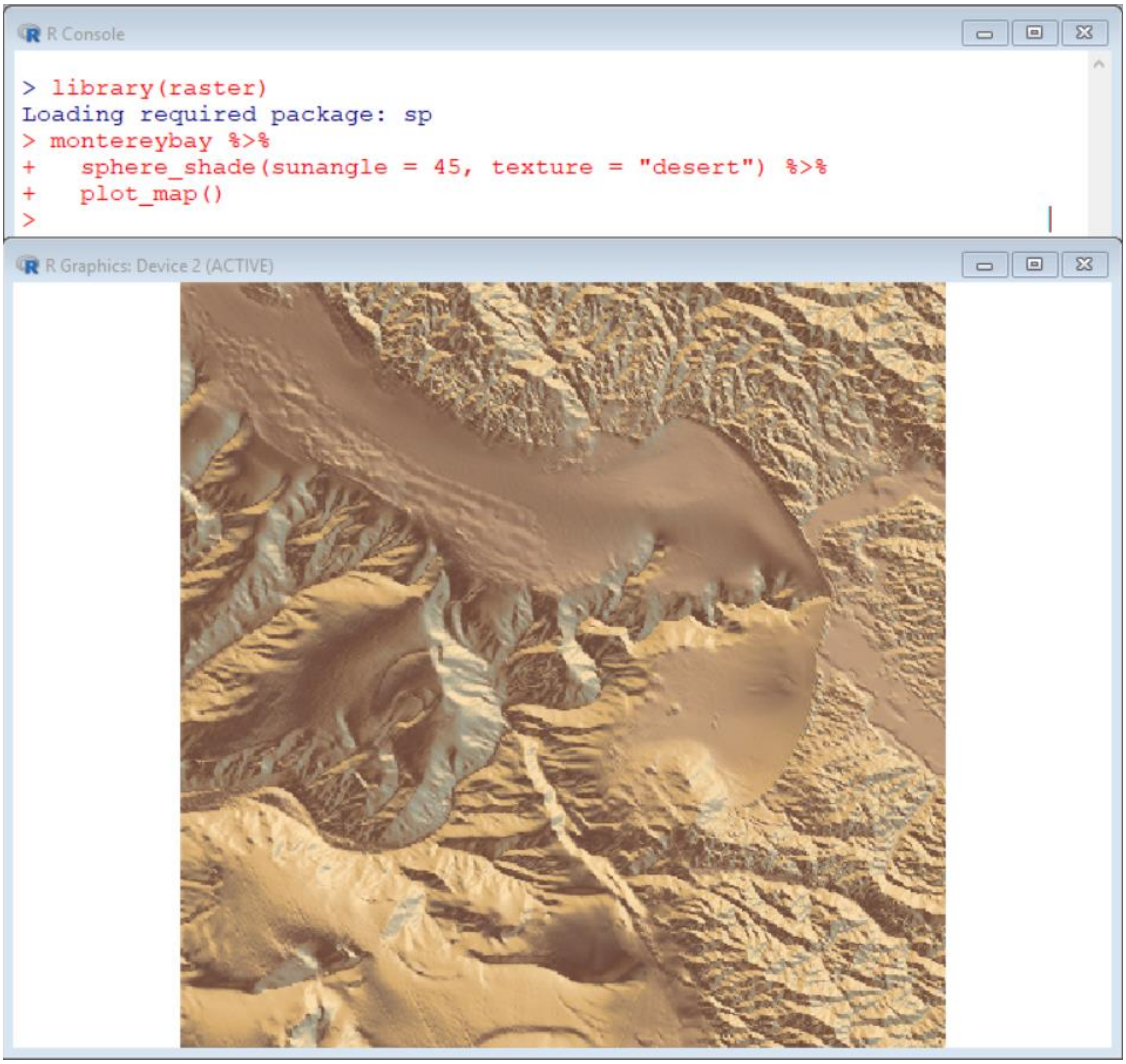


Figure 179 Graph de topographie hyperréel

On peut aussi ajouter une texture à l'aide du package **raster** et sa fonction **sphere_shade( )** et rajouter un effet d'ombre (soleil) et aussi afficher la carte en 2 dimensions avec la fonction **plot_map( )**:



Revenons à la 3D mais en préparant (sans l'afficher pour l'instant!!!!) la couche d'eau avec les fonctions **add_water( )** et **detect_water( )** définie selon la documentation par:

`detect_water` *Detect water*

Description

Detects bodies of water (of a user-defined minimum size) within an elevation matrix.

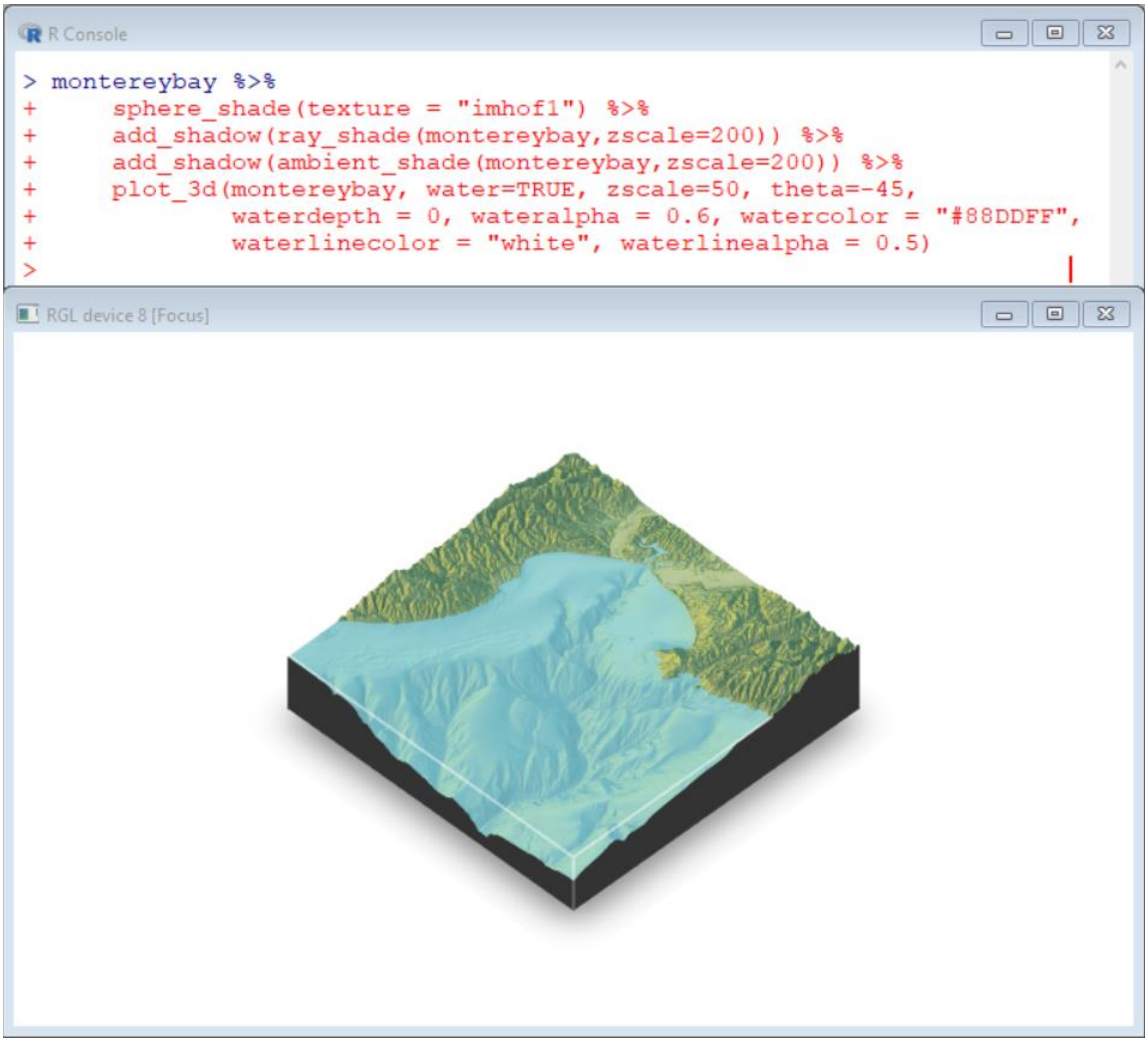
Usage

```
detect_water(heightmap, zscale = 1, cutoff = 0.999,
  min_area = length(heightmap)/400, normalvectors = NULL,
  keep_groups = FALSE, progbar = TRUE)
```

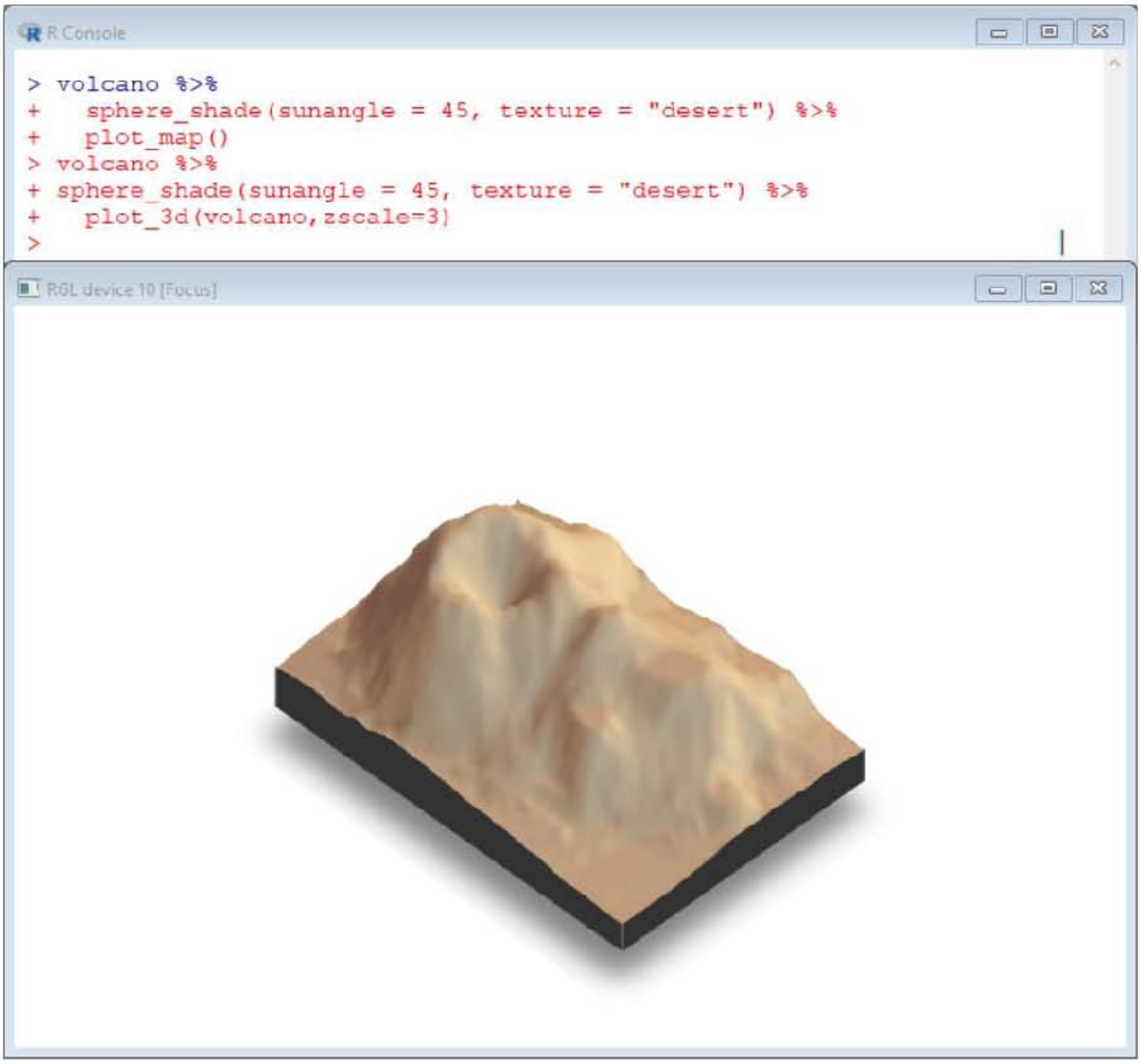
et d'effets d'ombres avec la fonction **add_shadow( )**:

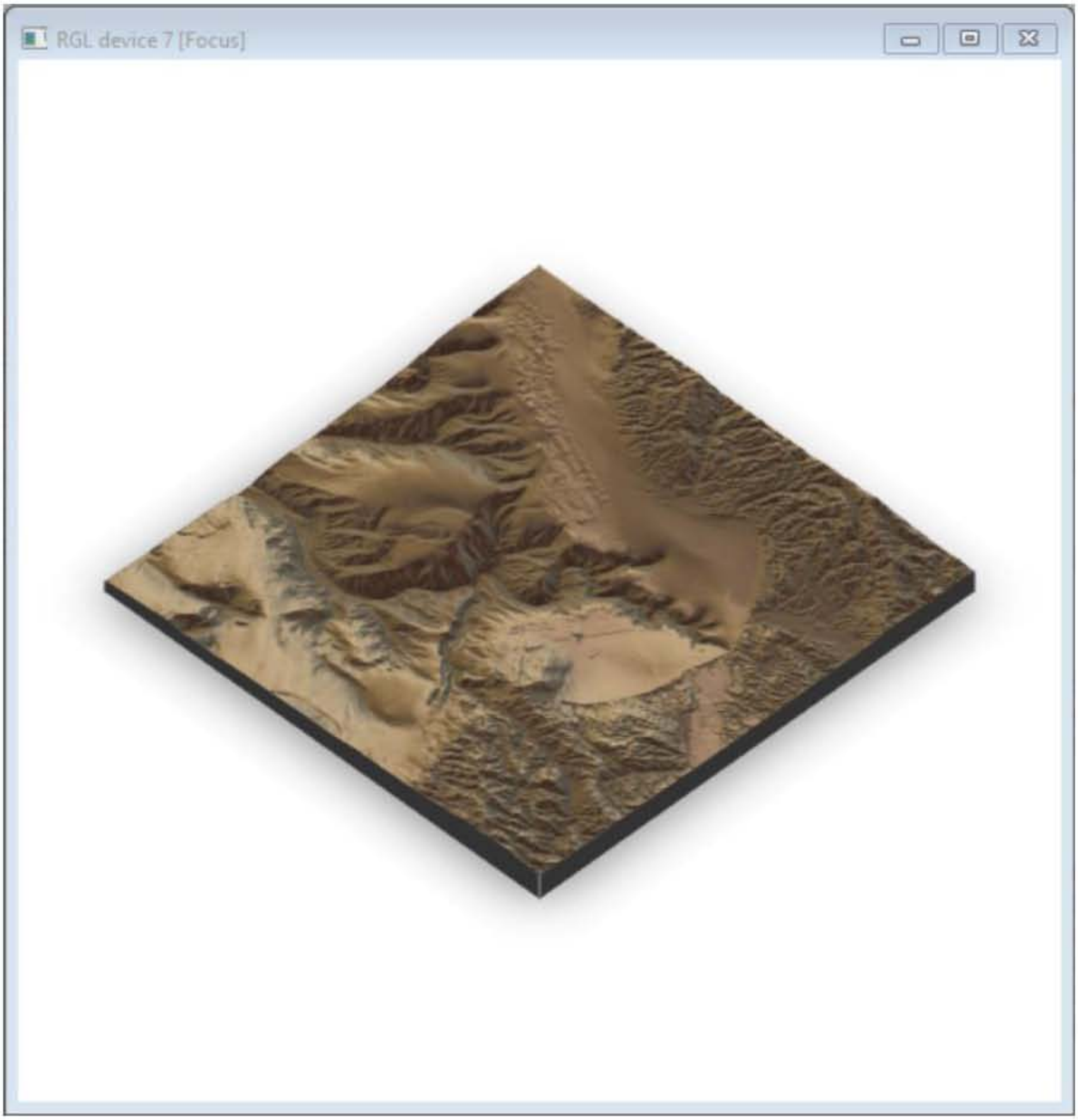


Et maintenant que la matrice contient une définition de la couche d'eau, nous pouvons l'afficher:



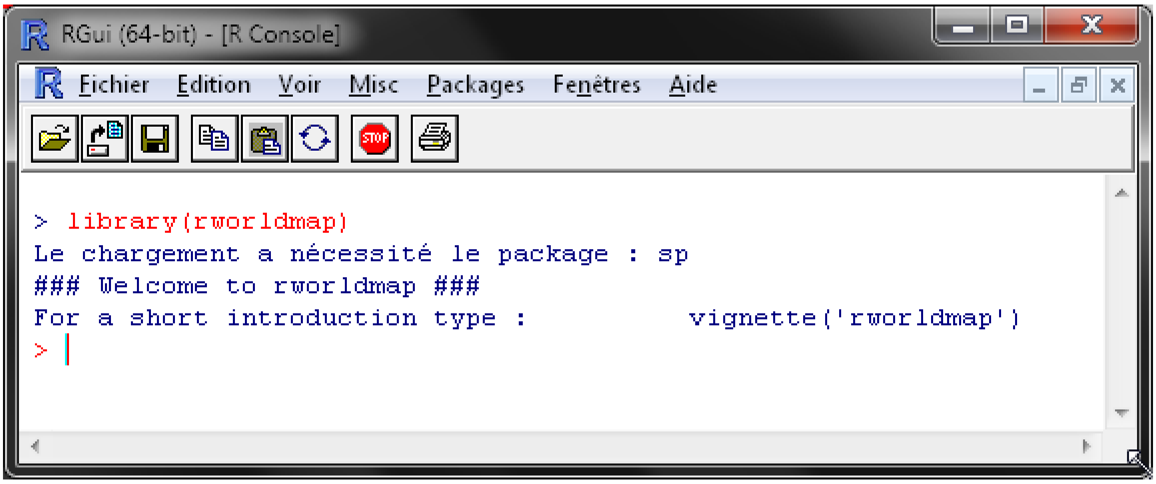
Je conseille également vivement au lecteur de s'amuser avec la matrice *volcano* native de R:


R Console
> volcano %>%
+   sphere_shade(sunangle = 45, texture = "desert") %>%
+   plot_map()
> volcano %>%
+ sphere_shade(sunangle = 45, texture = "desert") %>%
+   plot_3d(volcano,zscale=3)
>
RGL device 10 [Focus]

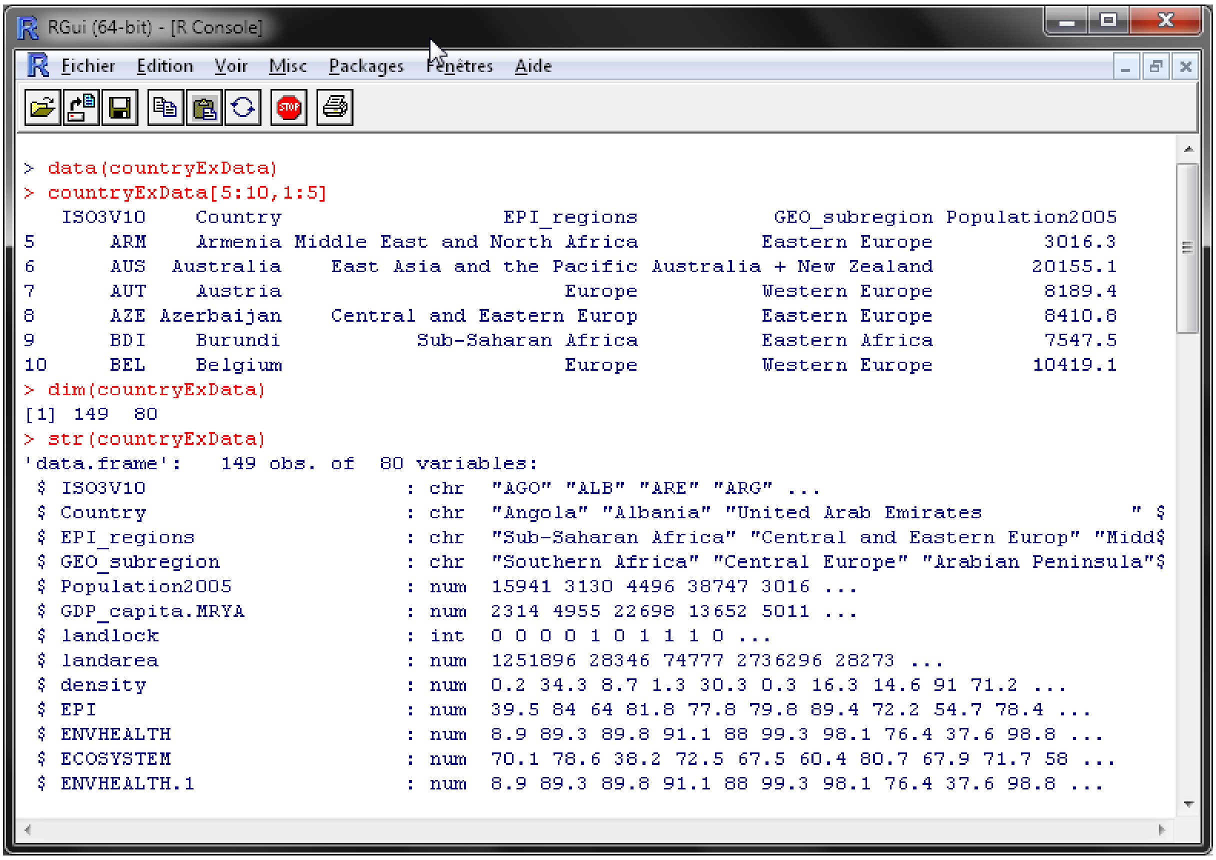

RGL device 7 [Focus]

Statistiques au niveau mondial (package rworldmap) avec projection cylindrique

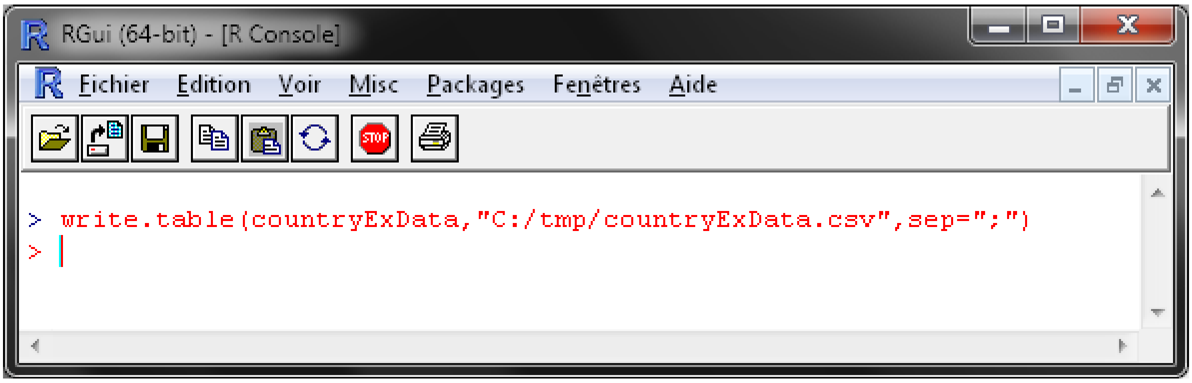
Le package le plus élégant que j'ai trouvé à ce jour pour faire des statistiques au niveau mondial est **rworldmap** (dont la documentation est en plus bien faite).



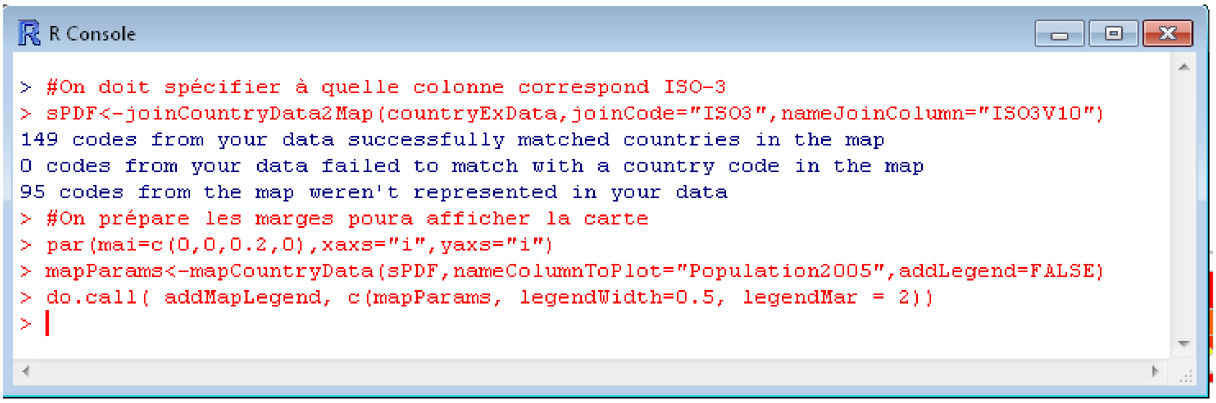
Nous allons travailler directement avec le jeu de données intégré à ce même package:



Que l'on peut facilement exporter pour son usage personnel (ceci afin de ne pas ressaisir la liste des pays et régions et sous régions les plus courants) par exemple avec la commande suivante et en faisant ensuite un coller dans un tableur:



Et nous commençons par afficher la population mondiale:



Ce qui donne:

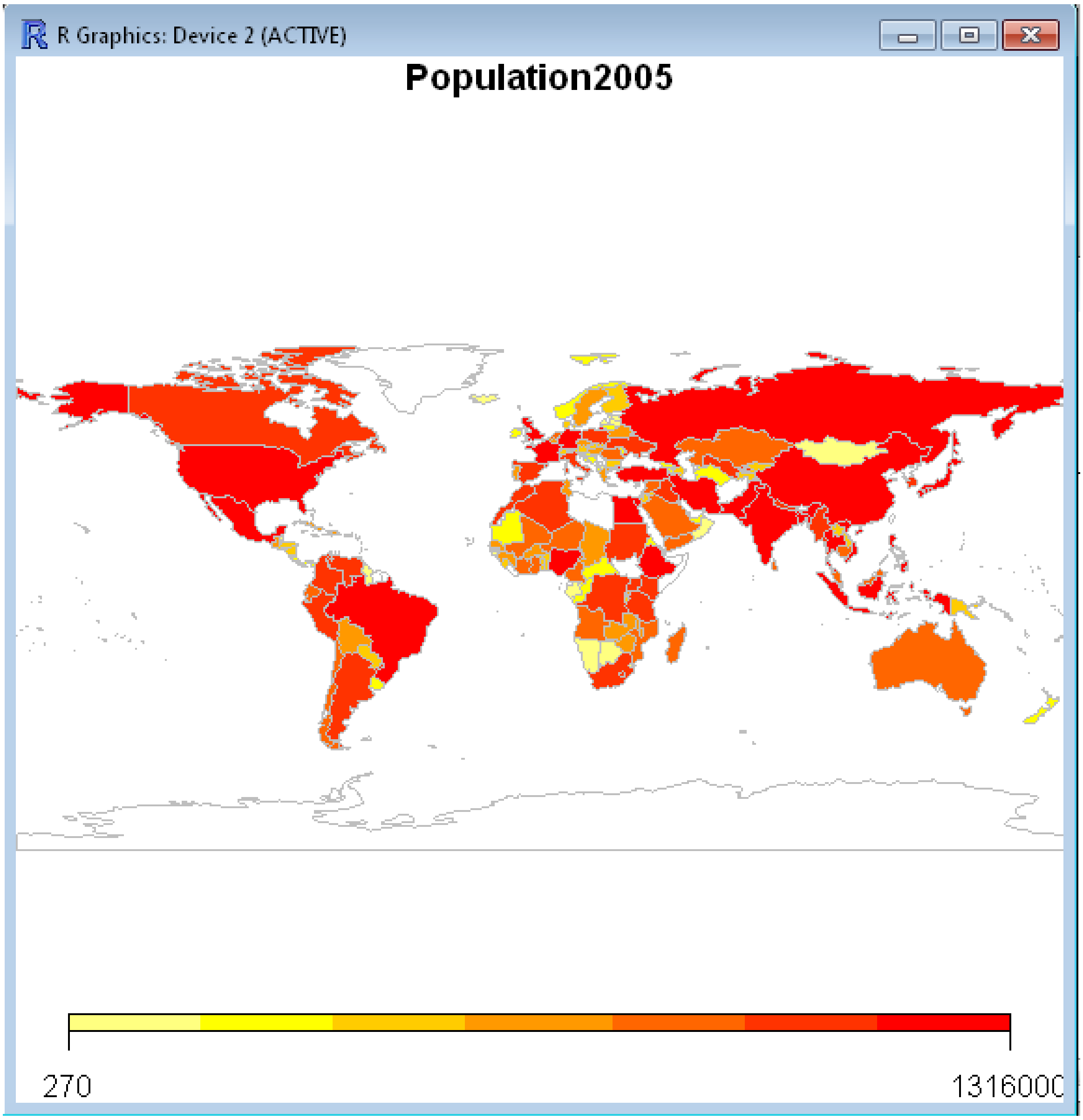
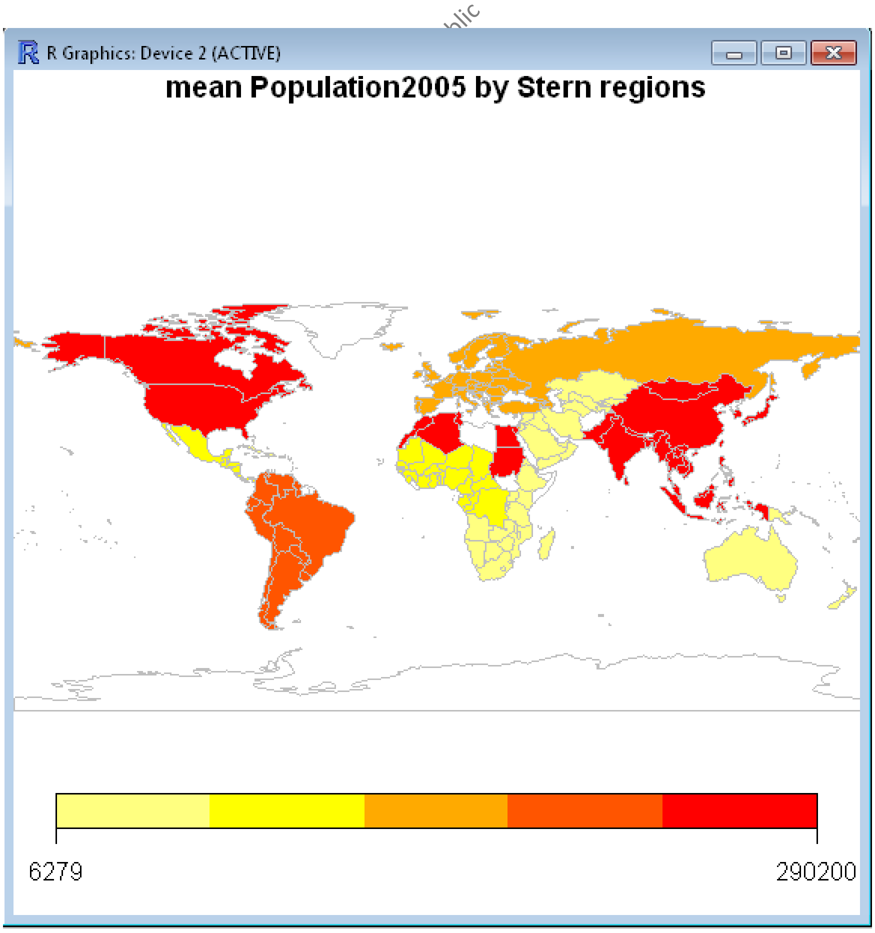


Figure 180 Graph Map-Monde Mercator avec zones géographiques coloriées

Ou encore en agrégant les données:

```
R Console
> #ça c'est juste pour avoir le résultat sous forme tabulaire
> sternGeoStat<-country2Region(inFile=countryExData,nameDataColumn="Population2005",
+ joinCode="ISO3",nameJoinColumn="ISO3V10",regionType="Stern",FUN="mean")
> print(sternGeoStat)
                meanPopulation2005byStern
Australasia                       6279.20
Caribbean                         6529.60
Central America                  18378.66
Central Asia                     11604.46
East Asia                              NA
Europe                           19373.14
North Africa                     36940.14
North America                   165240.55
South America                    34049.80
South Asia                      290200.78
South+E Africa                   18211.46
West Africa                      18492.85
West Asia                        18194.29
> #et maintenant l'équivalent sous forme de carte
> par(mai=c(0,0,0.2,0),xaxs="i",yaxs="i")
> mapByRegion(countryExData,nameDataColumn="Population2005", joinCode="ISO3",
+ nameJoinColumn="ISO3V10",regionType="Stern",FUN="mean")
149 codes from your data successfully matched countries in the map
0 codes from your data failed to match with a country code in the map
95 codes from the map weren't represented in your data
You asked for 7 quantiles, only 5 could be created in quantiles classification
> |
```

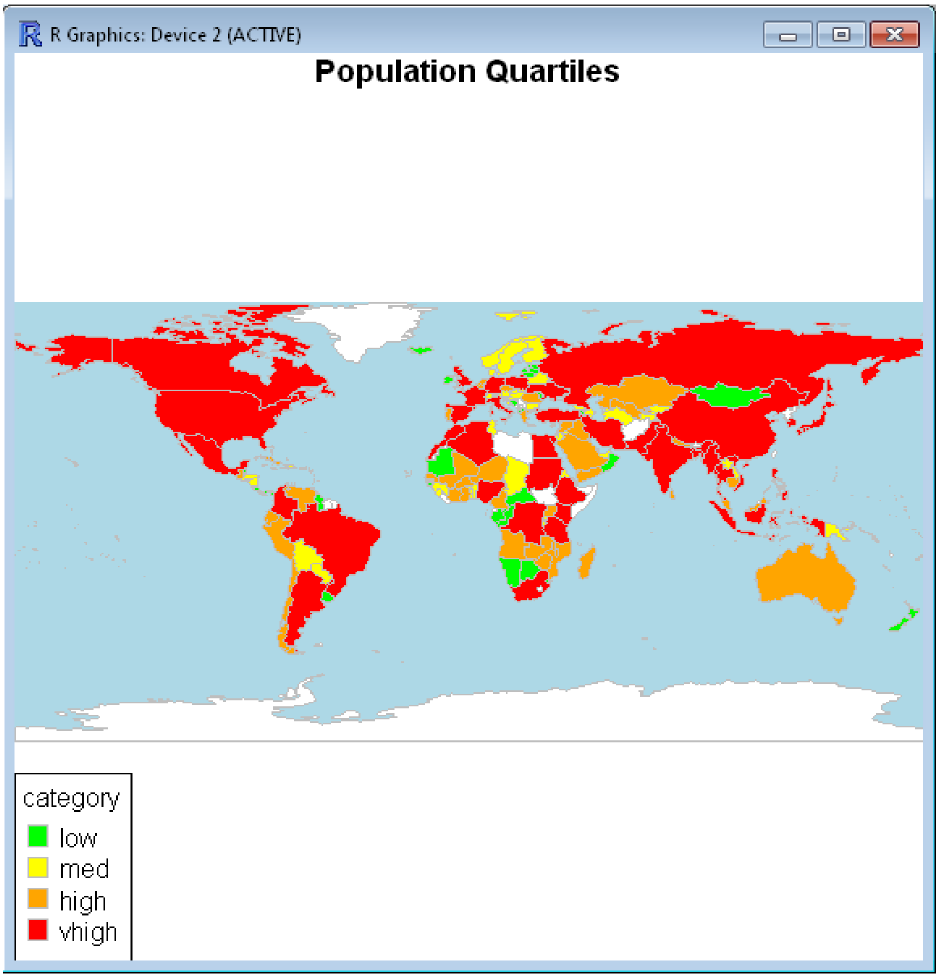
Ce qui donne:



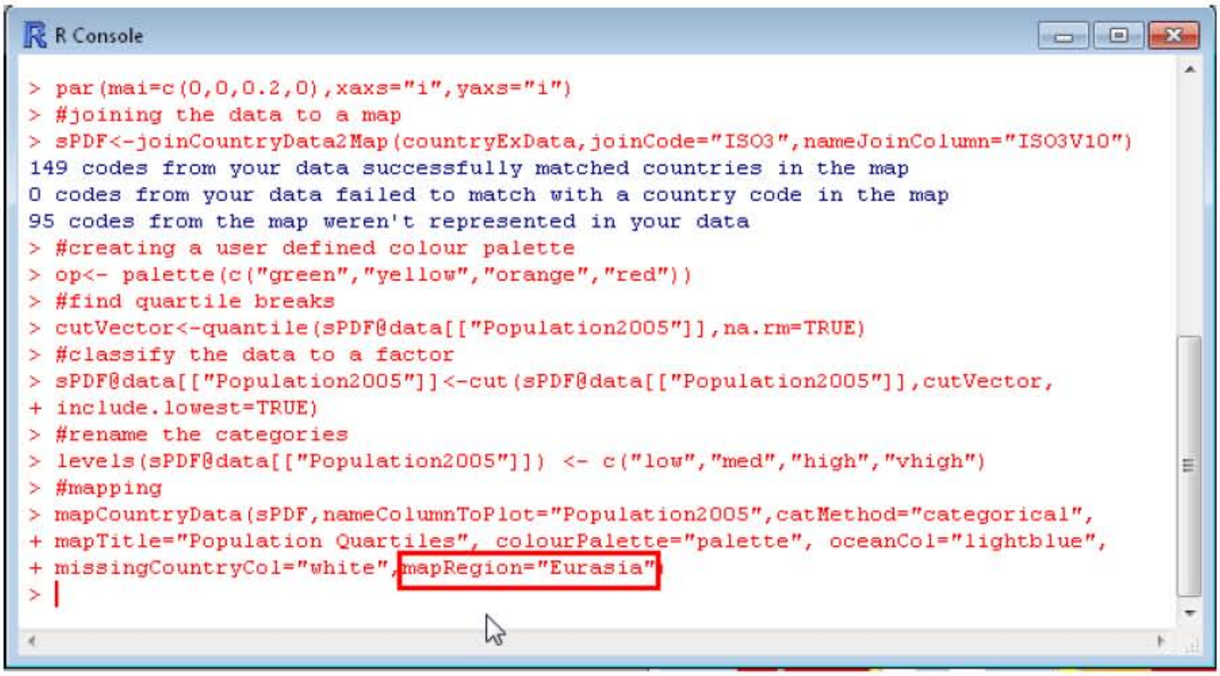
On peut aussi regrouper les données par quartiles:

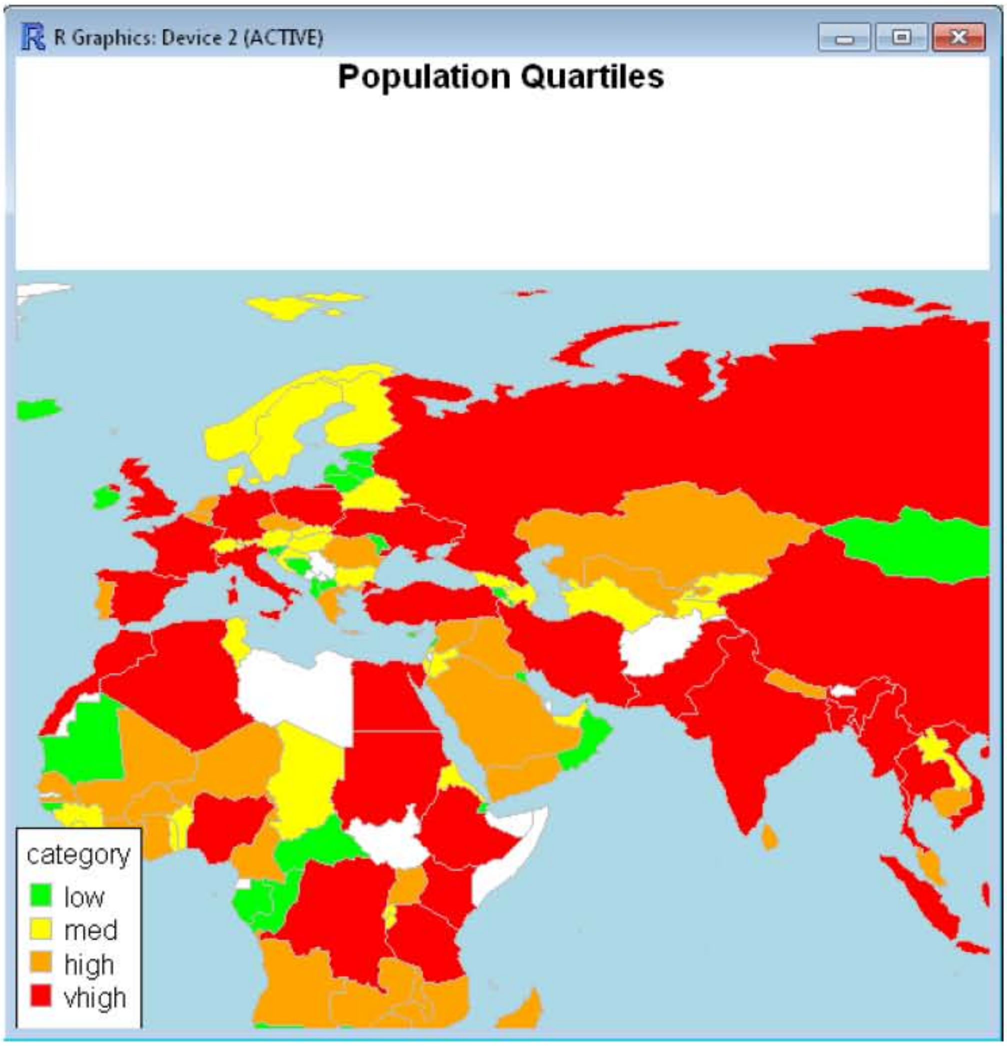
```
> par(mai=c(0,0,0.2,0),xaxs="i",yaxs="i")
> #joining the data to a map
> sPDF<-joinCountryData2Map(countryExData,joinCode="ISO3",nameJoinColumn="ISO3V10")
149 codes from your data successfully matched countries in the map
0 codes from your data failed to match with a country code in the map
95 codes from the map weren't represented in your data
> #creating a user defined colour palette
> op<- palette(c("green","yellow","orange","red"))
> #find quartile breaks
> cutVector<-quantile(sPDF@data[["Population2005"]],na.rm=TRUE)
> #classify the data to a factor
> sPDF@data[["Population2005"]]<-cut(sPDF@data[["Population2005"]],cutVector,
+ include.lowest=TRUE)
> #rename the categories
> levels(sPDF@data[["Population2005"]]) <- c("low","med","high","vhigh")
> #mapping
> mapCountryData(sPDF,nameColumnToPlot="Population2005",catMethod="categorical",
+ mapTitle="Population Quartiles", colourPalette="palette", oceanCol="lightblue",
+ missingCountryCol="white")
> 
```

ce qui donne:



et en rajoutant juste un argument nous pouvons zoomer sur une région donnée du monde:





Et si l'on possède des données avec des coordonnées pour les pays, il est possible de faire un diagramme à bulle sur la map monde:

```
R Console
> setwd("C:/")
> GeoStat<-read.csv("WorldBarPieBubble.csv",header=T,sep=";")
> par(mai=c(0,0,0,0))
> mapBubbles(GeoStat,nameX="LON",nameY="LAT",nameZSize="Population_2005",
+ nameZColour="MAP_COLOR",colourPalette="rainbow",oceanCol="lightblue",landCol="wheat")
> 
```

Ce qui donne:

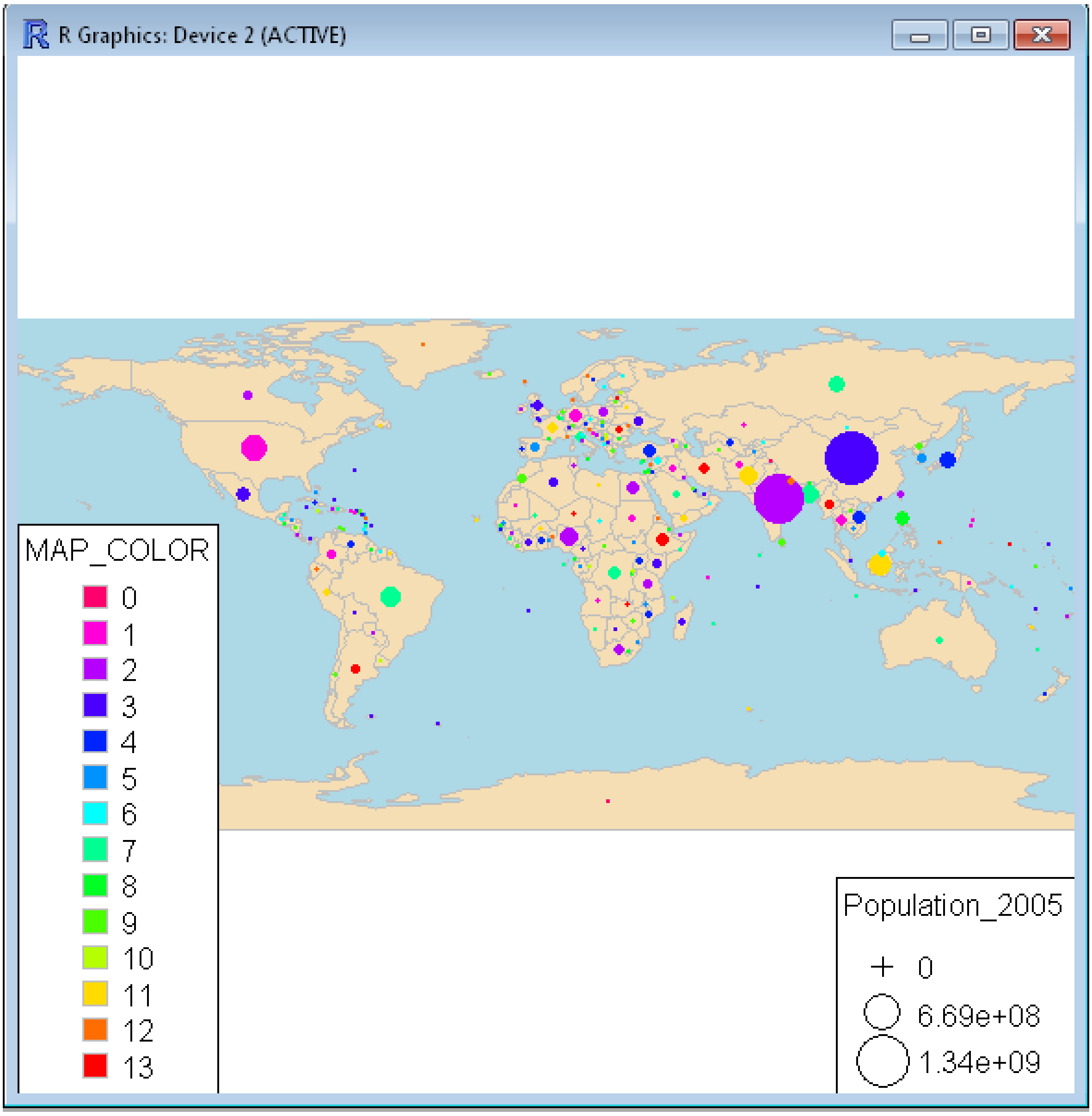


Figure 181 Graph de Map-Monde de Mercator avec bulles localisées

Statistiques au niveau mondial (package ggplot) avec projection cylindrique

Nous allons d'abord charger les packages et construire le jeu de données:

```
> library(rvest)
> library(ggplot2)
> library(dplyr)
>
> hci <- read_html("http://reports.weforum.org/human-capital-report-2016/rankings/") %>%
+   html_node("table") %>% html_table() %>%
+   rename(country = Economy, rank = `Overall Rank`, human_capital = `Overall Score`) %>%
+   mutate(country = recode(country,
+                           'Russian Federation' = 'Russia',
+                           'United Kingdom' = 'UK',
+                           'United States' = 'USA',
+                           'Korea, Rep.' = 'South Korea',
+                           'Kyrgyz Republic' = 'Kyrgyzstan',
+                           'Slovak Republic' = 'Slovakia',
+                           'Macedonia, FYR' = 'Macedonia',
+                           'Iran, Islamic Rep.' = 'Iran',
+                           'Trinidad and Tobago' = 'Trinidad',
+                           'Lao PDR' = 'Laos',
+                           'CÛte d'Ivoire' = 'Ivory Coast'))
> #ou vous pouvez charger le fichier HCI.csv
>
```

Ce qui donne:

```
> hci
   rank         country human_capital
1     1         Finland         85.86
2     2          Norway         84.64
3     3     Switzerland         84.61
4     4           Japan         83.44
5     5          Sweden         83.29
6     6     New Zealand         82.79
7     7         Denmark         82.47
8     8     Netherlands         82.18
9     9          Canada         81.95
10   10         Belgium         81.59
11   11         Germany         81.55
12   12         Austria         81.52
13   13       Singapore         80.94
14   14         Ireland         80.79
15   15         Estonia         80.63
16   16        Slovenia         80.33
17   17          France         80.32
18   18       Australia         80.08
19   19              UK         80.04
20   20         Iceland         79.74
21   21       Lithuania         79.34
```

et ensuite on y va pour le graphique:

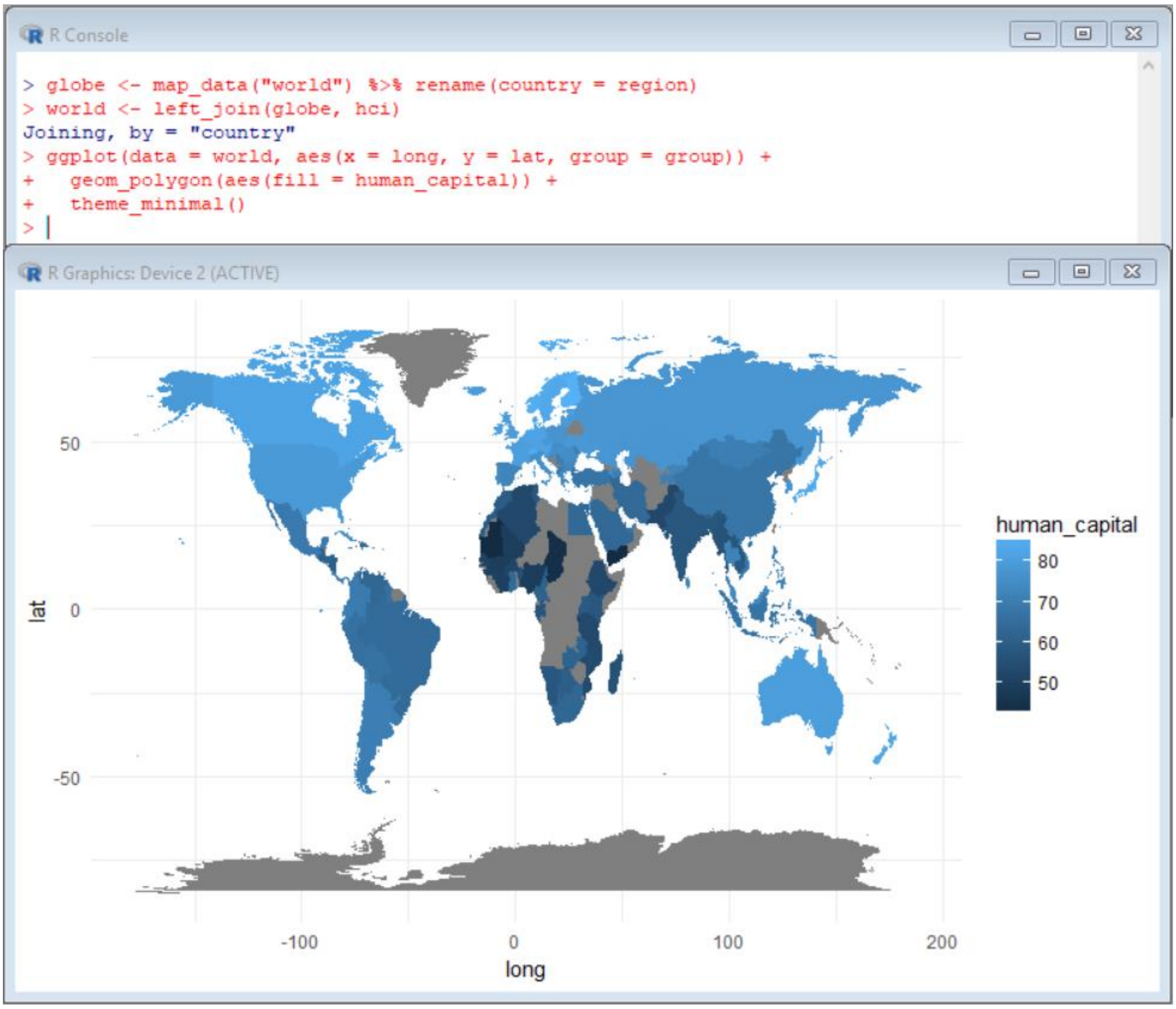


Figure 182 Graph Map-Monde de Mercator avec zones géographiques (ggplot)

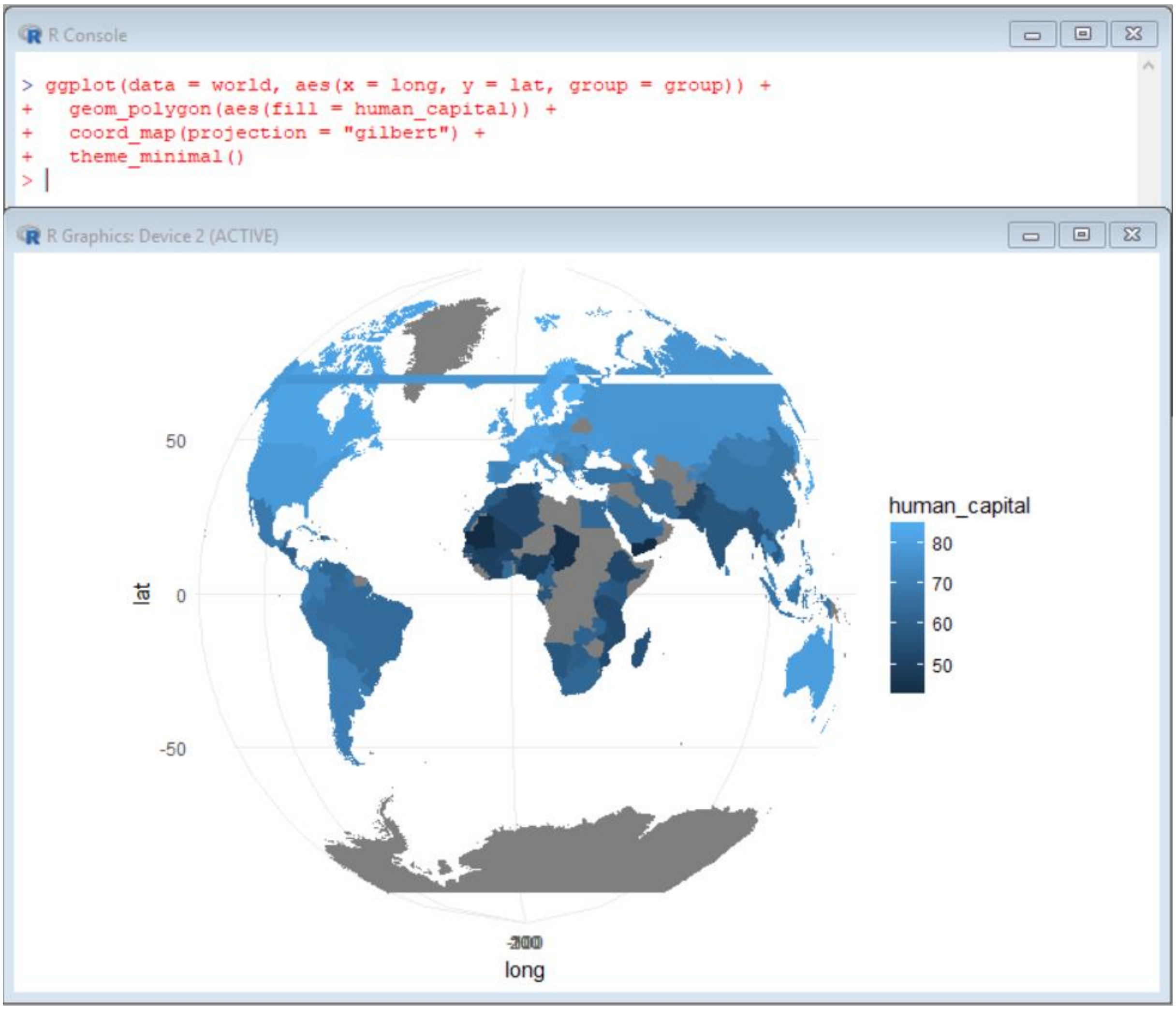


Figure 183 Graph Map-Monde de Gilbert avec zones géographiques (ggplot)

Statistiques au niveau mondial (package plot3D) avec projection cylindrique

Voilà une commande avec le package qui va en ravir plus d'un! Sous l'hypothèse que nous ayons une matrice dont le nombre de lignes est de 359 et le nombre de colonnes de 180 (correspondant respectivement aux angles de longitudes et latitudes) avec pour chaque composante une valeur physique mesurée, alors avec le package **plot3D**, nous pouvons obtenir en utilisant la commande **image2D( )** le résultat suivant (plus besoin de MATLAB pour cela... Yeeees!):

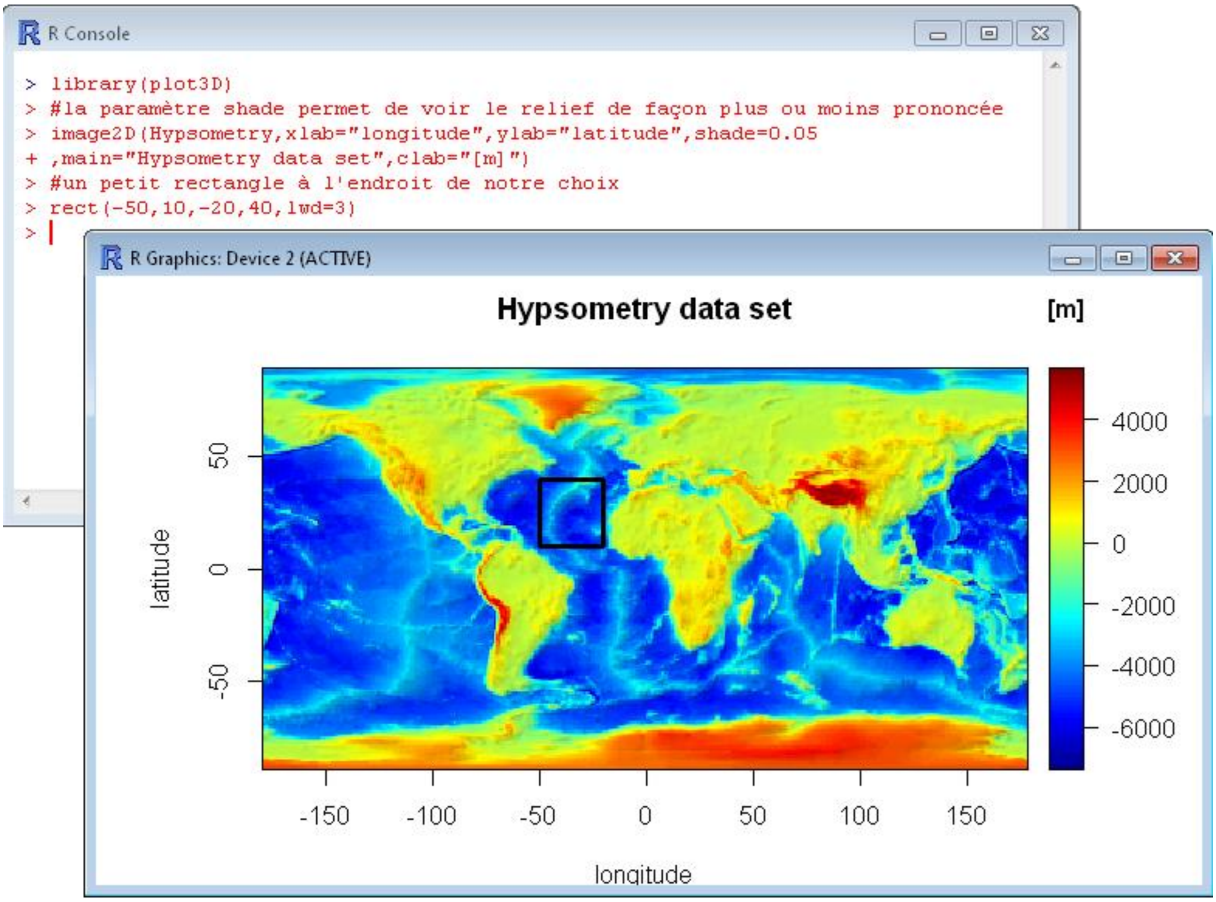


Figure 184 Graph Map-Monde de Mercator hyperréaliste hypsométrique

Maintenant toujours avec le même package zoomons en perspective sur la région mise en évidence:

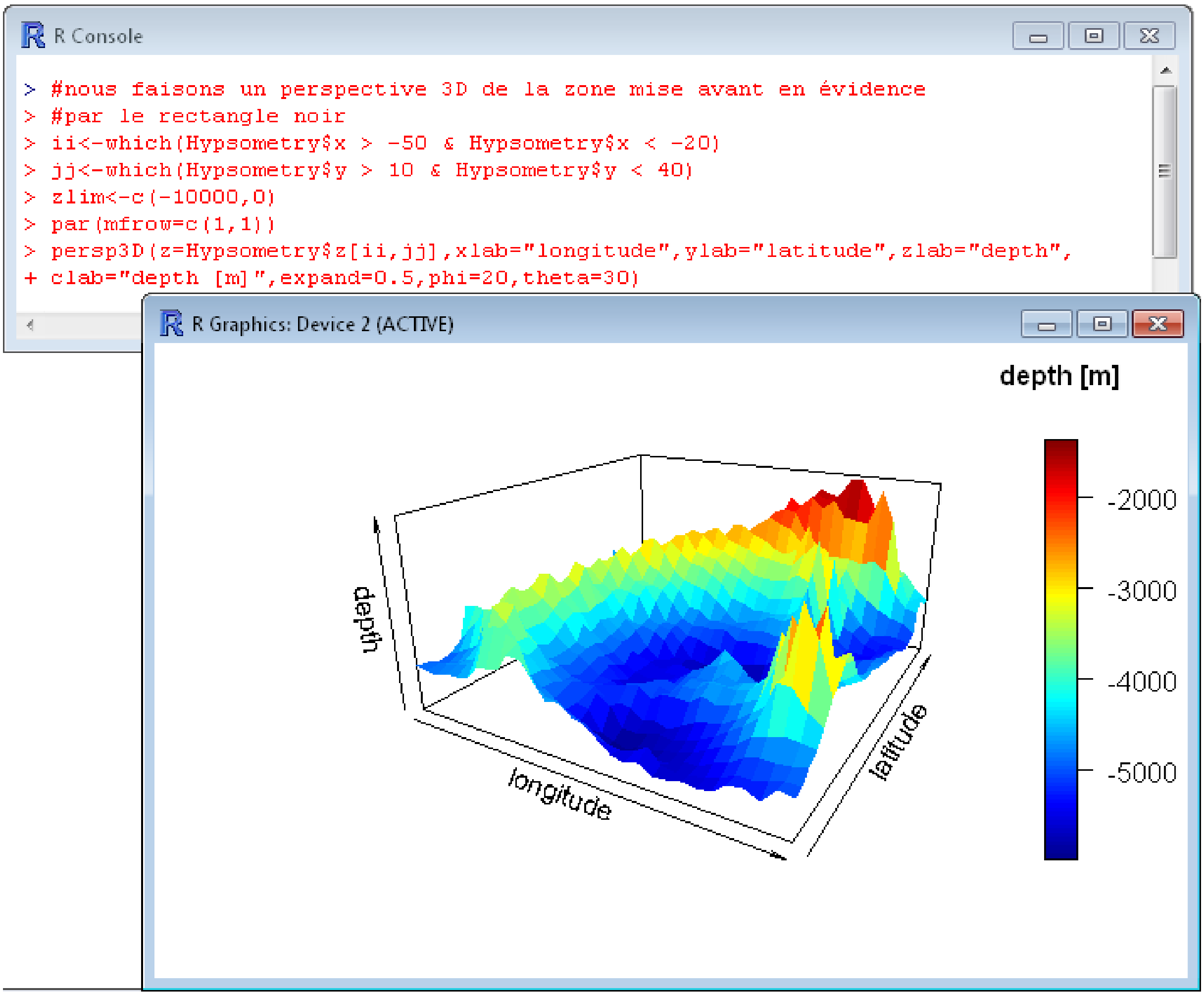
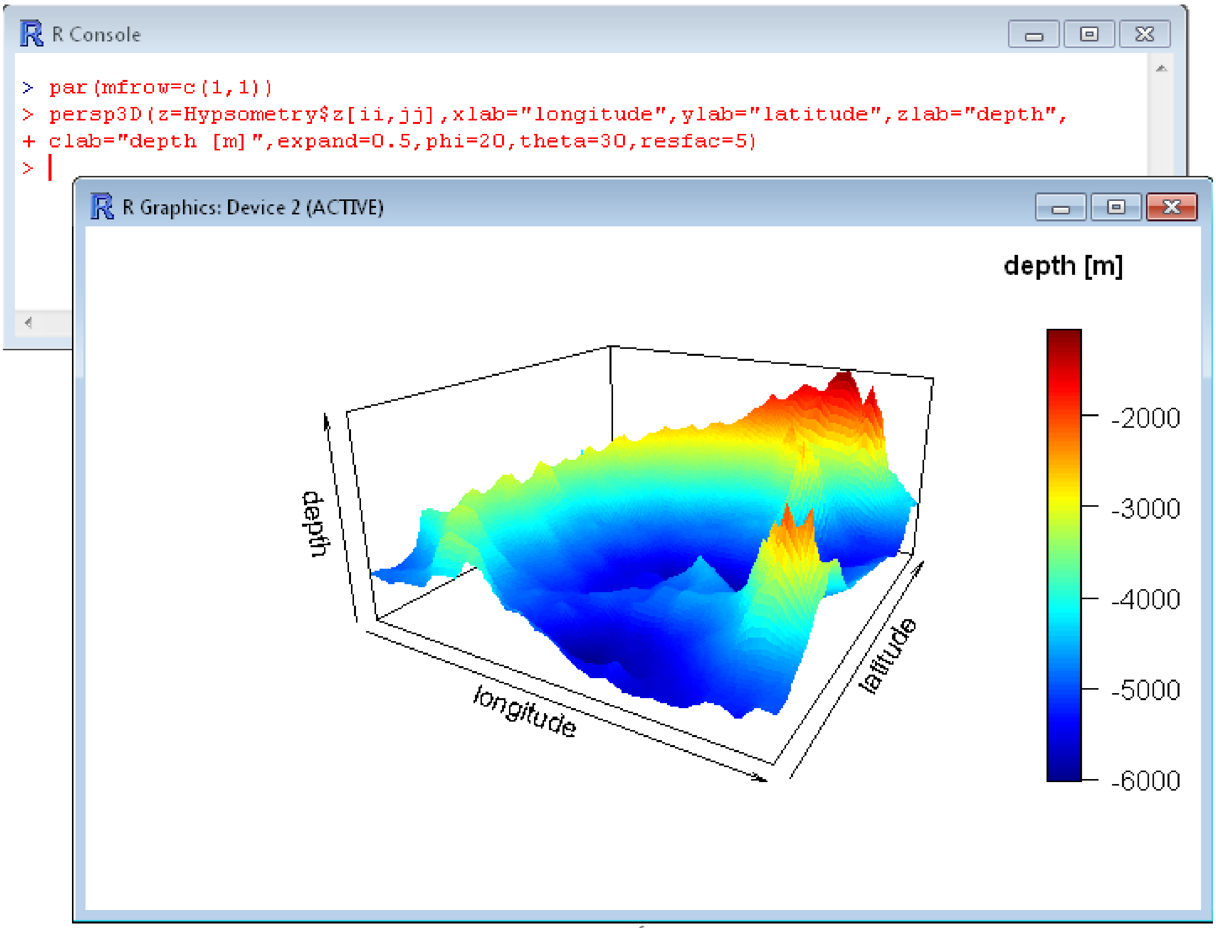


Figure 185 Graph bassin hypsométrique

Pas mal... mais nous allons faire un peu mieux en ajoutant un seul paramètre de résolution:



Bon c'est pas mal du tout (là aussi plus besoin de MATLAB pour cela!!!!). Faisons encore mieux en ajoutant les isoclines:

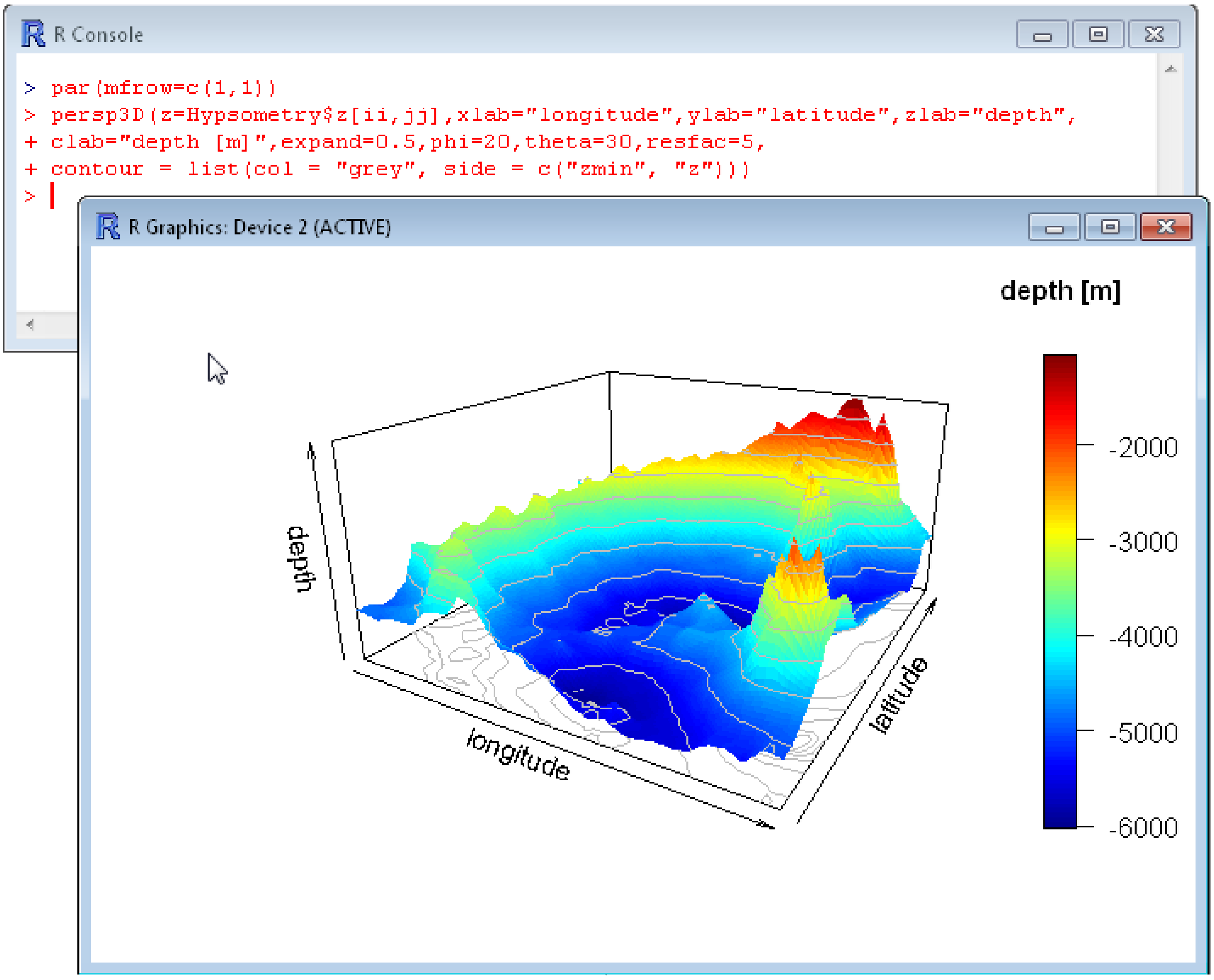
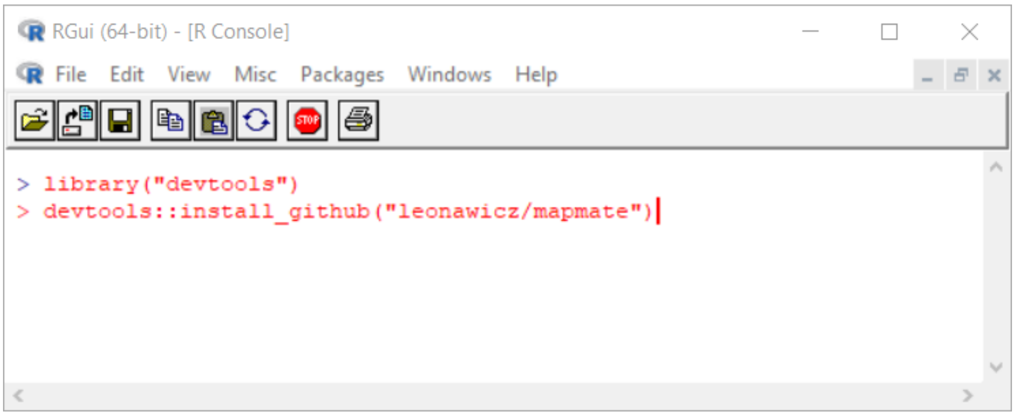


Figure 186 Graph bassin hypsométrique avec isoclines

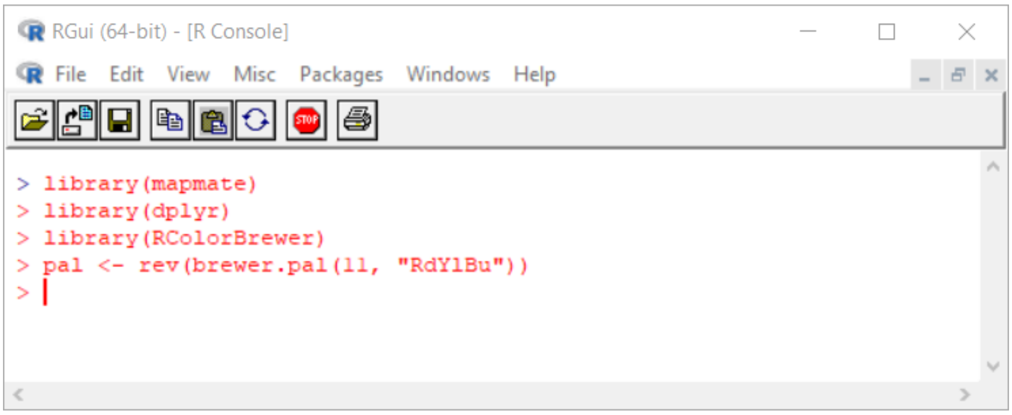
Statistiques au niveau mondial sans projection

Avec le package mapmate

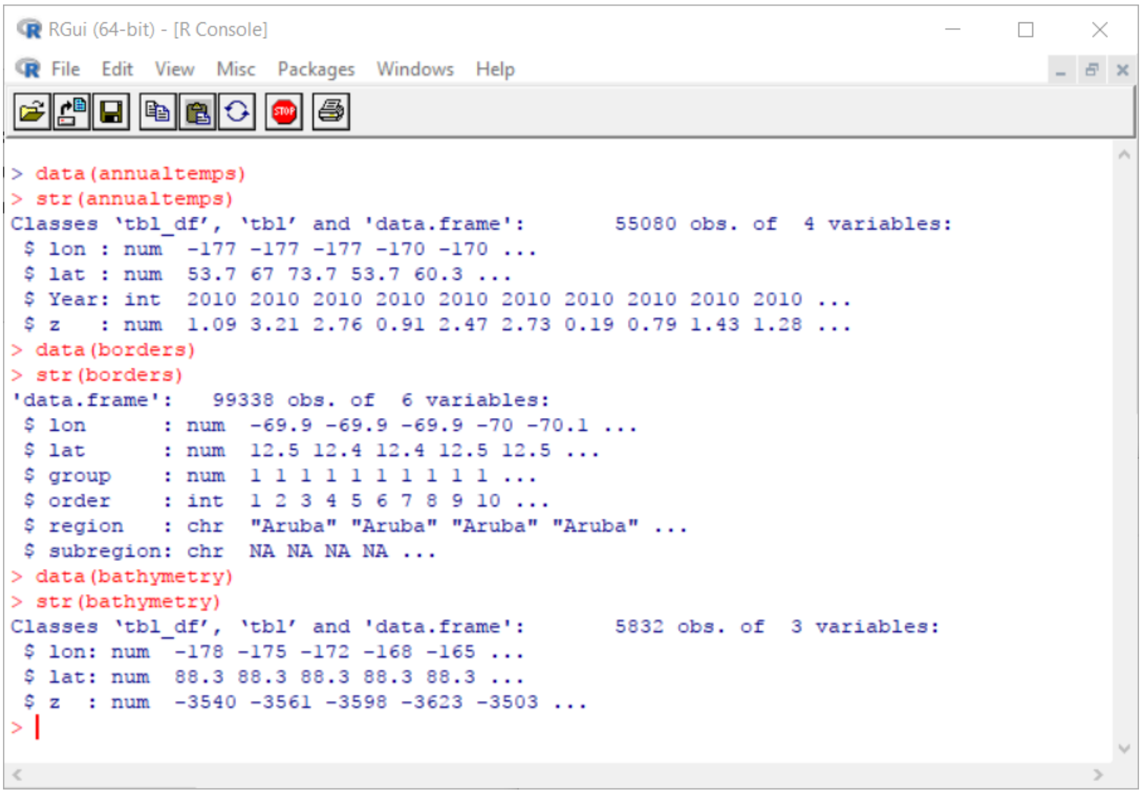
Le package **mapmate** est actuellement que sur GitHub, nous allons donc devoir charger le package **devtools** dans un premier temps (**mais attention!!! Sous Microsoft Windows, l'installation se passe souvent mal si le compte utilisateur contient des espaces!**):



Ensuite, nous chargeons les packages dont nous allons avoir besoin pour l'exemple et nous créons également une palette de couleurs au passage:

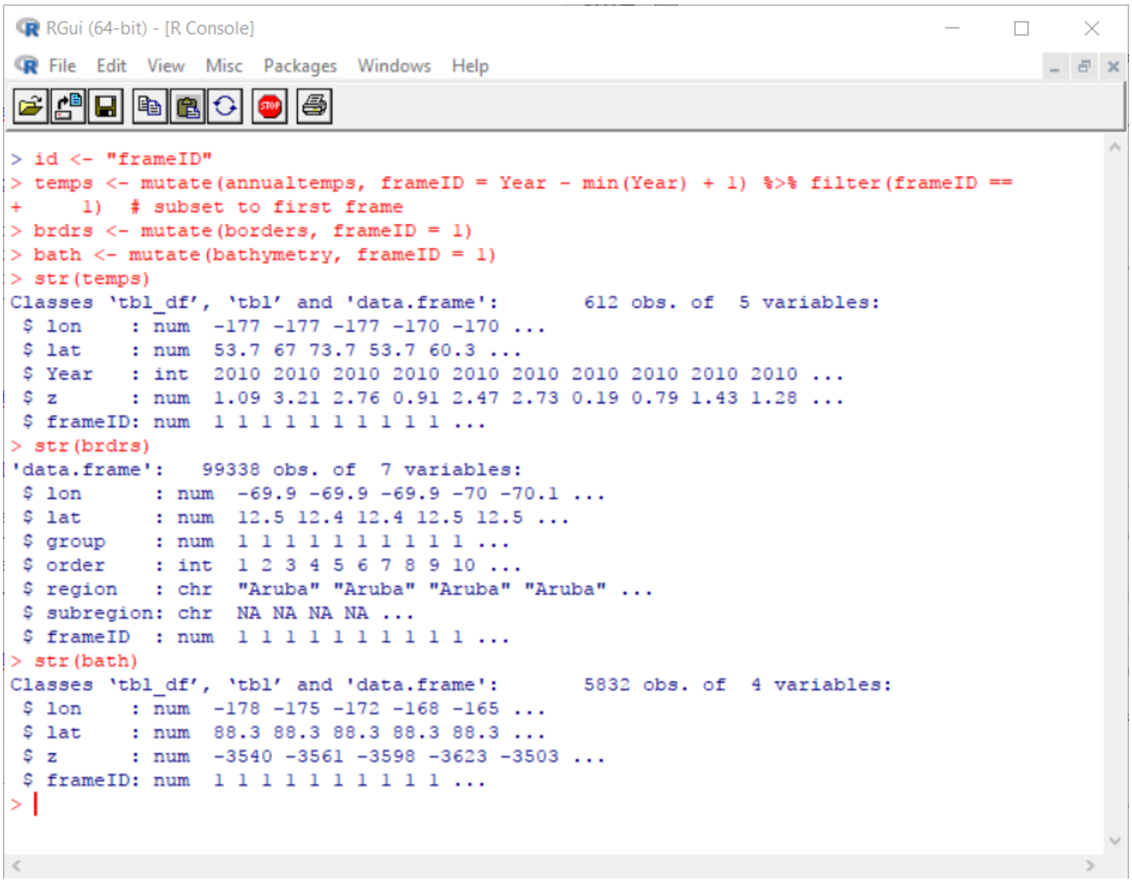


Pour montrer comment le page **mapmate** fonctionner, nous allons utiliser les trois jeux de données suivants:



Trivialement… le jeu de données *borders* est le plus important pour tracer les frontières sur la planète!

Ensuite on transforme un tout petit peu les données avec **dplyr**:



On y va pour le premier graphique avec la fonction **save_map( )** :

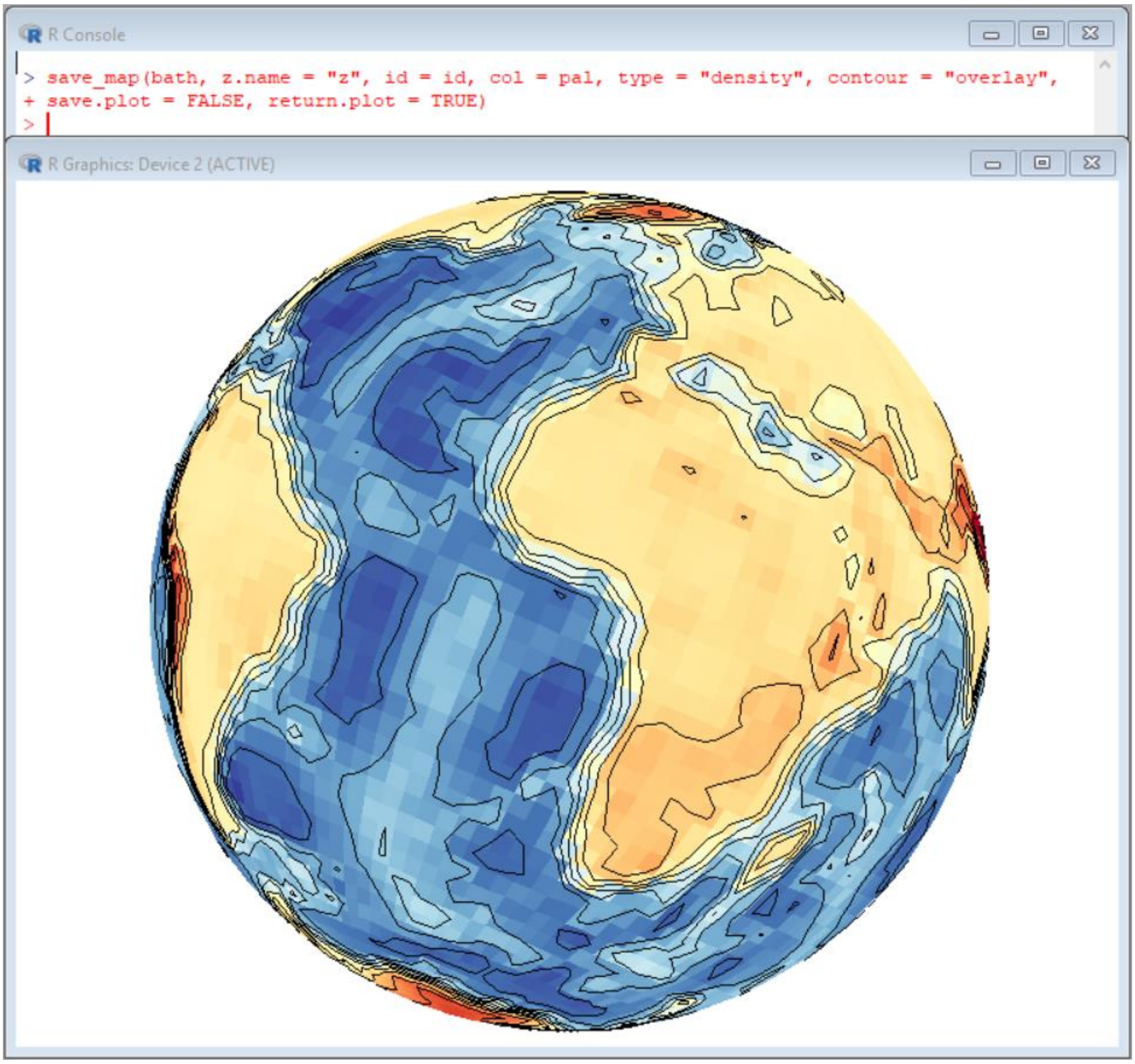
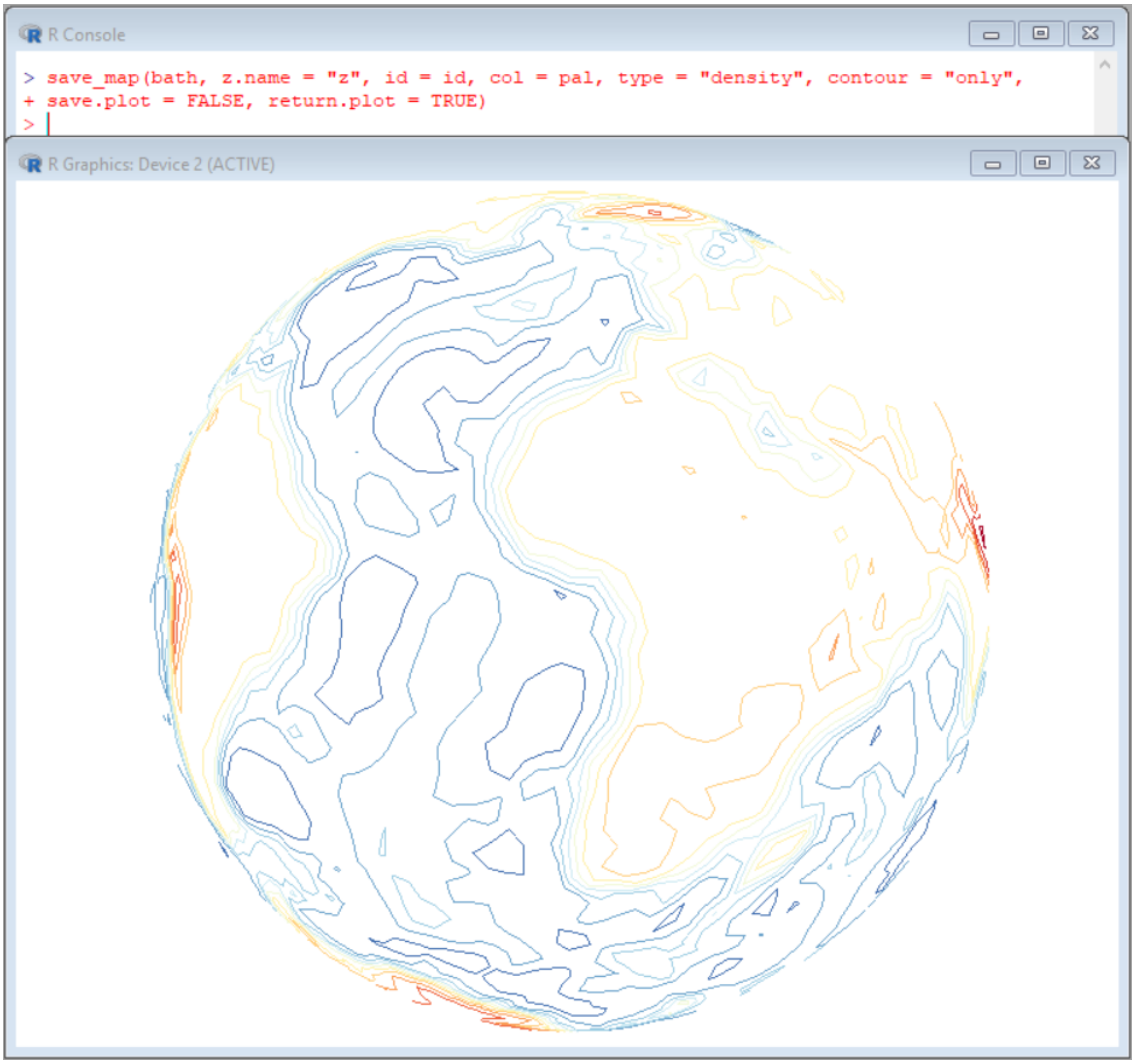
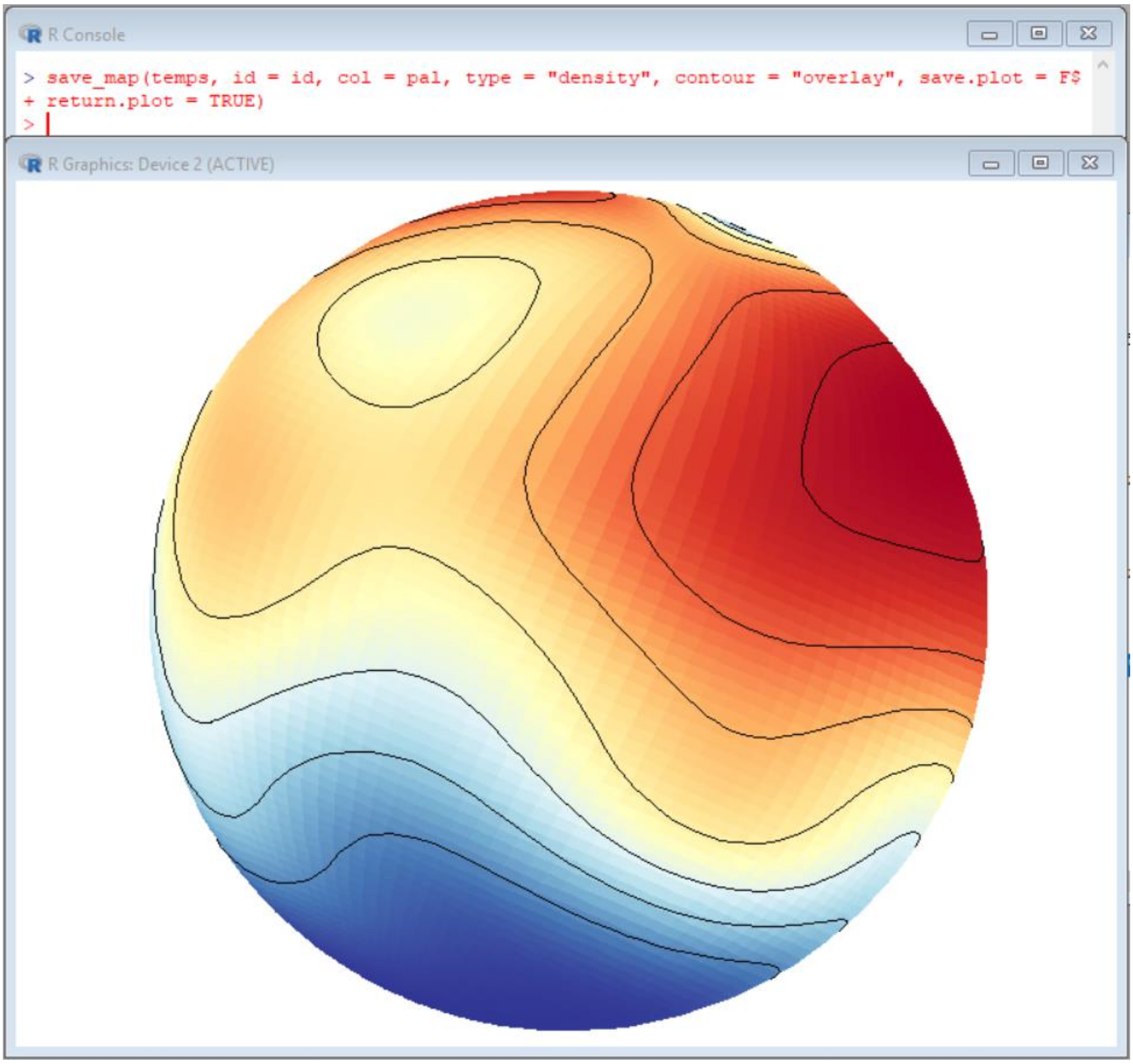


Figure 187 Graph globle Mondial avec iso-courbes et densités de couleurs

Ou en changeant le paramètre *contour*:



ou encore en changeant avec un jeu de données ayant beaucoup plus de points:



Pour plus d'exemples et d'informations, le lecteur pourra se reporter à l'adresse suivante:

https://leonawicz.github.io/mapmate/articles/usage_and_limitations.html

Avec le package rworldmap

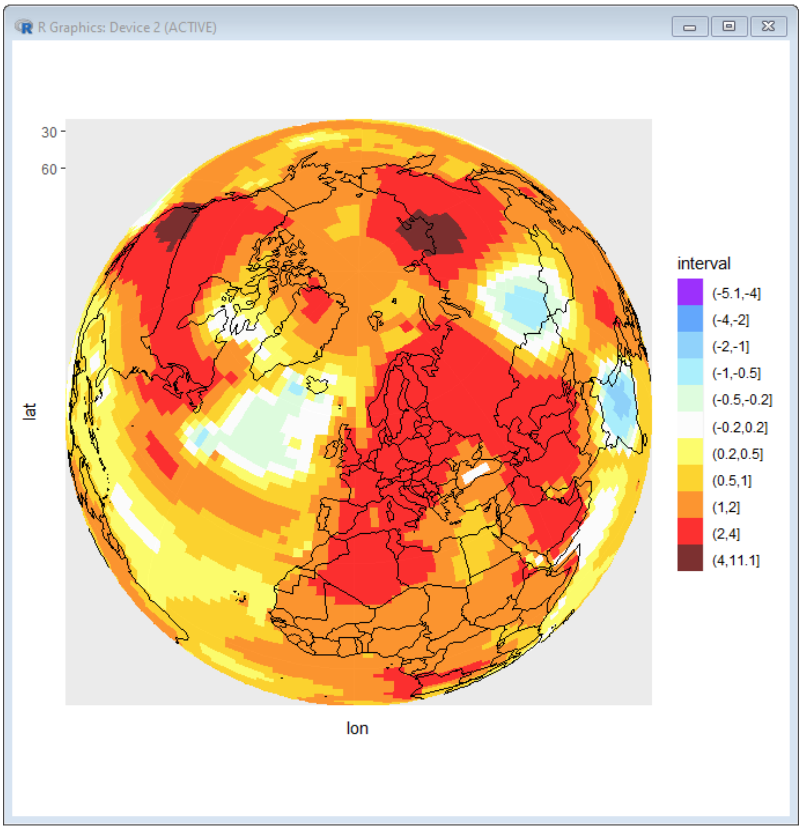
Ce qui suit est simplement une copie du code disponible ici:

https://egallic.fr/en/maps-with-r/

Voyons d'abord le graphique lui-même:

```
> library("rworldmap")
Loading required package: sp
### Welcome to rworldmap ###
For a short introduction type :          vignette('rworldmap')
> library("ggplot2")
>
> #World map nécessaire pour la suite
> worldMap <- getMap()
> world.points <- fortify(worldMap)
Regions defined for each Polygons
> world.points$region <- world.points$id
> world.df <- world.points[,c("long","lat","group", "region")]
>
> temp <- read.table("amaps.txt", skip=1, header = TRUE, na.strings="9999.0000")
> str(temp)
'data.frame':   16200 obs. of  5 variables:
 $ i         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ j         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ lon       : num  -179 -177 -175 -173 -171 -169 -167 -165 -163 -161 ...
 $ lat       : num  -89 -89 -89 -89 -89 -89 -89 -89 -89 -89 ...
 $ array.i.j.: num  -2.34 -2.34 -2.34 -2.34 -2.34 ...
> breaks <- c(-5.1, -4, -2, -1, -.5, -.2, .2, .5, 1, 2, 4, 11.1)
> lesCouleurs <- cbind(
+   val = levels(cut(range(breaks), breaks = breaks)),
+   col = c("#8600FF", "#3F94FE", "#77CAFD", "#99EEFF", "#D9FFD9", "#FFFFFF",
+           "#FFFF4C", "#FFCC00", "#FF7E00", "#FF0000", "#5E0000")
+ )
> lesCouleurs <- data.frame(lesCouleurs, stringsAsFactors = FALSE)
> colnames(lesCouleurs) <- list("val", "col")
> # Colours in 8 digits, the last two digits are for transparency
> lesCouleurs$col <- paste(lesCouleurs$col,"FF", sep = "")
> temp$interval <- cut(temp$array.i.j, breaks = breaks)
>
> rotate_map <- function(angle = -74){
+   ggplot() +
+     geom_tile(data = temp, aes(x = lon, y = lat, fill = interval), alpha = 0.8) +
+     scale_fill_manual("interval", breaks = lesCouleurs$val, values = lesCouleurs$col) +
+     geom_path(data = world.df, aes(x = long, y = lat, group = group)) +
+     scale_y_continuous(breaks = (-2:2) * 30) +
+     scale_x_continuous(breaks = (-4:4) * 45) +
+     coord_map("ortho", orientation=c(61, angle, 0))
+ }
>
> rotate_map(1)
>
```

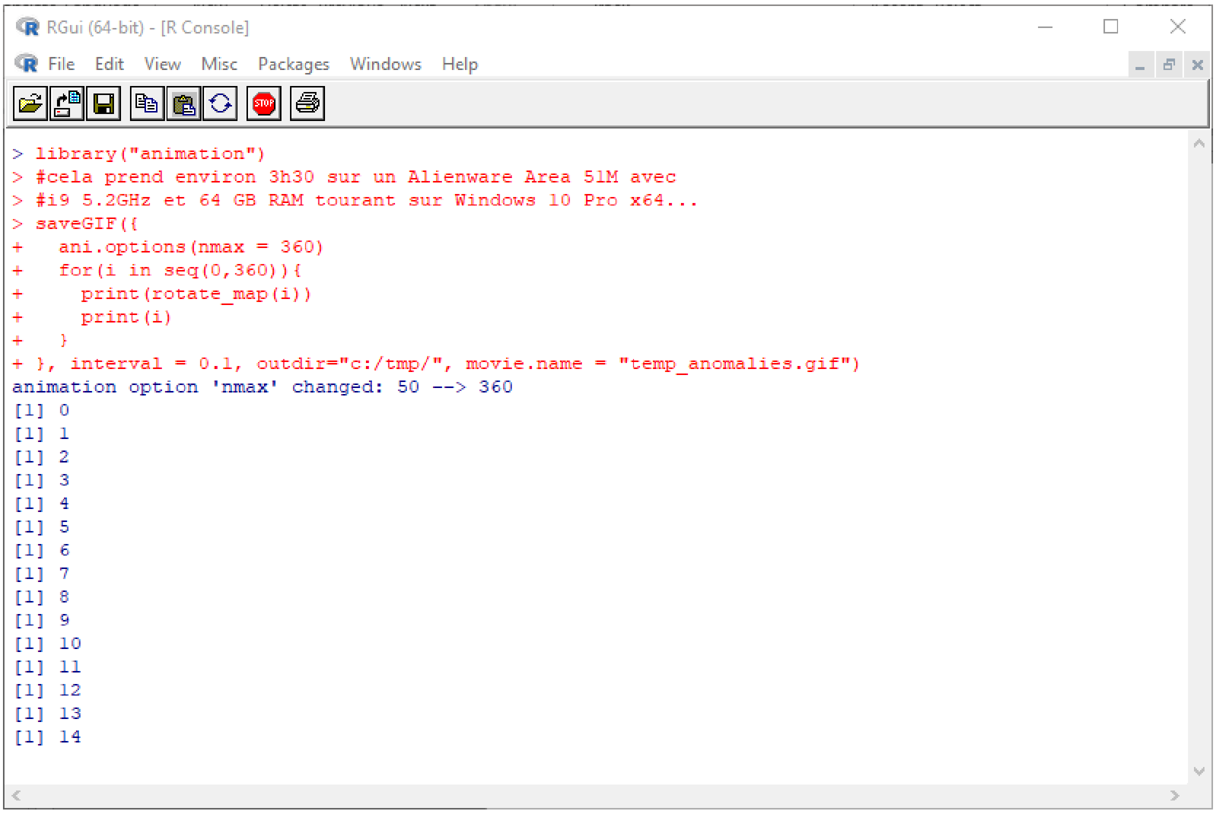
Ce qui donne:



On peut aussi créer une animation dont le résultat est visible ici:

www.sciences.ch/tmp/temp_anomalies.gif

en utilisant le code suivant:



```
> library("animation")
> #cela prend environ 3h30 sur un Alienware Area 51M avec
> #i9 5.2GHz et 64 GB RAM tourant sur Windows 10 Pro x64...
> saveGIF({
+   ani.options(nmax = 360)
+   for(i in seq(0,360)){
+     print(rotate_map(i))
+     print(i)
+   }
+ }, interval = 0.1, outdir="c:/tmp/", movie.name = "temp_anomalies.gif")
animation option 'nmax' changed: 50 --> 360
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
[1] 13
[1] 14
```

Points au niveau mondial (package ggOceanMaps) avec projection orthogonale

Bon précisons d'abord que ce package est très vaste en possibilités. Pour plus d'informations, le lecteur pourra se référa à la page web de la documentation:

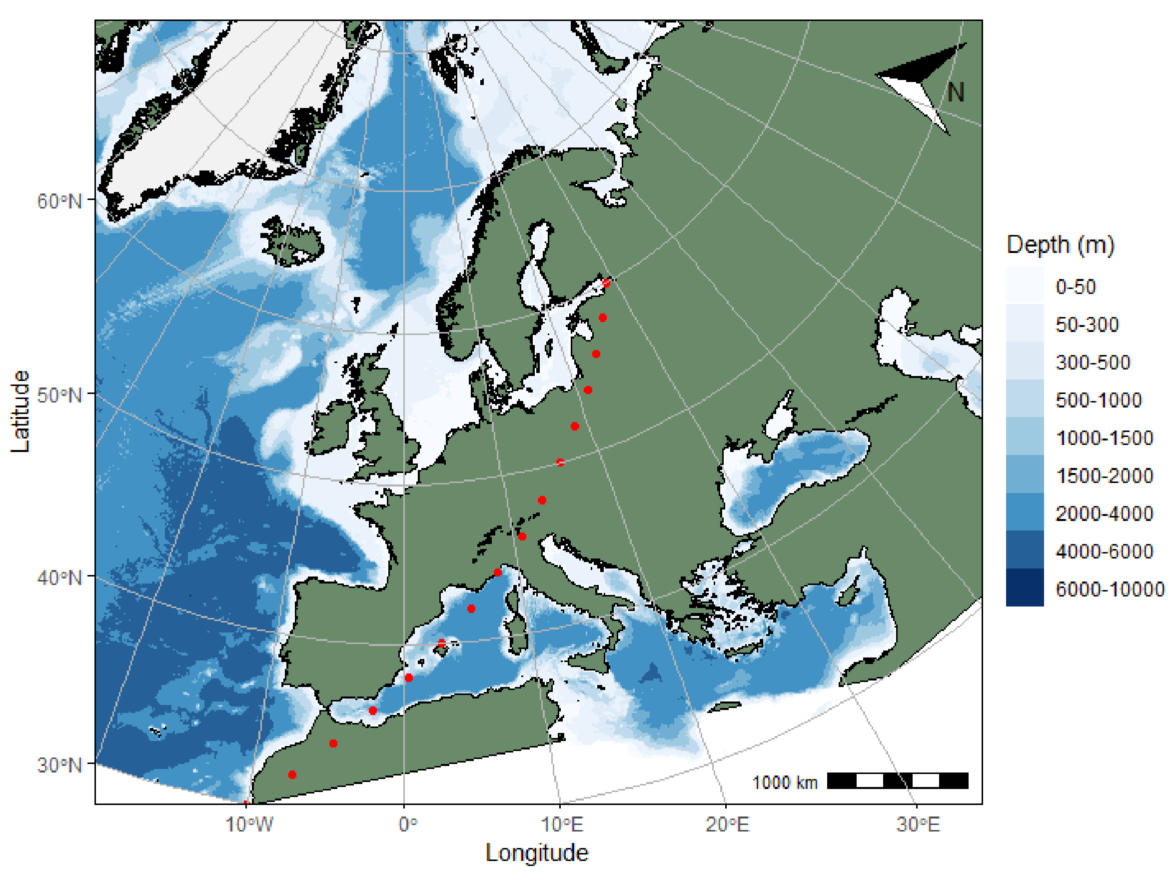
https://mikkovihtakari.github.io/ggOceanMaps/

Voyons maintenant un exemple:

R Console

```
> #devtools::install_github("MikkoVihtakari/ggOceanMaps")
> library("ggOceanMaps")
>
> #un petit jeu de données comme exemple d'ajouts de points
> dt <- data.frame(lon =seq(-10,+30,2.6) , lat = seq(+30,+60,2))
>
> basemap(limits = c(-20, +40, +30, +80), bathymetry = TRUE, glaciers = TRUE,
+          land.col="darkseagreen4",lon.interval = 10,lat.interval = 10)+
+        xlab("Longitude")+ylab("Latitude")+
+ geom_point(data = transform_coord(dt), aes(x = lon, y = lat), color = "red")+
+ annotation_scale(location = "br")+
+ annotation_north_arrow(location = "tr", which_north = "true")
Using lon and lat as longitude and latitude columns, respectively.
projection transformed from EPSG:4326 to EPSG:3995
Scale on map varies by more than 10%, scale bar may be inaccurate
> |
```

R Graphics: Device 2 (ACTIVE)



Statistiques au niveau Européen régional

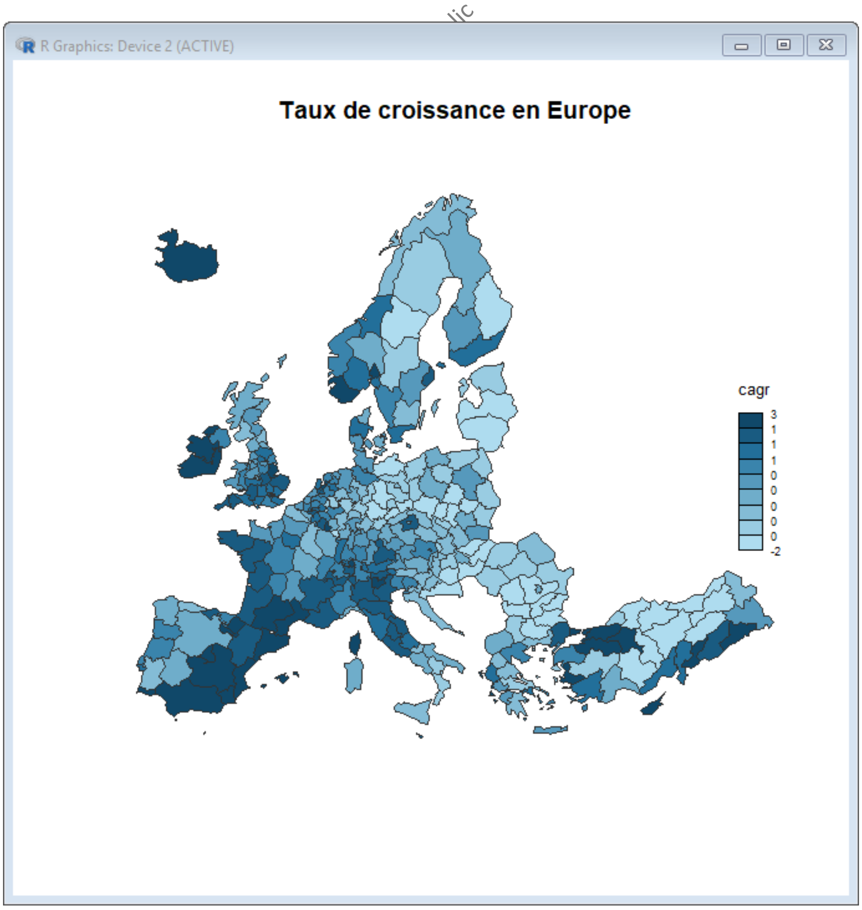
Je dois de présenter cet exemple grâce à une participante à mes cours. À noter que l'exemple est 100% inspiré du lient suivant de Timothée Giraud (CNRS - UMS RIATE):

http://wukan.ums-riate.fr/ifsttar/

Voici le code qui utilise la fonction **choroLayer( )** du package cartagrophy:

```
> library(cartography)
> data(nuts2006)
> head(nuts2.df)
       id emp2008 act2008 unemp2008 birth_2008 death_2008 gdpppps1999 gdpppps2008 pop1999 pop2008
383  AT11   137.0   142.1       5.1       2215       3067        4139        5695  276732  281185
384  AT12   792.3   820.3      28.0      14085      15616       28799       40998 1530745 1596538
385  AT13   784.6   841.4      56.7      17375      15702       51431       68722 1542252 1674909
789  AT21   263.4   272.8       9.4       4718       5385       10886       14632  560938  560322
1192 AT22   586.5   607.0      20.5      10255      11725       23436       32457 1183374 1204947
368  AT31   714.1   733.4      19.2      13540      11966       30468       43300 1366578 1406664
>
> # Calcul du taux de croissance annuel moyen
> nuts2.df$cagr <- 100 * (((nuts2.df$pop2008/nuts2.df$pop1999)^(1/9)) - 1)
> choroLayer(spdf = nuts2.spdf, df = nuts2.df, var = "cagr",legend.pos = "right")
> title("Taux de croissance en Europe")
>
```

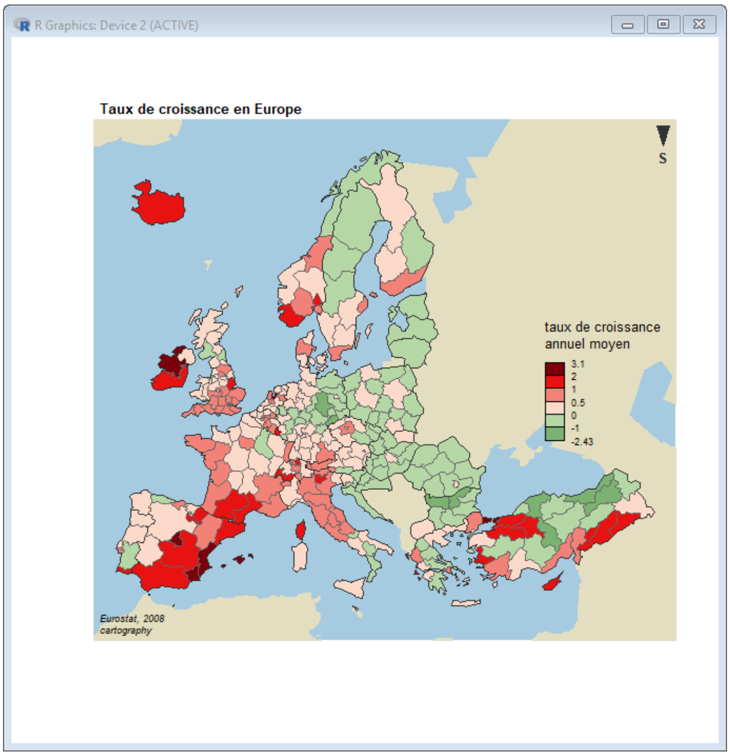
Ce qui donne:



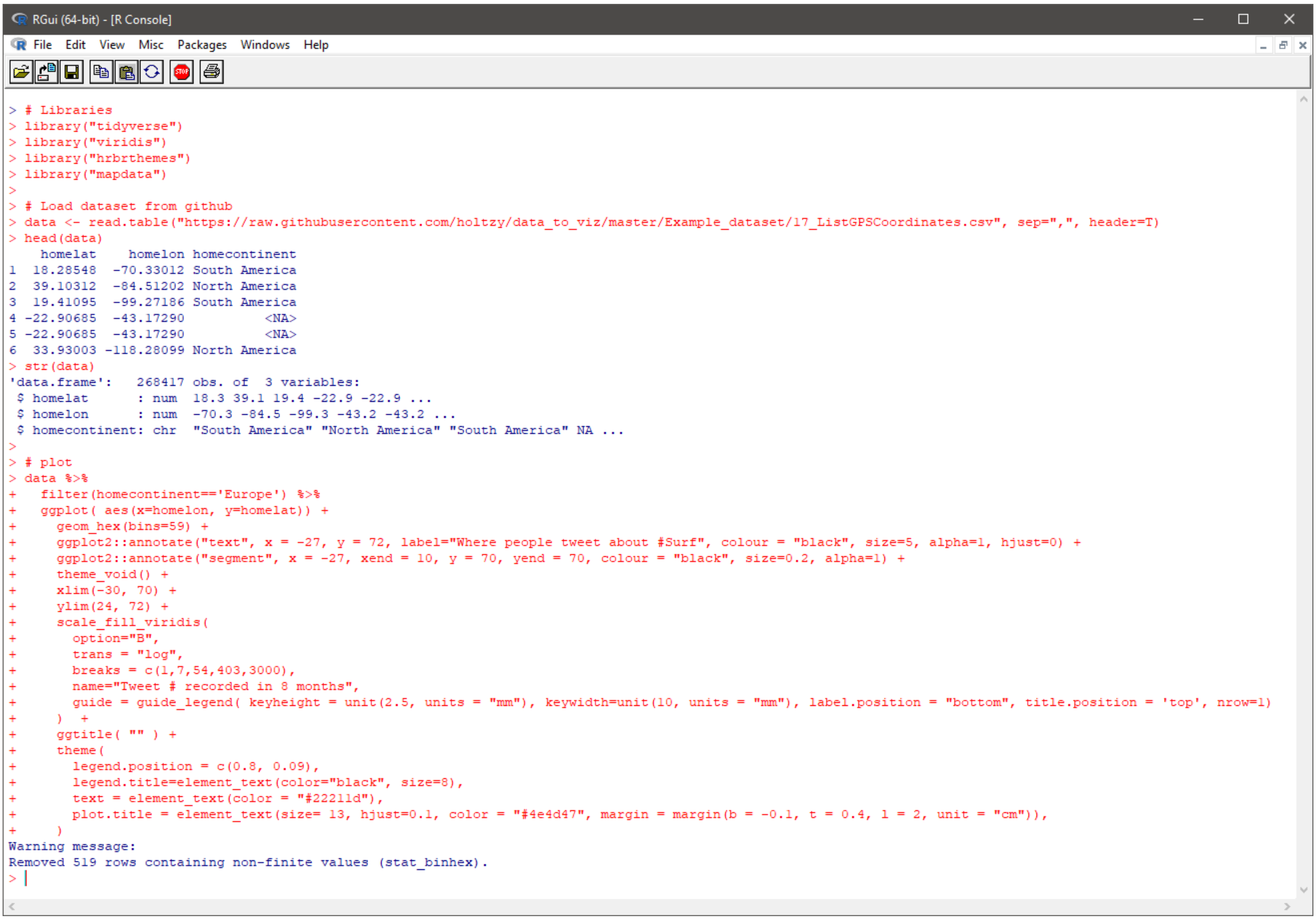
On peut faire quelque chose de très différent aussi:

```
> # Construire une palette de couleurs
> cols <- carto.pal(pal1="green.pal",n1=2,pal2="red.pal",n2 = 4)
>
> # Affichage de couches d'habillage
> plot(nuts0.spdf, border = NA, col = NA, bg = "#A6CAE0")
> plot(world.spdf, col = "#E3DEBF", border = NA, add = TRUE)
>
> # Cartographie du taux de croissance annuel moyen
> choroLayer(spdf = nuts2.spdf, df = nuts2.df, var = "cagr",
+     breaks = c(-2.43, -1, 0, 0.5, 1, 2, 3.1), col = cols,
+     border = "grey40", lwd = 0.5, legend.pos = "right",
+     legend.title.txt = "taux de croissance\nannuel moyen",
+     legend.values.rnd = 2, add = TRUE)
>
> # Affichage de couches d'habillage
> plot(nuts0.spdf, border = "grey20", lwd = 0.75, add = TRUE)
>
> # Ajout des titres, légende, sources, etc.
> layoutLayer(title = "Taux de croissance en Europe",
+     author = "cartography", sources = "Eurostat, 2008",
+     frame = TRUE, col = NA, scale = NULL, coltitle = "black",
+     south = TRUE)
>
```

Ce qui donnera:



Ou encore avec le package ggplot, nous pouvons faire des cartes hexabin:



```
> # Libraries
> library("tidyverse")
> library("viridis")
> library("hrbrthemes")
> library("mapdata")
>
> # Load dataset from github
> data <- read.table("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/17_ListGPSCoordinates.csv", sep=",", header=T)
> head(data)
    homelat    homelon homecontinent
1  18.28548  -70.33012 South America
2  39.10312  -84.51202 North America
3  19.41095  -99.27186 South America
4 -22.90685  -43.17290          <NA>
5 -22.90685  -43.17290          <NA>
6  33.93003 -118.28099 North America
> str(data)
'data.frame':   268417 obs. of  3 variables:
 $ homelat      : num  18.3 39.1 19.4 -22.9 -22.9 ...
 $ homelon      : num  -70.3 -84.5 -99.3 -43.2 -43.2 ...
 $ homecontinent: chr  "South America" "North America" "South America" NA ...
>
> # plot
> data %>%
+   filter(homecontinent=='Europe') %>%
+   ggplot( aes(x=homelon, y=homelat)) +
+     geom_hex(bins=59) +
+     ggplot2::annotate("text", x = -27, y = 72, label="Where people tweet about #Surf", colour = "black", size=5, alpha=1, hjust=0) +
+     ggplot2::annotate("segment", x = -27, xend = 10, y = 70, yend = 70, colour = "black", size=0.2, alpha=1) +
+     theme_void() +
+     xlim(-30, 70) +
+     ylim(24, 72) +
+     scale_fill_viridis(
+       option="B",
+       trans = "log",
+       breaks = c(1,7,54,403,3000),
+       name="Tweet # recorded in 8 months",
+       guide = guide_legend( keyheight = unit(2.5, units = "mm"), keywidth=unit(10, units = "mm"), label.position = "bottom", title.position = 'top', nrow=1)
+     )  +
+     ggtitle( "" ) +
+     theme(
+       legend.position = c(0.8, 0.09),
+       legend.title=element_text(color="black", size=8),
+       text = element_text(color = "#22211d"),
+       plot.title = element_text(size= 13, hjust=0.1, color = "#4e4d47", margin = margin(b = -0.1, t = 0.4, l = 2, unit = "cm")),
+     )
Warning message:
Removed 519 rows containing non-finite values (stat_binhex).
>
```

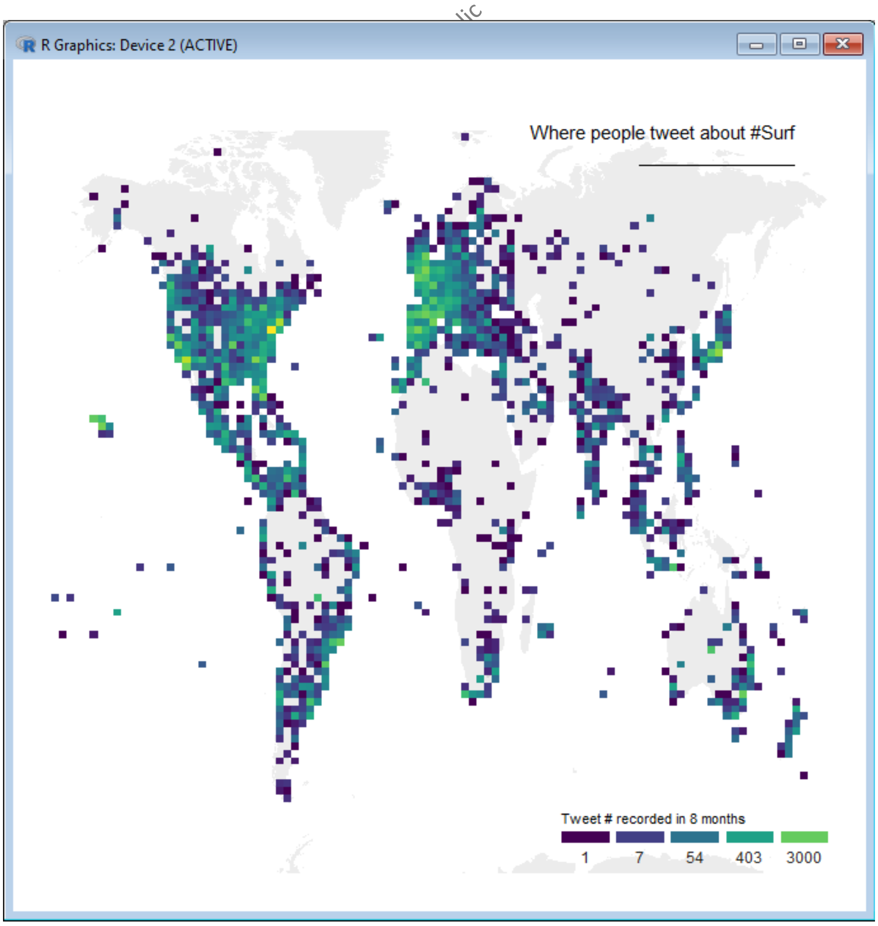
Ce qui donne:



ou avec la carte du monde et le même jeu de données:

```
> # Libraries
> library("tidyverse")
> library("viridis")
> library("hrbrthemes")
> library("mapdata")
>
> # Load dataset from github
> data <- read.table("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/17_ListGPSCoordinates.csv", sep=",", header=T)
>
> # Get the world polygon
> world <- map_data("world")
>
> # plot
> ggplot(data, aes(x=homelon, y=homelat)) +
+     geom_polygon(data = world, aes(x=long, y = lat, group = group), fill="grey", alpha=0.3) +
+     geom_bin2d(bins=100) +
+     ggplot2::annotate("text", x = 175, y = 80, label="Where people tweet about #Surf", colour = "black", size=4, alpha=1, hjust=1) +
+     ggplot2::annotate("segment", x = 100, xend = 175, y = 73, yend = 73, colour = "black", size=0.2, alpha=1) +
+     theme_void() +
+     ylim(-70, 80) +
+     scale_fill_viridis(
+       trans = "log",
+       breaks = c(1,7,54,403,3000),
+       name="Tweet # recorded in 8 months",
+       guide = guide_legend( keyheight = unit(2.5, units = "mm"), keywidth=unit(10, units = "mm"), label.position = "bottom", title.position = 'top', nrow=1)
+     )  +
+     ggtitle( "" ) +
+     theme(
+       legend.position = c(0.8, 0.09),
+       legend.title=element_text(color="black", size=8),
+       text = element_text(color = "#22211d"),
+       plot.title = element_text(size= 13, hjust=0.1, color = "#4e4d47", margin = margin(b = -0.1, t = 0.4, l = 2, unit = "cm")),
+     )
Warning message:
Removed 4 rows containing non-finite values (stat_bin2d).
>
```

Ce qui donne:

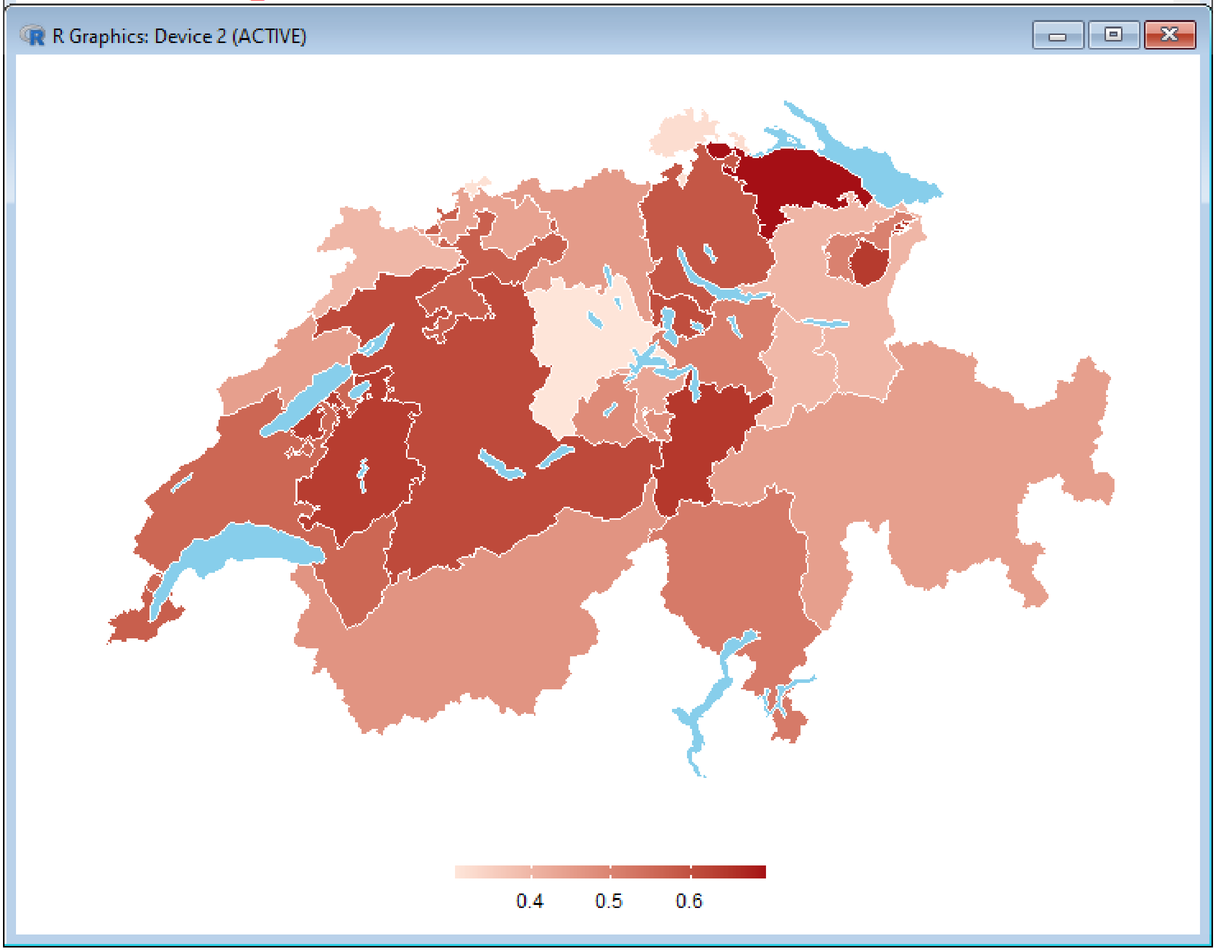


Package dédié pour la Suisse: RSwissMaps

Voilà un package qui sera très utile pour mes clients vu que je travaille et fournit des conseils principalement en Suisse. Les médias en sont très friands surtout les votations puisque nous sommes un des rares pays étant une démocratie directe et donc des votations sont une chose fréquente chez nous.

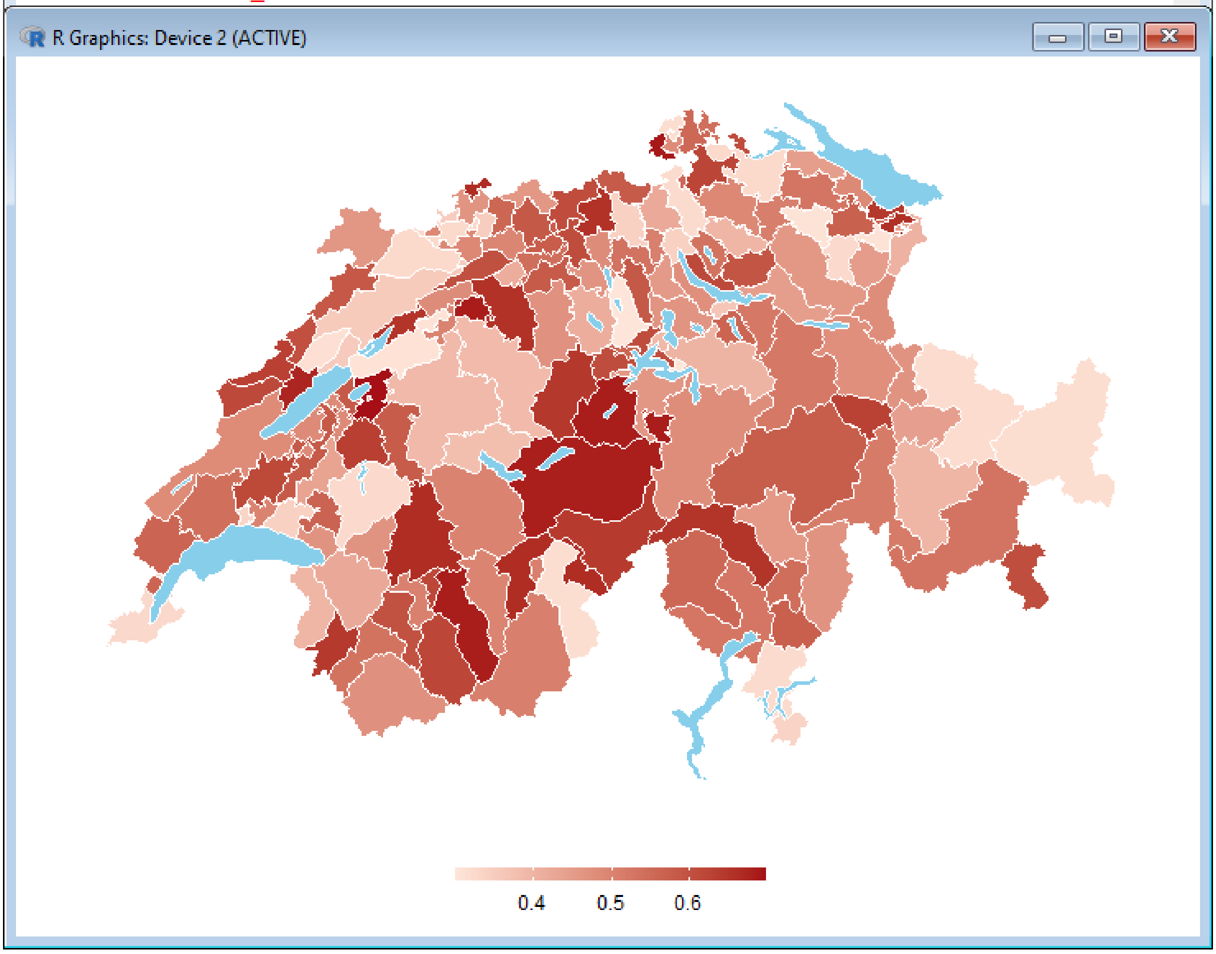
Voyons donc cela d'abord avec les cantons à l'aide de la fonction **can.plot( )**:

```
R Console
> library("RSwissMaps")
> # Generating sample data:
> dt <- can.template(2016)
> for(i in 1:nrow(dt)){dt$values[i] <- sample(c(300:700), 1)/1000}
> head(dt,10)
   bfs_nr name values
1       1   ZH  0.587
2       2   BE  0.611
3       3   LU  0.306
4       4   UR  0.642
5       5   SZ  0.511
6       6   OW  0.485
7       7   NW  0.431
8       8   GL  0.394
9       9   ZG  0.604
10     10   FR  0.635
> # Plotting sample data:
> can.plot(dt$bfs_nr, dt$values, 2016)
```



Voyons maintenant les districts avec la fonction **dis.plot( )**:

```
> library("RSwissMaps")
> # Generating sample data:
> dt <- dis.template(2016)
> for(i in 1:nrow(dt)){dt$values[i] <- sample(c(300:700), 1)/1000}#
> head(dt,10)
   bfs_nr                name values
1     101    Bezirk Affoltern  0.453
2     102  Bezirk Andelfingen  0.623
3     103       Bezirk Bülach  0.321
4     104    Bezirk Dielsdorf  0.420
5     105       Bezirk Hinwil  0.593
6     106       Bezirk Horgen  0.475
7     107       Bezirk Meilen  0.615
8     108    Bezirk Pfäffikon  0.397
9     109        Bezirk Uster  0.533
10    110   Bezirk Winterthur  0.490
> #Plotting sample data:
> dis.plot(dt$bfs_nr, dt$values, 2016)
```

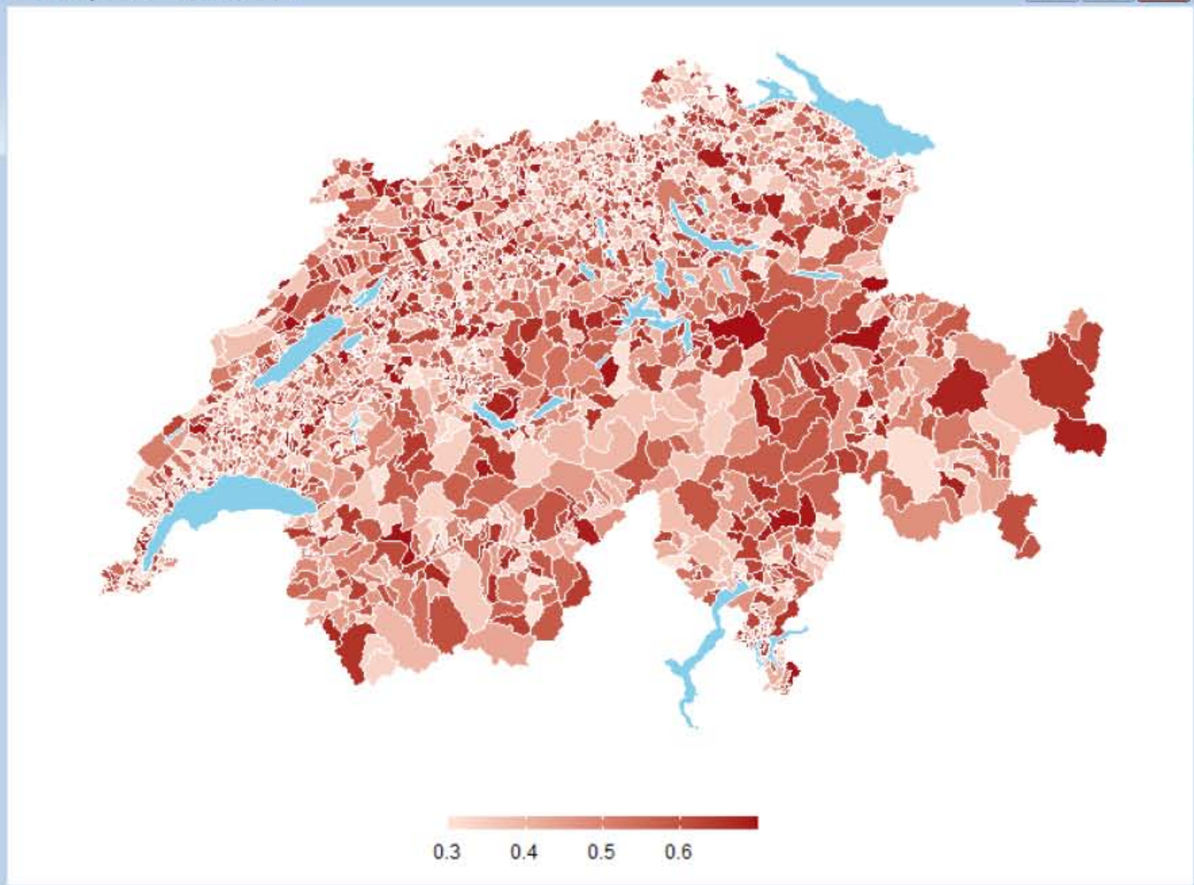


Ou encore au niveau des communes avec la fonction **mun.plot( )**:

R Console

```
> library("RSwissMaps")
> # Generating sample data:
> dt <- mun.template(2016)
> for(i in 1:nrow(dt)){dt$values[i] <- sample(c(300:700), 1)/1000}
> head(dt,10)
   bfs_nr                name values
1       1     Aeugst am Albis  0.474
2       2 Affoltern am Albis  0.576
3       3          Bonstetten  0.309
4       4     Hausen am Albis  0.424
5       5           Hedingen  0.467
6       6     Kappel am Albis  0.329
7       7              Knonau  0.358
8       8         Maschwanden  0.405
9       9     Mettmenstetten  0.699
10     10           Obfelden  0.409
> # Plotting sample data:
> mun.plot(dt$bfs_nr, dt$values, 2016)
Warning message:
Use of `data_combined$data` is discouraged. Use `data` instead.
>
```

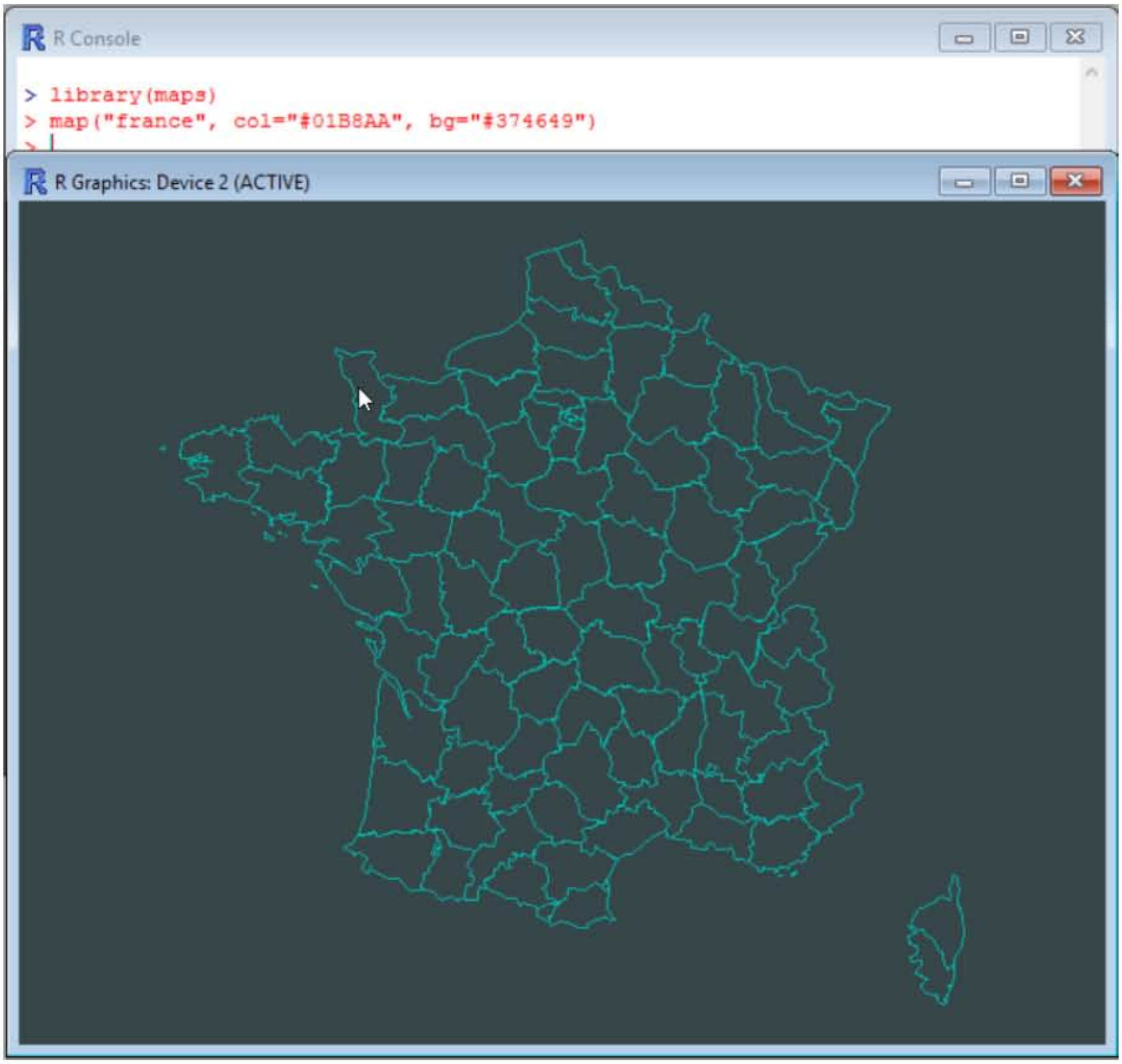
R Graphics: Device 2 (ACTIVE)



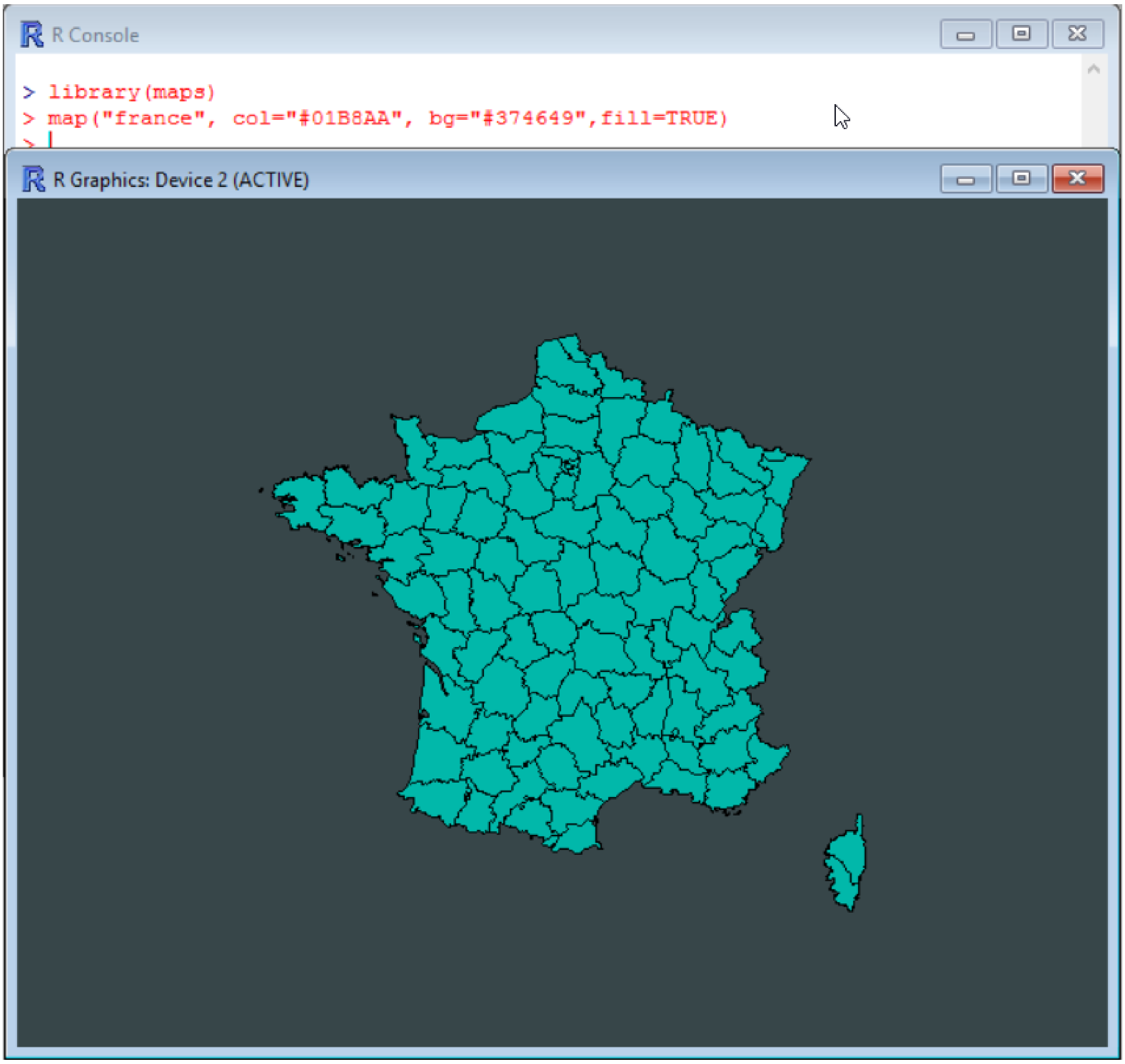
Géostatistiques planes ponctuelles (package maps)

Le but ici va être de découvrir le package **maps** que nous retrouverons juste plus loin et qui permet d'avoir des cartes avec frontières des régions/départements ou cantons dont nous pouvons représenter des points à l'aide des latitudes et longitudes.

Commençons d'abord par rapport à des notions esthétique de base:



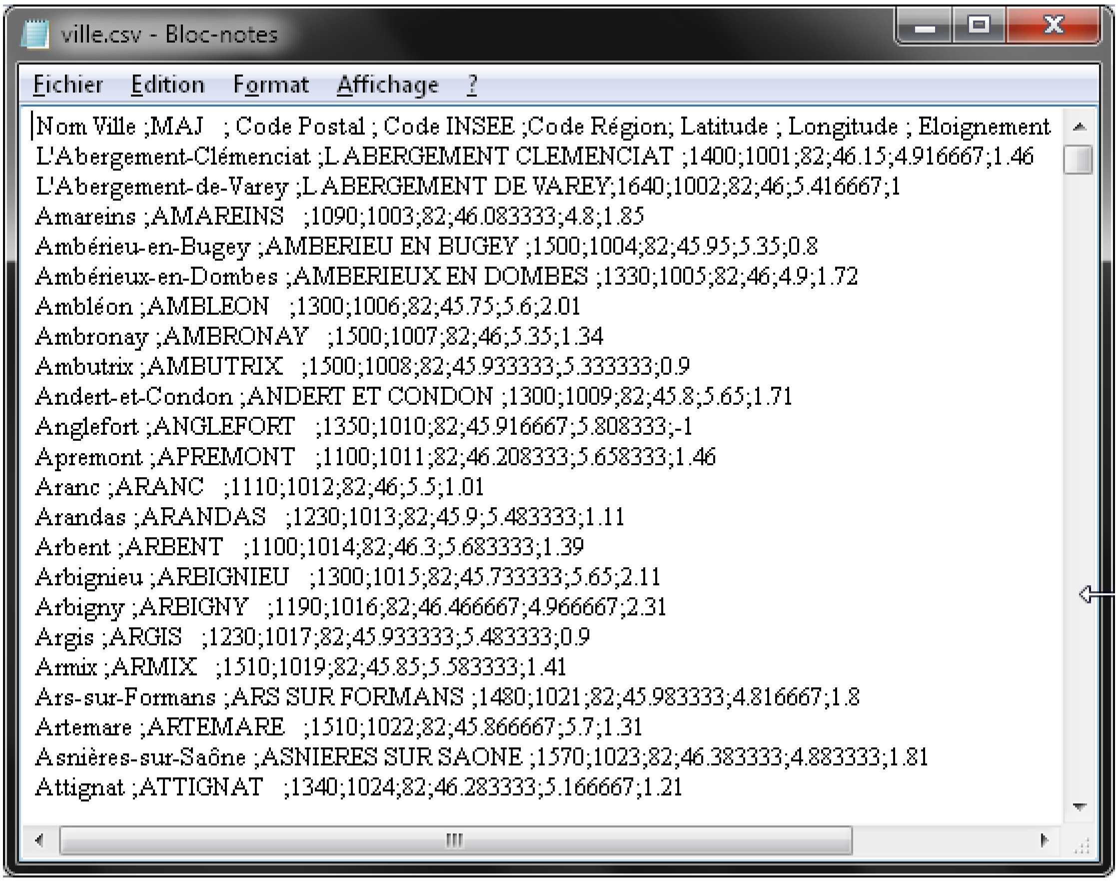
ou encore:



L'exemple ci-dessous est repris du post de l'excellentissime blog de Arthur CHARPENTIER de Novembre 2010 avec son autorisation:

http://freakonometrics.hypotheses.org/2160

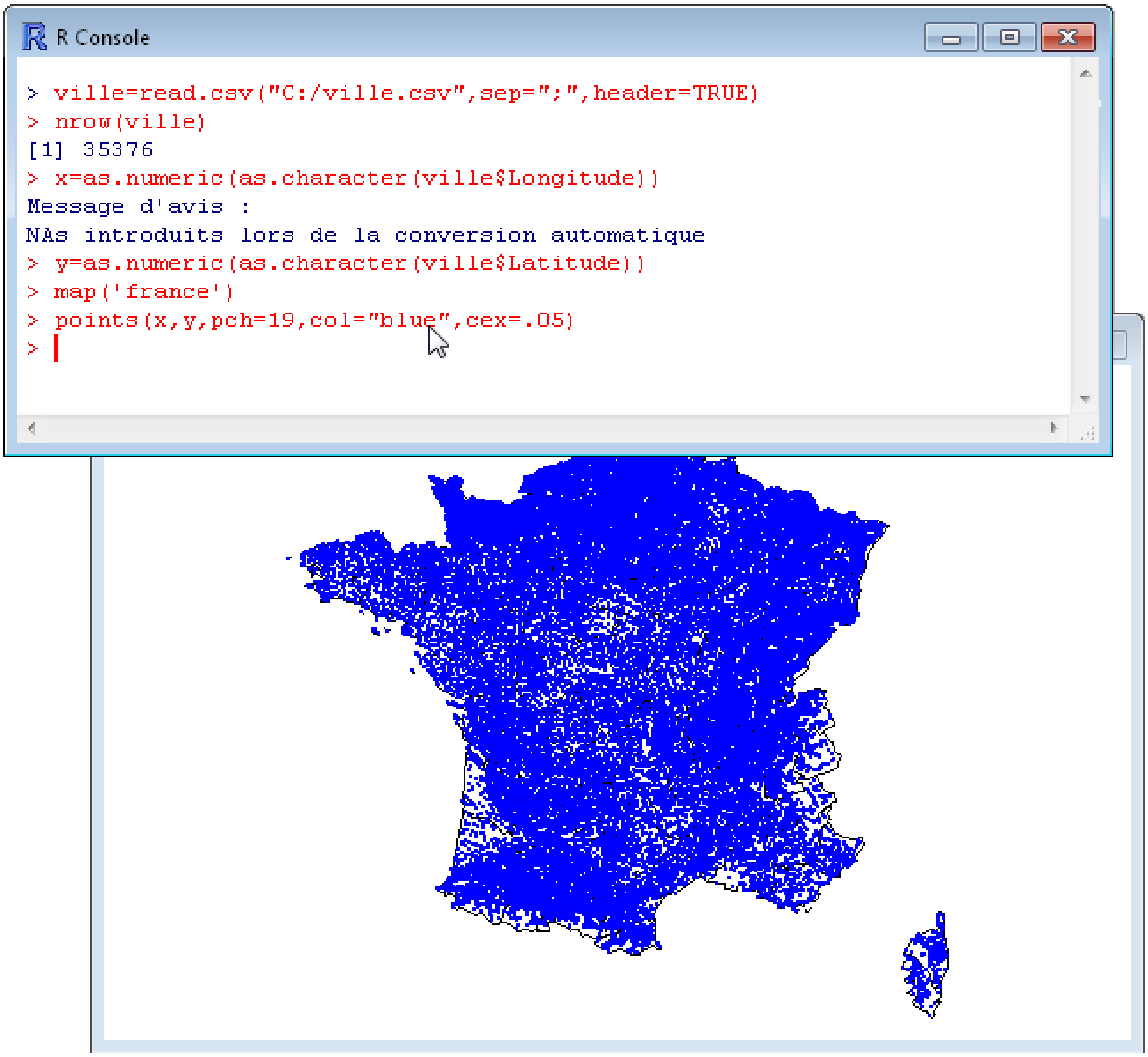
Nous utiliserons le fichier suivant qui contient la liste de 32'250 communes français avec leur nom et latitude/longitude (le package ne permet pas à ce jour de faire la Suisse):



D'abord nous chargeons le package **maps**:



Si nous voulons représenter toutes les communes, nous aurons alors:



Et maintenant en prenant que les villes qui comprennent l'expression "-sur-Mer" (comme quoi les techniques de filtrage que nous avons apprises ne sont pas inutiles!):

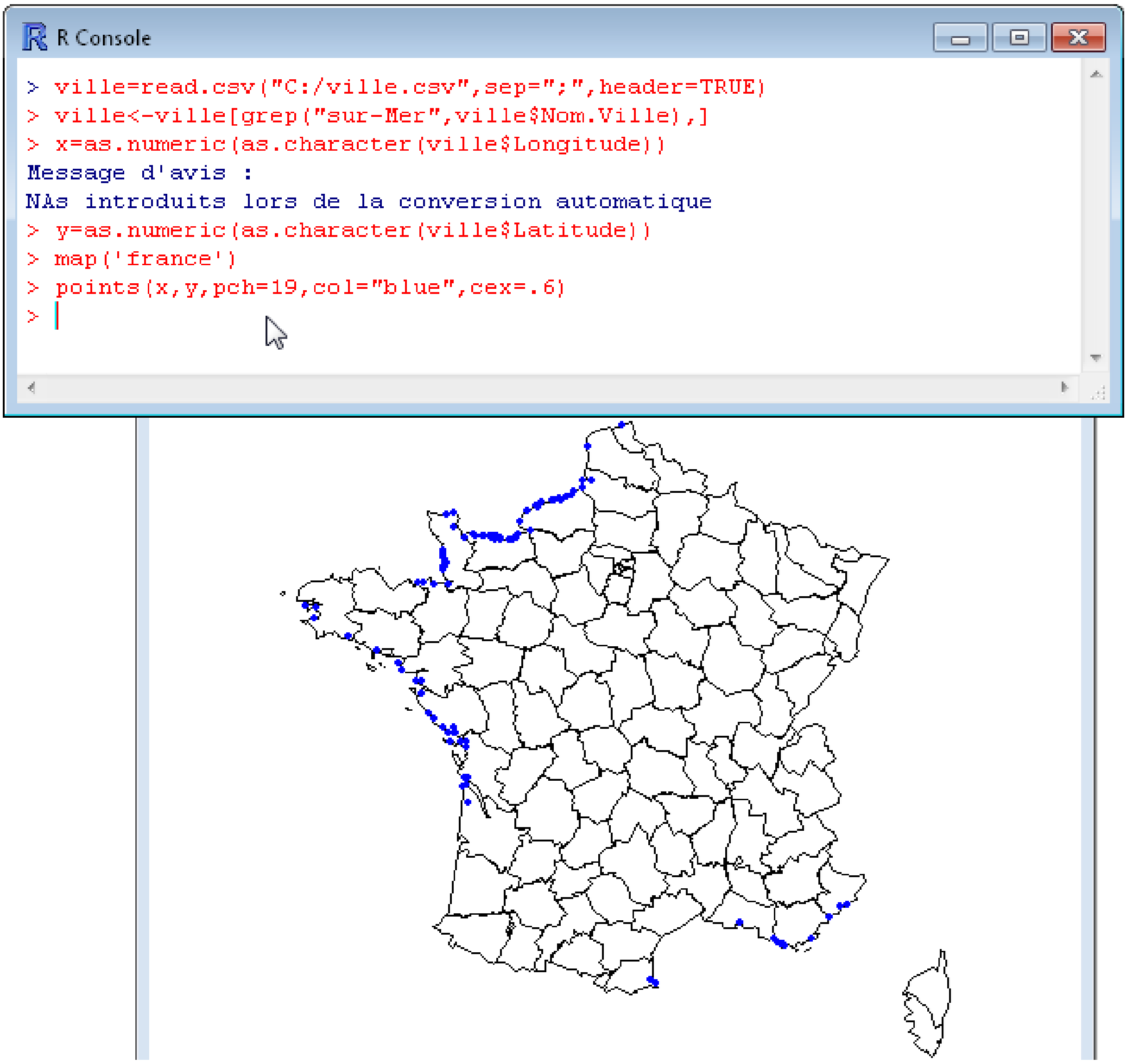
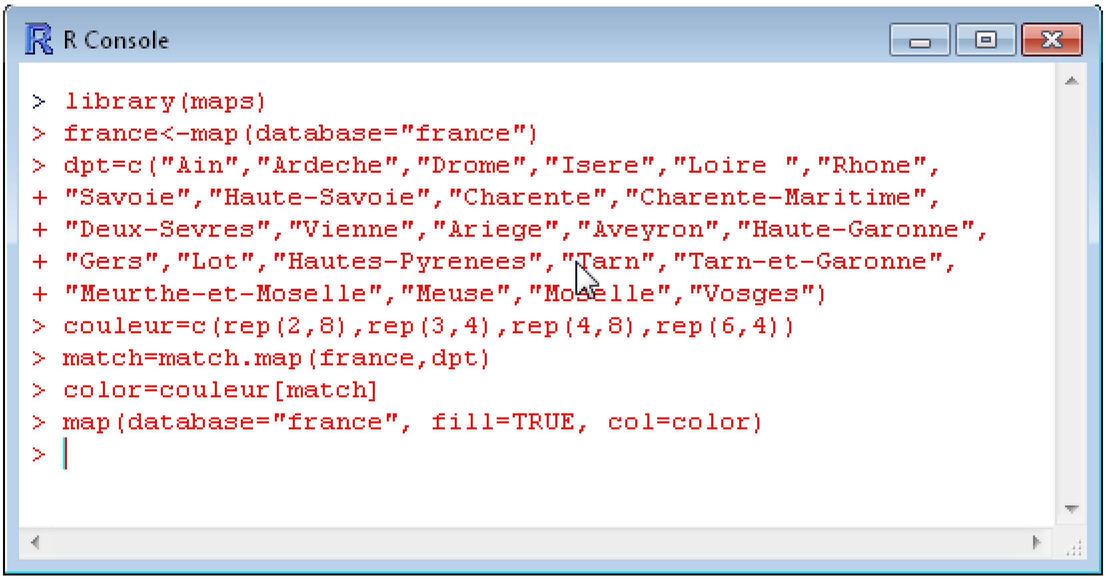


Figure 188 Graphe carte pays avec points

Et aussi d'Arthur CHARPENTIER et du même excellentissime blog mais de Novembre 2011:

http://freakonometrics.hypotheses.org/2308

nous pouvons contrôler les couleurs des régions (ce qui n'est pas faisable à ce jour avec la Suisse):



```
> library(maps)
> france<-map(database="france")
> dpt=c("Ain","Ardeche","Drome","Isere","Loire ","Rhone",
+ "Savoie","Haute-Savoie","Charente","Charente-Maritime",
+ "Deux-Sevres","Vienne","Ariege","Aveyron","Haute-Garonne",
+ "Gers","Lot","Hautes-Pyrenees","Tarn","Tarn-et-Garonne",
+ "Meurthe-et-Moselle","Meuse","Moselle","Vosges")
> couleur=c(rep(2,8),rep(3,4),rep(4,8),rep(6,4))
> match=match.map(france,dpt)
> color=couleur[match]
> map(database="france", fill=TRUE, col=color)
>
```

Ce qui donne:

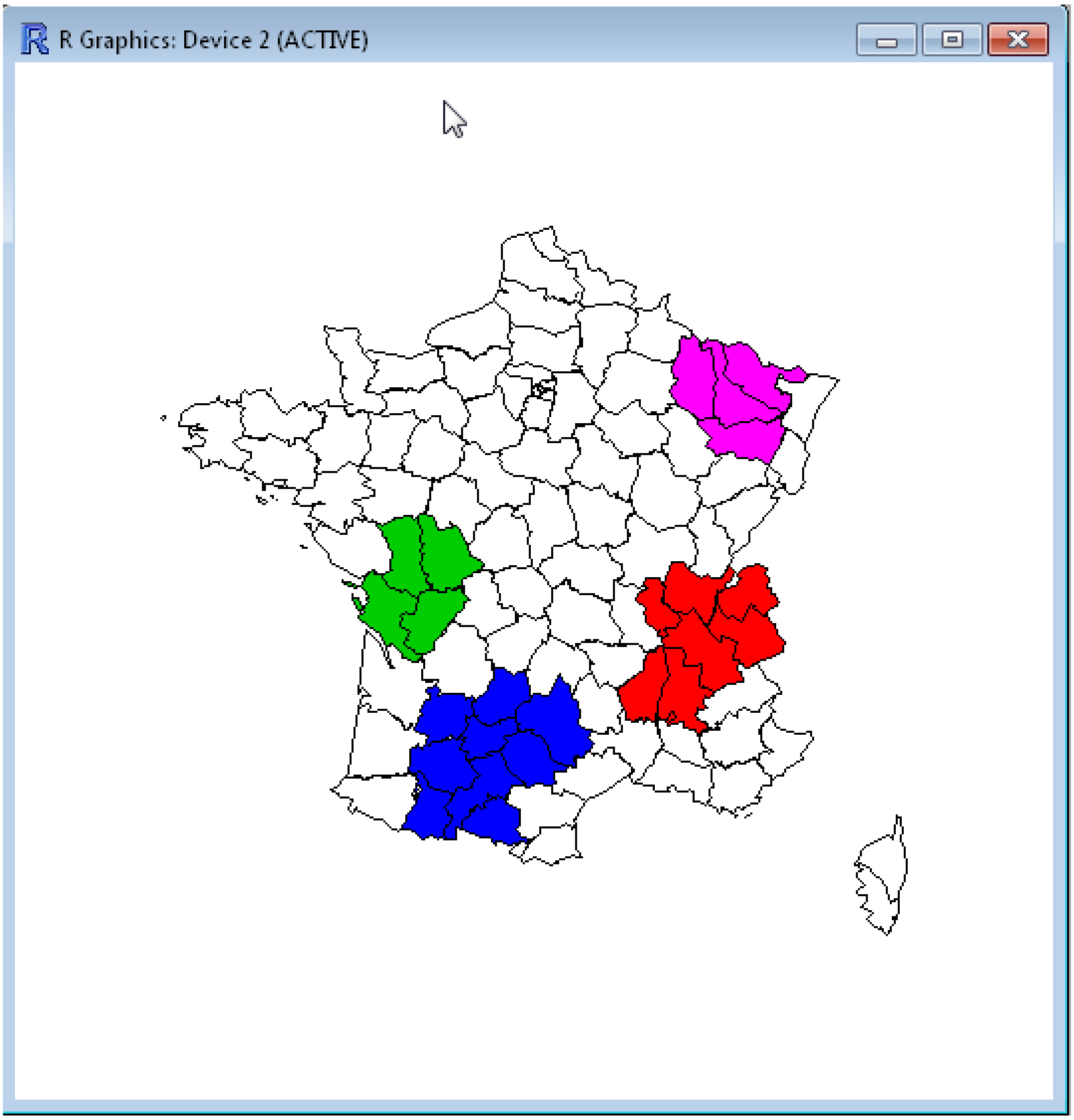
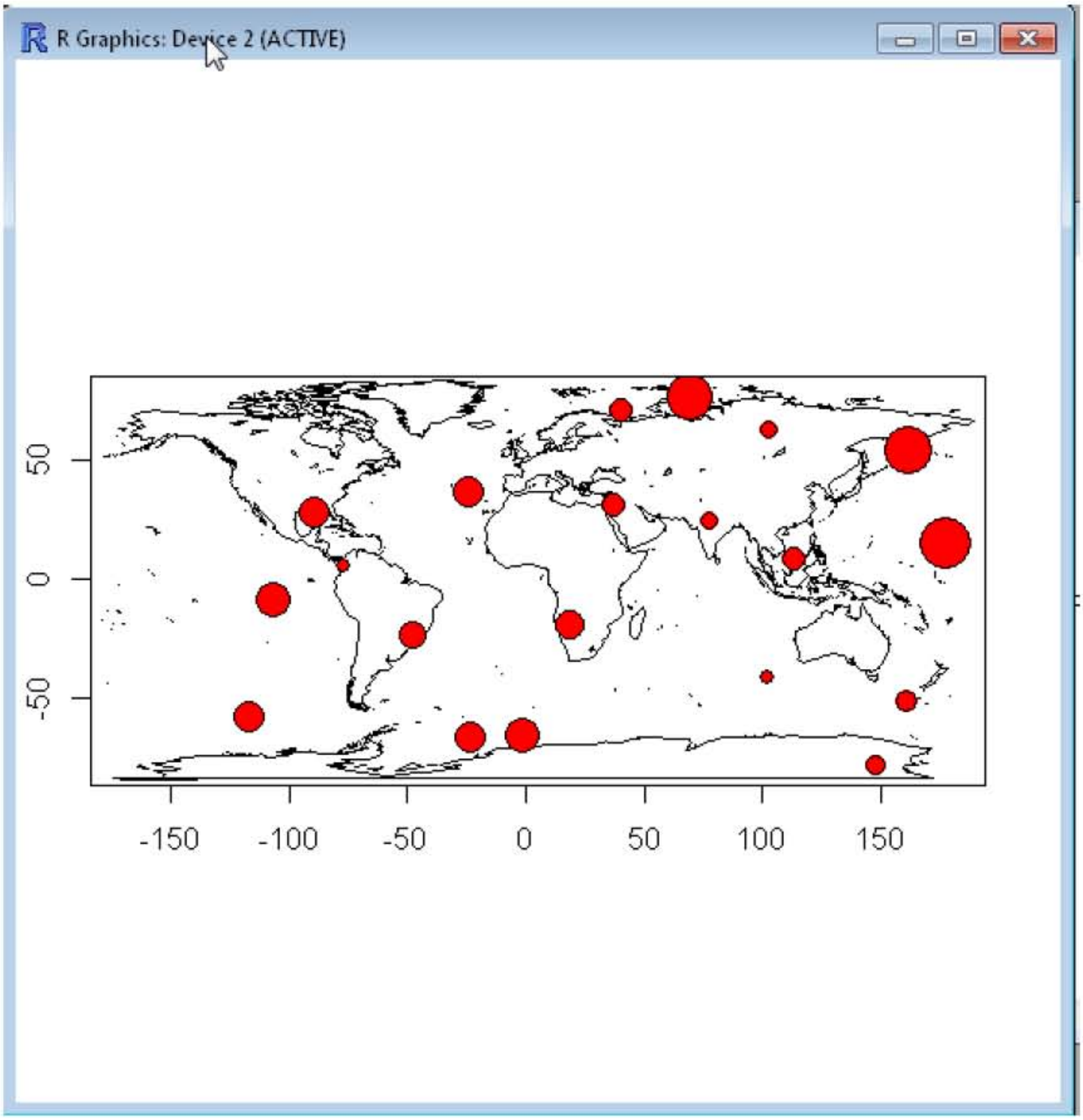


Figure 189 Graphe carte pays avec zones géographiques

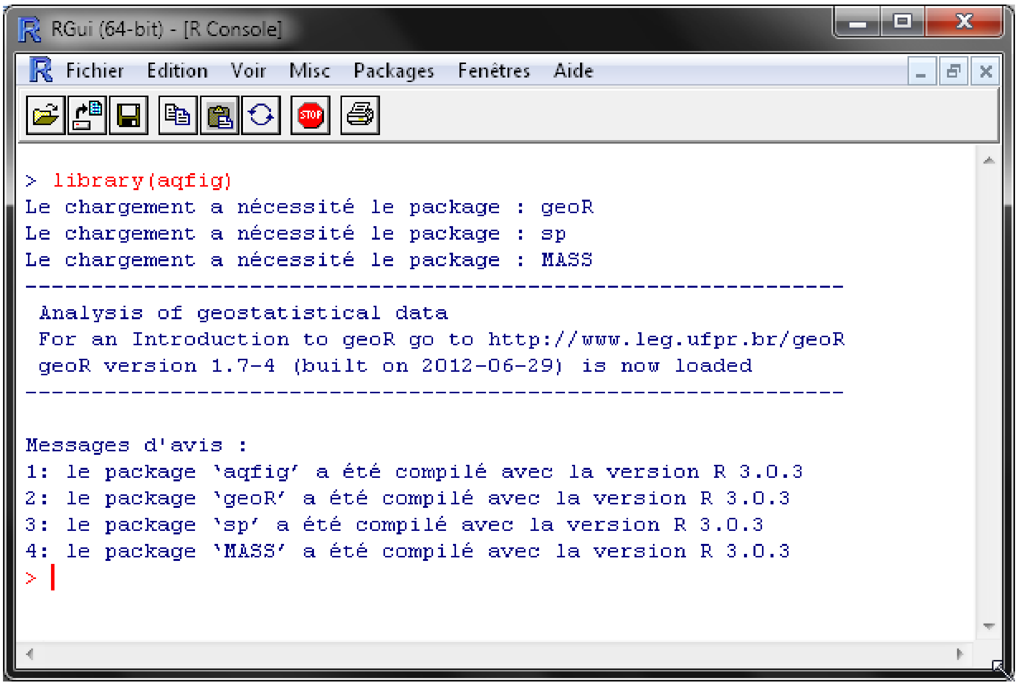
Sinon toujours avec le même package des points sur une carte dont le diamètre dépend de l'amplitude des données:

```
R Console
> library(maps)
> map("world", interior = FALSE)
> #on le fait avec des variables aléatoires car on
> #a compris le principe de l'import *.csv ou autre
> lon<-runif(20,-180,180)
> lat<-runif(20,-90,90)
> radius=runif(40,1,4)
> #avec les points pch=21 on peut contrôler la couleur de
> #bordure et de fond
> points(lon,lat,pch=21,cex=radius,col="black",bg="red")
> map.axes()
>
```

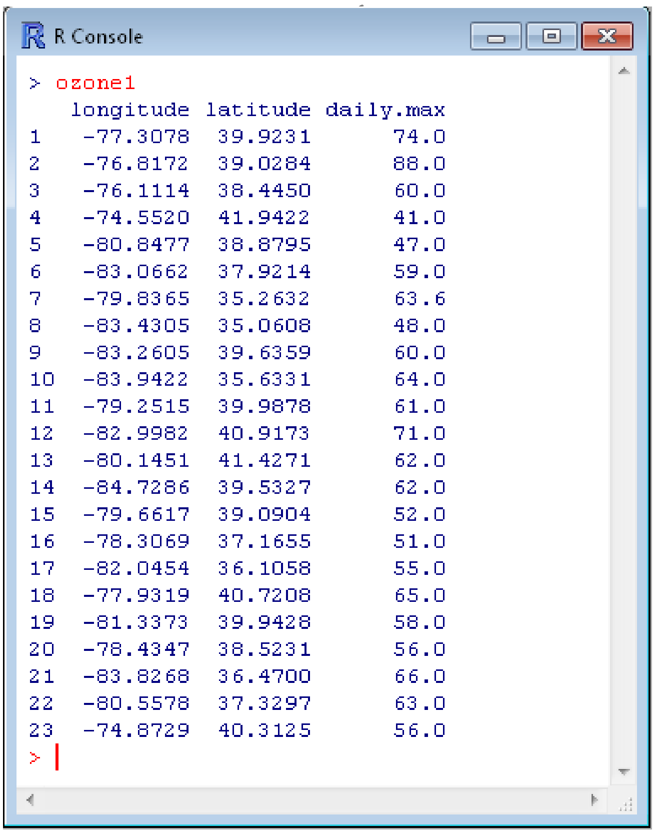

R Graphics: Device 2 (ACTIVE)
50
0
-50
-150
-100
-50
0
50
100
150

Géostatistiques planes avec gradients (packages aqfig et maps)

Nous chargeons le package **aqfig**:



Considérons les données suivantes du package **aqfig**:



En utilisant les commandes **plot3d.points( )** et **vertical.image.legend( )** du package **aqfig** et la commande **map( )** du package **maps**:

```
> data(ozone1)
> col.rng <- rev(rainbow(n=10, start=0, end=4/6))
> z.rng <- c(40, 90)
> plot3d.points(x=ozone1$longitude, y=ozone1$latitude, z=ozone1$daily.max,
+               xlab="longitude", ylab="latitude", zlim=z.rng, col=col.rng,
+               cex.min=0.5, cex.max=1.5)
> ## To verify, label the points with their concentrations.
> text(ozone1$longitude, ozone1$latitude+0.15, ozone1$daily.max, cex=0.7)
> ## If maps package is available, put on state lines.
> if (require("maps"))  map("state", add=TRUE, col="lightgray")
> ## Put on legend.
> vertical.image.legend(col=col.rng, zlim=z.rng)
> |
```

Nous obtenons alors un grand classique:

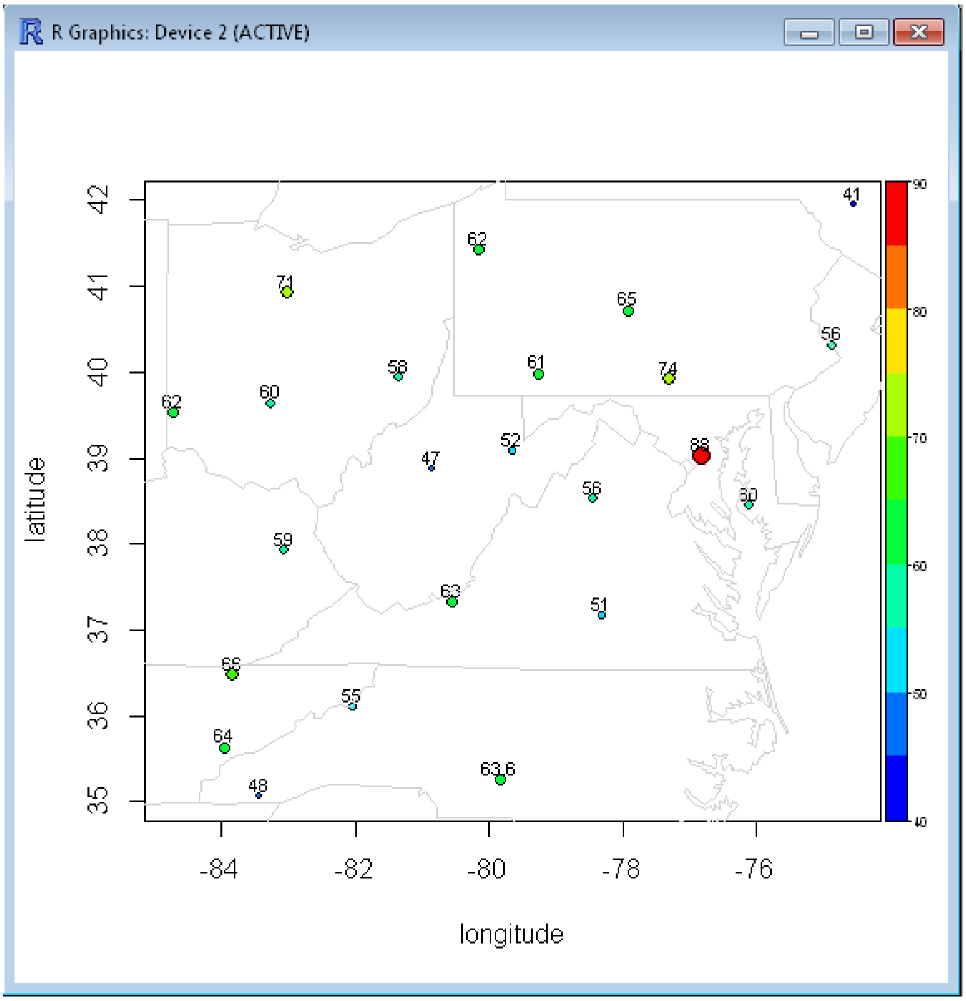


Figure 190 Graphe carte avec zone latitude-longitude définies avec points et amplitudes

Utilisation du format NetCDF avec projection cylindrique localisée

NetCDF (Network Common Data Form) est constitué, d'une part, d'un ensemble de bibliothèques logicielles et d'autre part, d'un format de données « auto-documenté », indépendant de l'architecture matérielle qui permet la création, l'accès et le partage de données scientifiques stockées sous la forme de tableaux. Le site internet du projet est hébergé par le Unidata program à l'University Corporation for Atmospheric Research (UCAR). Ils sont par ailleurs les mainteneurs principaux des programmes, des spécifications, etc. Il s'agit d'un format ouvert.

Le format, à l'origine, avait pour modèle conceptuel le format CDF issu de la NASA mais il a depuis divergé de façon telle qu'aujourd'hui ces deux formats ne sont plus compatibles.

Ce format est couramment utilisé dans des applications de climatologie, de météorologie et d'océanographie (ex., prévision météorologique, changement climatique et applications des S.I.G.).

Il s'agit d'un format de choix pour les entrées/sorties de nombreux S.I.G. et pour l'échange de données scientifiques. Le site définit le format NetCDF comme "une interface pour un accès aux données orienté tableaux et une bibliothèque qui fournit une implémentation de cette interface. La bibliothèque netCDF définit en sus un format de représentation des données scientifiques indépendant de l'architecture machine".

Nous allons utiliser pour cet exemple le package **ncdf4** et charger un jeu de données local avec la fonction **nc_open( )** :

```
> library(ncdf4)
> bom <- nc_open("c:/tmp/sd_pct_Actual_month.nc")
> print(bom) # Inspect the data
File c:/tmp/sd_pct_Actual_month.nc (NC_FORMAT_NETCDF4):

     2 variables (excluding dimension variables):
        int time_bounds[nv,time]   (Chunking: [2,1])
        float sd_pct[longitude,latitude,time]   (Chunking: [32,32,16])  (Compression: shuffle,level 1)
[1] ">>>> WARNING <<<  attribute least_significant_digit is an 8-byte value, but R"
[1] "does not support this data type. I am returning a double precision"
[1] "floating point, but you must be aware that this could lose precision!"
            least_significant_digit: 3
            units: fullness
            name: sd_pct
            long_name: Total moisture in deep soil layer (100-500 cm): averaged across both HRUs (mm)
            _FillValue: -999
            standard_name: sd_pct

     4 dimensions:
        time  Size:213   *** is unlimited ***
            bounds: time_bounds
            long_name: time
            standard_name: time
            calendar: gregorian
            units: days since 1900-01-01
            name: time
        nv  Size:2
        latitude  Size:681
            units: degrees_north
            name: latitude
            long_name: latitude
            standard_name: latitude
        longitude  Size:841
            units: degrees_east
            name: longitude
            long_name: longitude
            standard_name: longitude

    1 global attributes:
        var_name: sd_pct
>
```

Ensuite nous extrayons les données qui nous intéressent:

```
> lon <- ncvar_get(bom, "longitude")
> lat <- ncvar_get(bom, "latitude")
> dates <- as.Date("1900-01-01") + ncvar_get(bom, "time")
> moisture <- ncvar_get(bom, "sd_pct")
> dimnames(moisture) <- list(lon, lat, dates)
>
```

Ensuite allons-y pour l'afficher:

```
> library(tidyverse)
> library(RColorBrewer)
> library(reshape2)
>
> d <- "2017-07-31"
> m <- moisture[, , which(dates == d)] %>%
+         melt(varnames = c("lon", "lat")) %>%
+         subset(!is.na(value))
>
> ggplot(m, aes(x = lon, y = lat, fill = value)) + borders("world") +
+     geom_tile() +
+     scale_fill_gradientn(colors = brewer.pal(9, "Blues")) +
+     labs(title = "Total moisture in deep soil layer (100-500 cm)",
+     subtitle = format(as.Date(d), "%d %B %Y")) +
+     xlim(range(lon)) + ylim(range(lat)) + coord_fixed()
>
```

Ce qui donne:

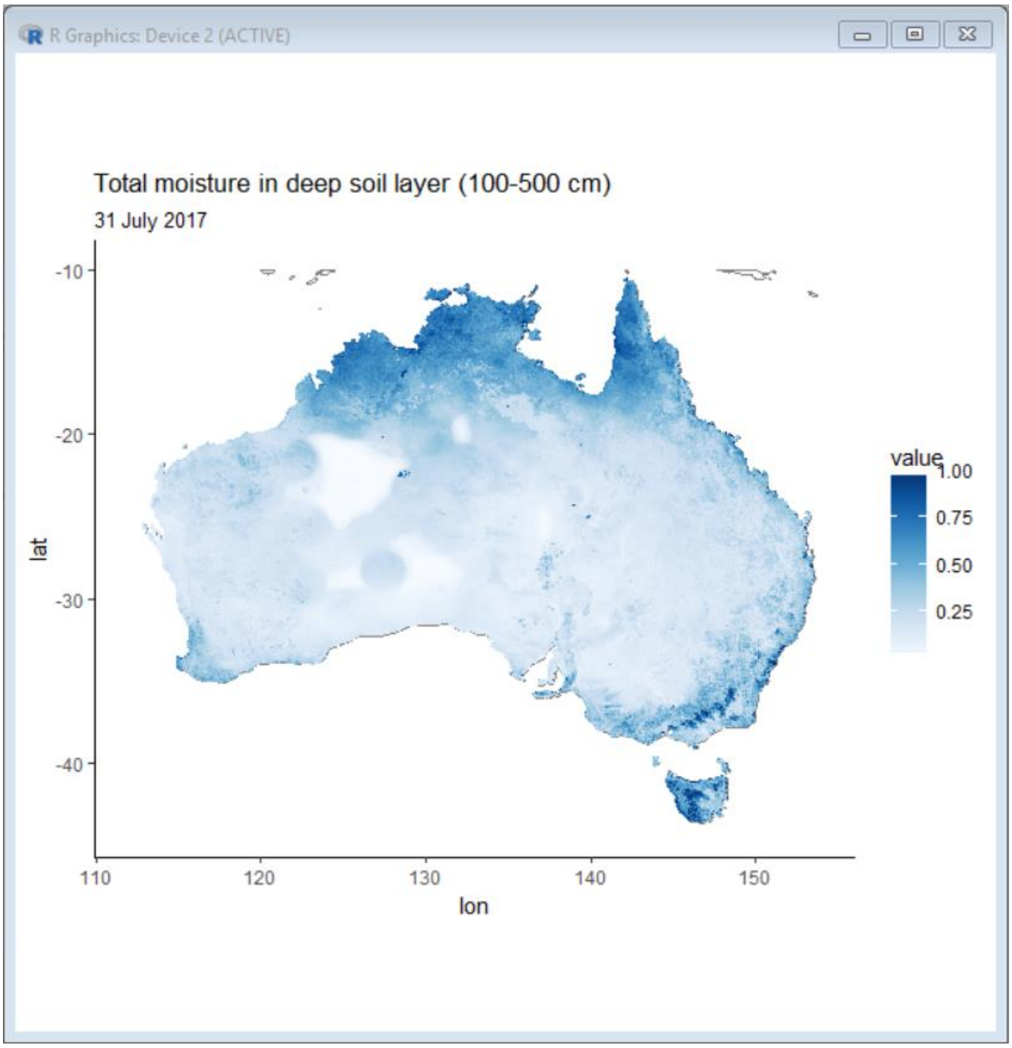
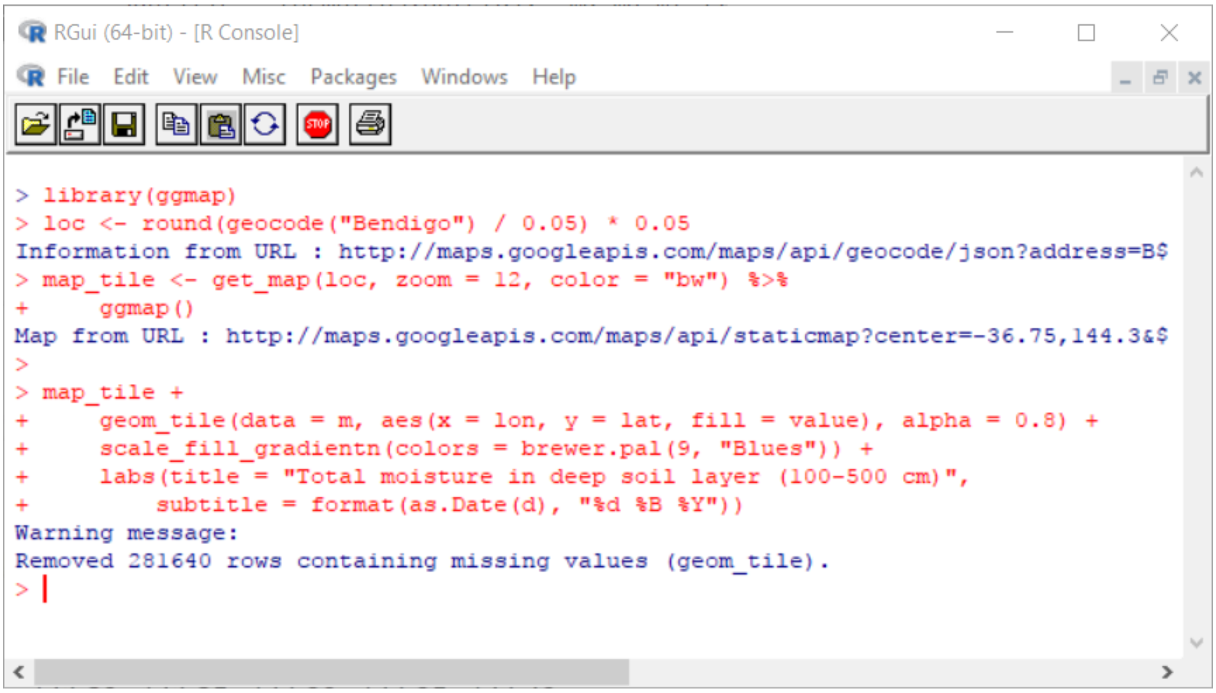
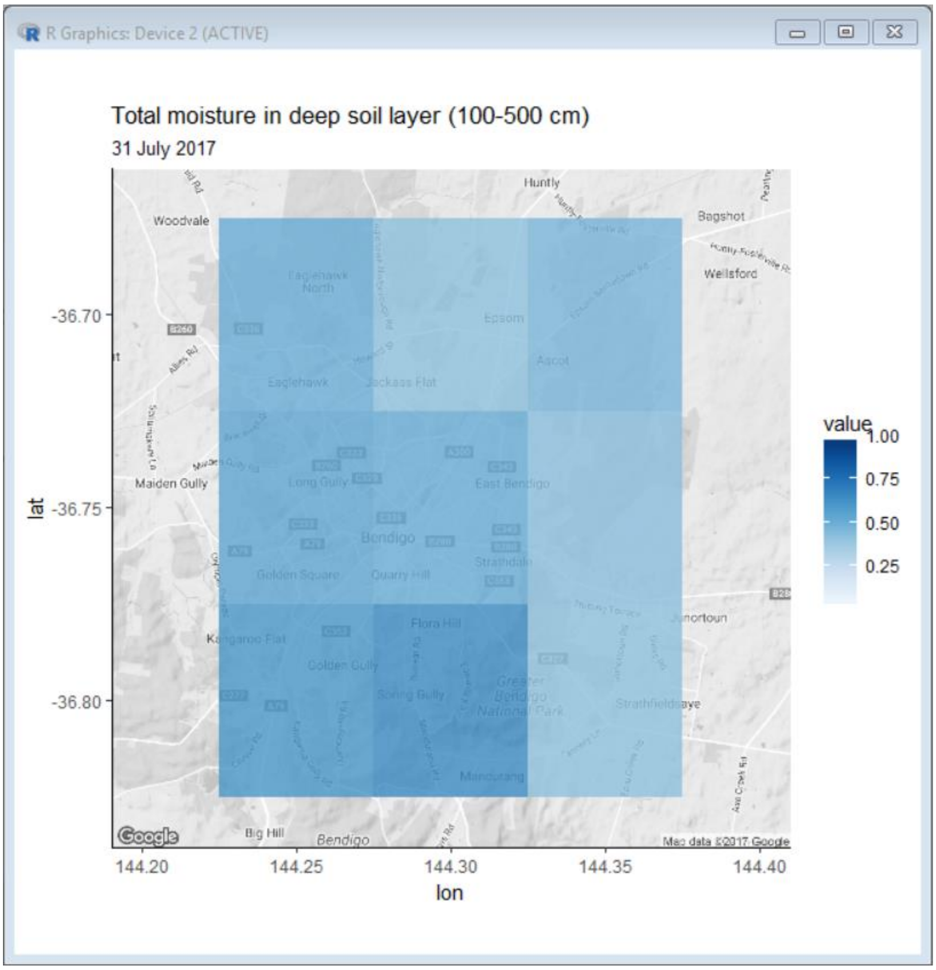


Figure 191 Graph à carte NCDF

On peut zoomer avec le package **ggmap**:



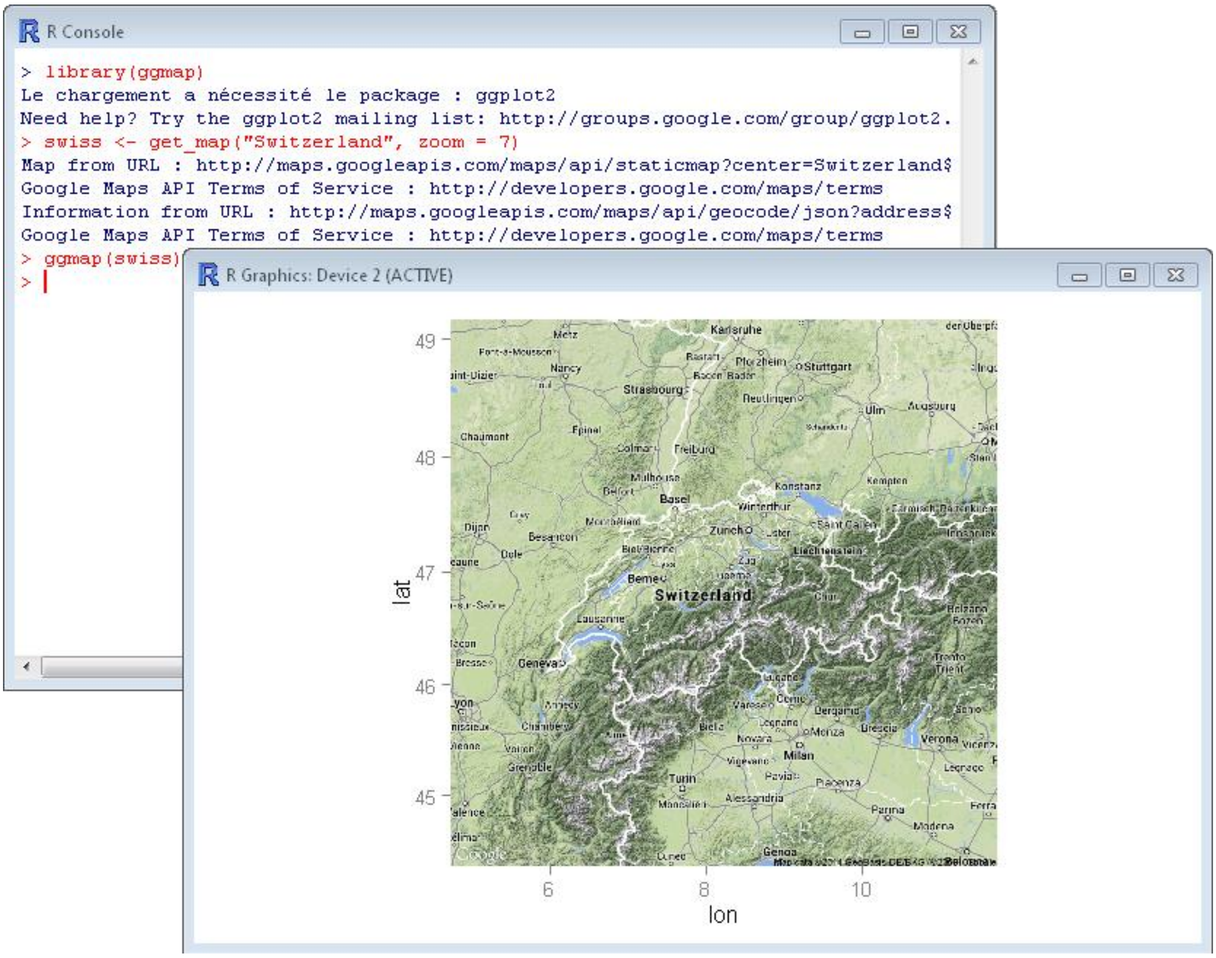
Ce qui donne:



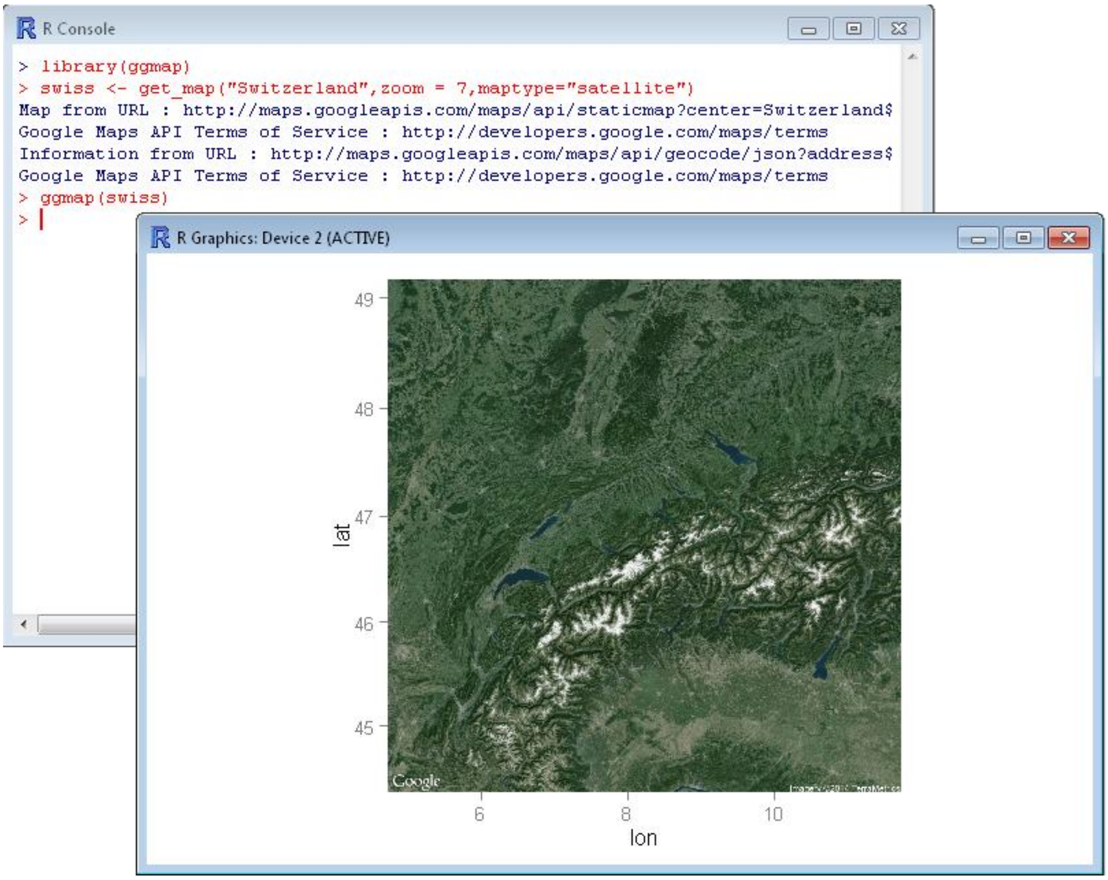
Google maps/Open maps (package ggmap)

Attention!!! Quand vous travaillez avec ce package, choisissez un répertoire de travail qui soit bien en lecture **et** écriture!

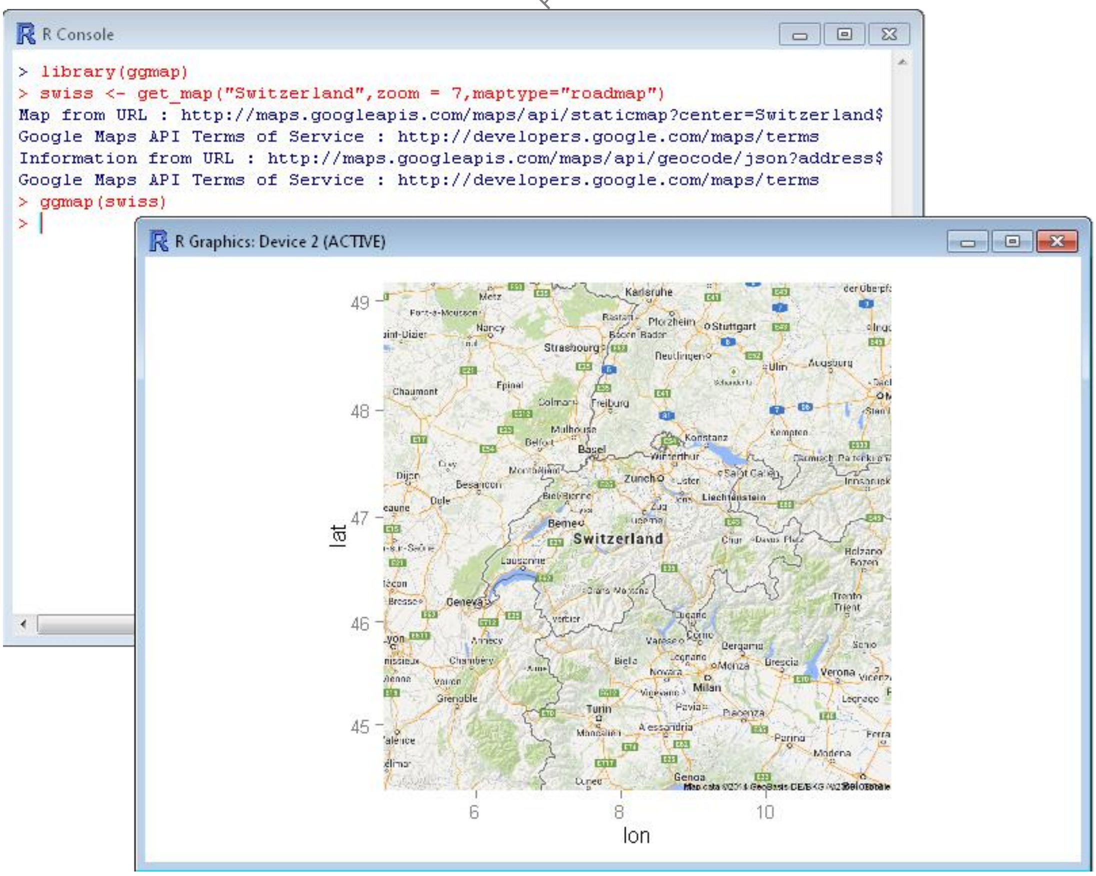
Nous allons voir ici quelques possibilités du package **ggmap**. Commençons par apprendre à afficher des cartes tout simplement avec différents styles en utilisant les commandes **get_map( )** et **ggmap( )** :



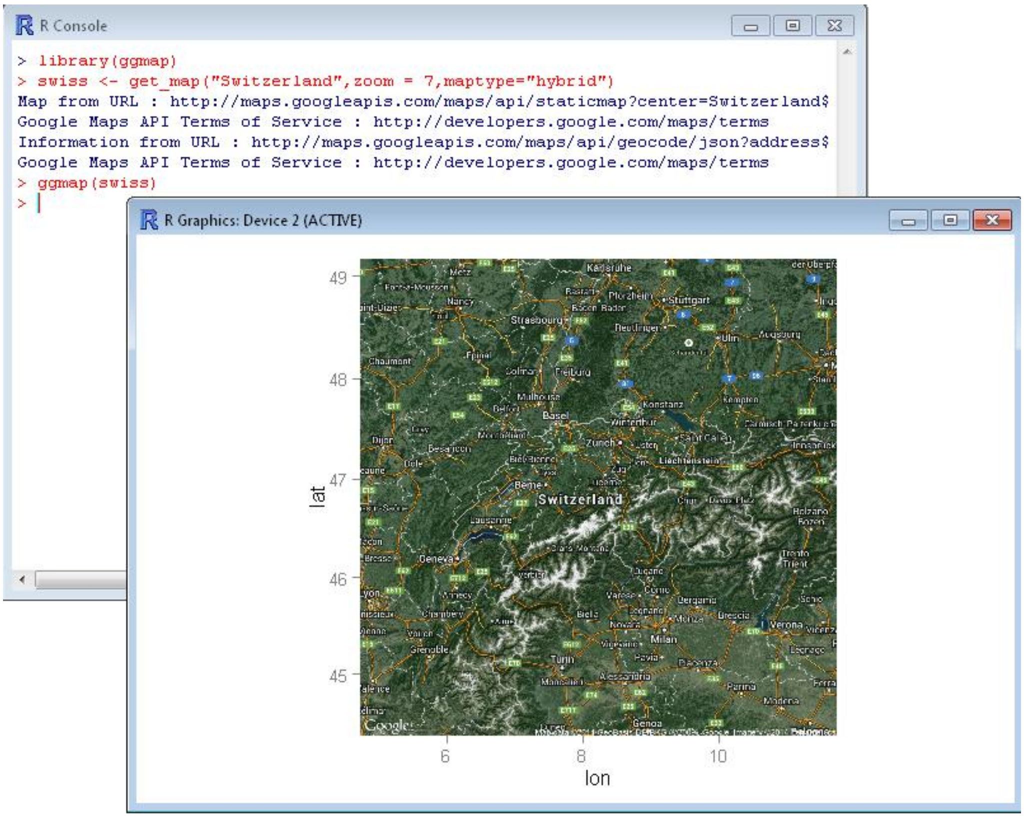
ou avec une vue satellite:



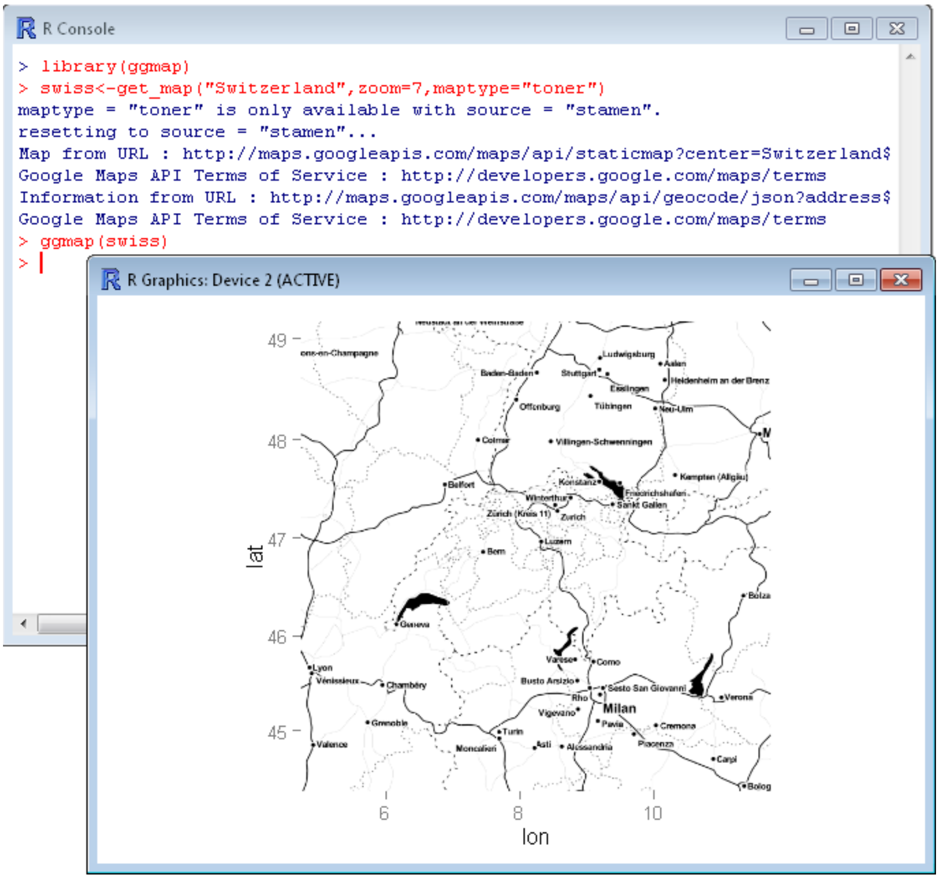
ou de type carte routière:



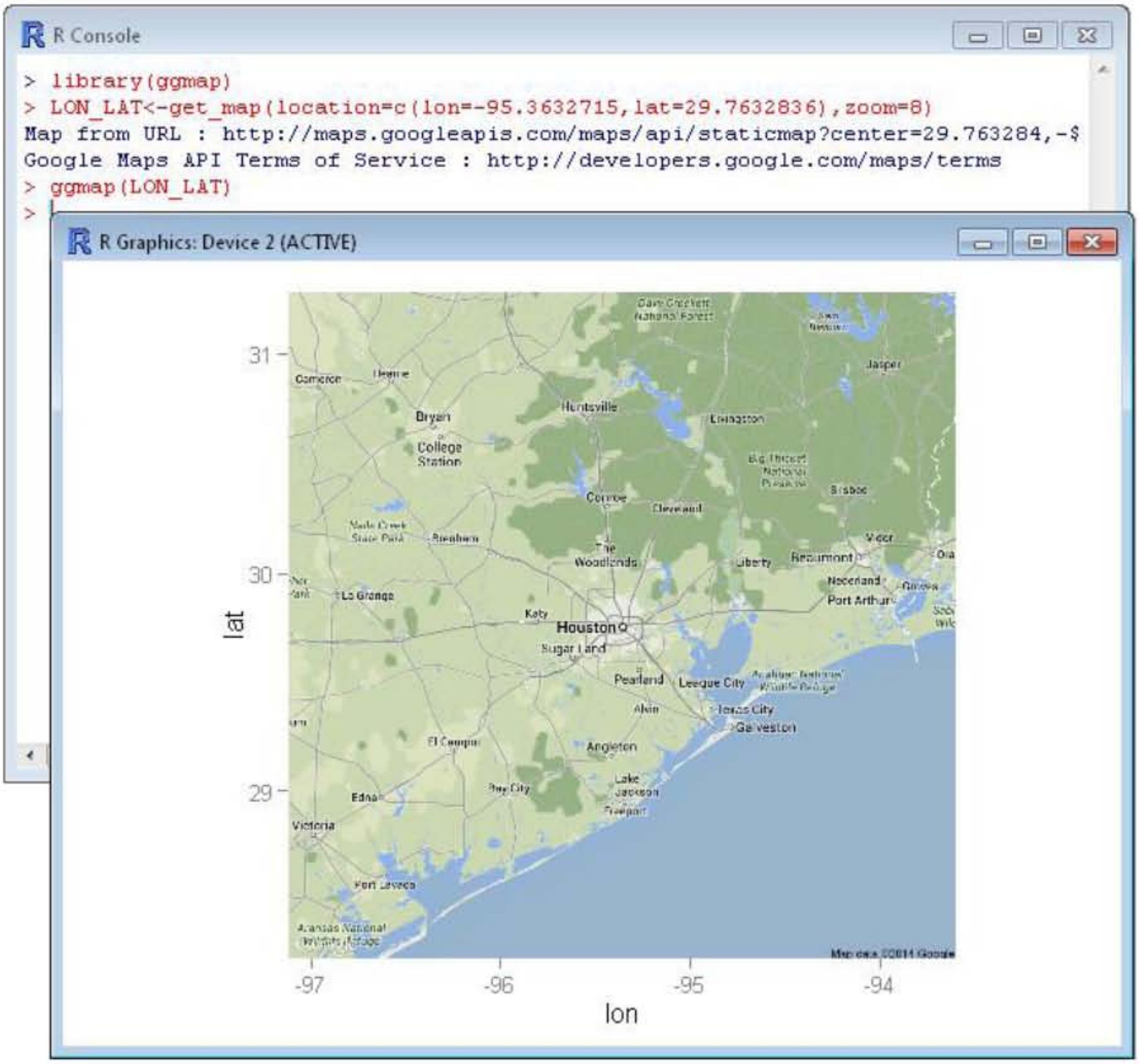
ou avec un style hybride/carte routière:



ou avec un style militaire:



Maintenant choisissons le lieu avec des coordonnées en latitude/longitude:



Apprenons à mettre un point sur un lieu de la carte avec la commande **geom_point( )** :

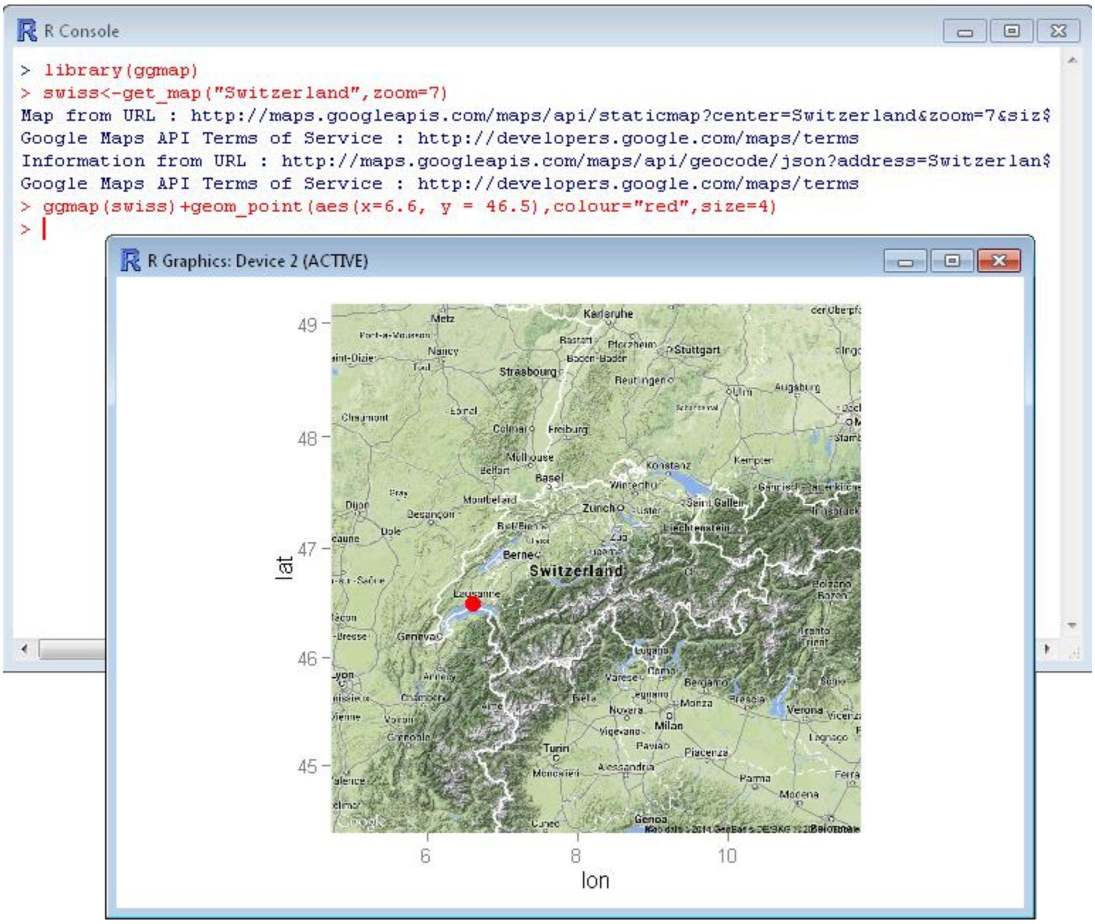
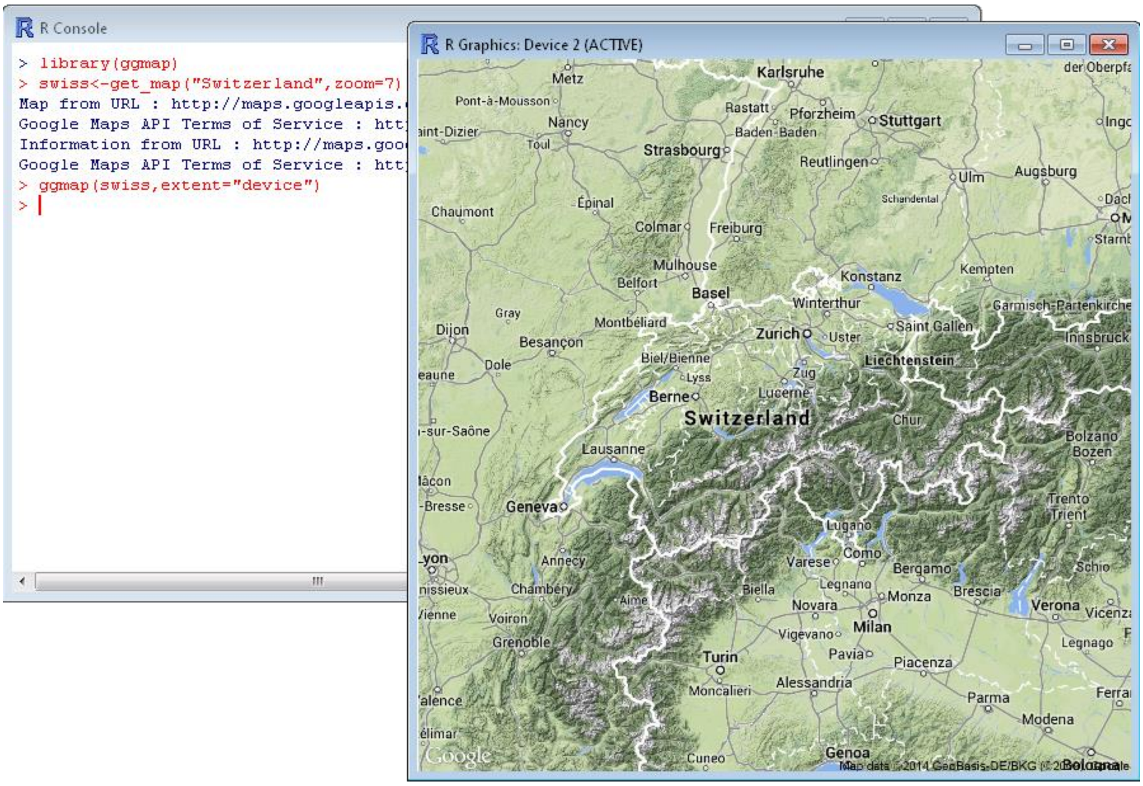
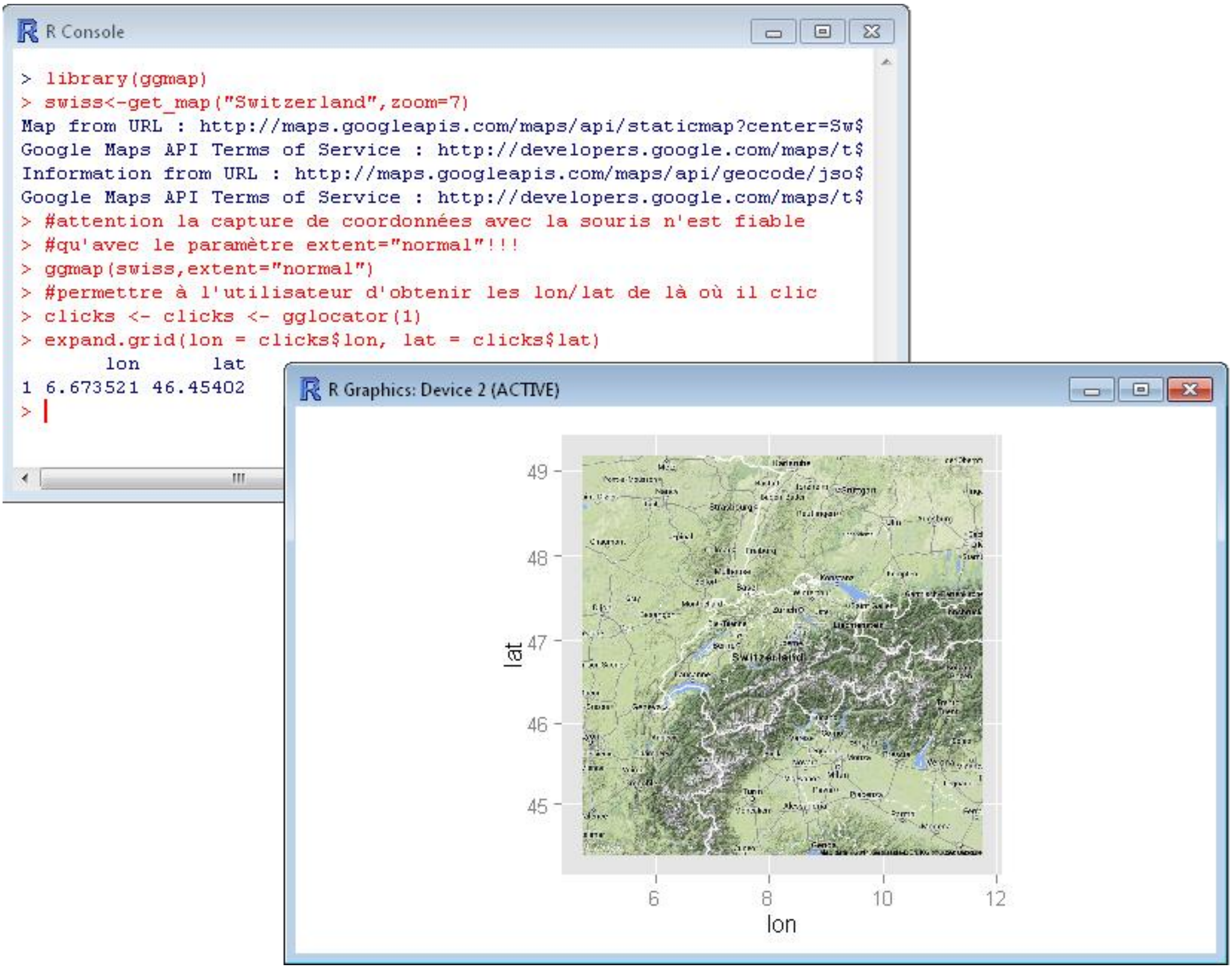


Figure 192 Graph Google Maps avec points

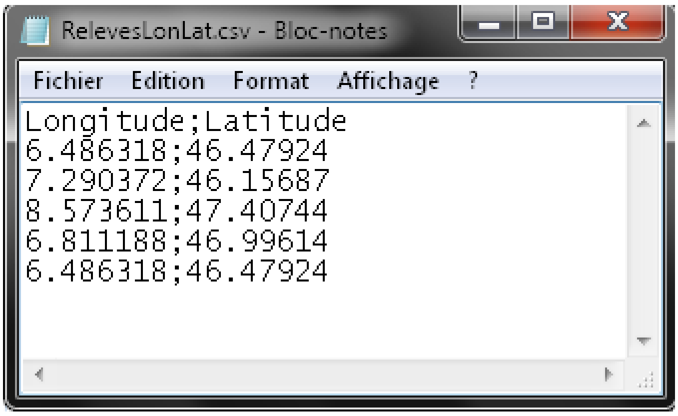
ou faire en sorte que la carte prenne tout la fenêtre:



Nous pouvons aussi "piquer" des coordonnées de carte avec la commande **gglocater( )** où entre parenthèse nous pouvons choisir le nombre de points cliquables selon l'exemple ci-dessous:



Nous pouvons également tracer des polygones avec la commande **geom_poly( )** et des tracés avec **geom_path( )** ou écrire des textes avec la commande **annotate( )** en utilisant par exemple comme source le fichier suivant:



et le script suivant:

```
R Console
> library(ggmap)
> mydata<-read.csv("C:/RelevesLonLat.csv",header=T,sep=";")
> swiss<-get_map("Switzerland",zoom=8)
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Switzerland&zoom=8&size$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Switzerland$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
> ggmap(swiss)+
+ geom_polygon(data=mydata,aes(x=Longitude,y=Latitude),alpha=0.3,colour="red",fill="red")+
+ geom_path(data=mydata,aes(x=Longitude,y=Latitude),colour="white",alpha=.7,size=3)+
+ annotate("point",x=7.257885,y=46.79049,size=7)+
+ annotate("text",x=7.257885,y=46.79049,label="Golden Swiss Area",colour="white",size=3)
> |
```

Nous obtenons:

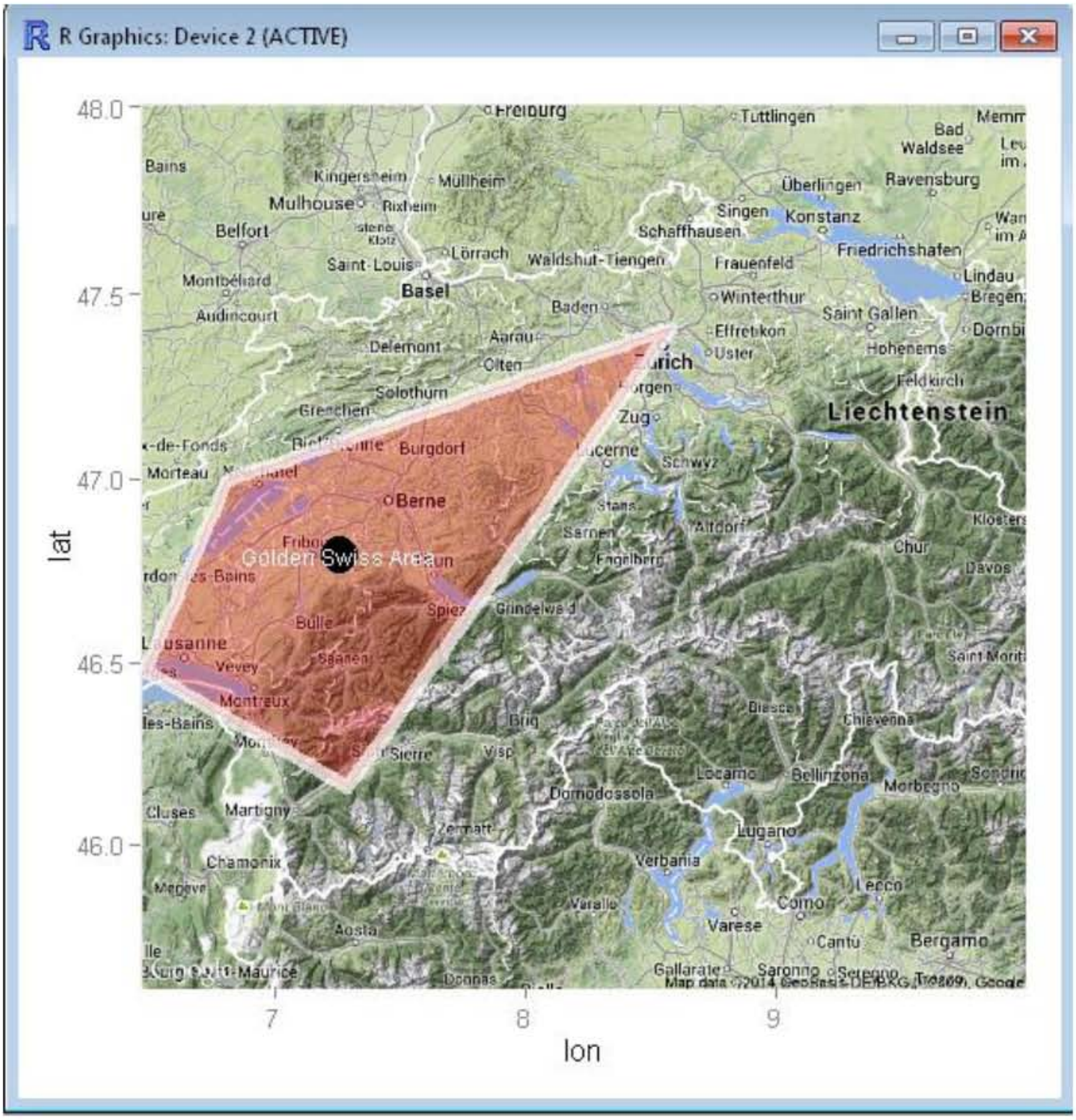
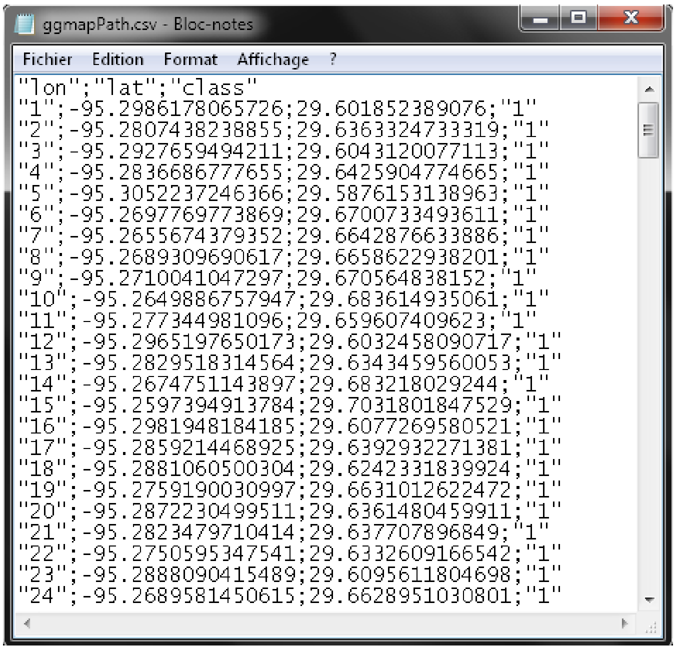


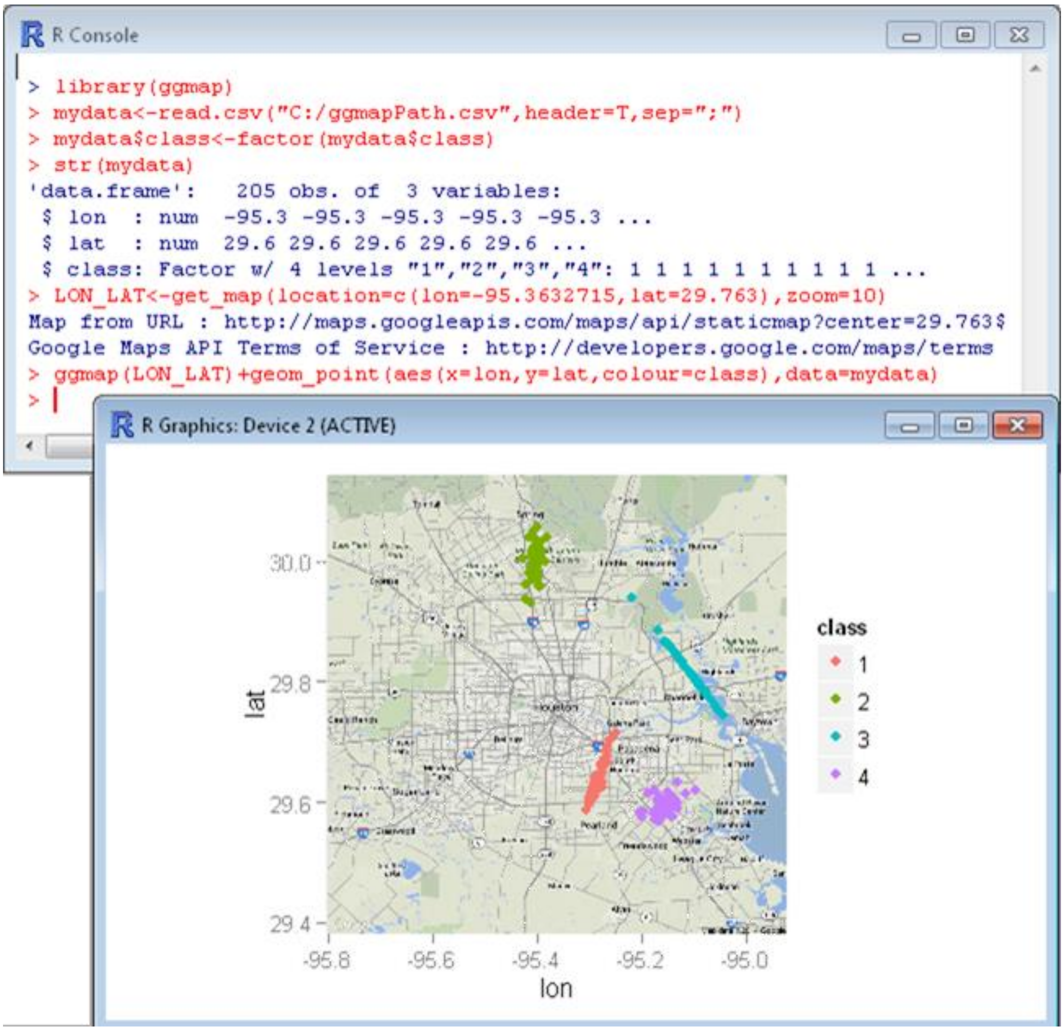
Figure 193 Graph Google Maps avec polygone

En s'inspirant de l'aide du package **ggmap** et en utilisant le fichier suivant:



```
"lon";"lat";"class"
"1";-95.2986178065726;29.601852389076;"1"
"2";-95.2807438238855;29.6363324733319;"1"
"3";-95.2927659494211;29.6043120077113;"1"
"4";-95.2836686777655;29.642590477 4665;"1"
"5";-95.3052237246366;29.5876153138963;"1"
"6";-95.2697769773869;29.6700733493611;"1"
"7";-95.2655674379352;29.6642876633886;"1"
"8";-95.2689309690617;29.6658622938201;"1"
"9";-95.2710041047297;29.670564838152;"1"
"10";-95.2649886757947;29.6836149350 61;"1"
"11";-95.277344981096;29.659607409623;"1"
"12";-95.2965197650173;29.6032458090717;"1"
"13";-95.2829518314564;29.6343459560053;"1"
"14";-95.2674751143897;29.683218029244;"1"
"15";-95.2597394913784;29.7031801847529;"1"
"16";-95.2981948184185;29.6077269580521;"1"
"17";-95.2859214468925;29.6392932271381;"1"
"18";-95.2881060500304;29.6242331839924;"1"
"19";-95.2759190030997;29.6631012622472;"1"
"20";-95.2872230499511;29.6361480459911;"1"
"21";-95.2823479710414;29.637707896849;"1"
"22";-95.2750595347541;29.6332609166542;"1"
"23";-95.2888090415489;29.6095611804698;"1"
"24";-95.2689581450615;29.6628951030801;"1"
```

Nous pouvons indiquer des relevés sur une carte en utilisant à nouveau **geom_point( )** :



Ou avec des densités de relevées grâce à des isoclines de densités locales en utilisant la commande **stat_density2d( )** :

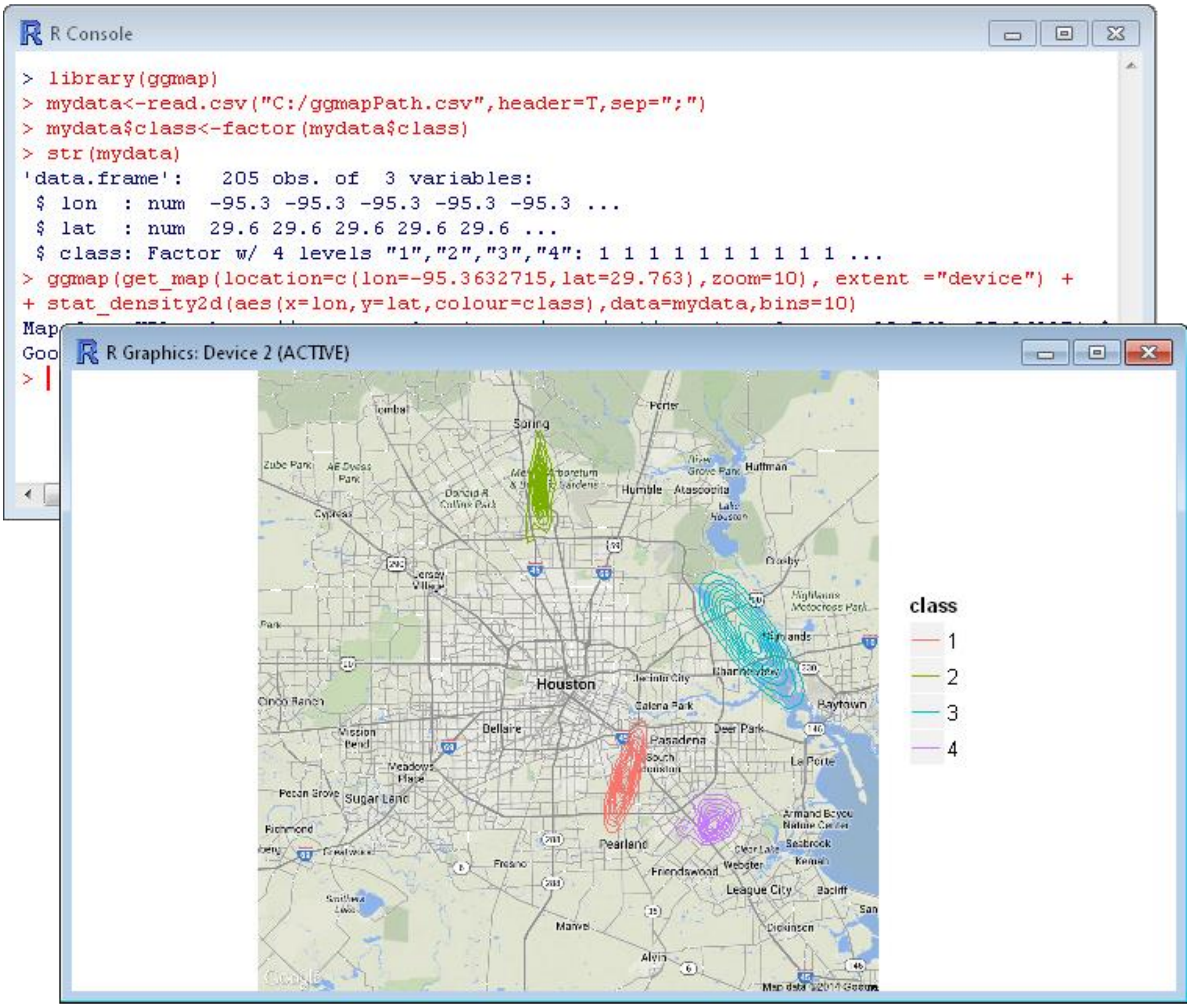


Figure 194 Graph Google Maps avec courbes de densité

ou avec un autre visuel:

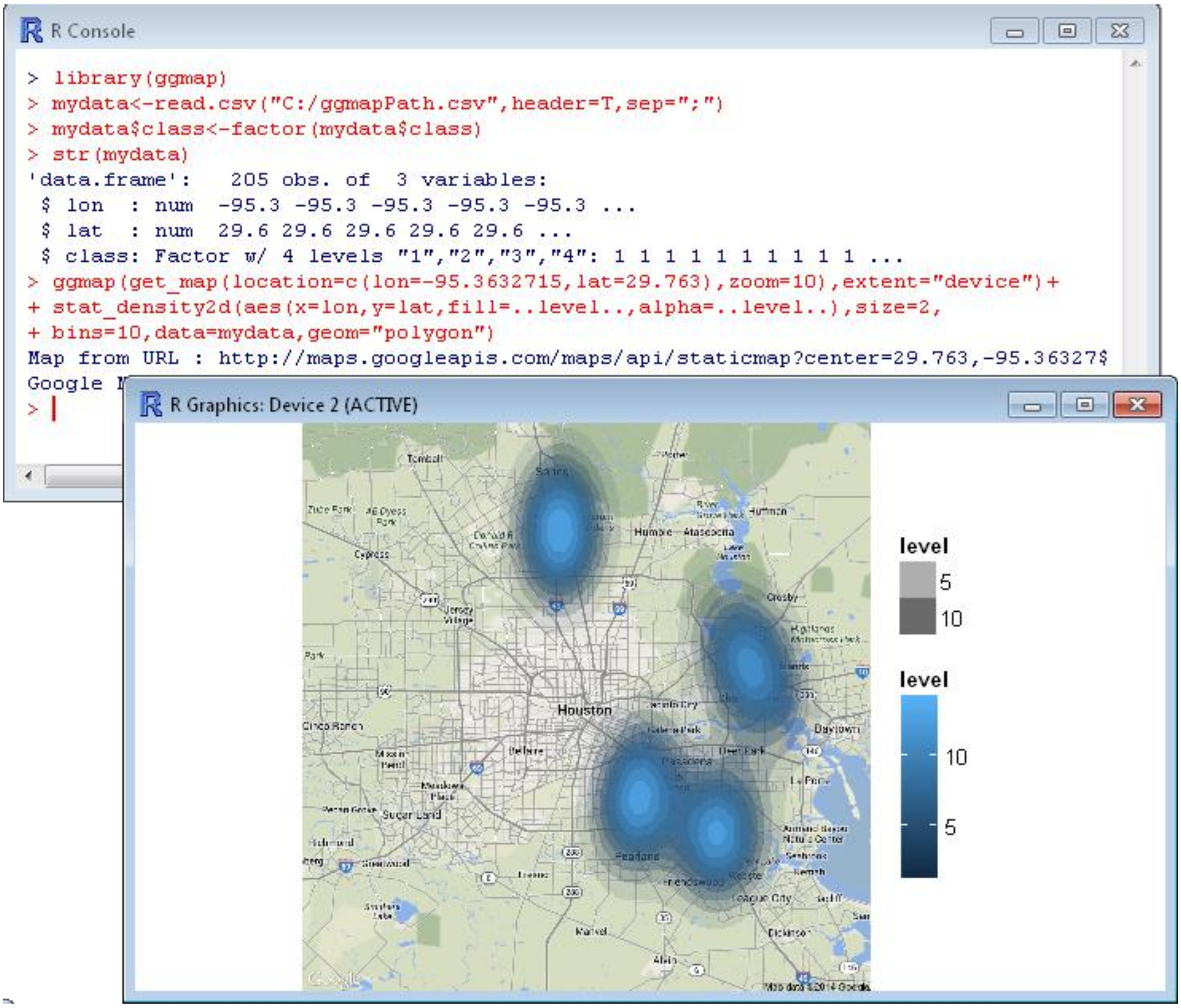


Figure 195 Graph Google Maps avec cartes de densité

Ou encore afficher des barres de données en des lieux précis avec le fichier de données suivant:

```
"University";"Count";"Category";"lat";"long"
"1";"EPFL";3;"Physics";46.8;6.8
"2";"EPFL";5;"Maths";46.8;6.8
"3";"EPFL";8;"Bioscience";46.8;6.8
"4";"EPFL";4;"Social Science";46.8;6.8
"5";"EPFZ";7;"Physics";47.5;8.8
"6";"EPFZ";6;"Maths";47.5;8.8
"7";"EPFZ";6;"Bioscience";47.5;8.8
"8";"EPFZ";1;"Social Science";47.5;8.8
"9";"UNIGE";5;"Physics";46.3;6.3
"10";"UNIGE";3;"Maths";46.3;6.3
"11";"UNIGE";10;"Bioscience";46.3;6.3
"12";"UNIGE";7;"Social Science";46.3;6.3
"13";"HEC St-Gall";4;"Physics";47.3;10
"14";"HEC St-Gall";6;"Maths";47.3;10
"15";"HEC St-Gall";8;"Bioscience";47.3;10
"16";"HEC St-Gall";4;"Social Science";47.3;10
```

et le script utilisant la commande **geom_subplot( )** :

```
> library(ggplot2)
> library(ggmap)
> library(ggsubplot)
> mydata<-read.csv("C:/ggmapBars.csv",header=T,sep=";")
> swiss <- get_map("Switzerland",zoom=7)
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Switzerland&zoo$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Swi$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
> p <- ggmap(swiss,extent="device",legend="topleft")
> p+geom_subplot(data=mydata,mapping=aes(x =long,y=lat,group=University,
+ subplot=geom_bar(aes(x=Category,y=Count,color=Category,stat= "identity"))))
Mapping a variable to y and also using stat="bin".
```

Ce qui donnera:

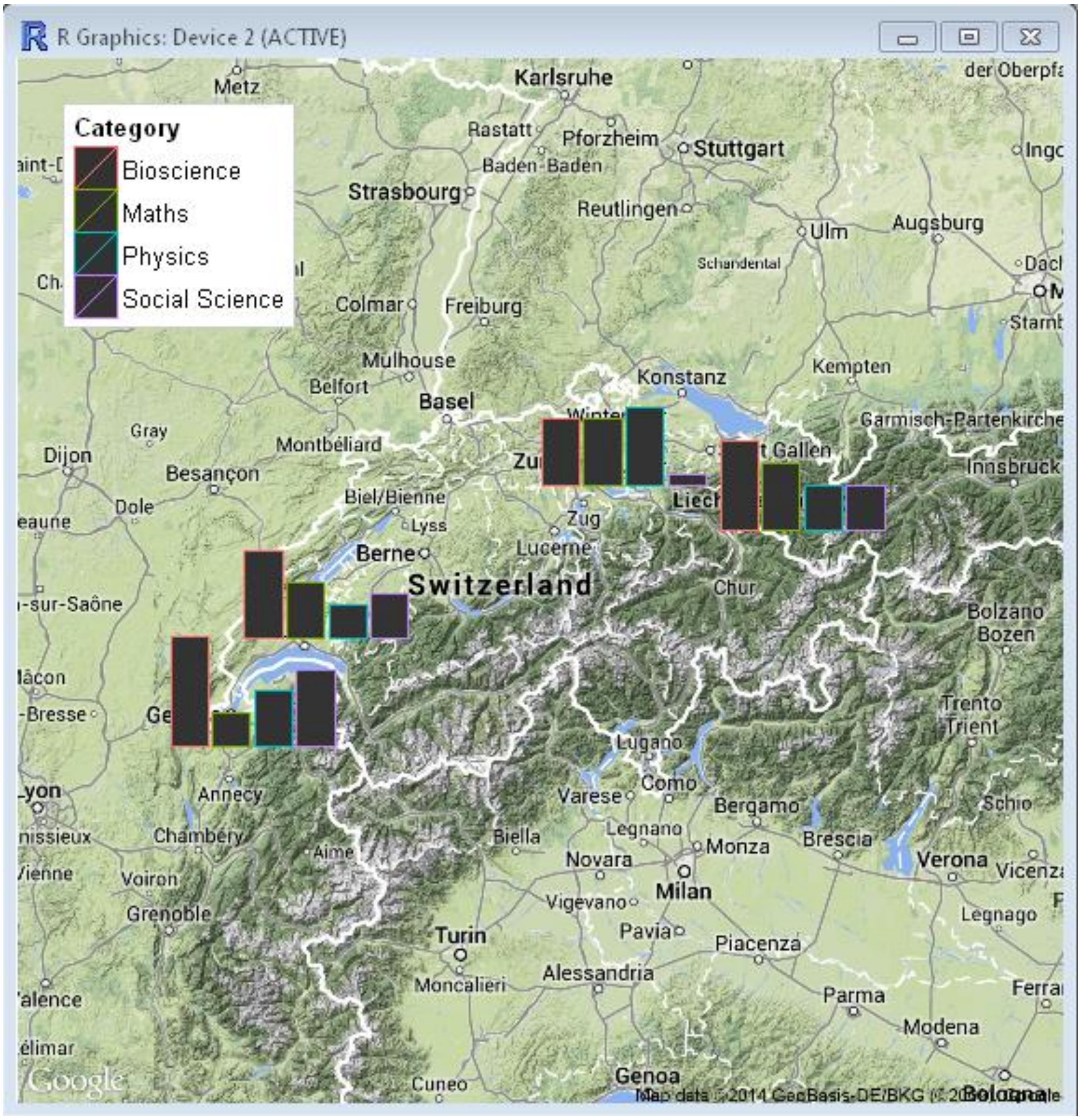
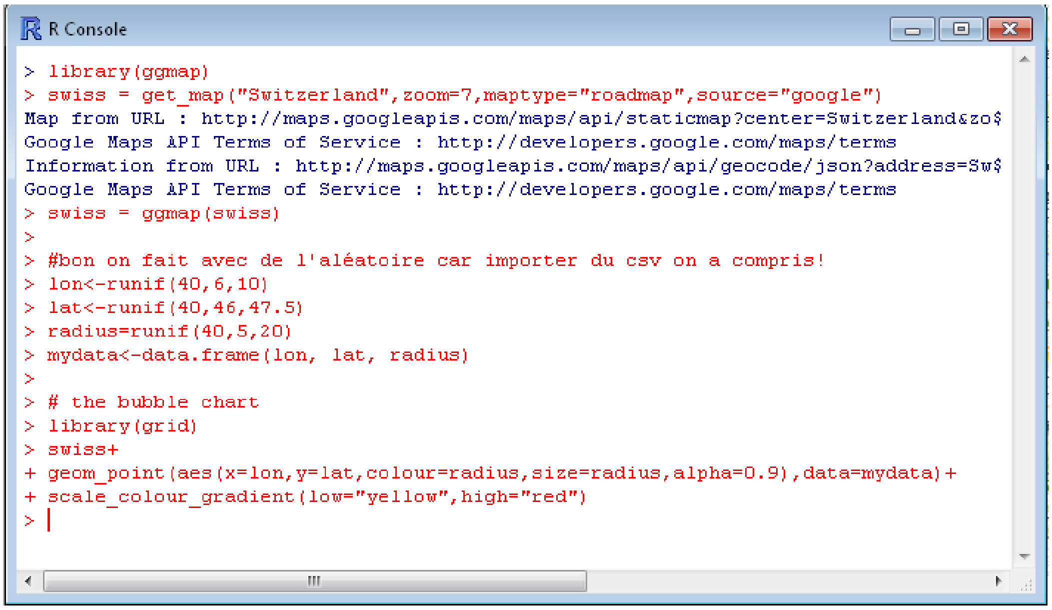


Figure 196 Graph Google Maps avec diagrammes à barres

On peut aussi s'amuser avec faire des cartes à bulles (remarquez les 0.9 dans la légende qui est mal compris par R car il s'agit de l'alpha... il faudra le masquer a priori par un rectangle blanc):



Ce qui donne:

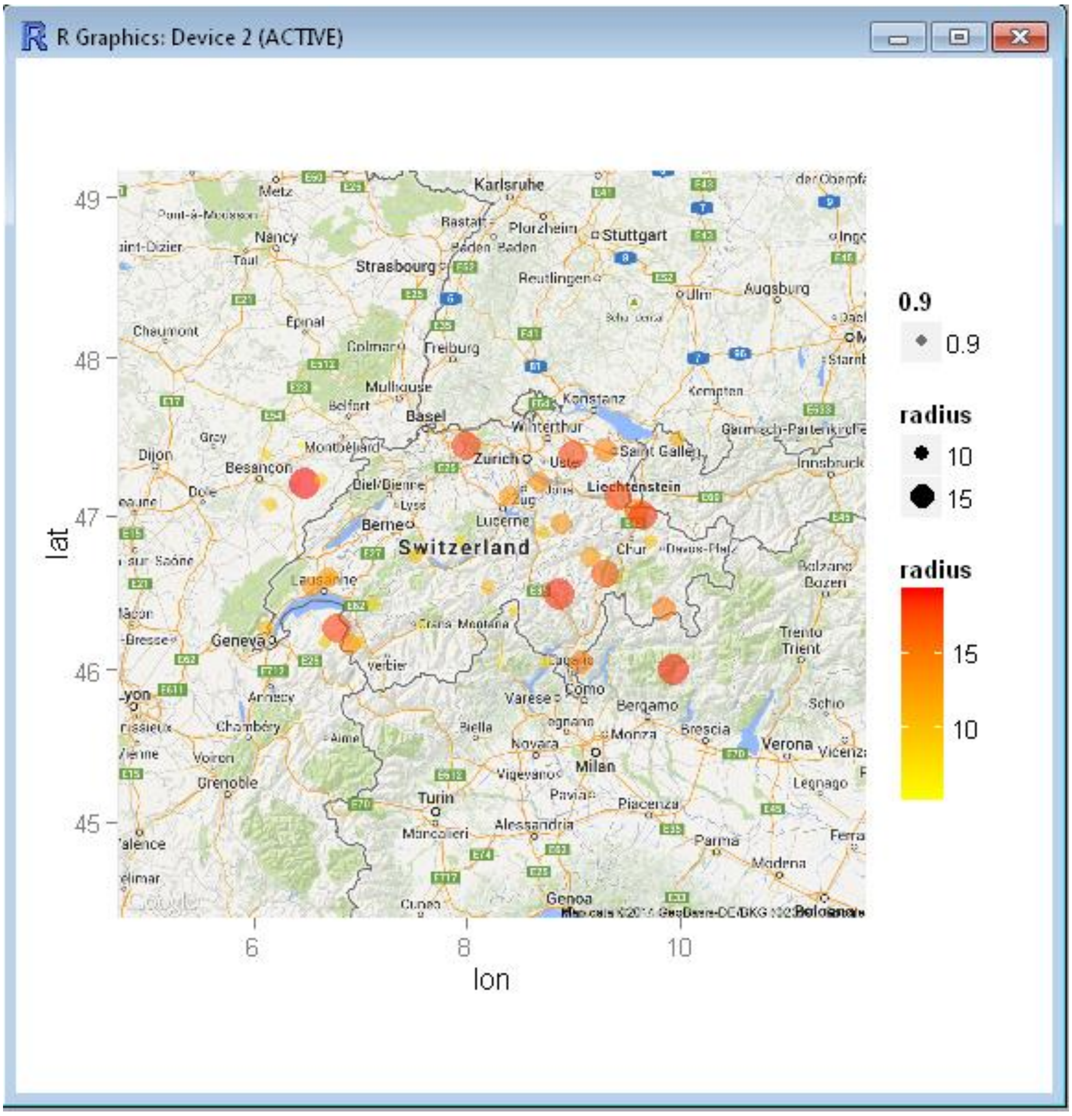
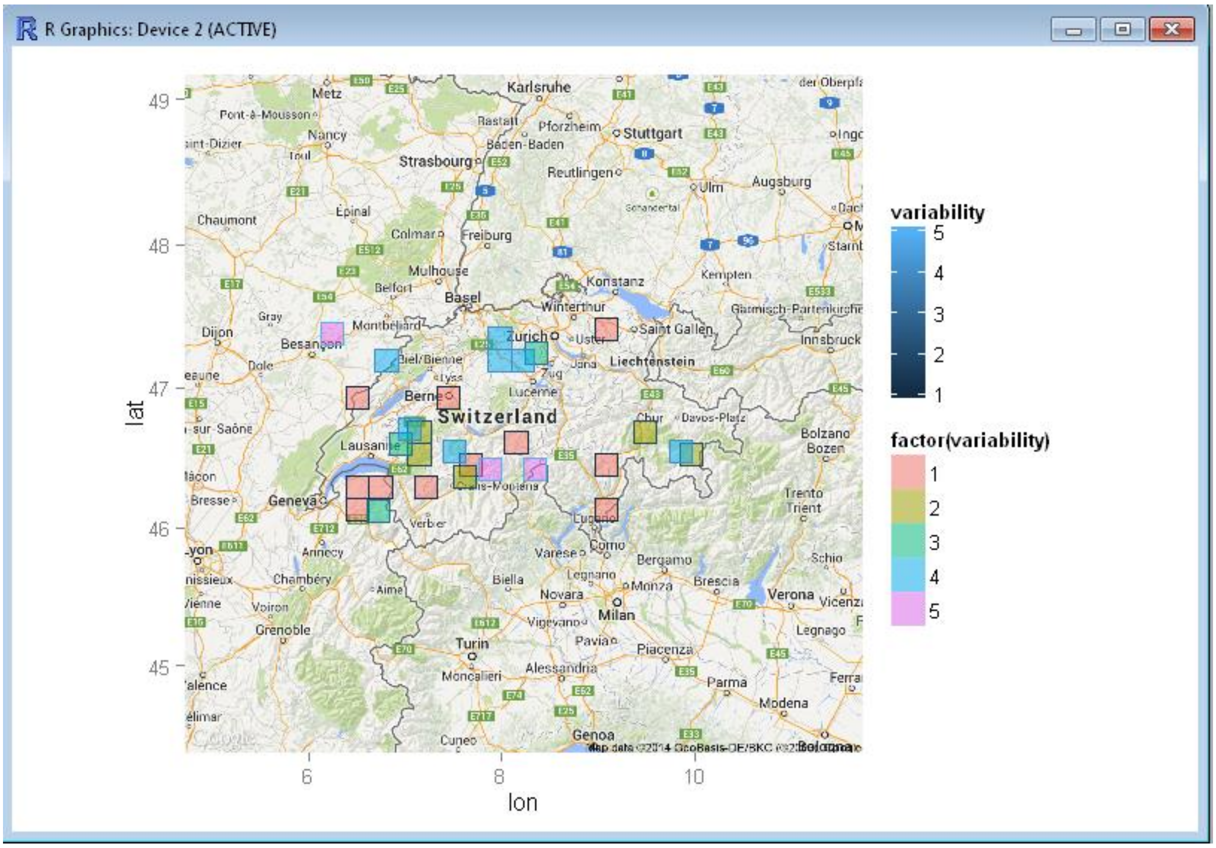


Figure 197 Graph Google Maps avec disques

Ou dans un autre style en utilisant la commande **stat_bind2d( )** :

```
> library(ggmap)
> #on fait aussi avec de l'aléatoire
> n<-30
> lon<-runif(n,6,10)
> lat<-runif(n,46,47.5)
> variability=sample(c(1:5),n,replace=TRUE)
> mydata<-data.frame(lon,lat,variability)
>
> swiss<-get_map("Switzerland",zoom=7,maptype="roadmap",source="google")
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Switzerland&zoo$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Swi$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
> swiss<-ggmap(swiss)
>
> mydata<-data.frame(lon,lat,variability)
> swiss+stat_bin2d(aes(x=lon,y=lat,colour=variability,fill=factor(variability)),
+ size=0.10,alpha=0.5,data=mydata)
>
```

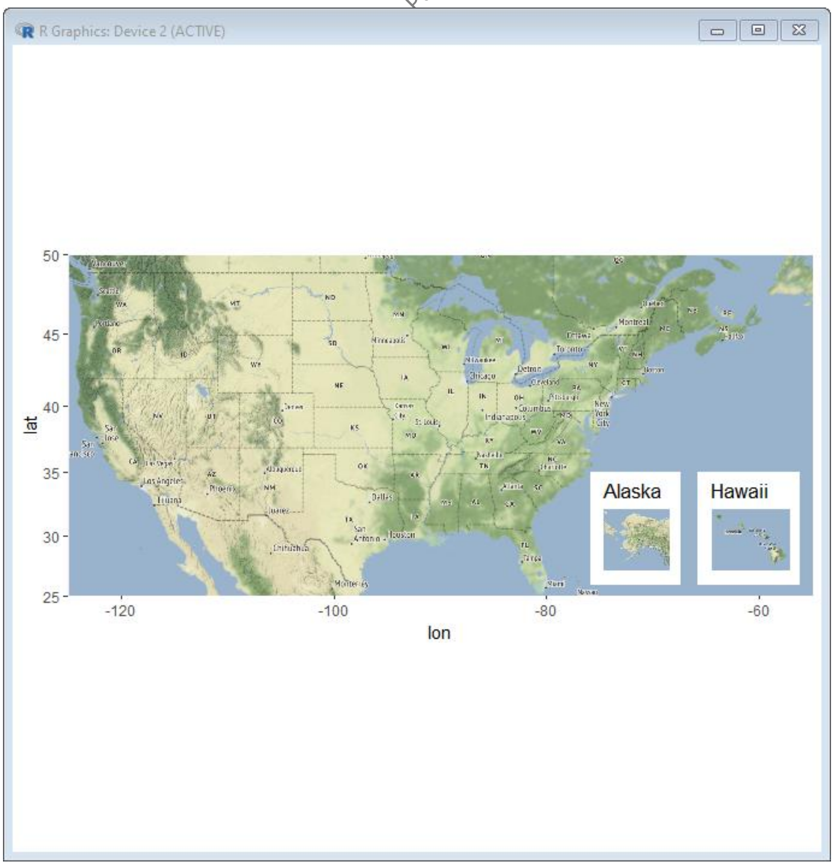
Ce qui donne:



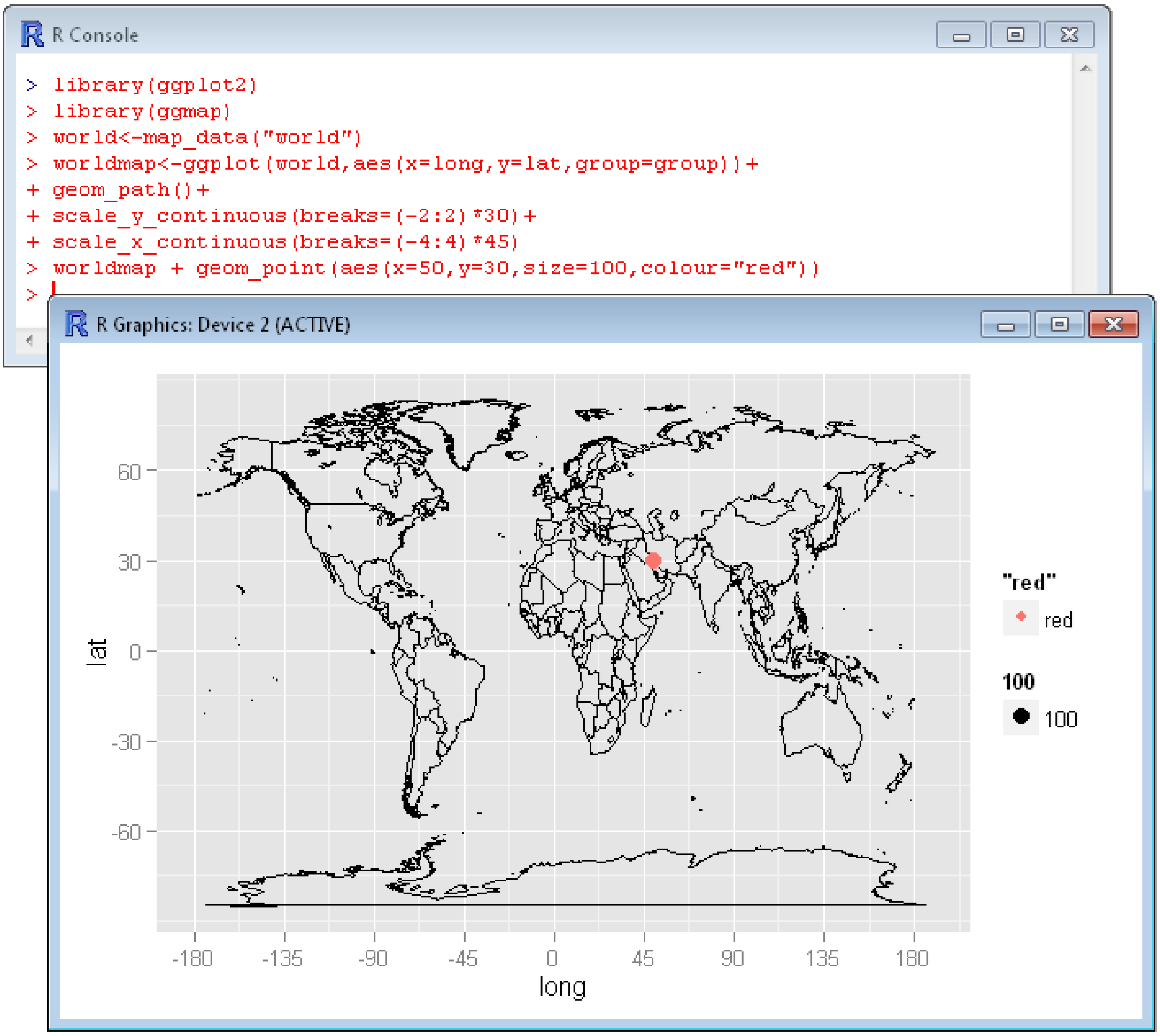
On peut également afficher des cartes détaillées dans un carte:

```
> library("ggmap")
>
> usa_bbox <- c(left = -125, bottom = 25, right = -55, top = 50)
> usa_main_map <- get_stamenmap(usa_bbox, zoom = 5, maptype = "terrain")
> #on prend la carte globale des USA
> p_main <- ggmap(usa_main_map)
>
>
> #on construit le premier sous-ensemble qui sera l'Alaska
> alaska_bbox <- c(left = -180, bottom = 50, right = -128, top = 72)
> alaska_map <- get_stamenmap(alaska_bbox, zoom = 5, maptype = "terrain")
> p_alaska <- ggmap(alaska_map) + labs(title = "Alaska") +
+     theme(axis.title = element_blank(), axis.text  = element_blank(),
+           axis.ticks = element_blank())
>
>
> #on construit le deuxième sous-ensemble qui sera Hawai
> hawaii_bbox <- c(left = -160, bottom = 18.5, right = -154.5, top = 22.5)
> hawaii_map <- get_stamenmap(hawaii_bbox, zoom = 6, maptype = "terrain")
> p_hawaii <- ggmap(hawaii_map) +
+     labs(title = "Hawaii") +
+     theme(axis.title = element_blank(),
+           axis.text  = element_blank(),
+           axis.ticks = element_blank())
>
>
> #on fusionne le tout
> library("grid")
> p_main +
+     inset(ggplotGrob(p_alaska), xmin = -76.7, xmax = -66.7, ymin = 26, ymax = 35) +
+     inset(ggplotGrob(p_hawaii), xmin = -66.5, xmax = -55.5, ymin = 26, ymax = 35)
>
```

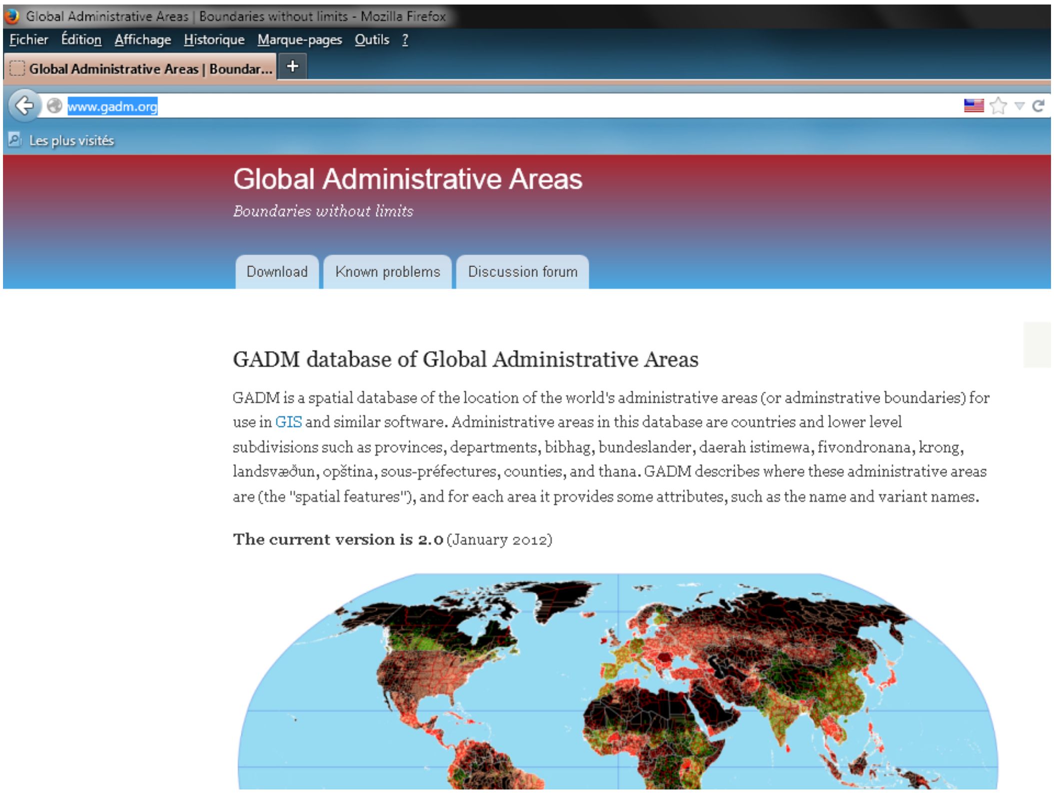
Ce qui donne:



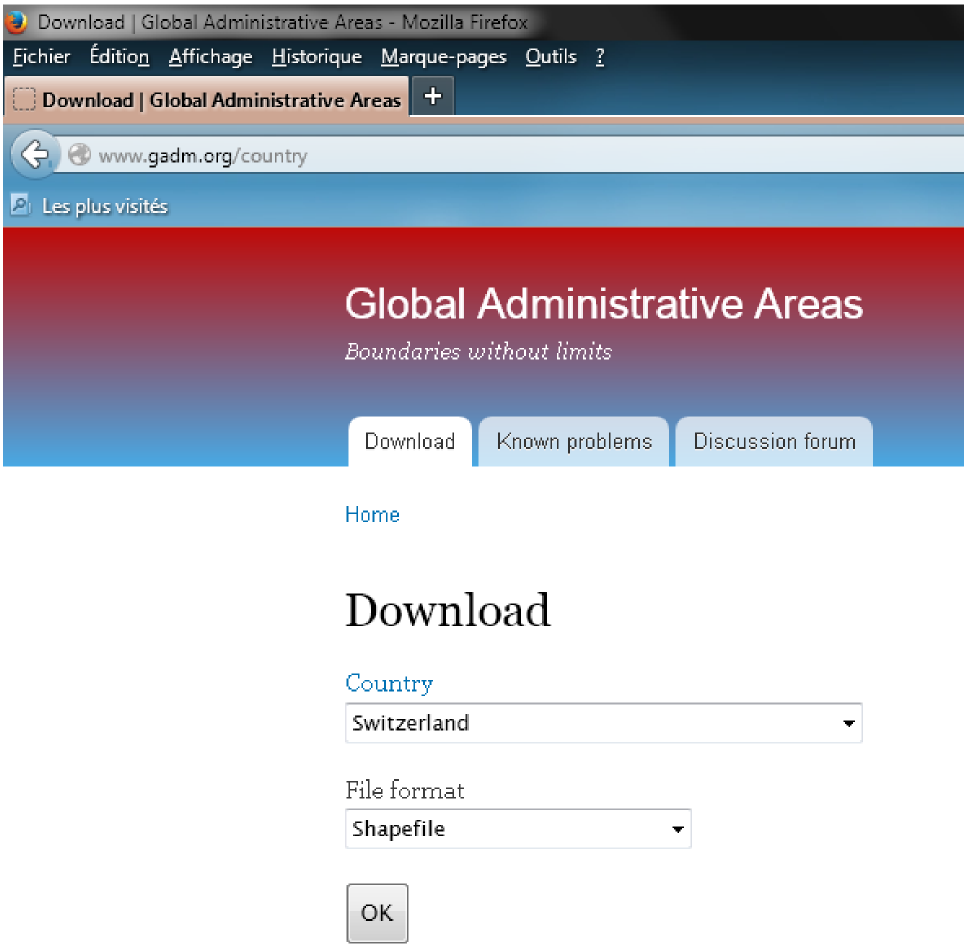
Au niveau mondial avec un seul point, nous pouvons faire par exemple en utilisant les commandes **map_data( )** et **ggplot( )** :



Passons maintenant à un cas très utile dans la meilleure démocratie directe du monde. D'abord téléchargez sur le site web suivant http://www.gadm.org :



et téléchargez le fichier *.zip:



Décompressez tout le dossier *.zip dans un dossier unique en préservant tous les fichiers d'une même famille de niveau de détails:

Nom	Date	Type	Taille	Mots-clés
CHE_adm0.csv	20.12.2011 14:42	Fichier CSV	1 Ko	
CHE_adm0.dbf	20.12.2011 11:58	Fichier DBF	2 Ko	
CHE_adm0.prj	20.12.2011 11:58	Fichier PRJ	1 Ko	
CHE_adm0.shp	20.12.2011 11:58	Fichier SHP	205 Ko	
CHE_adm0.shx	20.12.2011 11:58	Fichier SHX	1 Ko	
CHE_adm1.csv	20.12.2011 14:55	Fichier CSV	4 Ko	
CHE_adm1.dbf	20.12.2011 12:48	Fichier DBF	13 Ko	
CHE_adm1.prj	20.12.2011 12:48	Fichier PRJ	1 Ko	
CHE_adm1.shp	20.12.2011 12:48	Fichier SHP	765 Ko	
CHE_adm1.shx	20.12.2011 12:48	Fichier SHX	1 Ko	
CHE_adm2.csv	20.12.2011 15:06	Fichier CSV	15 Ko	
CHE_adm2.dbf	20.12.2011 13:21	Fichier DBF	97 Ko	
CHE_adm2.prj	20.12.2011 13:21	Fichier PRJ	1 Ko	
CHE_adm2.shp	20.12.2011 13:21	Fichier SHP	1'718 Ko	
CHE_adm2.shx	20.12.2011 13:21	Fichier SHX	2 Ko	
CHE_adm3.csv	20.12.2011 15:15	Fichier CSV	292 Ko	
CHE_adm3.dbf	20.12.2011 13:36	Fichier DBF	1'671 Ko	
CHE_adm3.prj	20.12.2011 13:36	Fichier PRJ	1 Ko	
CHE_adm3.shp	20.12.2011 13:36	Fichier SHP	6'700 Ko	
CHE_adm3.shx	20.12.2011 13:36	Fichier SHX	22 Ko	

Ensuite, nous utilisons les packages **rgdal**, **maptools**, **plyr**, **RColorBrewer**, **ggplot2** avec la série de commandes suivantes (merci à Ahmadou Dicko!) en plusieurs étapes (car c'est un peu long).

1ière étape on charge les packages et on lit le contenu de fichier *.shp[20] en utilisant **readOGR( )** et **fortify( )** :

[20] Les fichiers *.shp (shapefiles) sont des fichiers au format ArcView et sont donc de facto le standard pour les fichiers SIG (bien que ce soit un format propriétaire). La plupart des cartes au monde sont dans ce format.

R Console

```
> library(rgdal)
> library(maptools)
> library(ggplot2)
> library(plyr)
> library(RColorBrewer)
> #on lit le fichier *.shp
> ch <- readOGR("C:/tmp",layer="CHE_adm1")
OGR data source with driver: ESRI Shapefile
Source: "C:/tmp", layer: "CHE_adm1"
with 26 features and 9 fields
Feature type: wkbPolygon with 2 dimensions
> # convertit les données en data frame
> suisse.ggmap<-fortify(ch,"NAME_1")
Regions defined for each Polygons
> # pour avoir les bon accents sur des noms comme Zürich
> suisse.ggmap$id<-iconv(suisse.ggmap$id,from="latin1",to="utf8")
> 
```

On génère des données fictives pour l'exemple (on a compris le principe de l'import de *.csv):

```
> # jeux de données factice qui peut tout aussi bien provenir d'un *.csv externe
> n<-length(unique(suisse.ggmap$id));n
[1] 26
> taux<-sample(1:100,size=n,replace=TRUE);taux
 [1] 30  6 99 94 28 81 39 24  2 40 20 12 92 11 69 93 64 35 26 67 94 70 43 82 50 60
> inscrit<-sample(2000:30000,size= n,replace=TRUE)
> suisse.emploi<-data.frame(canton=unique(suisse.ggmap$id),emploi=taux,inscrit=inscrit);suisse.emploi
   canton emploi inscrit
1       0     30   20530
2       1      6   10533
3       2     99   14125
4       3     94   15985
5       4     28   25324
6       5     81    5896
7       6     39   14549
8       7     24    5213
9       8      2    2305
10      9     40   23986
11     10     20   18271
12     11     12    7052
13     12     92    5520
14     13     11   11752
15     14     69   24142
16     15     93   16180
17     16     64   13181
18     17     35    8396
19     18     26   13361
20     19     67    3128
21     20     94   29600
22     21     70    8674
23     22     43   21495
24     23     82   26693
25     24     50    3046
26     25     60    4466
>
```

On fait quelques préparatifs sur les données:

```
> # on fusionne les deux bases
> suisse.fusion <- merge(suisse.ggmap, suisse.emploi, by.x = "id", by.y = "canton")
> # on reordonne
> suisse.fusion <- suisse.fusion[order(suisse.fusion$order), ]
>
> # et on prépare des points sur le centre des cantons
> mid_range<-function(x){mean(range(x,na.rm=TRUE))}
> centre.canton<-ddply(suisse.fusion,.(id),colwise(mid_range,.(lat,long)))
>
```

Et on trace en cumulant presque tout le savoir cumulé jusqu'ici concernant les cartes:

```
> # maintenant on peut dessiner
> ggplot(suisse.fusion, aes(long,lat)) +
+ geom_polygon(aes(group=group,fill=emploi),colour="grey60",size=.3) +
+ geom_point(data=centre.canton,aes(size=inscrit),colour="purple",alpha=.7) +
+ scale_fill_continuous(name = "taux en %",low="white",high="steelblue")+coord_map()
>
> # on peut discretiser la variable taux aussi :
> suisse.fusion<-mutate(suisse.fusion, taux_discr = cut(emploi,breaks=c(0,10,20,30,40,50,60,70,80,90,100),
+ labels=c("<10","10-20","20-30","30-40","40-50","50-60","60-70","70-80","80-90","90>"),right=FALSE))
>
> #utilisation de RColorBrewer, on retrouve le nombre de sectionnement du pourcentage=9
> couleur <- brewer.pal(10, "Spectral")
> ggplot(suisse.fusion, aes(long, lat))+
+ geom_polygon(aes(group=group, fill = taux_discr),colour="grey60",size=.3) +
+ geom_point(data = centre.canton, aes(size = inscrit),colour ="purple",alpha=.7) +
+ scale_fill_manual(name = "taux en %", values = couleur) +
+ coord_map()
>
```

Ce qui donne au final:

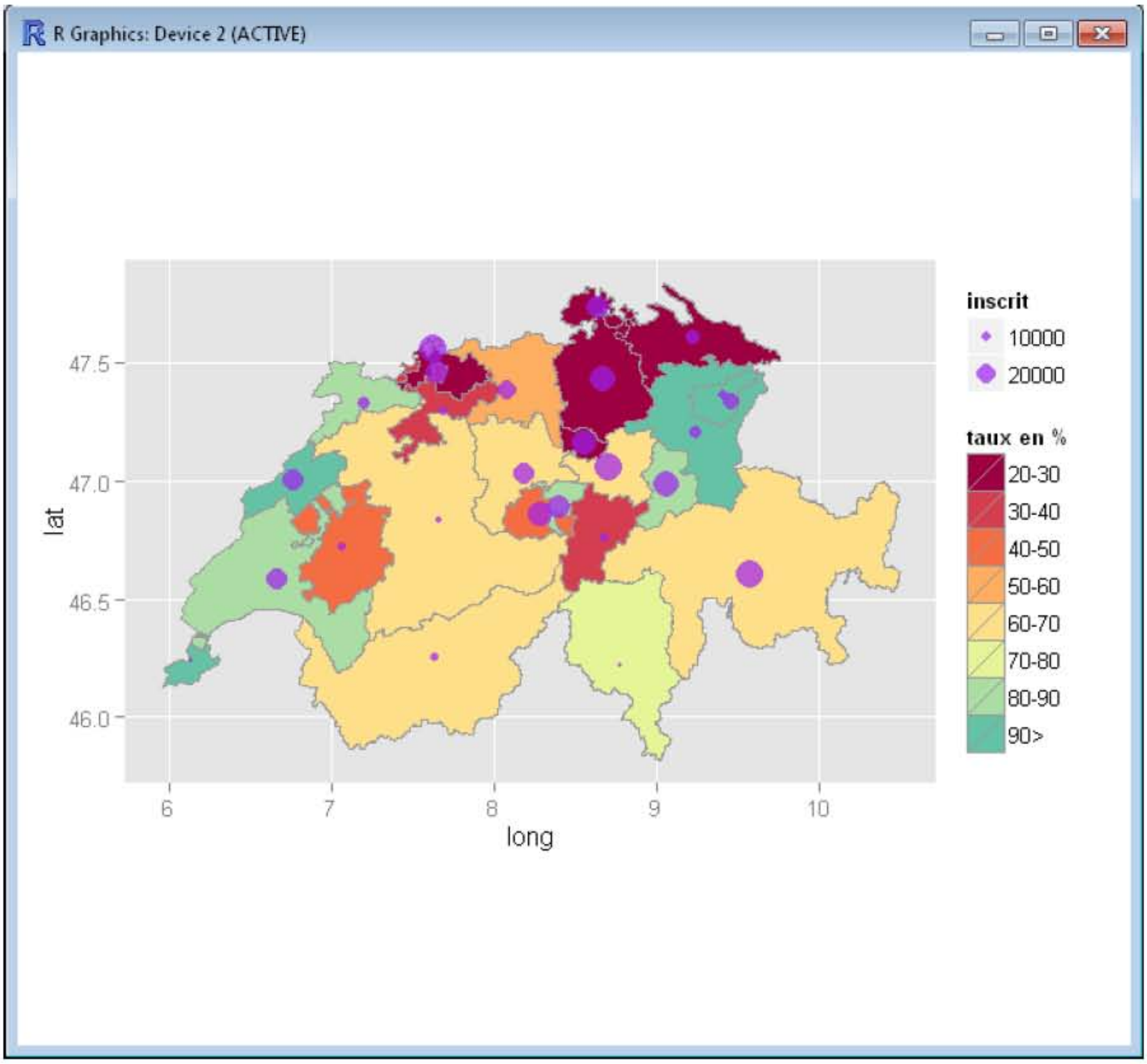


Figure 198 Graph Map (Suisse) avec zones et disques (ggplot)

Avec le package **plotKML** et un fichier *.gpx généré par les GPS (ou tout autre appareil emportant cette technologie) nous pouvons tracer le chemin parcouru par un élément mobile.

Nous partons du code XML suivant (fichier GPS_path.gpx):

```
GPS_path.gpx
1  <?xml version="1.0" encoding="UTF-8"?>
2  <gpx version="1.0">
3      <name>Example gpx</name>
4      <wpt lat="46.57638889" lon="8.89263889">
5          <ele>2372</ele>
6          <name>LAGORETICO</name>
7      </wpt>
8      <trk><name>Example gpx</name><number>1</number><trkseg>
9          <trkpt lat="46.57608333" lon="8.89241667"><ele>2376</ele><time>2007-10-14T10:09:57Z</time></trkpt>
10         <trkpt lat="46.57619444" lon="8.89252778"><ele>2375</ele><time>2007-10-14T10:10:52Z</time></trkpt>
11         <trkpt lat="46.57641667" lon="8.89266667"><ele>2372</ele><time>2007-10-14T10:12:39Z</time></trkpt>
12         <trkpt lat="46.57650000" lon="8.89280556"><ele>2373</ele><time>2007-10-14T10:13:12Z</time></trkpt>
13         <trkpt lat="46.57638889" lon="8.89302778"><ele>2374</ele><time>2007-10-14T10:13:20Z</time></trkpt>
14         <trkpt lat="46.57652778" lon="8.89322222"><ele>2375</ele><time>2007-10-14T10:13:48Z</time></trkpt>
15         <trkpt lat="46.57661111" lon="8.89344444"><ele>2376</ele><time>2007-10-14T10:14:08Z</time></trkpt>
16     </trkseg></trk>
17 </gpx>
```

Ensuite, nous écrivons le code suivant dans R utilisant la commande **readGPX( )** du package **plotKML** et la fonction get_googlmap( ) et ggmap( ) du package… **ggmap**:

```
R Console
> setwd("C:/tmp")
> LON_LAT<-get_map(location=c(lon=8.89263889,lat=46.5763889),zoom=8)
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=46.576389,8.892639&zoom=8&siz$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
> files<-dir(pattern="GPS_path.gpx")
> index<-c();latitude<-c();longitude<-c()
> for (i in 1:length(files)){
+ route<-readGPX(files[i])
+ location<-route$tracks[[1]][[1]]
+ index<-c(index,rep(i, dim(location)[1]))
+ latitude<-c(latitude, location$lat)
+ longitude<-c(longitude, location$lon)
+ }
> routes<-data.frame(cbind(longitude,latitude));routes
  longitude latitude
1  8.892417 46.57608
2  8.892528 46.57619
3  8.892667 46.57642
4  8.892806 46.57650
5  8.893028 46.57639
6  8.893222 46.57653
7  8.893444 46.57661
> map<-get_googlemap(center=c(8.89263889,46.5763889),zoom=16,markers=routes,path=routes,scale=2)
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=46.576389,8.892639&zoom=16&si$
Google Maps API Terms of Service : http://developers.google.com/maps/terms
> ggmap(map)
> |
```

Ce qui donne (**attention suite à un abus des utilisateurs Google peut exiger la création d'une clé api, la procédure est facile à suivre le cas échéant!**):

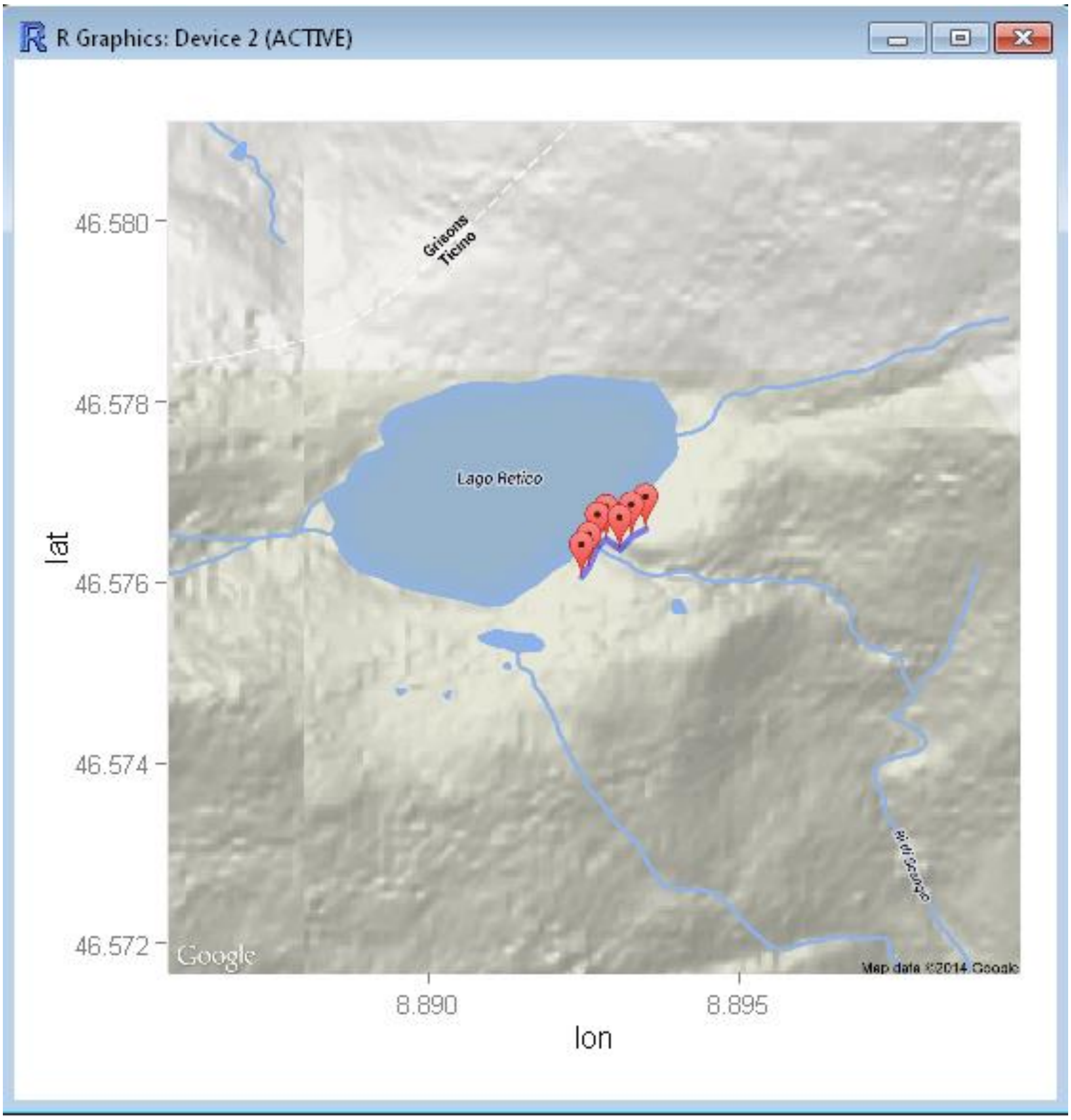
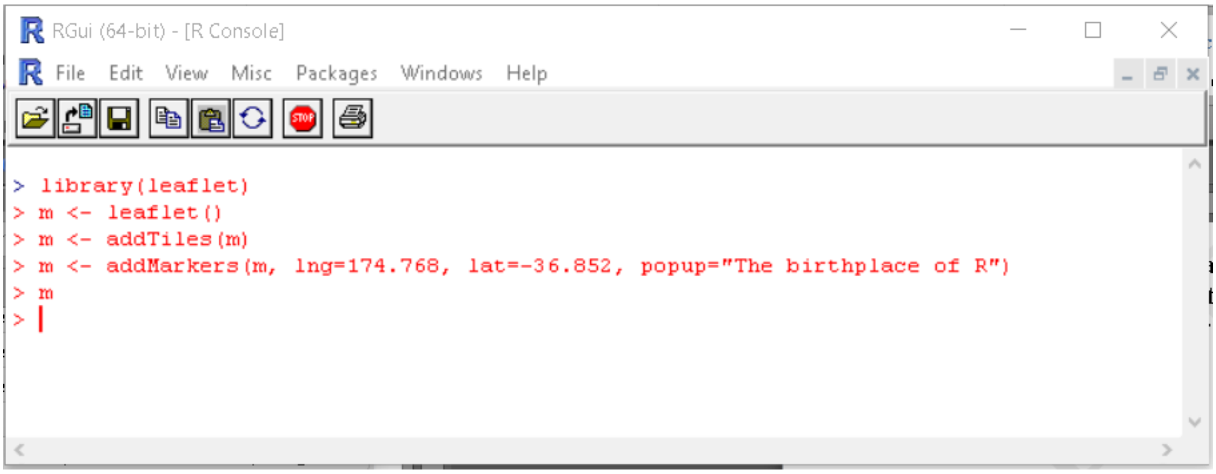
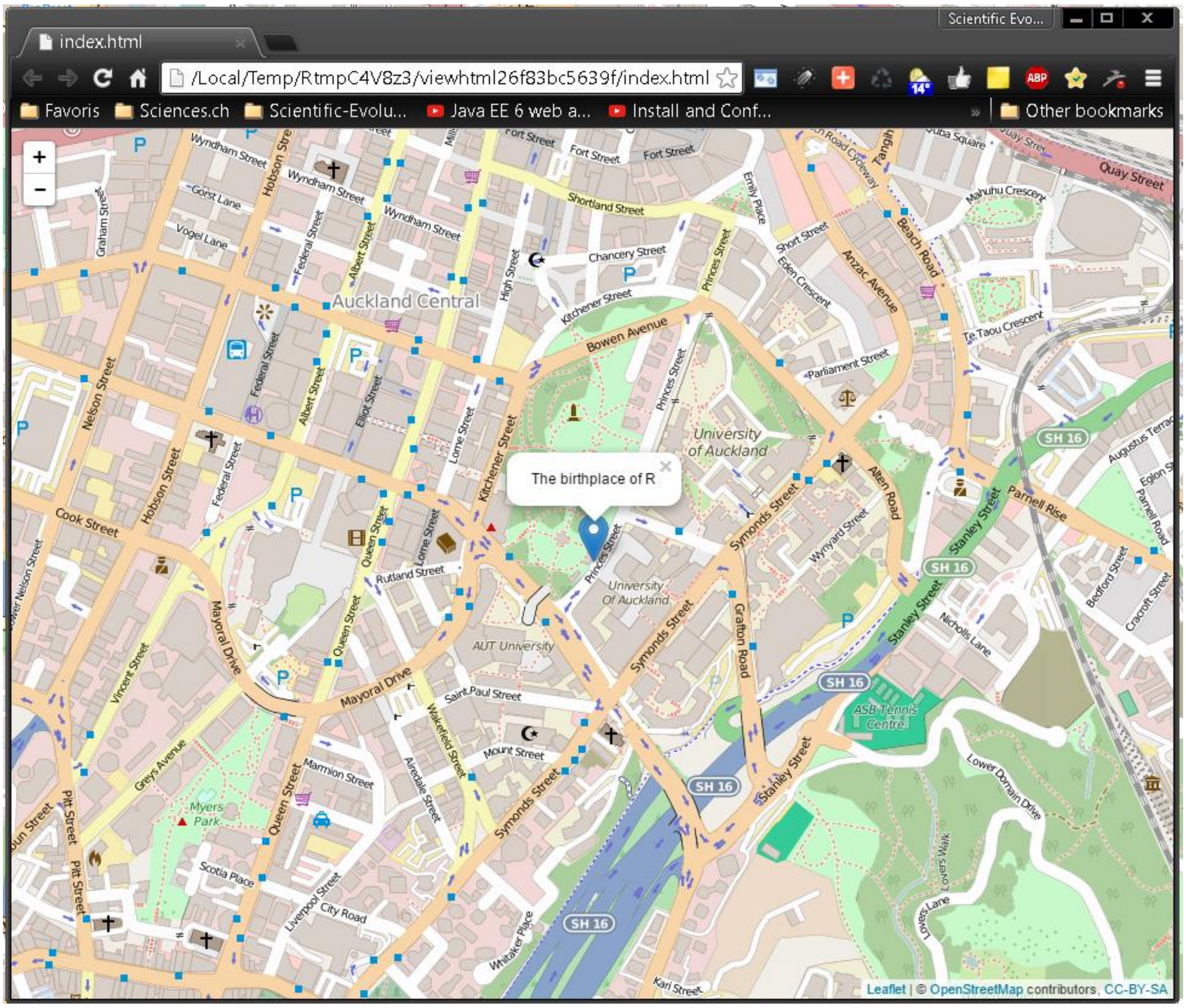


Figure 199 Graph Google Maps avec trajectoire GPS

Voyons maintenant un package sympathique nommé **leaflet** qui génère une page web html locale interactive utilisant Google Map. Voici un exemple pris du site de l'équipe de développement qui suffit à en montrer un aperçu du potentiel:



Ce qui donne:



Si on a une adresse internet avec une icône (une petite image), nous pouvons choisir le logo à mettre sur la carte:

```
> library(leaflet)
> fileUrl <- "http://www.graphicsfuel.com/wp-content/uploads/2012/09/emoticon-happy.png"
> download.file(fileUrl,"radar.png", mode = 'wb')
trying URL 'http://www.graphicsfuel.com/wp-content/uploads/2012/09/emoticon-happy.png'
Content type 'image/png' length 66448 bytes (64 KB)
downloaded 64 KB

> RadarICON <- makeIcon(  iconUrl = fileUrl,   iconWidth = 40, iconHeight = 40)
> m <- leaflet()
> m <- addTiles(m)
> m <- addMarkers(m, lng=174.768, lat=-36.852,icon = RadarICON,popup="The birthplace of R")
> m
```

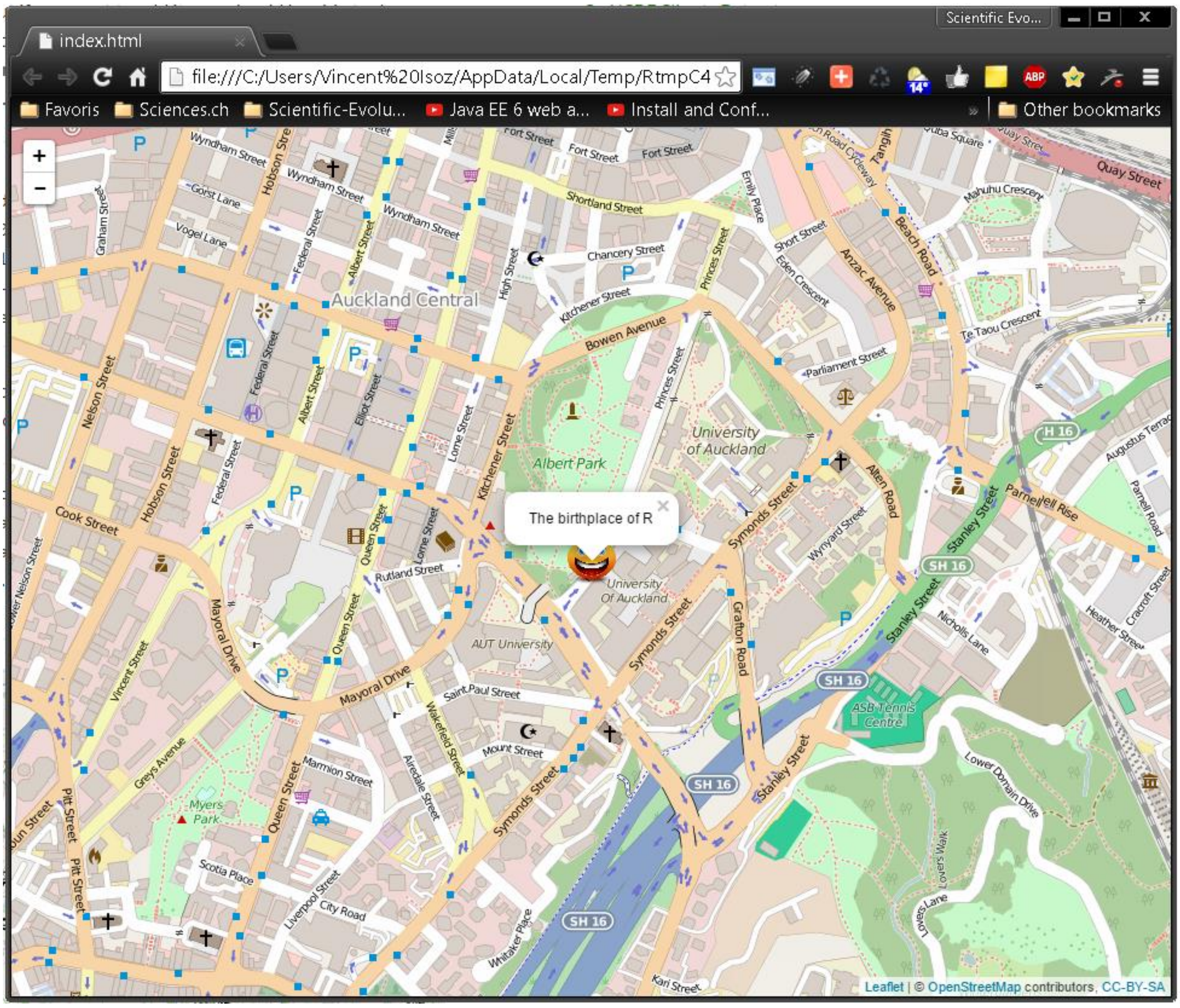


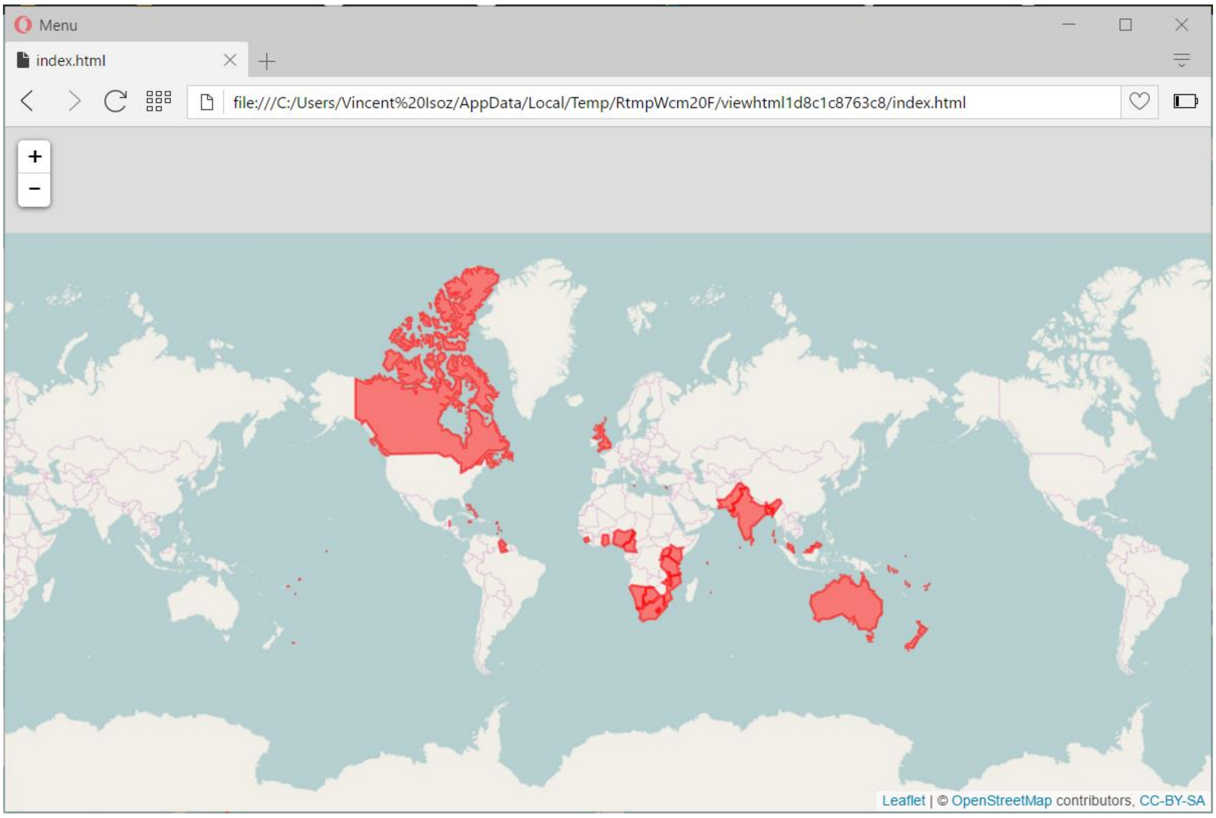
Figure 200 Graph Google Maps avec icônes et légendes

ou encore:

```
> library(rgdal)
> library(leaflet)
>
> download.file(file.path('http://www.naturalearthdata.com/http/',
+                         'www.naturalearthdata.com/download/50m/cultural',
+                         'ne_50m_admin_0_countries.zip'),
+               f <- tempfile())
trying URL 'http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/cultural/ne_$
Content type 'application/x-zip-compressed' length 817686 bytes (798 KB)
downloaded 798 KB

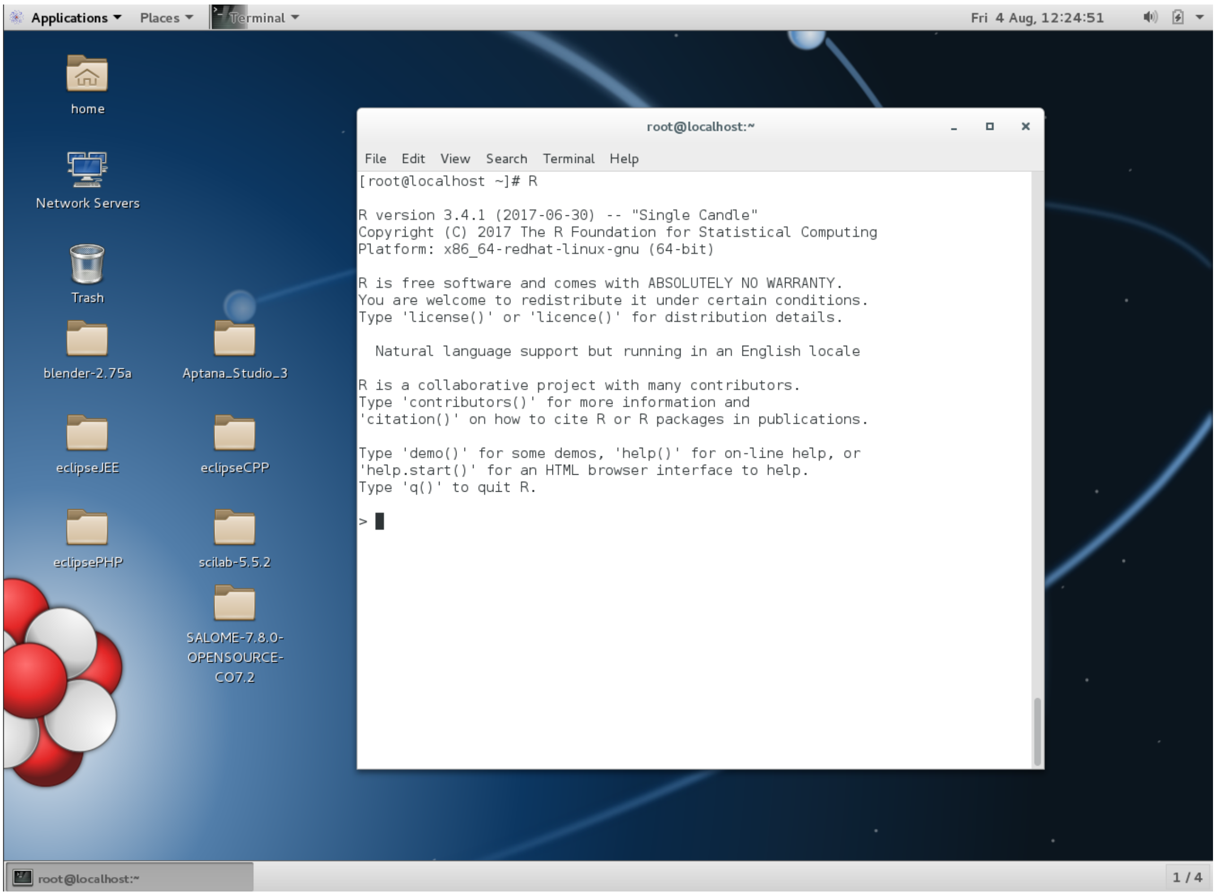
> unzip(f, exdir=tempdir())
>
> world <- readOGR(tempdir(), 'ne_50m_admin_0_countries', encoding='UTF-8')
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\VINCEN~1\AppData\Local\Temp\RtmpWcm20F", layer: "ne_50m_admin_0_countries"
with 241 features
It has 63 fields
>
> commonwealth <- c("Antigua and Barb.", "Australia", "Bahamas", "Bangladesh",
+   "Barbados", "Belize", "Botswana", "Brunei", "Cameroon", "Canada", "Cyprus",
+   "Dominica", "Fiji", "Ghana", "Grenada", "Guyana", "India", "Jamaica", "Kenya",
+   "Kiribati", "Lesotho", "Malawi", "Malaysia", "Maldives", "Malta", "Mauritius",
+   "Mozambique", "Namibia", "Nauru", "New Zealand", "Nigeria", "Pakistan", "Papua
+   New Guinea", "Rwanda", "St. Kitts and Nevis", "Saint Lucia", "St. Vin. and
+   Gren.", "Samoa", "Seychelles", "Sierra Leone", "Singapore", "Solomon Is.",
+   "South Africa", "Sri Lanka", "Swaziland", "Tanzania", "Tonga", "Trinidad and
+   Tobago", "Tuvalu", "Uganda", "United Kingdom", "Vanuatu", "Zamibia")
>
> leaflet() %>% addTiles() %>%
+   addPolygons(data=subset(world, name %in% commonwealth), weight=2,fillOpacity = 0.5,color="red")
```

Ce qui donne:



Pavage de sphère et statistiques

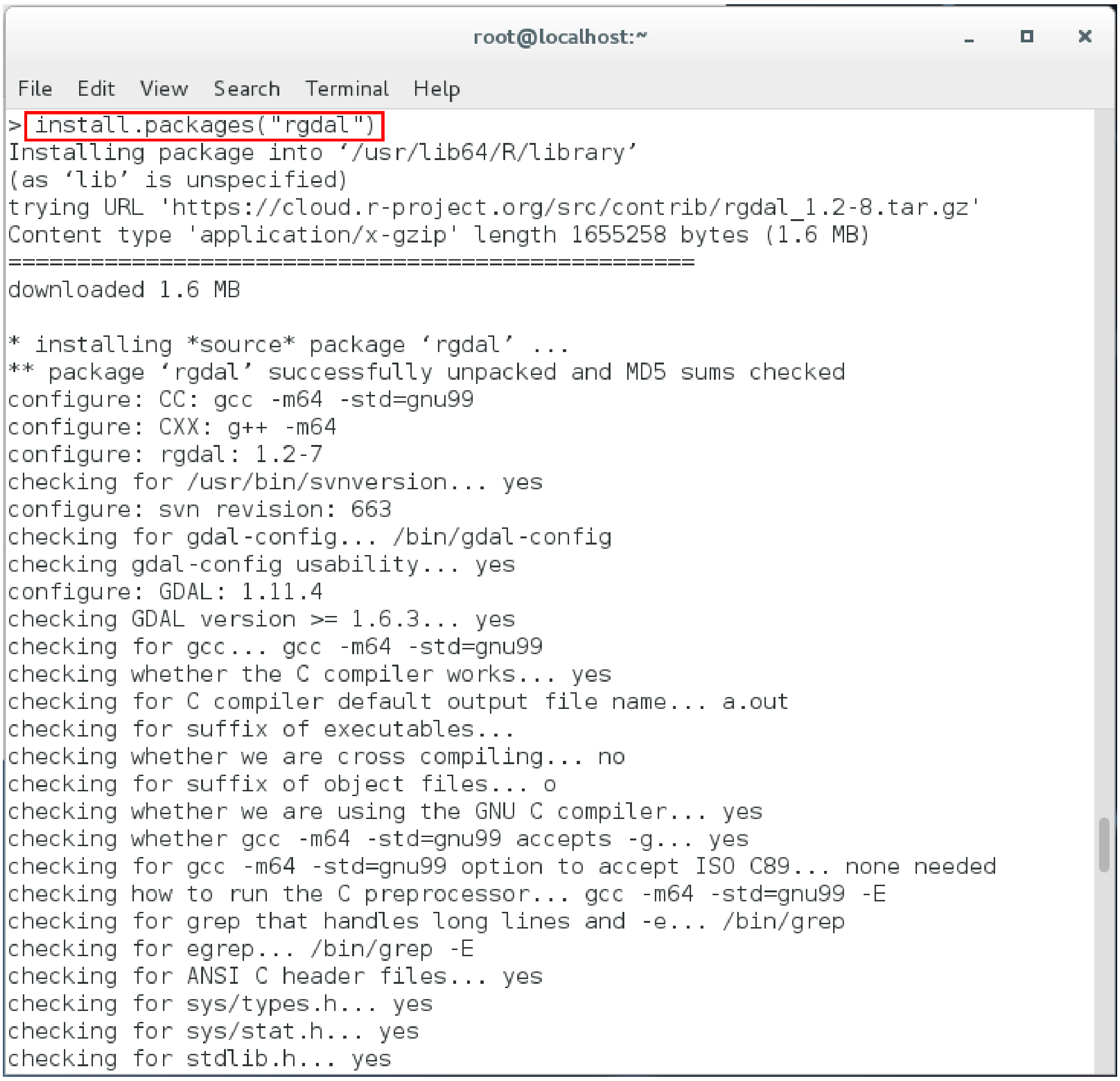
Le type de diagrammes que nous allons voici ici nécessite à ce jour Linux. Les exemples exécutés ci-dessous ont été fait sous Scientific Linux 7.3 avec R 3.4.1:



Avant de pouvoir exécuter les commandes ci-dessous il va vous falloir installer les composants suivant dans Scientific Linux:

```
yum install gdal gdal-devel
yum install proj
yum install proj-devel
yum install proj-nad
yum install proj-epsg
```

et après seulement on peut installer et charger le package de base nécessaire pour la suite qu'est **rgdal**:



```
root@localhost:~
File Edit View Search Terminal Help
> install.packages("rgdal")
Installing package into ‘/usr/lib64/R/library’
(as ‘lib’ is unspecified)
trying URL 'https://cloud.r-project.org/src/contrib/rgdal_1.2-8.tar.gz'
Content type 'application/x-gzip' length 1655258 bytes (1.6 MB)
==================================================
downloaded 1.6 MB

* installing *source* package ‘rgdal’ ...
** package ‘rgdal’ successfully unpacked and MD5 sums checked
configure: CC: gcc -m64 -std=gnu99
configure: CXX: g++ -m64
configure: rgdal: 1.2-7
checking for /usr/bin/svnversion... yes
configure: svn revision: 663
checking for gdal-config... /bin/gdal-config
checking gdal-config usability... yes
configure: GDAL: 1.11.4
checking GDAL version >= 1.6.3... yes
checking for gcc... gcc -m64 -std=gnu99
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc -m64 -std=gnu99 accepts -g... yes
checking for gcc -m64 -std=gnu99 option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -m64 -std=gnu99 -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
```

Ensuite seulement on peut installer le package **dggridR**:

```
admin@localhost:~
File Edit View Search Terminal Help
> install.packages("dggridR")
Installing package into '/home/admin/R/x86_64-redhat-linux-gnu-library/3.4'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/src/contrib/dggridR_1.0.1.tar.gz'
Content type 'application/x-gzip' length 2175240 bytes (2.1 MB)
==================================================
downloaded 2.1 MB

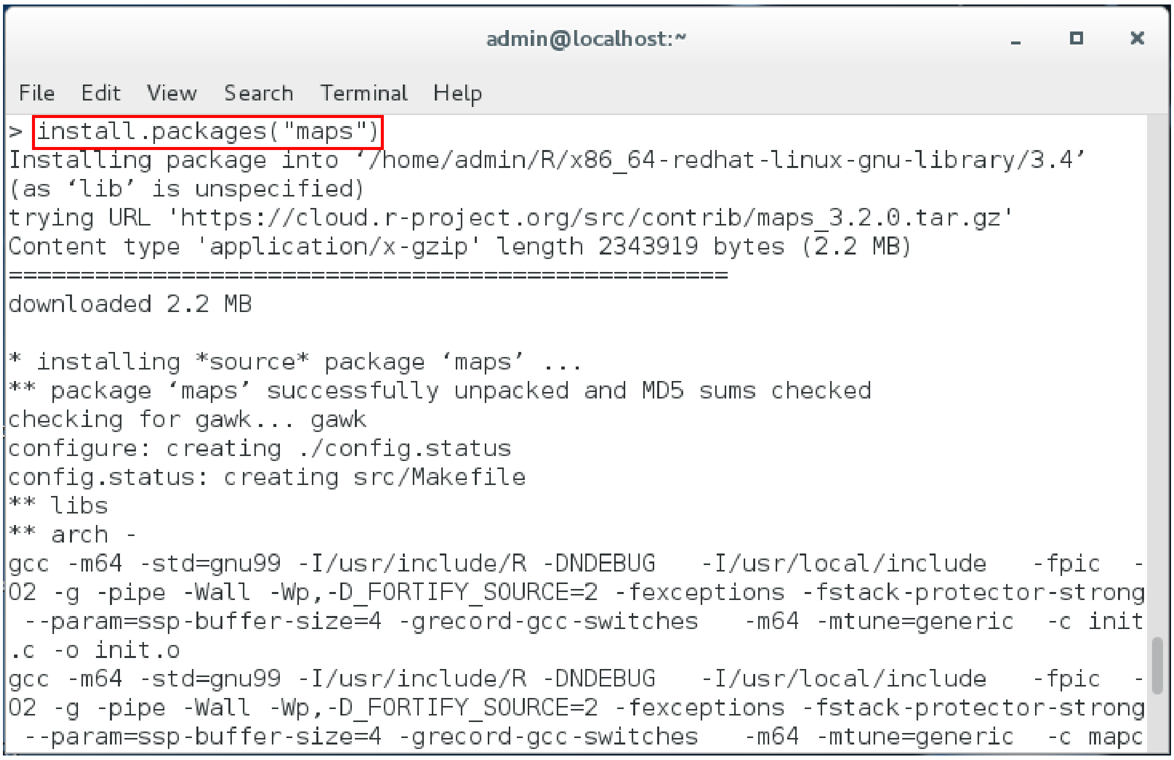
* installing *source* package 'dggridR' ...
** package 'dggridR' successfully unpacked and MD5 sums checked
** libs
make[1]: Entering directory `/tmp/RtmpCYW05m/R.INSTALL17261669e939/dggridR/src/l
ib/shapelib-1.3.0'
gcc -m64 -std=gnu99 -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fst
ack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches   -m64 -mtu
ne=generic  -fpic -c shpopen.c
gcc -m64 -std=gnu99 -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fst
ack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches   -m64 -mtu
ne=generic  -fpic -c dbfopen.c
gcc -m64 -std=gnu99 -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fst
ack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches   -m64 -mtu
ne=generic  -fpic -c safileio.c
gcc -m64 -std=gnu99 -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fst
```

et le package **dplyr**:

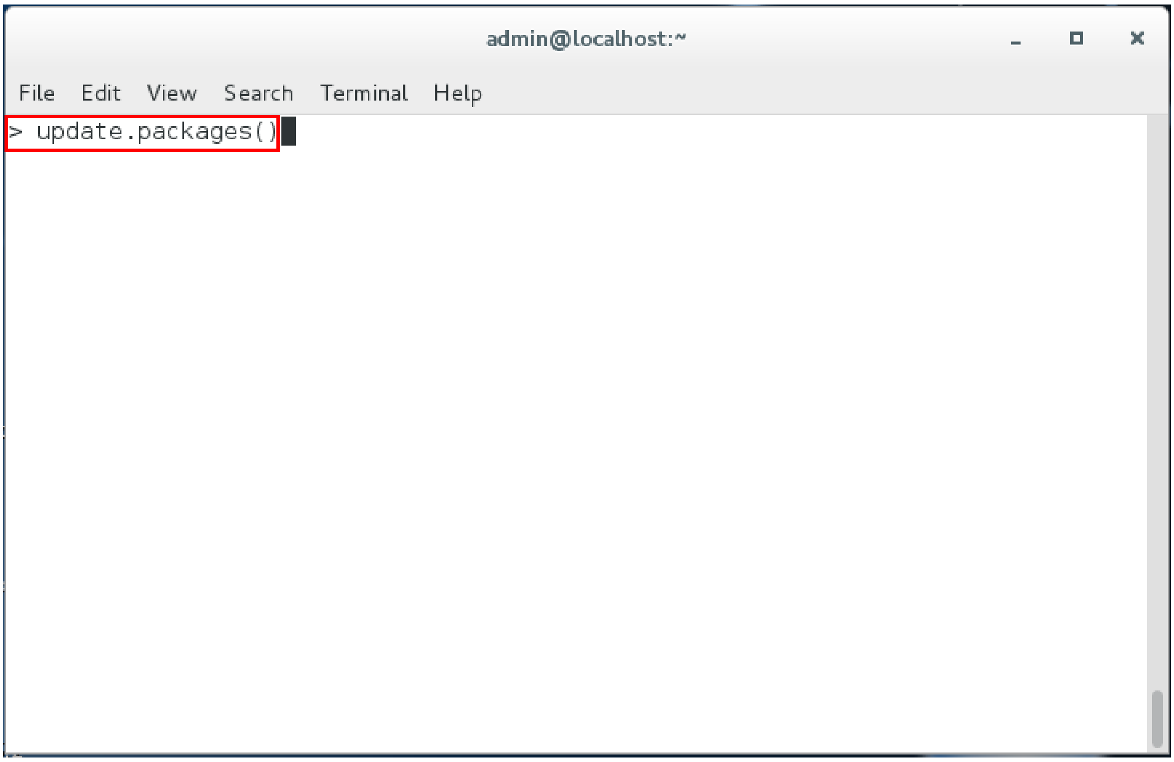
```
admin@localhost:~
File Edit View Search Terminal Help
> install.packages("dplyr")
Installing package into '/home/admin/R/x86_64-redhat-linux-gnu-library/3.4'
(as 'lib' is unspecified)
trying URL 'https://cloud.r-project.org/src/contrib/dplyr_0.7.2.tar.gz'
Content type 'application/x-gzip' length 692952 bytes (676 KB)
==================================================
downloaded 676 KB

* installing *source* package 'dplyr' ...
** package 'dplyr' successfully unpacked and MD5 sums checked
** libs
g++ -m64  -I/usr/include/R -DNDEBUG -I../inst/include -DCOMPILING_DPLYR -DBOOST_
NO_INT64_T -DBOOST_NO_INTEGRAL_INT64_T -DBOOST_NO_LONG_LONG -DRCPP_USING_UTF8_ER
ROR_STRING -I"/home/admin/R/x86_64-redhat-linux-gnu-library/3.4/Rcpp/include" -I
"/home/admin/R/x86_64-redhat-linux-gnu-library/3.4/BH/include" -I"/home/admin/R/
x86_64-redhat-linux-gnu-library/3.4/bindrcpp/include" -I"/home/admin/R/x86_64-re
dhat-linux-gnu-library/3.4/plogr/include" -I/usr/local/include   -fpic  -O2 -g -
pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param
=ssp-buffer-size=4 -grecord-gcc-switches   -m64 -mtune=generic  -c RcppExports.c
pp -o RcppExports.o
```

Assurez-vous d'installer aussi le package **maps**:



Ensuite, assurez-vous de lancer une mise à jour des packages:

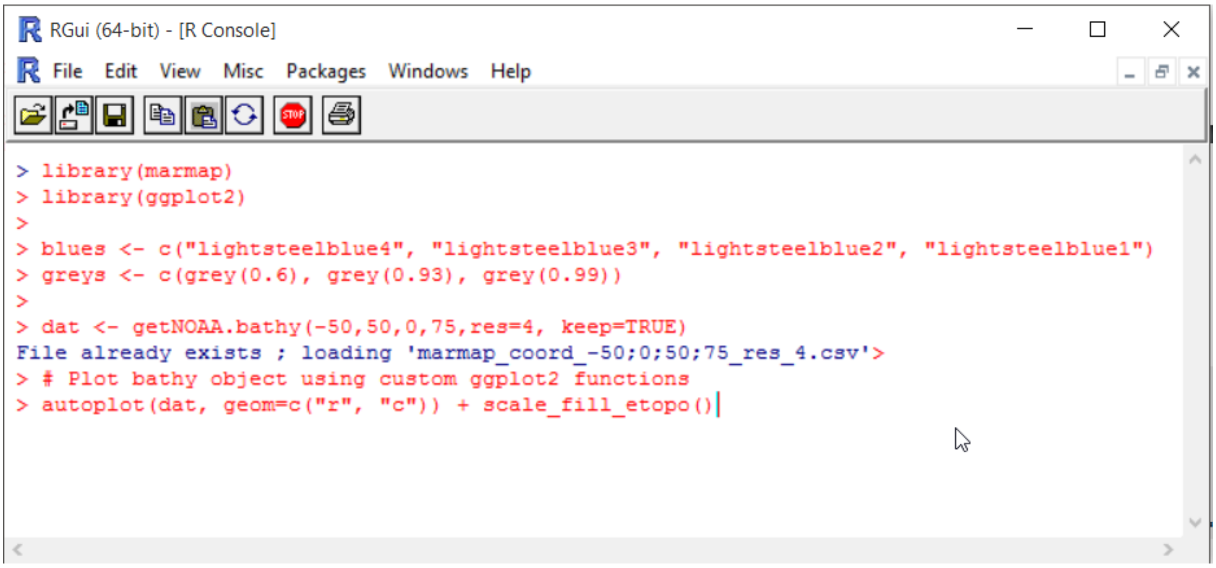


À terminer…

Topographie Marine

Nous allons voir ici qu'il existe un package spécifique pour la topologie Marine ce qui peut être utile pour l'industrie de la pêche, le recensement d'animaux marins, le traçage GPS ou l'exploration pétrolière.

L'idée consiste à utiliser le package **marmap**, les commandes sont assez intuitives pour ne pas avoir à être explicitées:



Ce qui donne:

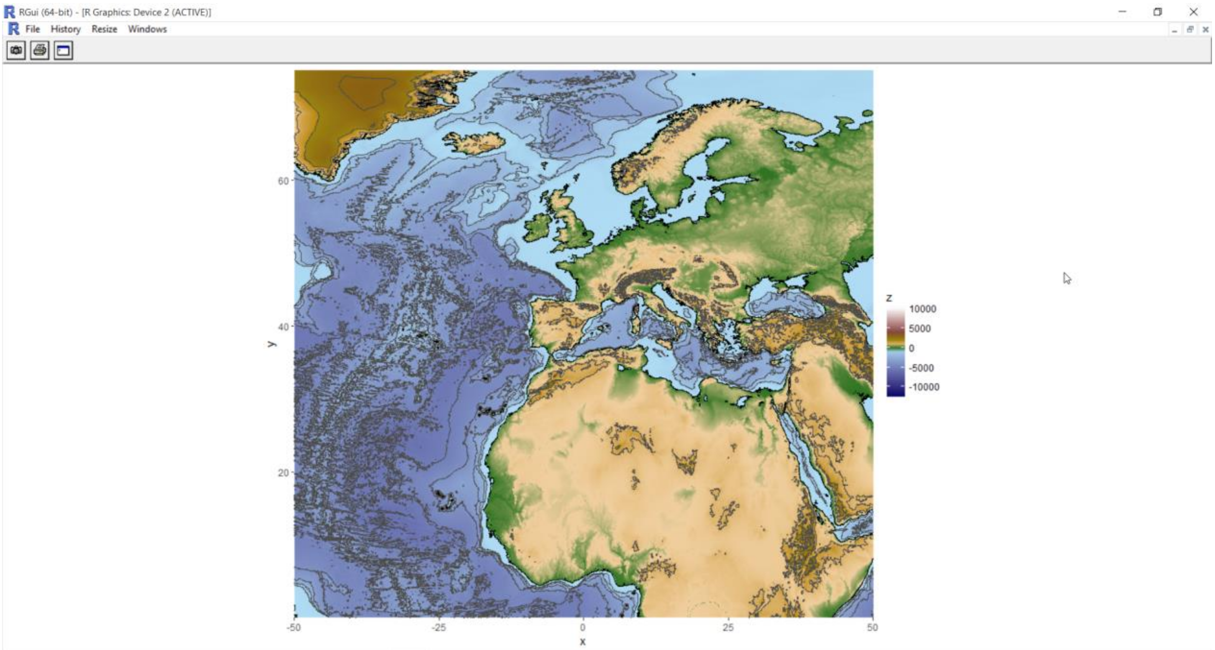


Figure 201 Graph de topographie marine

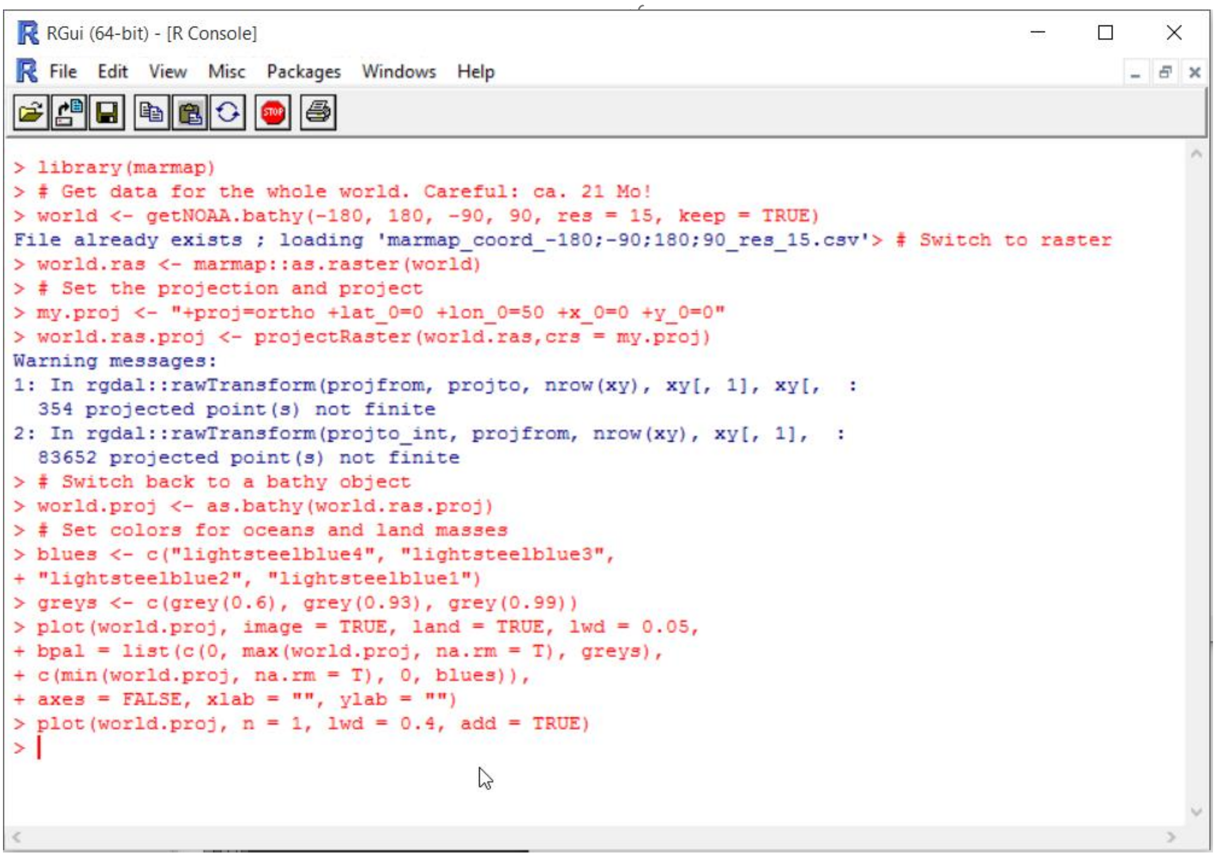
Attention!!! Le téléchargement des cartes depuis la National Oceanic And Atmospheric Administration:



en local peut prendre beauuuuucoup de temps suivant les latitudes et longitudes choisies.

ToDo: Représenter un point sur la carte, relier des points entre eux.

Nous pouvons aussi représenter la planète entière:



ToDo: Tourner la Terre, Enlever les courbes.

Cartogrammes

Les cartogrammes (choroplèthe) sont des cartes géographiques qui montrent non seulement l'amplitude d'une donnée via sa couleur mais également son poids via la taille déformée de la zone géographique correspondante.

Cartogrammes contigus

Pour faire un tel graphique, nous allons devoir installer le package **topogRam** qui est basé sur D3.js et qui peut à ce jour être installé que depuis GitHub (**mais attention!!! Sous Microsoft Windows, l'installation se passe souvent mal si le compte utilisateur contient des espaces!**). Voyons cela en l'installant d'abord:

```
> library("devtools")
> install_github("pvictor/topogRam")
Downloading GitHub repo pvictor/topogRam@master
from URL https://api.github.com/repos/pvictor/topogRam/zipball/master
Installing topogRam
"C:/PROGRA~1/R/R-35~1.0/bin/x64/R" --no-site-file --no-environ --no-save --no-resto$
  Isoz/AppData/Local/Temp/RtmpuEVgSj/devtools428439a75744/pvictor-topogRam-4773202"$
  --install-tests

* installing *source* package 'topogRam' ...
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
  converting help for package 'topogRam'
    finding HTML links ... done
    frRegPop                                html
    topogRam-shiny                          html
    topogRam                                html
** building package indices
** testing if installed package can be loaded
*** arch - i386
*** arch - x64
* DONE (topogRam)
In R CMD INSTALL
> 
```

Ensuite nous faisons un exemple de démonstration avec la fonction **topogRam( )** :



Ce qui donnera dans un navigateur:

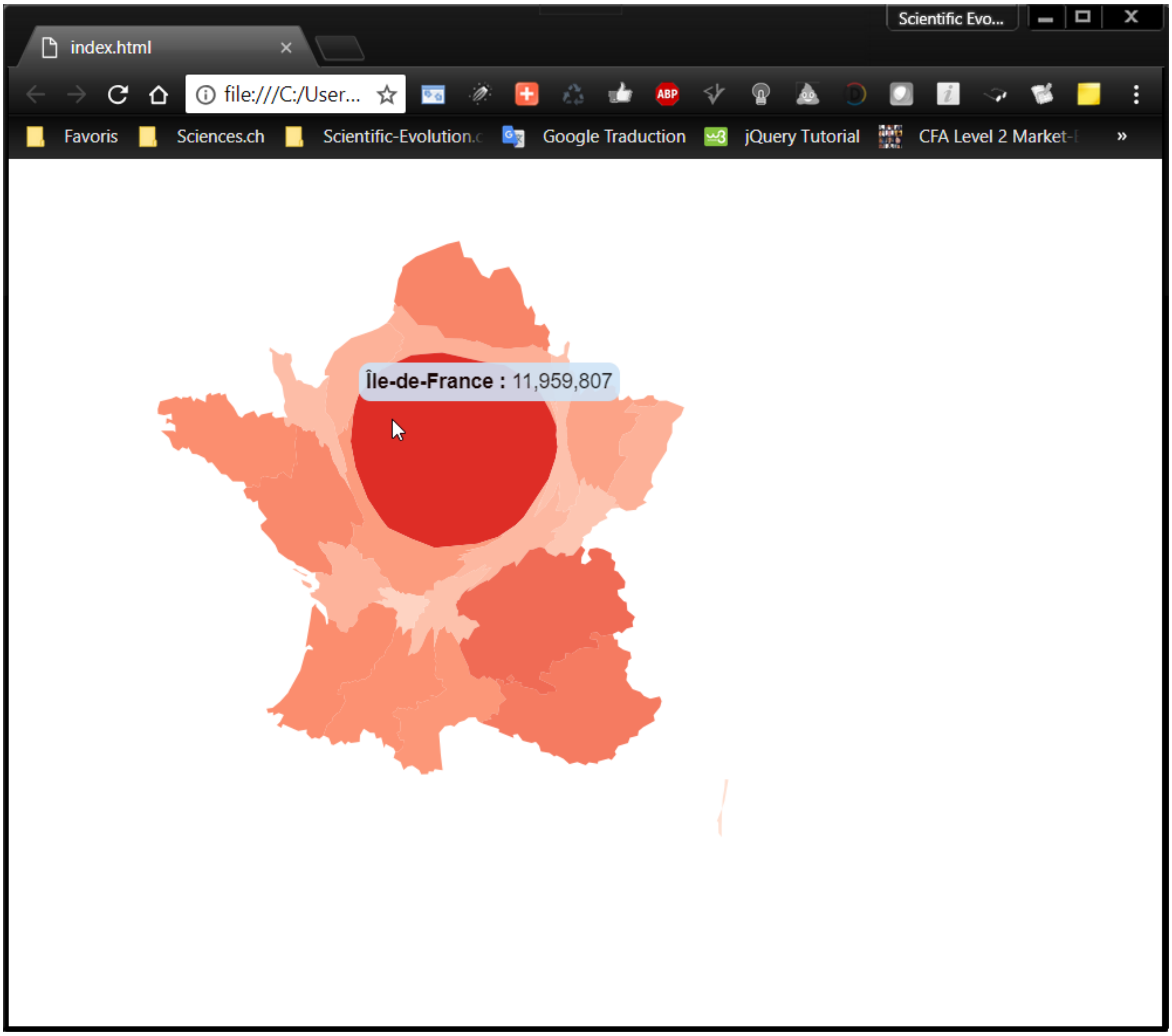
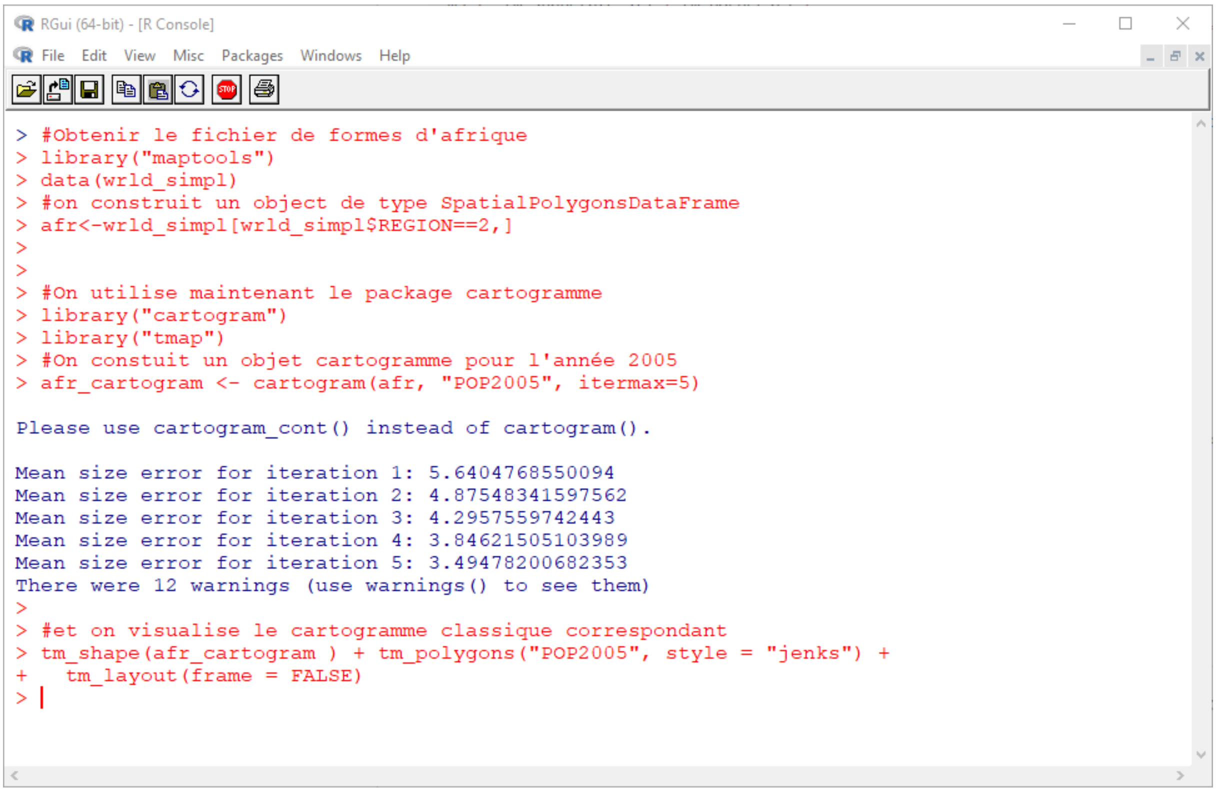


Figure 202 Graph de type cartogramme

Voici pour information ce que contient le jeu de données **frRegPop**:

```
> frRegPop
   id                     region  P13_POP C13_POP15P_CS1 C13_POP15P_CS$
1  11              Île-de-France 11959807           6075         29299$
2  21          Champagne-Ardenne  1339008          19834          3227$
3  22                   Picardie  1927142          12942          4381$
4  23            Haute-Normandie  1849652           9297          4401$
5  24                     Centre  2570548          22528          6768$
6  25            Basse-Normandie  1478712          19288          4416$
7  26                  Bourgogne  1642687          20702          4733$
8  31         Nord-Pas-de-Calais  4060741          14647          8159$
9  41                   Lorraine  2345197          11851          5355$
10 42                     Alsace  1868183           6727          4434$
11 43              Franche-Comté  1177096           9729          3009$
12 52           Pays de la Loire  3660852          41011          9788$
13 53                   Bretagne  3258707          39023          9400$
14 54           Poitou-Charentes  1789779          24588          5559$
15 72                  Aquitaine  3316889          34220         11492$
16 73              Midi-Pyrénées  2954157          39815          9928$
17 74                   Limousin   737509          13600          2196$
18 82                Rhône-Alpes  6399927          34040         20091$
19 83                   Auvergne  1357668          22982          4201$
20 91      Languedoc-Roussillon  2729721          23231          9982$
21 93 Provence-Alpes-Côte d'Azur  4953675          18420         18158$
22 94                      Corse   320208           2741          1471$
   C13_POP15P_CS8
1         1660825
2          178748
3          260826
4          235810
5          282176
6          168773
7          189510
8          666832
9          341050
10         243406
11         141813
12         394227
13         380459
14         196584
```

Sinon, sans passer par une page web, nous pouvons utiliser les packages **maptools**, **cartogram** et **tmap** avec principalement les fonction **cartogram( )**, **tm_shape( )**, **tm_polyongs( )** et **tm_layout( )**:

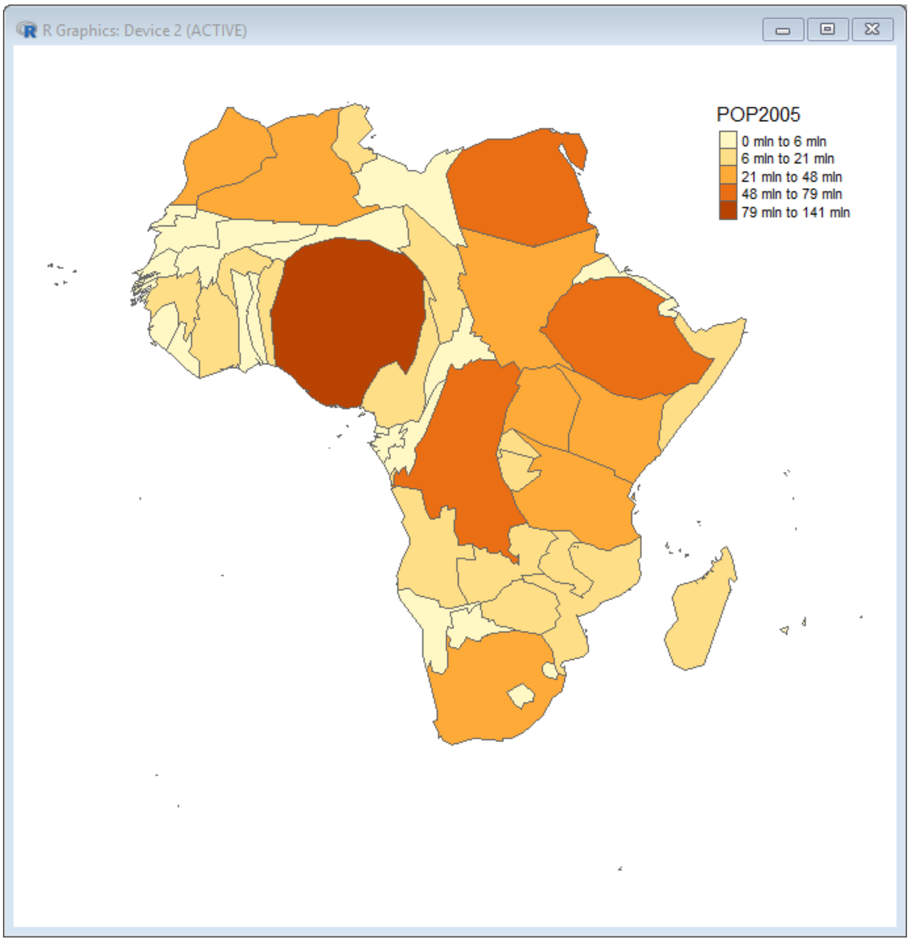


```
> #Obtenir le fichier de formes d'afrique
> library("maptools")
> data(wrld_simpl)
> #on construit un object de type SpatialPolygonsDataFrame
> afr<-wrld_simpl[wrld_simpl$REGION==2,]
>
>
> #On utilise maintenant le package cartogramme
> library("cartogram")
> library("tmap")
> #On constuit un objet cartogramme pour l'année 2005
> afr_cartogram <- cartogram(afr, "POP2005", itermax=5)

Please use cartogram_cont() instead of cartogram().

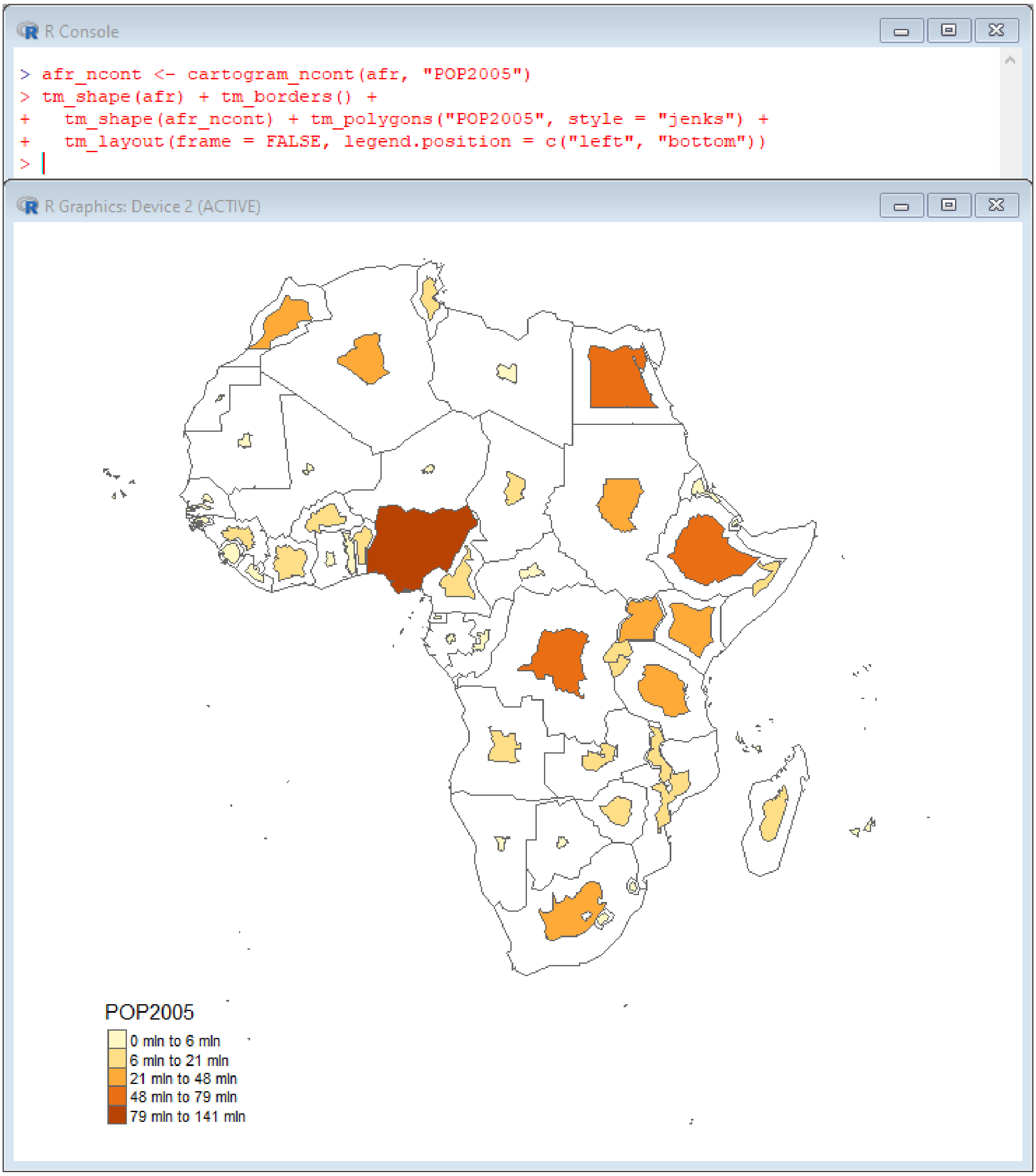
Mean size error for iteration 1: 5.6404768550094
Mean size error for iteration 2: 4.87548341597562
Mean size error for iteration 3: 4.2957559742443
Mean size error for iteration 4: 3.84621505103989
Mean size error for iteration 5: 3.49478200682353
There were 12 warnings (use warnings() to see them)
>
> #et on visualise le cartogramme classique correspondant
> tm_shape(afr_cartogram ) + tm_polygons("POP2005", style = "jenks") +
+   tm_layout(frame = FALSE)
>
```

La difficulté majeure est en réalité de créer ou modifier les objets de type SpatialPolyongDataFrame. Le script ci-dessous donnera:



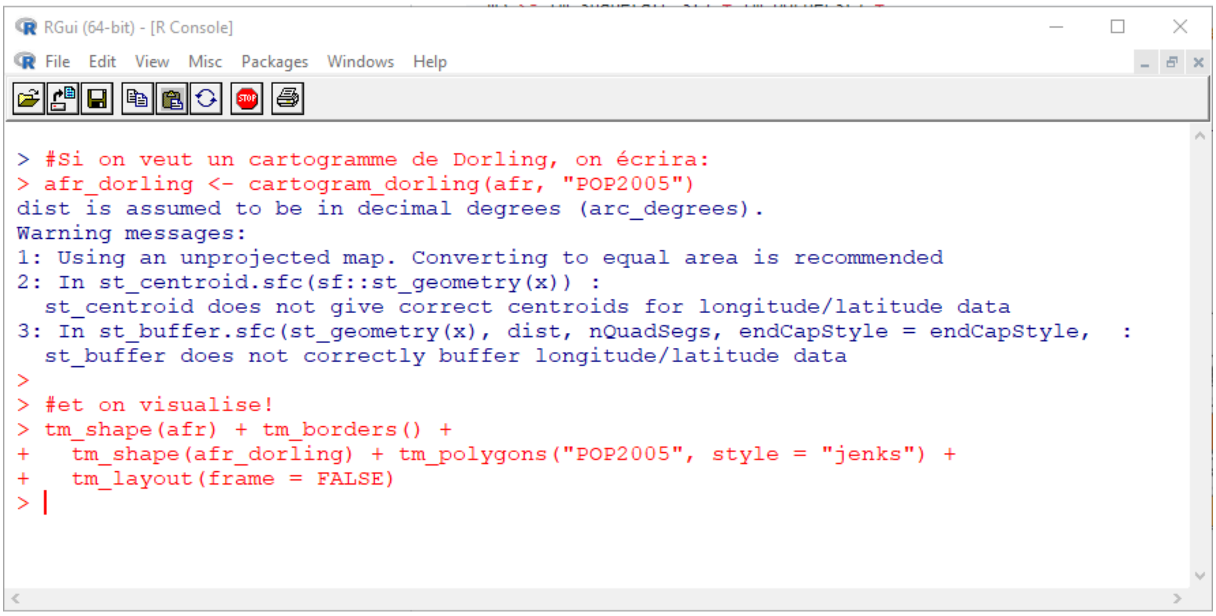
Cartogrammes non-contigus

Alors voici un type de Cartogramme que je n'ai jamais vu de ma vie être utilisé par mes clients:

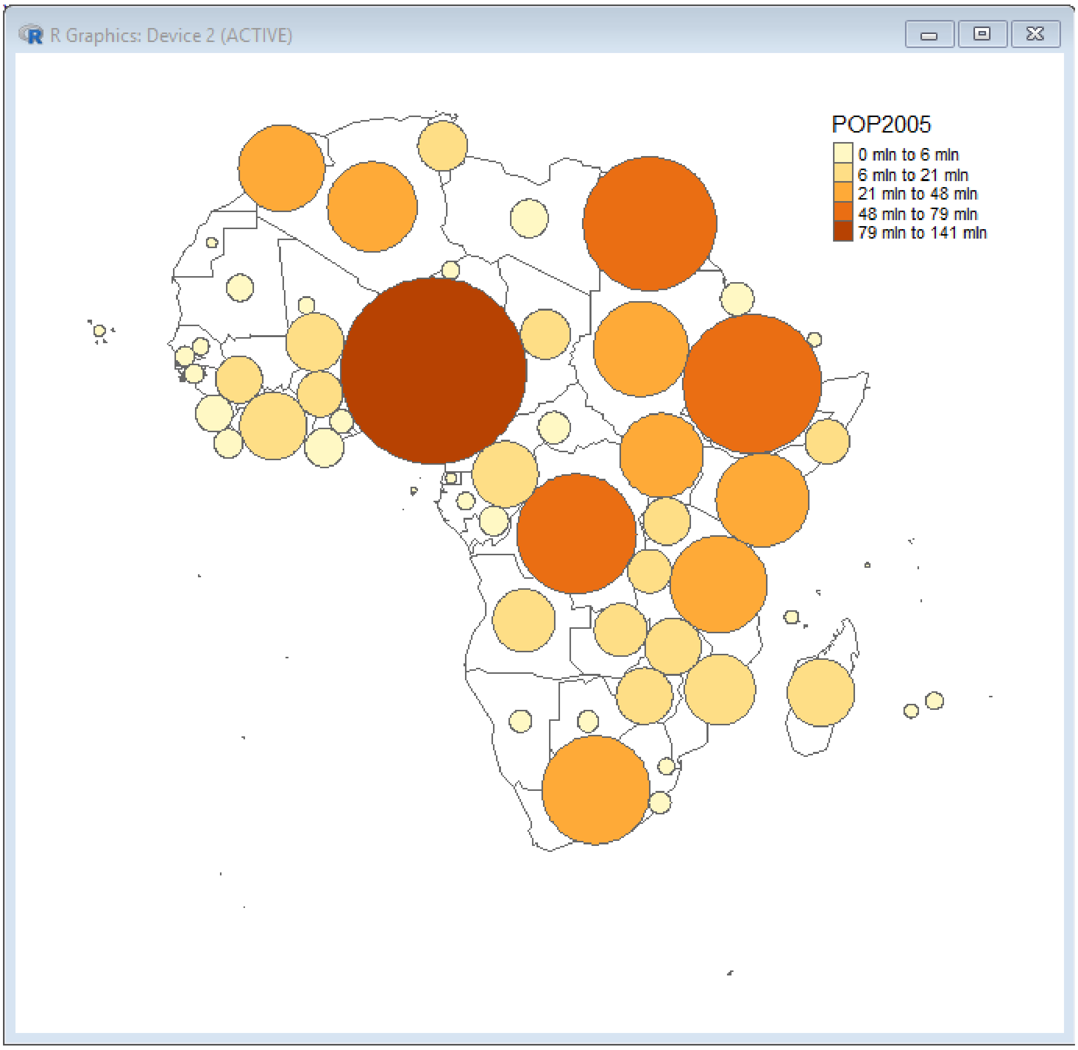


Cartogrammes de Dorling

On peut aussi faire un cartogramme de Dorling avec la fonction **cartogram_dorling( )**:



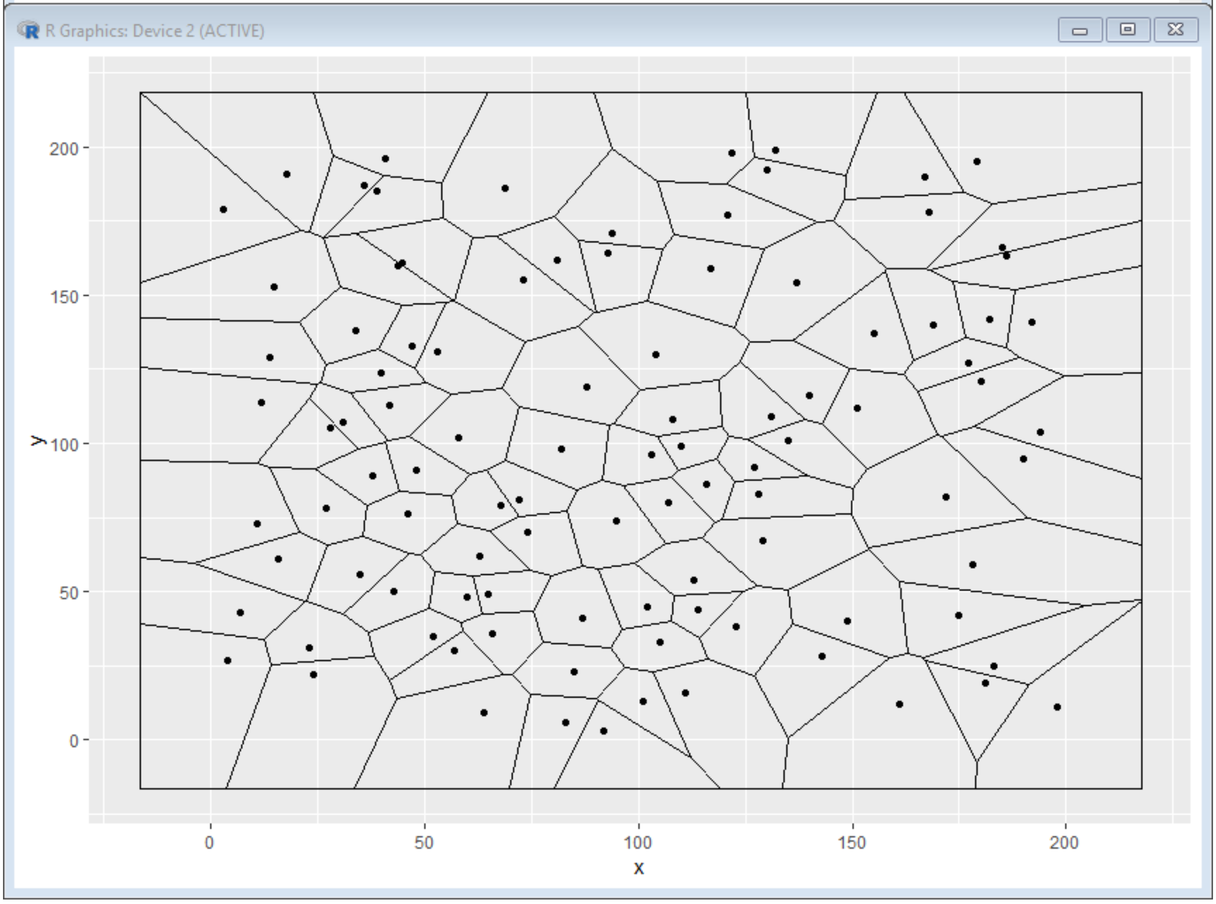
Ce qui donnera:



Diagrammes de Voronoï

Nous allons voici ici un diagramme sorti du contexte des SIG (car il ne s'applique pas seulement à ce domaine!). Perso je n'en ai jamais eu besoin à titre professionnel mais un contact sur LinkedIn me l'a demandé. Donc voici comment faire un diagramme de Voronoì avec le package **ggvoronoi** et son paramètre **stat_voronoi( )**:

```
> library(ggvoronoi)
> set.seed(0)
> x <- sample(1:200,100)
> y <- sample(1:200,100)
> points <- data.frame(x, y,
+                      distance = sqrt((x-100)^2 + (y-100)^2))
> circle <- data.frame(x = 100*(1+cos(seq(0, 2*pi, length.out = 2500))),
+                      y = 100*(1+sin(seq(0, 2*pi, length.out = 2500))),
+                      group = rep(1,2500))
>
>
> ggplot(points,aes(x,y))+stat_voronoi(geom="path")+geom_point()
> 
```



Ceux qui veulent en savoir plus pourront se rendre sur la vignette du package:

https://cran.r-project.org/web/packages/ggvoronoi/vignettes/ggvoronoi.html

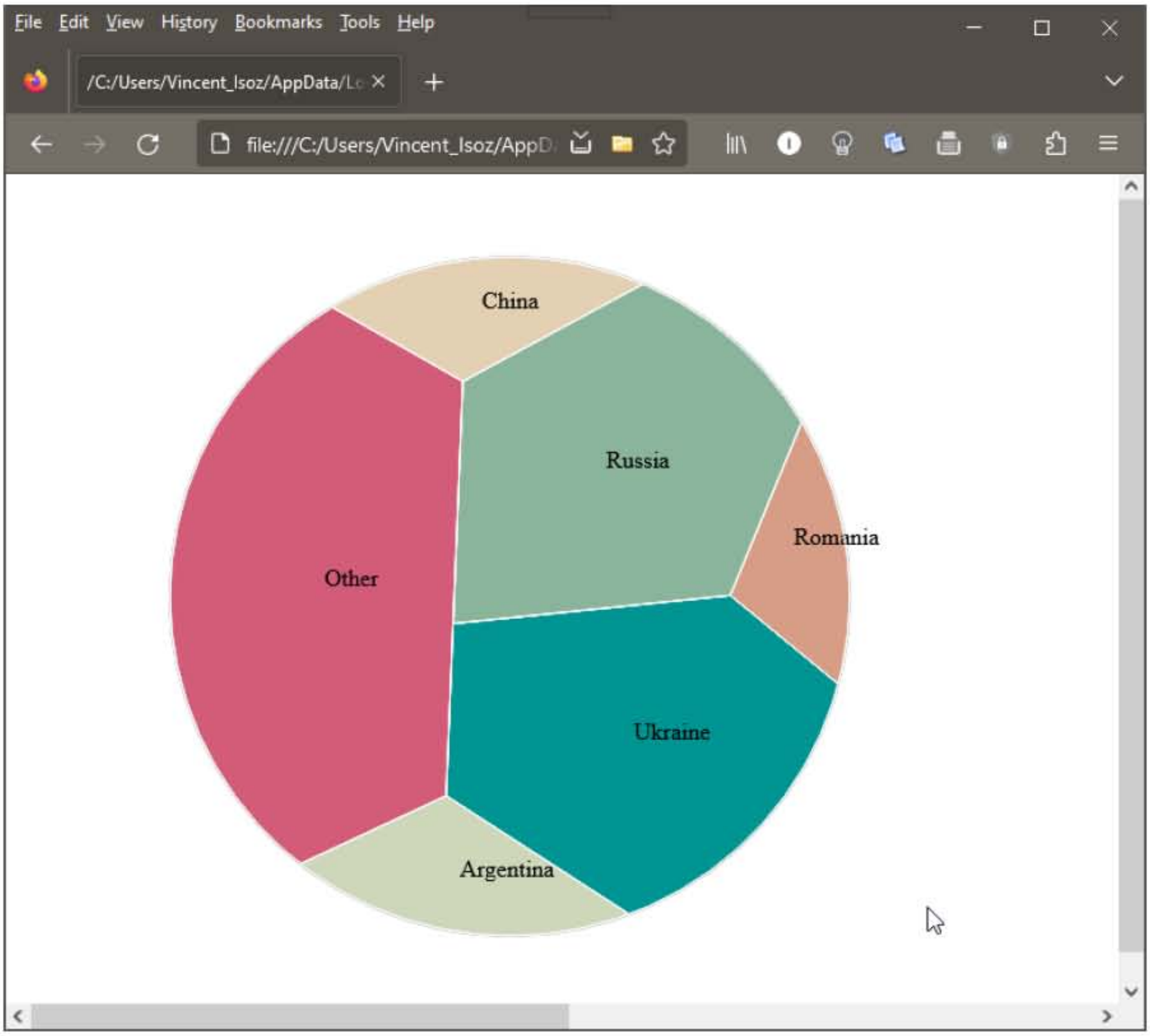
Sinon pour ceux qui souhaitent faire des diagrammes de Voronoi circulaires, nous avec le package **voronoiTreemap** :

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> library("voronoiTreemap")
> df <- data.frame(country = c("Ukraine", "Russia", "Argentina", "China","Romania", "Other"),
+                  prod = c(11.0, 10.6, 3.1, 2.4, 2.1, 15.3))
> (vor <- data.frame(h1 = 'World',
+                   h2 = c('Europe', 'Europe', 'Americas', 'Asia',
+                          'Europe', 'Other'),
+                   h3 = df$country,
+                   color = hcl.colors(nrow(df), palette = 'TealRose'),
+                   weight = df$prod,
+                   codes = df$country))
     h1       h2        h3   color weight     codes
1 World   Europe   Ukraine #009593   11.0   Ukraine
2 World   Europe    Russia #8AB49B   10.6    Russia
3 World Americas Argentina #CED7BA    3.1 Argentina
4 World     Asia     China #E4D1B3    2.4     China
5 World   Europe   Romania #D79D86    2.1   Romania
6 World    Other     Other #D35C79   15.3     Other
> (vt <- vt_input_from_df(vor))
          levelName
1  World
2   ¦--Europe
3   ¦   ¦--Ukraine
4   ¦   ¦--Russia
5   ¦   °--Romania
6   ¦--Americas
7   ¦   °--Argentina
8   ¦--Asia
9   ¦   °--China
10  °--Other
11      °--Other
> vt_d3(vt_export_json(vt))
> 
```

qui donnera:


File Edit View History Bookmarks Tools Help
/C:/Users/Vincent_Isoz/AppData/L
file:///C:/Users/Vincent_Isoz/AppD
China
Russia
Romania
Other
Ukraine
Argentina

Chemins (analyse qualitative de flux)

Le but de cet exemple est de montrer comment il est possible de représenter élégamment les trajectoires (chemins) parcourus par un ensemble d'entités mobiles (migrations, transports, analyse de flux) pendant une période de temps donnée, dans une région donnée.

Nous avons les données suivantes de lieu de domicile et de travail d'individus incluant différentes colonnes avec les moyens de transport (uniquement les trois premières colonnes vont nous intéresser):

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Area of usual residence	Area of workplace	All categories: Method of travel	Work mainly at or from home	Underground, metro, light rail, tram	Train	Bus, minibus or coach	Taxi	Other method of travel to work	F10	F11	F12	F13	F14
2	E02000001	E02000001	1506	0	73	41	32	9	1	8	1	33	1304	4
3	E02000001	E02000014	2	0	2	0	0	0	0	0	0	0	0	0
4	E02000001	E02000016	3	0	1	0	2	0	0	0	0	0	0	0
5	E02000001	E02000025	1	0	0	1	0	0	0	0	0	0	0	0
6	E02000001	E02000028	1	0	0	0	0	0	0	1	0	0	0	0
7	E02000001	E02000051	1	0	1	0	0	0	0	0	0	0	0	0
8	E02000001	E02000053	2	0	2	0	0	0	0	0	0	0	0	0
9	E02000001	E02000057	1	0	1	0	0	0	0	0	0	0	0	0
10	E02000001	E02000058	1	0	0	0	0	0	0	0	0	0	1	0
11	E02000001	E02000059	1	0	0	0	0	1	0	0	0	0	0	0
12	E02000001	E02000061	2	0	0	2	0	0	0	0	0	0	0	0
13	E02000001	E02000078	1	0	0	1	0	0	0	0	0	0	0	0
14	E02000001	E02000094	1	0	0	0	0	0	0	0	0	0	1	0
15	E02000001	E02000100	2	0	2	0	0	0	0	0	0	0	0	0
16	E02000001	E02000104	1	0	1	0	0	0	0	0	0	0	0	0
17	E02000001	E02000114	1	0	1	0	0	0	0	0	0	0	0	0
18	E02000001	E02000119	6	0	6	0	0	0	0	0	0	0	0	0
19	E02000001	E02000121	1	0	1	0	0	0	0	0	0	0	0	0
20	E02000001	E02000124	1	0	0	0	0	0	0	0	0	1	0	0

Nous chargeons les packages nécessaires (déjà connus) et les données d'intérêt (3 premières colonnes):

R Console

```
> library(plyr)
> library(ggplot2)
> library(maptools)
> input<-read.table("C:/tmp/WorkingPath.csv", sep=",", header=T)
>
> #Nous avons seulement besoin des 3 premières colonnes
> #Déplacements des gens allant au travail
> input<- input[,1:3]
> names(input)<- c("origin", "destination","total")
>
```

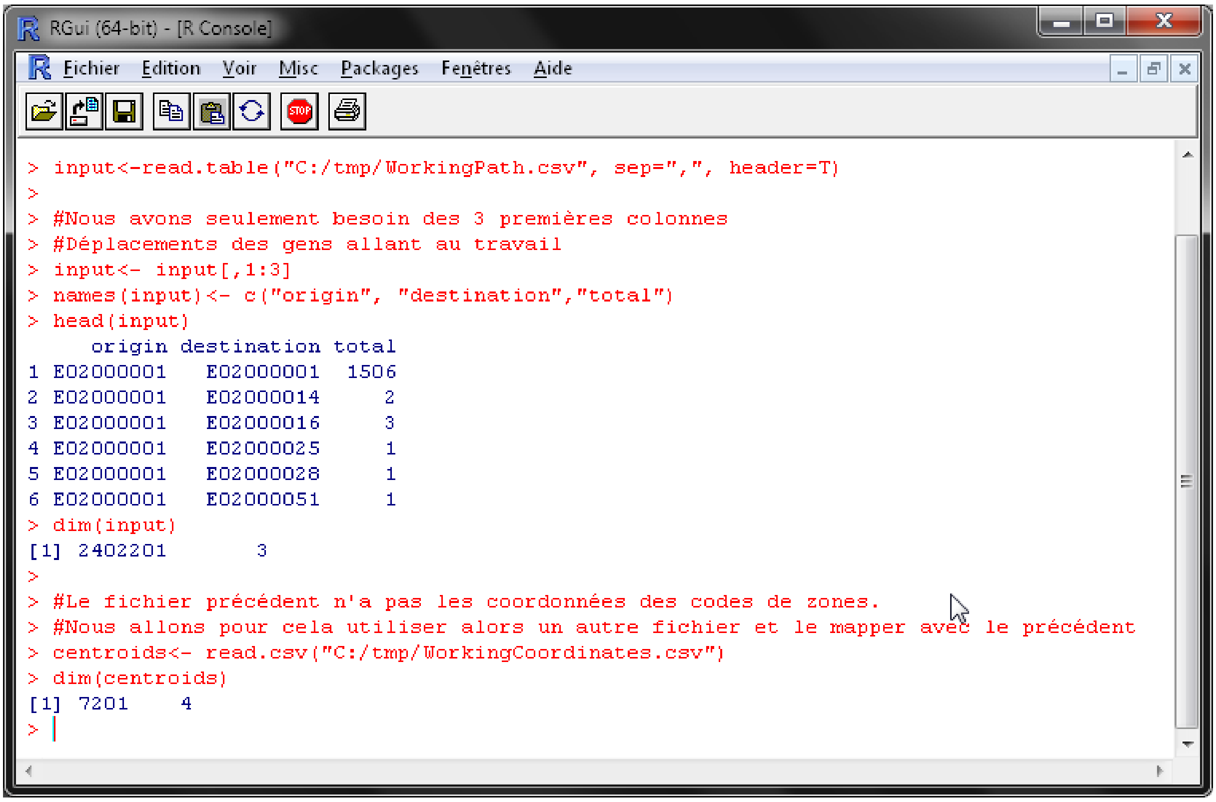
Un petit aperçu du dataset:

```
R Console
> head(input)
     origin destination total
1 E02000001   E02000001  1506
2 E02000001   E02000014     2
3 E02000001   E02000016     3
4 E02000001   E02000025     1
5 E02000001   E02000028     1
6 E02000001   E02000051     1
>
```

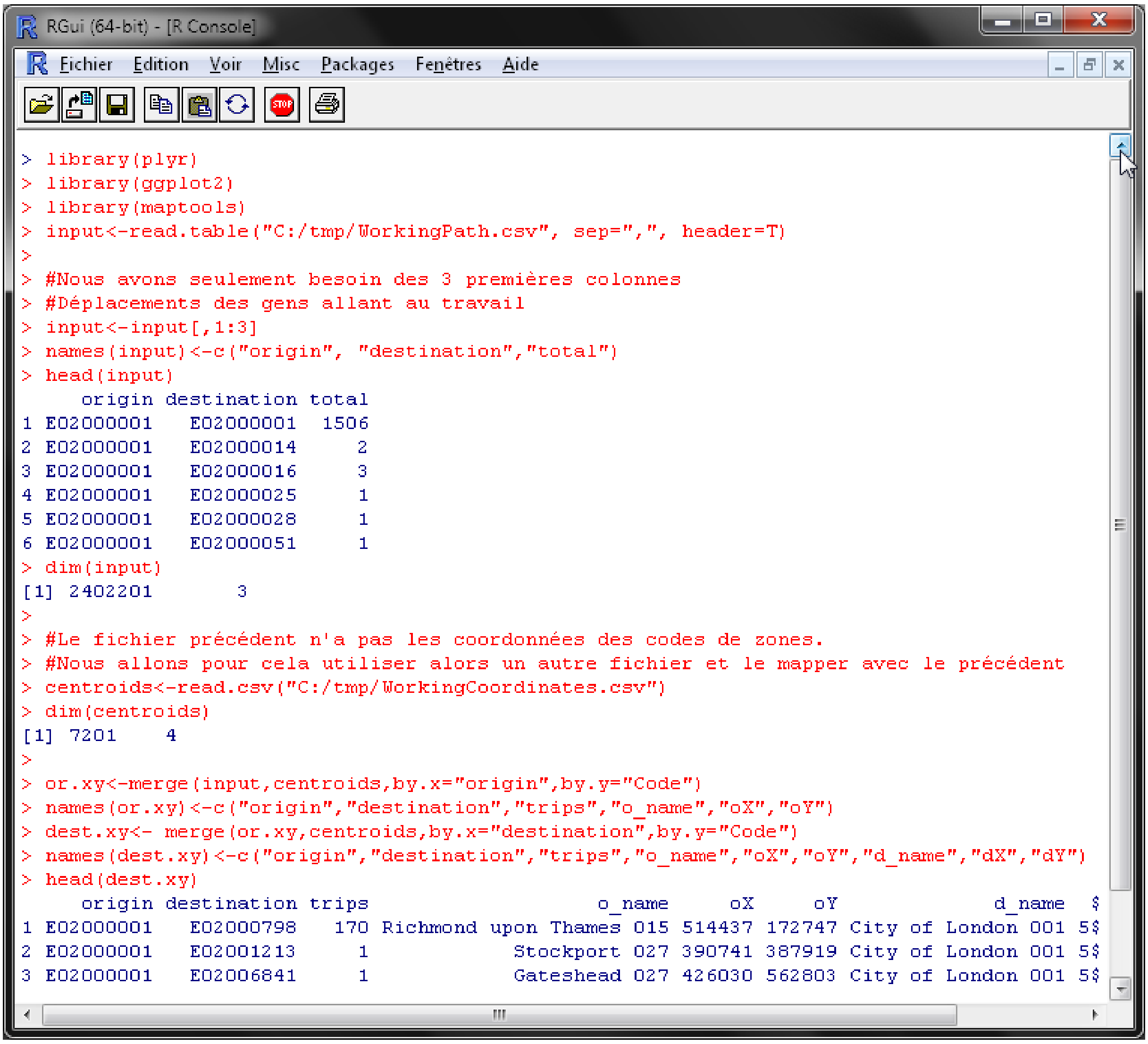
Le problème c'est que ce dernier fichier ne contient pas les coordonnées des villes/communes correspondant aux codes de zones. Pour cela nous allons utiliser le fichier suivant:

	A	B	C	D
1	Code	Name	East	North
2	E02000001	City of London 001	532269	181787
3	E02000002	Barking and Dagenham 001	547971	189394
4	E02000003	Barking and Dagenham 002	548374	188095
5	E02000004	Barking and Dagenham 003	551031	186870
6	E02000005	Barking and Dagenham 004	548671	186800
7	E02000007	Barking and Dagenham 006	549499	186298
8	E02000008	Barking and Dagenham 007	548115	185919
9	E02000009	Barking and Dagenham 008	547019	185763
10	E02000010	Barking and Dagenham 009	549156	185344
11	E02000011	Barking and Dagenham 010	550254	185071
12	E02000012	Barking and Dagenham 011	545702	184878
13	E02000013	Barking and Dagenham 012	547208	184746
14	E02000014	Barking and Dagenham 013	548913	184471
15	E02000015	Barking and Dagenham 014	549930	184481
16	E02000016	Barking and Dagenham 015	543991	184406
17	E02000017	Barking and Dagenham 016	544812	184395
18	E02000018	Barking and Dagenham 017	546754	184222
19	E02000019	Barking and Dagenham 018	547990	184035
20	E02000020	Barking and Dagenham 019	545363	183615
21	E02000021	Barking and Dagenham 020	549589	183619

et nous la chargeons aussi dans un dateset. Le tout donne jusqu'à maintenant:



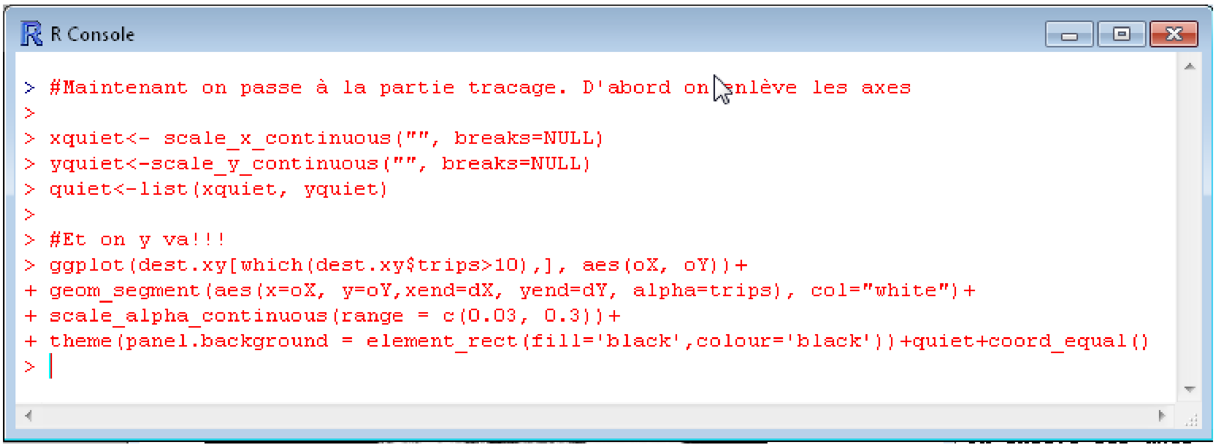
Ensuite, nous faisons un mappage classique avec la fonction **merge( )** déjà connue ce qui donne alors:



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> library(plyr)
> library(ggplot2)
> library(maptools)
> input<-read.table("C:/tmp/WorkingPath.csv", sep=",", header=T)
>
> #Nous avons seulement besoin des 3 premières colonnes
> #Déplacements des gens allant au travail
> input<-input[,1:3]
> names(input)<-c("origin", "destination","total")
> head(input)
     origin destination total
1 E02000001   E02000001  1506
2 E02000001   E02000014     2
3 E02000001   E02000016     3
4 E02000001   E02000025     1
5 E02000001   E02000028     1
6 E02000001   E02000051     1
> dim(input)
[1] 2402201       3
>
> #Le fichier précédent n'a pas les coordonnées des codes de zones.
> #Nous allons pour cela utiliser alors un autre fichier et le mapper avec le précédent
> centroids<-read.csv("C:/tmp/WorkingCoordinates.csv")
> dim(centroids)
[1] 7201    4
>
> or.xy<-merge(input,centroids,by.x="origin",by.y="Code")
> names(or.xy)<-c("origin","destination","trips","o_name","oX","oY")
> dest.xy<- merge(or.xy,centroids,by.x="destination",by.y="Code")
> names(dest.xy)<-c("origin","destination","trips","o_name","oX","oY","d_name","dX","dY")
> head(dest.xy)
     origin destination trips                  o_name     oX     oY             d_name  $
1 E02000001   E02000798   170 Richmond upon Thames 015 514437 172747 City of London 001 5$
2 E02000001   E02001213     1           Stockport 027 390741 387919 City of London 001 5$
3 E02000001   E02006841     1           Gateshead 027 426030 562803 City of London 001 5$
```

Et maintenant nous passons à la partie traçage:



```
R Console

> #Maintenant on passe à la partie tracage. D'abord on enlève les axes
>
> xquiet<- scale_x_continuous("", breaks=NULL)
> yquiet<-scale_y_continuous("", breaks=NULL)
> quiet<-list(xquiet, yquiet)
>
> #Et on y va!!!
> ggplot(dest.xy[which(dest.xy$trips>10),], aes(oX, oY))+
+ geom_segment(aes(x=oX, y=oY,xend=dX, yend=dY, alpha=trips), col="white")+
+ scale_alpha_continuous(range = c(0.03, 0.3))+
+ theme(panel.background = element_rect(fill='black',colour='black'))+quiet+coord_equal()
> |
```

Ce qui donne visuellement (après une attente certaine dépendant de la puisse de l'ordinateur car il y a quand même près de 2 millions de chemins/traits à dessiner).

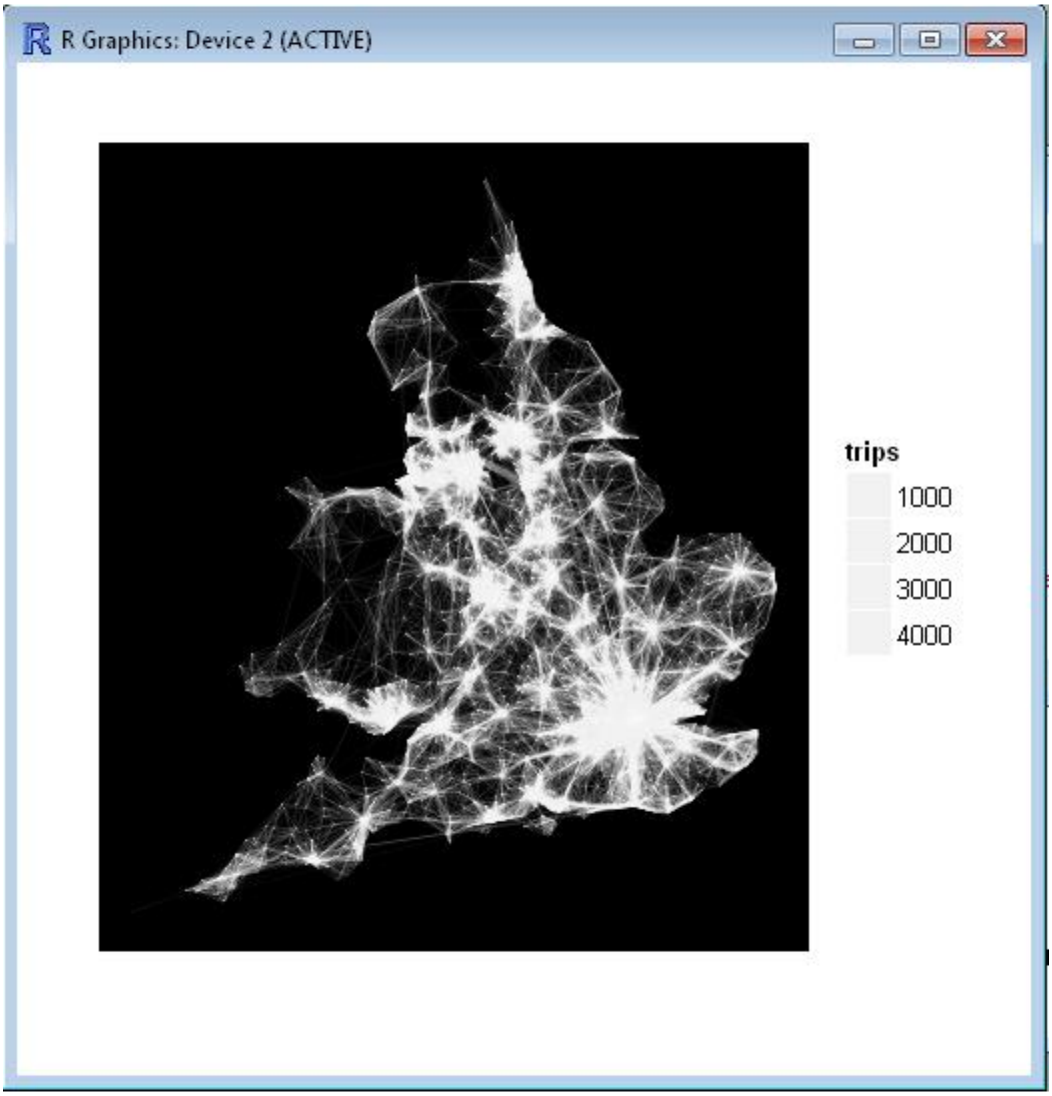


Figure 203 Graph de flux (trajectoires)

Reste plus qu'à exporter en PDF ou SVG pour avoir une image HD.

Trajet optimal (des moindres distances)

Le package **TSP** contient une commande **solve_TSP( )** qui à partir d'une matrice des distances permet de déterminer le trajet le plus court en distances entre de multiples points géographiques.

Rappelons d'abord comment construire une matrice de distance et montrons ensuite comment en faire un objet TSP avec la commande **TSP( )** et en retirer quelques informations

```
> lat<-c(27.685,29.343,23.495)
> lon<-c(46.344,46.233,46.23)
> coord<-data.frame(lat,lon)
> row.names(coord)<-c("Genève","Berne","Lausanne")
> coord
            lat    lon
Genève   27.685 46.344
Berne    29.343 46.233
Lausanne 23.495 46.230
> #on calcule les distances euclidiennes entre chaque objet
> dist(coord)
           Genève    Berne
Berne    1.661711
Lausanne 4.191551 5.848001
> #on crée un objet TSP pour le package TSP
> library(TSP)
> tspDist<-TSP(dist(coord))
> tspDist
object of class 'TSP'
3 cities (distance 'euclidean')
> n_of_cities(tspDist)
[1] 3
> labels(tspDist)
[1] "Genève"   "Berne"    "Lausanne"
> 
```

Pour la suite nous utiliserons une matrice des distances inclue dans le package **TSP** qui contient quelques informations de distances euclidiennes entre des grandes villes américaines.

Voyons donc ce que nous pouvons en tirer:

```
R Console
> library(TSP)
> library(rgeos)
> library(maps)
> library(sp)
> library(maptools)
>
> data("USCA312")
> data("USCA312_map")
>
> #calculate a tour
> tour <- solve_TSP(USCA312)
>
> #plot map
> plot(as(USCA312_coords,"Spatial"),axes=TRUE)
> plot(USCA312_basemap,add=TRUE,col="gray")
>
>
> #plot tour and add cities
> tour_line <- SpatialLines(list(Lines(list(
+     Line(USCA312_coords[c(tour, tour[1]),])),ID="1")))
>
> plot(tour_line,add=TRUE,col="red")
> points(USCA312_coords,pch=3,cex=0.4,col="black")
>
```

Ce qui donne:

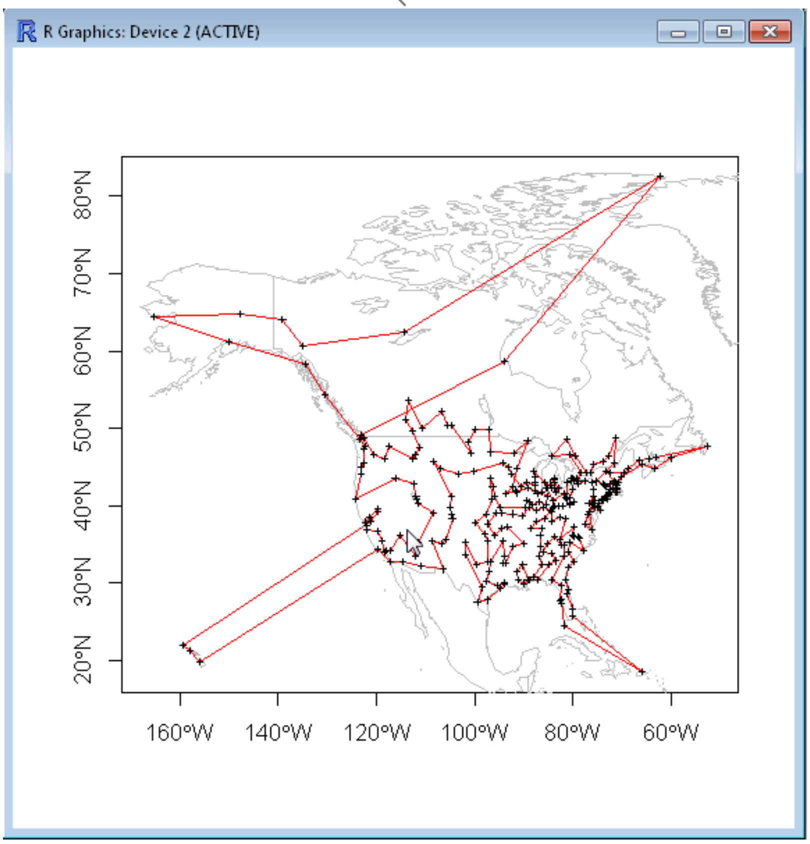


Figure 204 Graph de trajet optimal

Exemples 155.: Calendriers de densité

R 3.0.2

Voyons une petite représentative sympathique que certains apprécieront (à nouveau on peu faire beaucoup mieux avec un tableur mais bon...).

Considérons les données suivantes d'occupation des salles de conférences d'un petit centre de centre de formation:

```
Dates;Trainees
2008-01-07;8
2008-01-08;8
2008-02-15;7
2008-02-26;5
2008-03-03;5
2008-03-07;5
2008-03-10;6
2008-03-11;5
2008-03-14;6
2008-05-09;4
2008-05-13;6
2008-05-15;7
2008-05-20;4
2008-05-23;4
2008-05-26;10
2008-05-29;6
2008-05-30;6
2008-06-03;6
2008-06-05;5
```

Ensuite, nous allons utiliser le long script de Paul Bleicher comme une boîte noire:

```
##############################################################################
 #                           Calendar Heatmap
 #                                  by
 #                             Paul Bleicher
 # an R version of a graphic from:
 # http://stat-computing.org/dataexpo/2009/posters/wicklin-allison.pdf
 #  requires lattice, chron, grid packages
 #############################################################################
require(lattice); require(chron); require(grid)

## calendarHeat: An R function to display time-series data as a calendar he
## Copyright 2009 Humedica. All rights reserved.

## This program is free software; you can redistribute it and/or modify
## it under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 2 of the License, or
## (at your option) any later version.

## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.

## You can find a copy of the GNU General Public License, Version 2 at:
## http://www.gnu.org/licenses/gpl-2.0.html

#' An R function to display time-series data as a calendar heatmap
#'
#' This graphic originally appeared \href{http://stat-computing.org/dataexp
#' This function is included with the \code{makeR} package to support the
#' R-Bloggers demo. See \code{demo('makeR')} for more information.
#'
```

Remarquez que ce script utilise les package **lattice**, **chron** et **grid**!!!

Ensuite, dans R nous aurons:

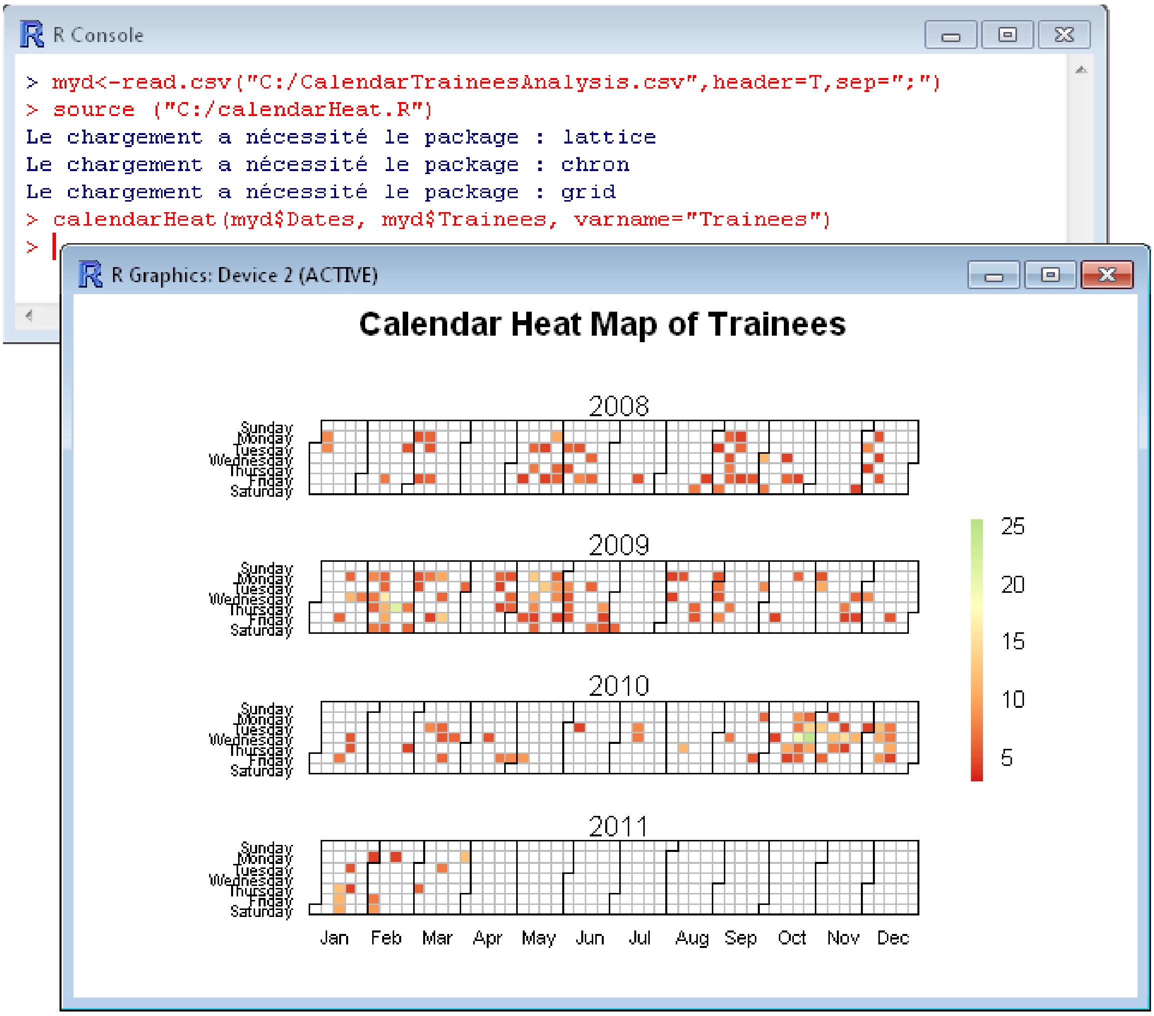
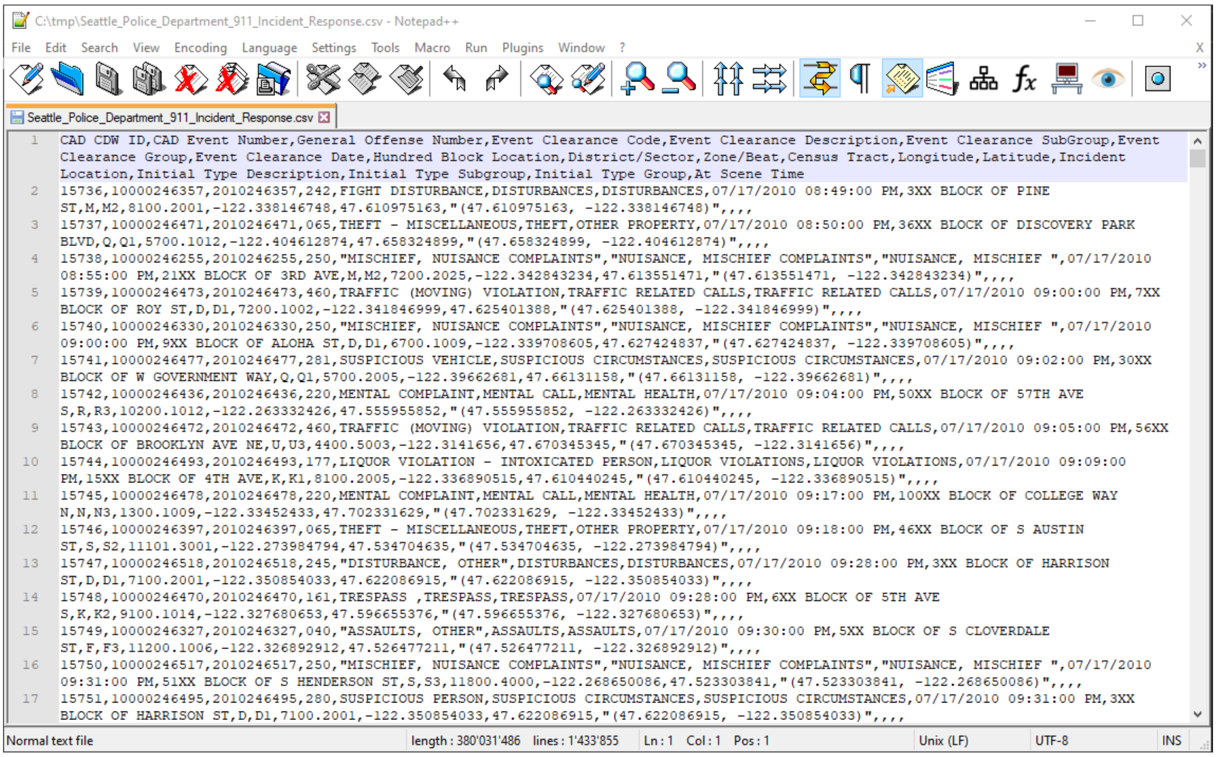
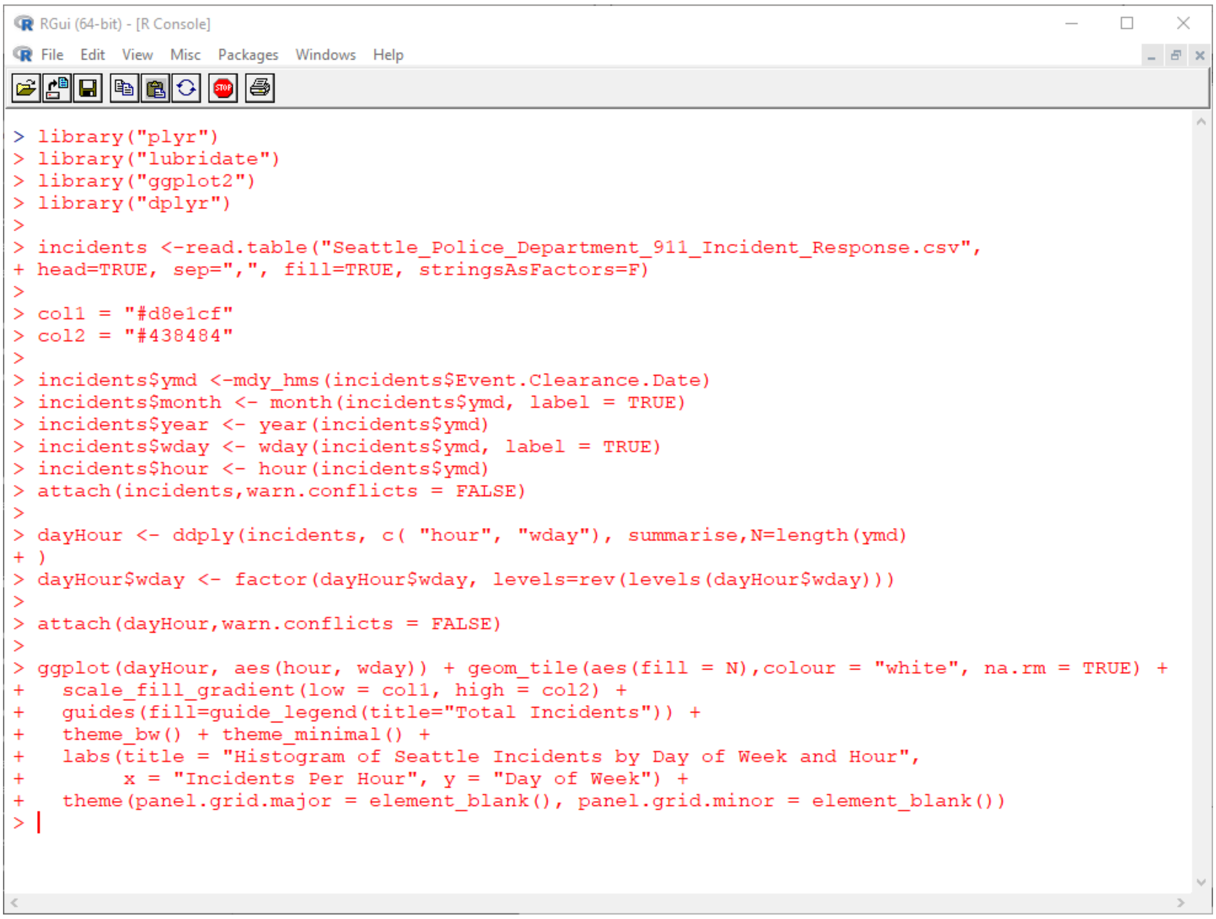


Figure 205 Graph de calendrier ave densité de couleurs d'événements

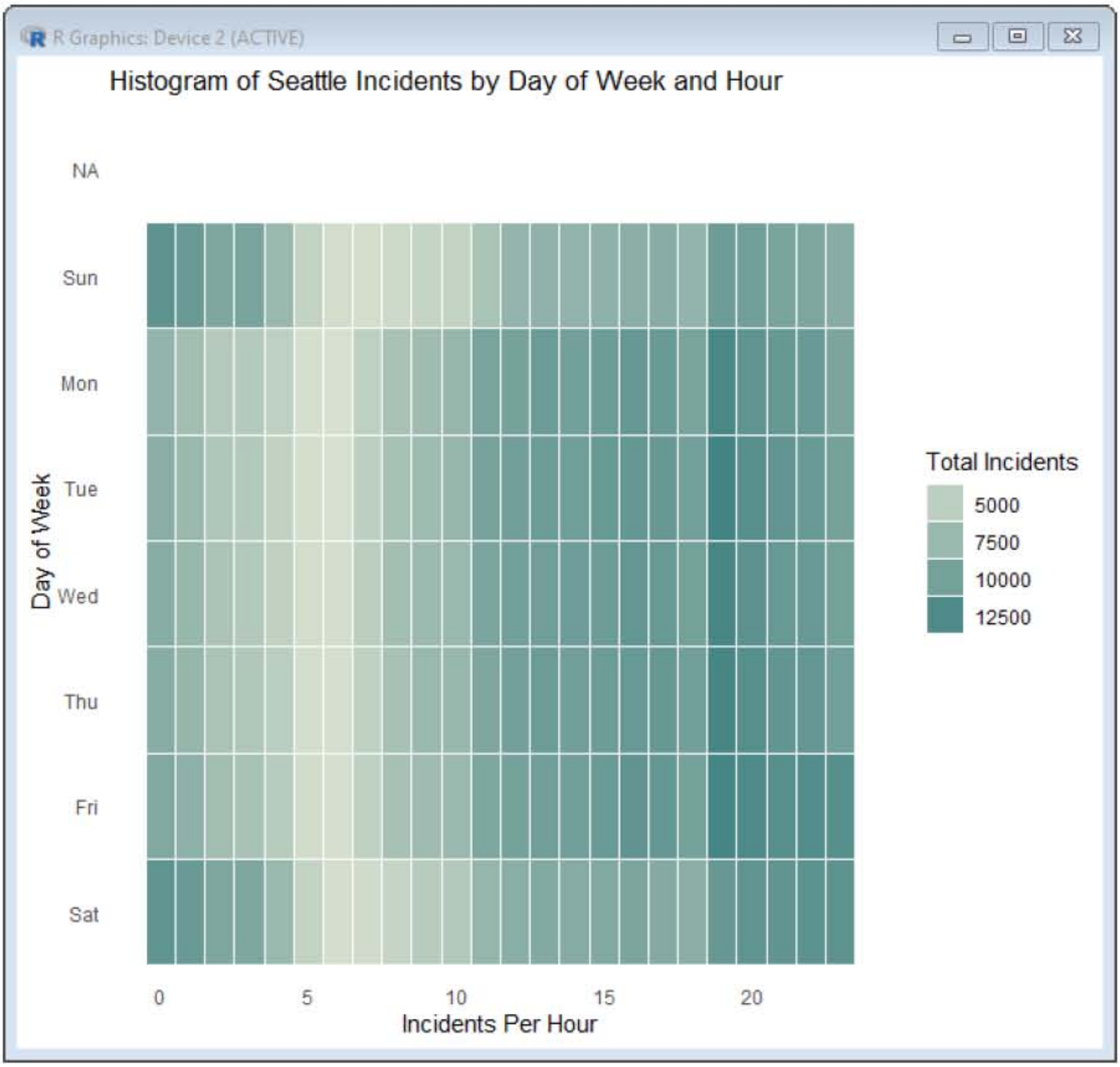
Dans la même genre très connu (surtout parce que c'est ainsi que Microsoft SharePoint affiche les statistiques de consultations des pages web), on a en utilisant le jeu de données *Seattle_Police_Department_911_Incident_Response.csv* de 371 MB et 1'400'000 lignes:



On utilise alors le code écrit par Laura Ellis:



Ce qui donne:

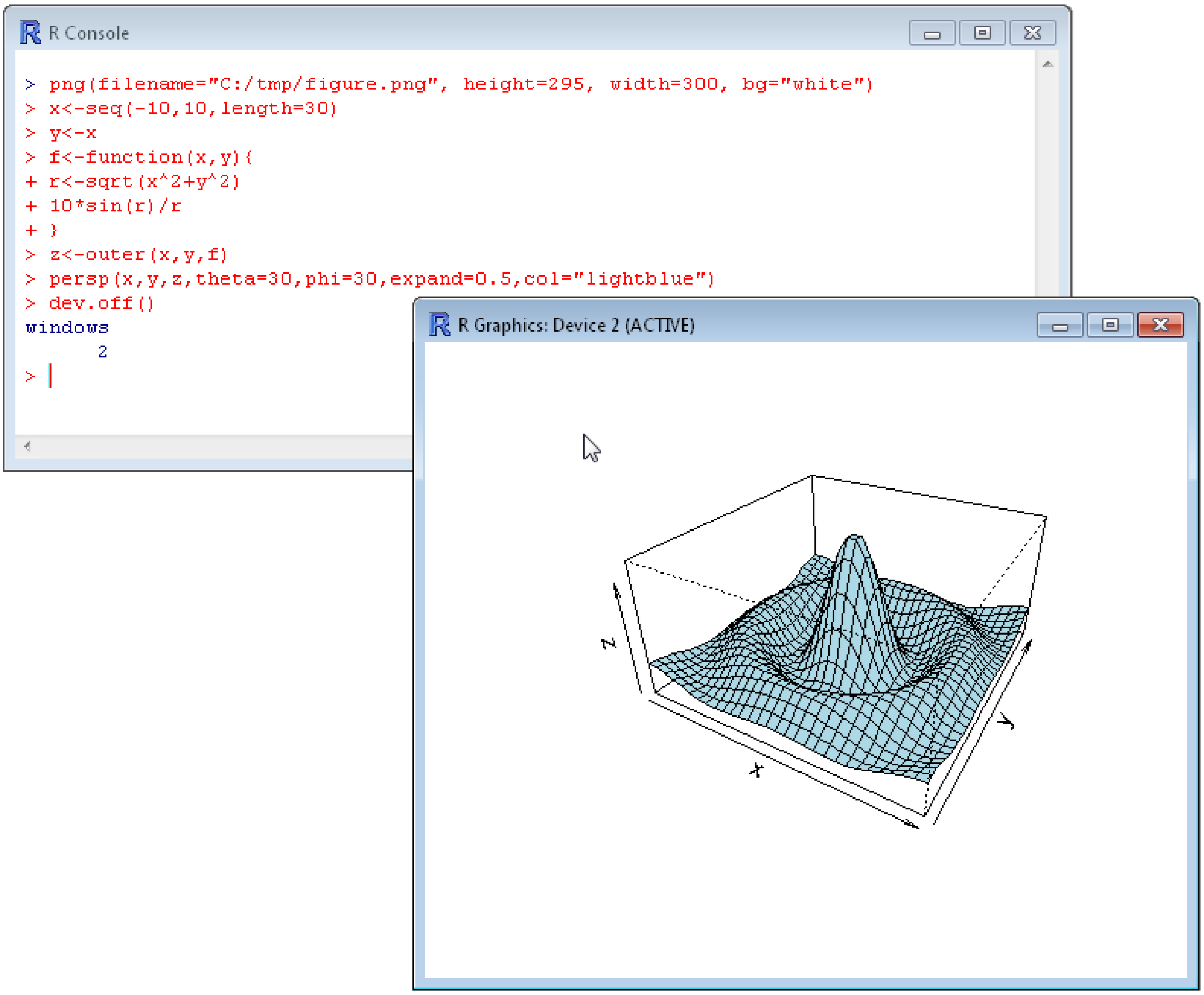

R Graphics: Device 2 (ACTIVE)
Histogram of Seattle Incidents by Day of Week and Hour
Day of Week
NA
Sun
Mon
Tue
Wed
Thu
Fri
Sat
0
5
10
15
20
Incidents Per Hour
Total Incidents
5000
7500
10000
12500

Exemples 156.: Exports de graphiques

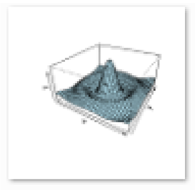
R 3.0.2

Exporter un graphique en tant qu'image ou pdf

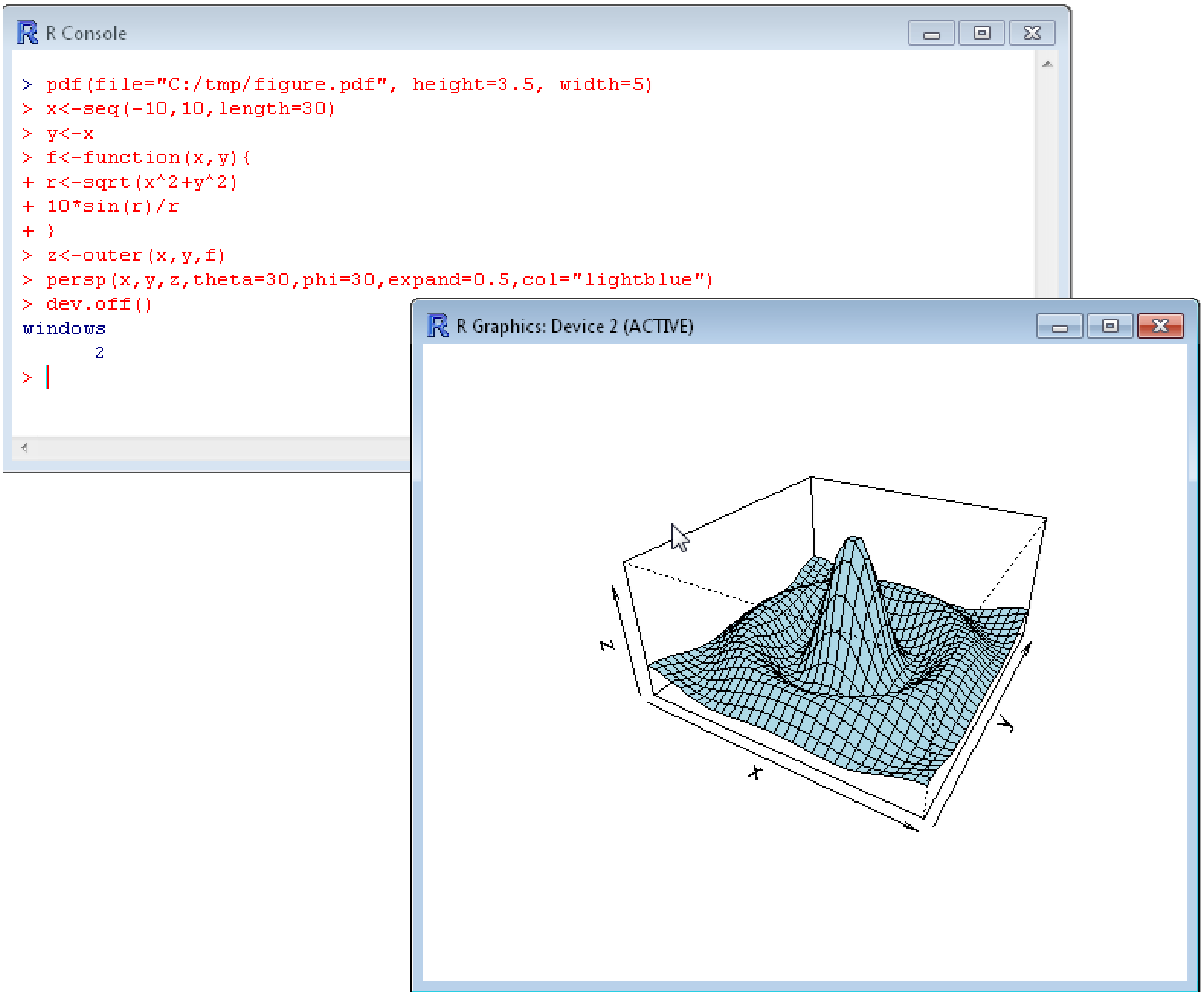
Lorsque vous faites des graphiques vous pouvez certes les exporter manuellement en faisant un clic droit et choisir une option d'export mais dans la majorité des cas, nous automatiserons de type de procédure dans des scripts. Voici un exemple (notez que l'export se **fait** avant le script du graphique et qu'il faut avoir la commande **dev.off( )** une fois terminé!!!!!):



Nous obtenons alors:



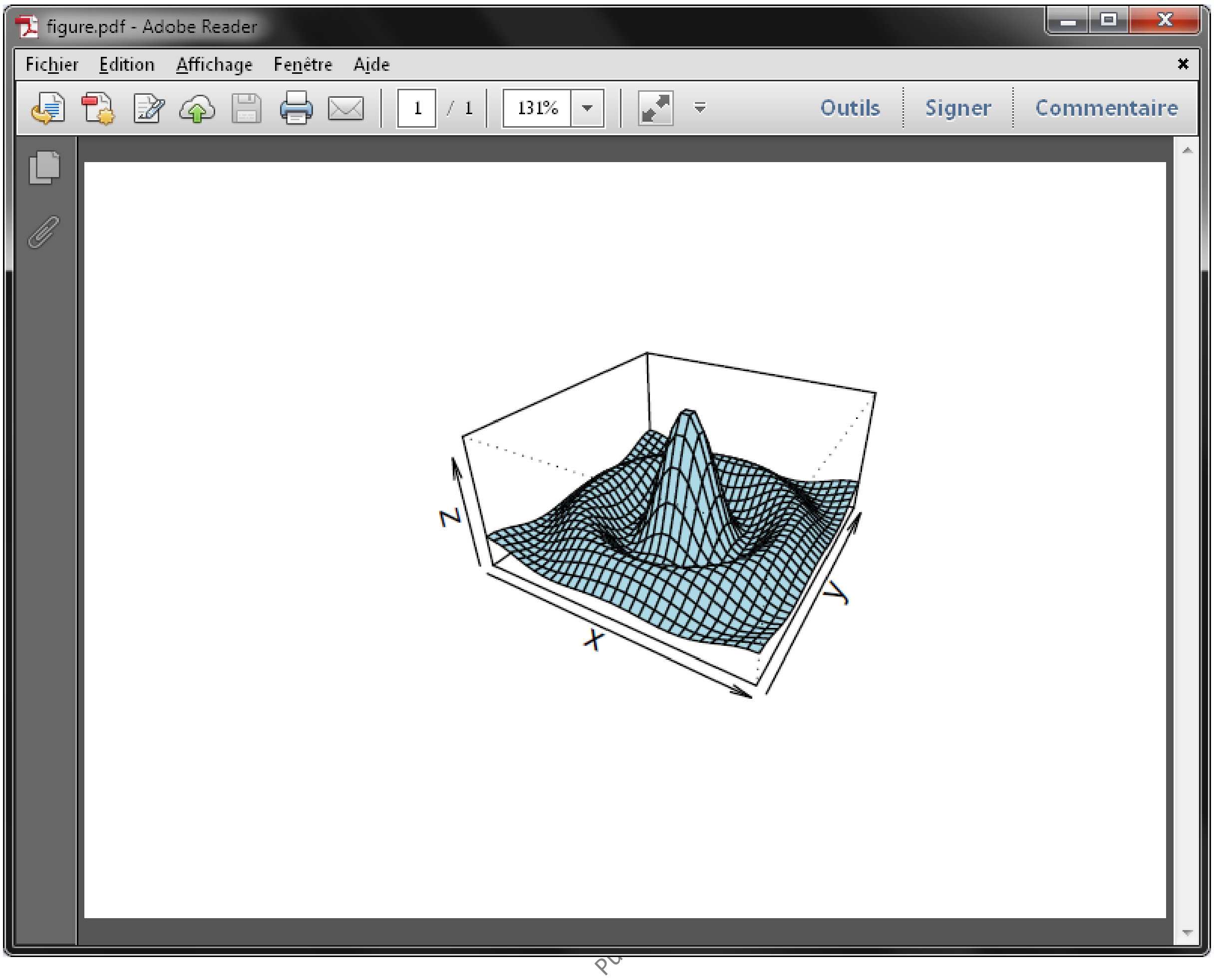
Et pour exporter en PDF (notez encore une fois que l'export se **fait** avant le script du graphique!!!!!):



Nous obtenons alors:



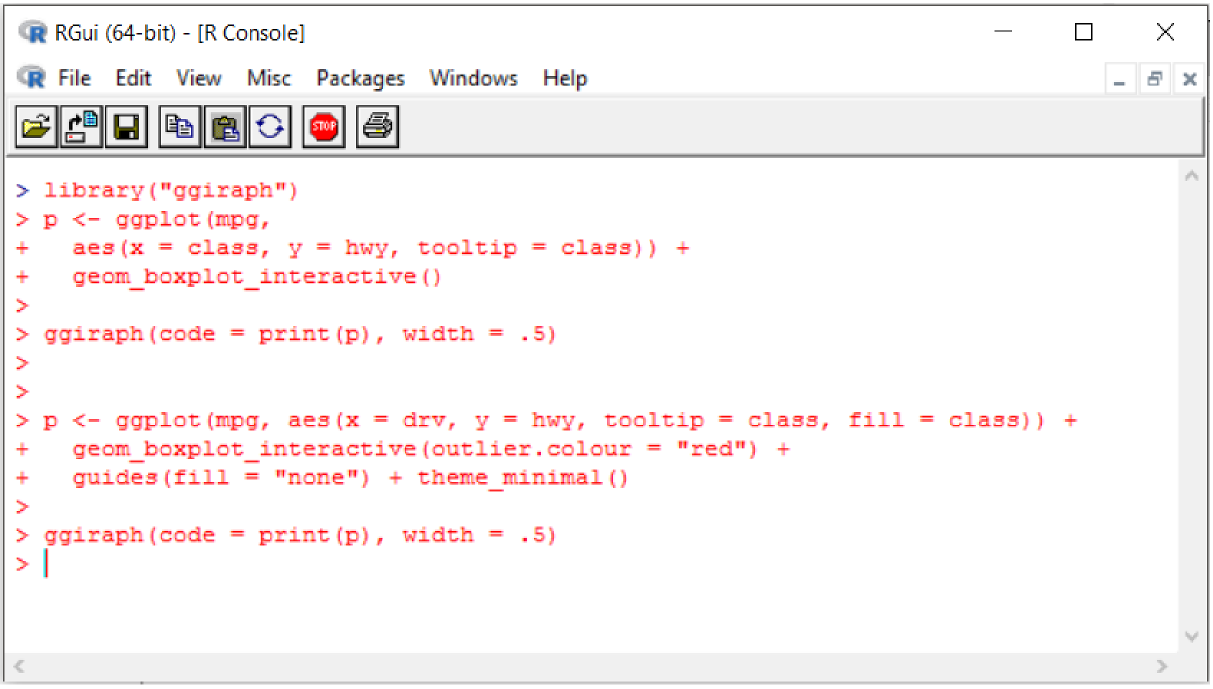
et si nous ouvrons:



L'export en PDF est d'une qualité visuelle remarquable! Sachant que l'on peut mettre plusieurs graphiques sur une même figure de sortie, ainsi que du texte et des résultats de calculs, on imagine aisément comme cela peut être utile pour automatiser du reporting en entreprise.

Export de graphiques ggplot en format web interactif

Avec le package **ggiraph** et sa commande correspondante il est possible d'obtenir des graphiques interactifs pour le web qui sont remarquables. Voyons les exemples fournis avec le package:



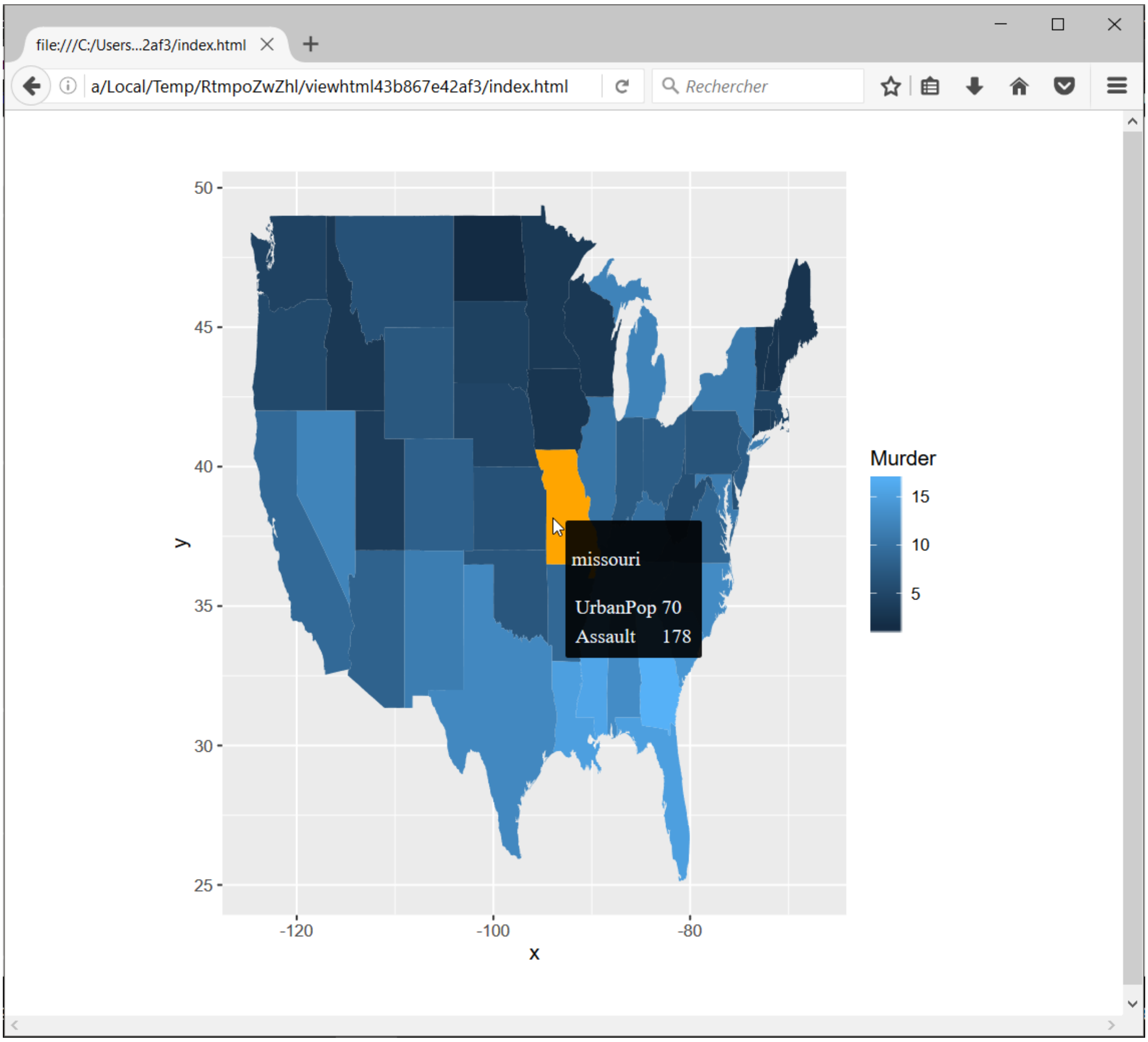
ce qui donne:



ou encore (dernier exemple):

```
> library("ggiraph")
> crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)
>
> # create tooltips and onclick events
> states_ <- sprintf("<p>%s</p>",
+                    as.character(crimes$state) )
> table_ <- paste0(
+   "<table><tr><td>UrbanPop</td>",
+   sprintf("<td>%.0f</td>", crimes$UrbanPop),
+   "</tr><tr>",
+   "<td>Assault</td>",
+   sprintf("<td>%.0f</td>", crimes$Assault),
+   "</tr></table>"
+ )
>
> onclick <- sprintf(
+   "window.open(\"%s%s\")",
+   "http://en.wikipedia.org/wiki/",
+   as.character(crimes$state)
+ )
>
>
> crimes$labs <- paste0(states_, table_)
> crimes$onclick = onclick
>
> if (require("maps") ) {
+   states_map <- map_data("state")
+   gg_map <- ggplot(crimes, aes(map_id = state))
+   gg_map <- gg_map + geom_map_interactive(aes(
+                   fill = Murder,
+                   tooltip = labs,
+                   data_id = state,
+                   onclick = onclick
+                 ),
+                 map = states_map) +
+     expand_limits(x = states_map$long, y = states_map$lat)
+   ggiraph(code = print(gg_map), width = .8)
+ }
>
```

Ce qui donne:



et ainsi de suite. Donc on voit que les possibilités sont considérables!

12. Probabilités et fonctions de distributions de probabilités

Il arrive fréquemment que l'on souhaite comparer des données mesurées dont nous faisons l'hypothèse qu'elle suive une loi de probabilité donnée, avec des vraies variables aléatoires issues de cette loi. Ne serait-ce que pour faire une représentation graphique de l'histogramme de chacun et de pouvoir les comparer.

Remarque: La génération de variables aléatoires est également utile dans le cadre de simulations de Monte-Carlo après avoir copier/coller ou exporter les données dans MS Excel.

Comme il s'agit d'une manipulation simple à effectuer dans R (afficher les histogrammes de différentes distributions), nous allons donc commencer par ce premier point.

D'abord signalons les conventions d'écriture de R pour les fonctions de distribution (***p*** pour la probabilité cumulée, ***q*** pour le quantile, ***d*** pour la densité et ***r*** pour réaliser une valeur aléatoire au hasard):

Distribution	Fonctions			
Beta	pbeta	qbeta	dbeta	rbeta
Binomial	pbinom	qbinom	dbinom	rbinom
Cauchy	pcauchy	qcauchy	dcauchy	rcauchy
Chi-Square	pchisq	qchisq	dchisq	rchisq
Exponential	pexp	qexp	dexp	rexp
F	pf	qf	df	rf
Gamma	pgamma	qgamma	dgamma	rgamma
Geometric	pgeom	qgeom	dgeom	rgeom
Hypergeometric	phyper	qhyper	dhyper	rhyper
Logistic	plogis	qlogis	dlogis	rlogis
Log Normal	plnorm	qlnorm	dlnorm	rlnorm
Negative Binomial	pnbinom	qnbinom	dnbinom	rnbinom
Multinomial	pmultinom	qmultinom	dmultinom	rmultinom
Normal	pnorm	qnorm	dnorm	rnorm
Poisson	ppois	qpois	dpois	rpois
Student t	pt	qt	dt	rt
Studentized Range	ptukey	qtukey	dtukey	rtukey
Uniform	punif	qunif	dunif	runif
Weibull	pweibull	qweibull	dweibull	rweibull
Wilcoxon Rank Sum Statistic	pwilcox	qwilcox	dwilcox	rwilcox
Wilcoxon Signed Rank Statistic	psignrank	qsignrank	dsignrank	rsignrank

Tableau 1 Syntaxe des différentes distributions de probabilités

Pour d'autres distribution et leur package respectif voir:

http://cran.r-project.org/web/views/Distributions.html

ou en particulier le package **extraDistr** que je recommande particulièrement!

Avant de commencer signalons au lecteur que pour chaque distribution il peut s'amuser à faire le graphique suivant très instructif (dans le cas ci-dessous nous l'avons fait qu'avec la loi Normale):

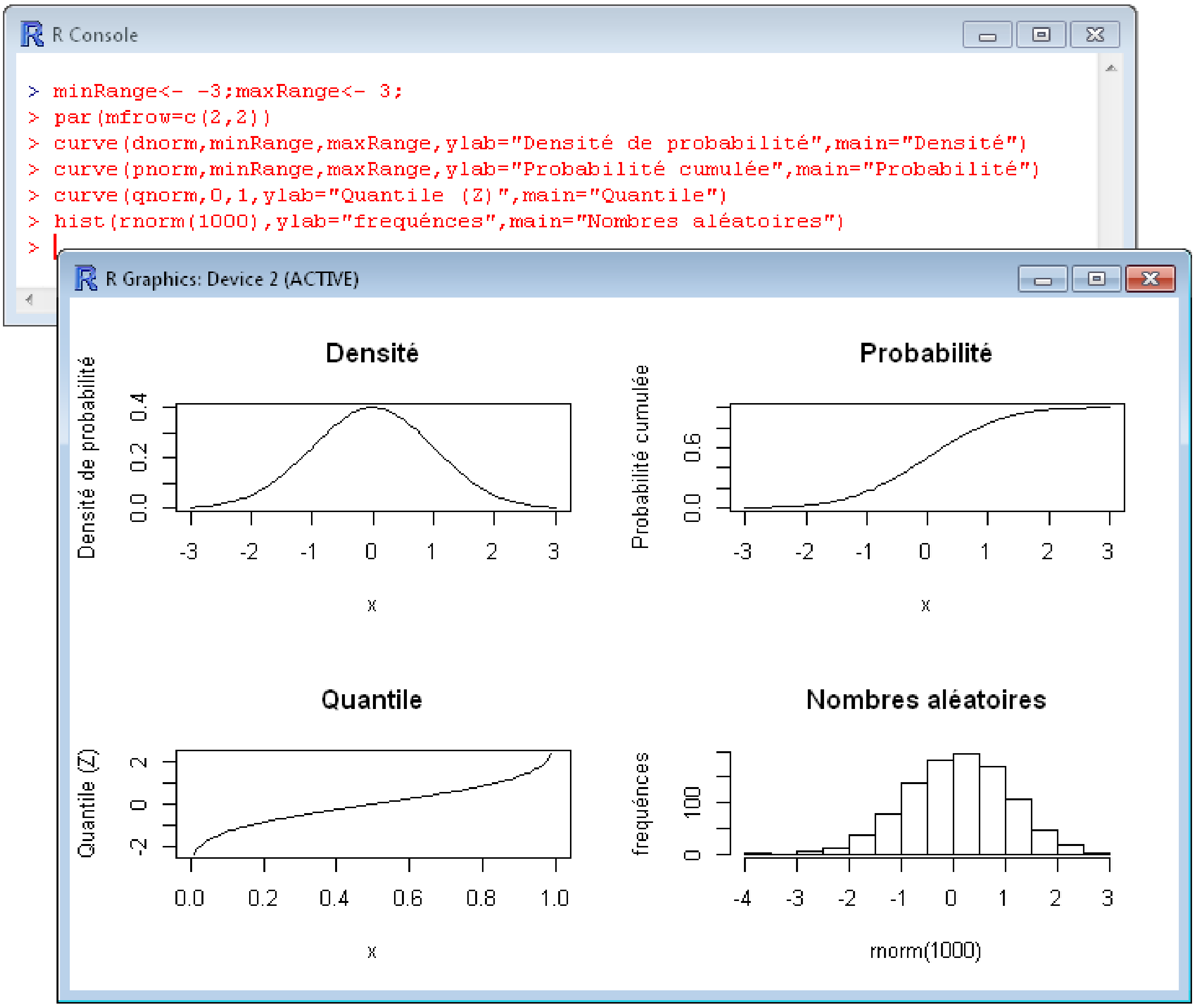
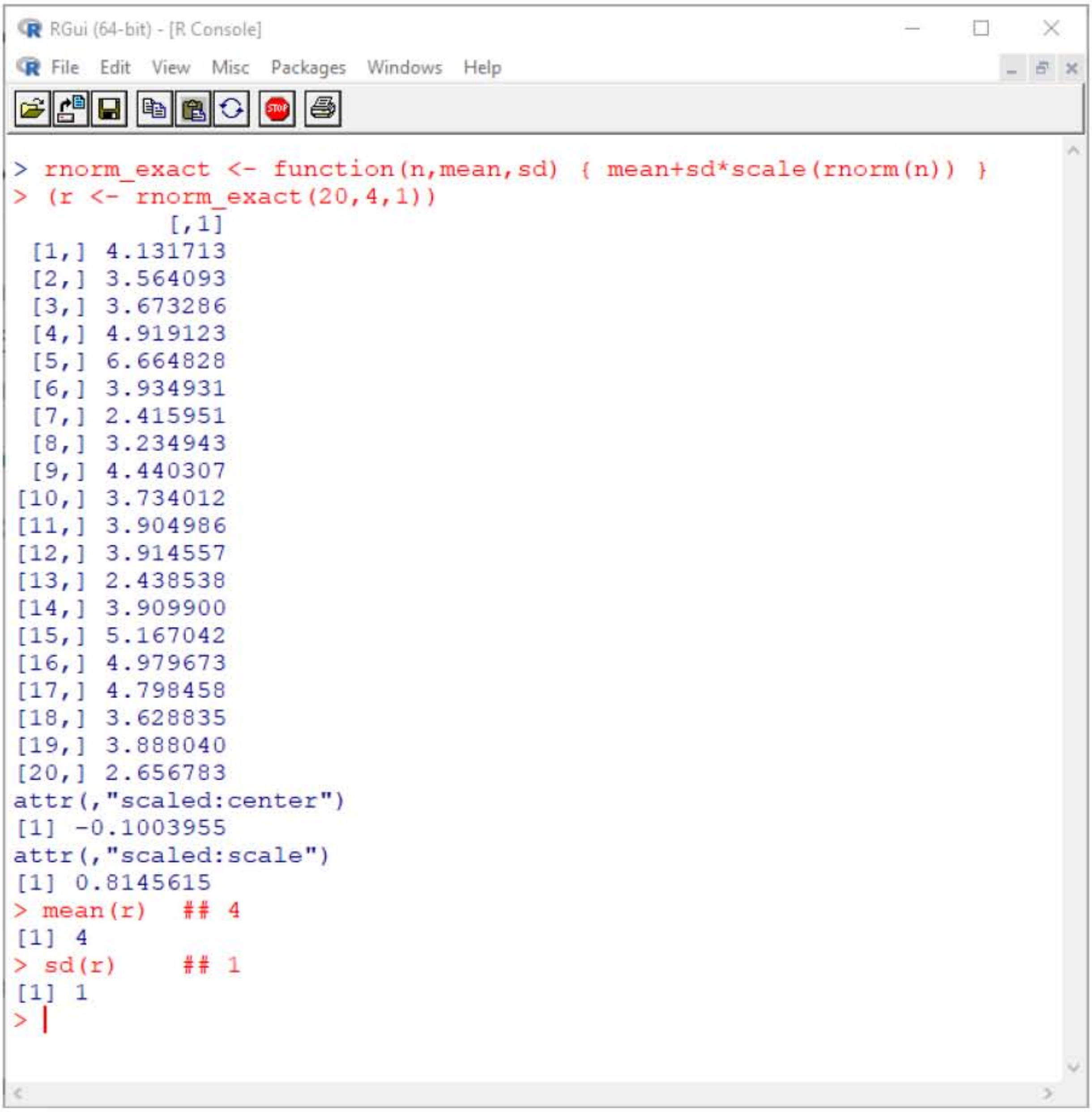


Figure 206 Graph de diagnostique de fonction de probabilité

Et surtout si votre objectif (très utile pédagogiquement pour des formations ou techniquement pour faire des essais d'algorithmes) est de créer une variable aléatoire Normale dont un échantillon aléatoire donne <u>exactement</u> un moyenne et une espérance précise:

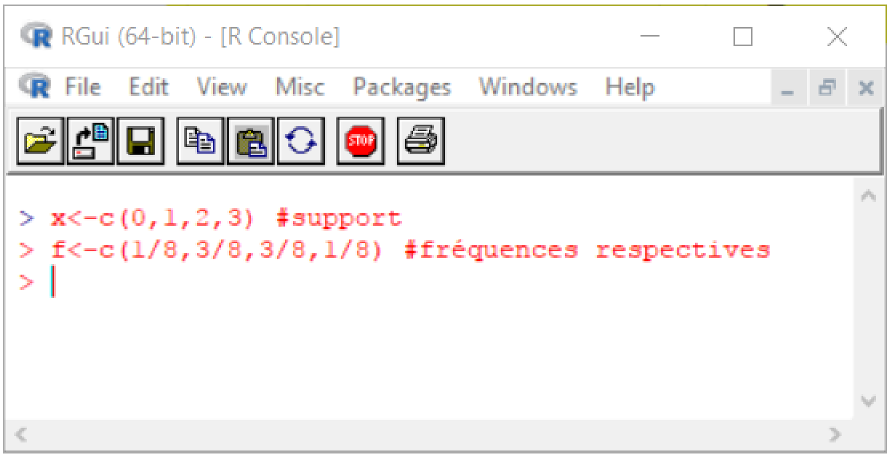

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> rnorm_exact <- function(n,mean,sd) { mean+sd*scale(rnorm(n)) }
> (r <- rnorm_exact(20,4,1))
[,1]
[1,] 4.131713
[2,] 3.564093
[3,] 3.673286
[4,] 4.919123
[5,] 6.664828
[6,] 3.934931
[7,] 2.415951
[8,] 3.234943
[9,] 4.440307
[10,] 3.734012
[11,] 3.904986
[12,] 3.914557
[13,] 2.438538
[14,] 3.909900
[15,] 5.167042
[16,] 4.979673
[17,] 4.798458
[18,] 3.628835
[19,] 3.888040
[20,] 2.656783
attr(,"scaled:center")
[1] -0.1003955
attr(,"scaled:scale")
[1] 0.8145615
> mean(r) ## 4
[1] 4
> sd(r) ## 1
[1] 1
>

Exemple 157.: Loi discrète personnalisée

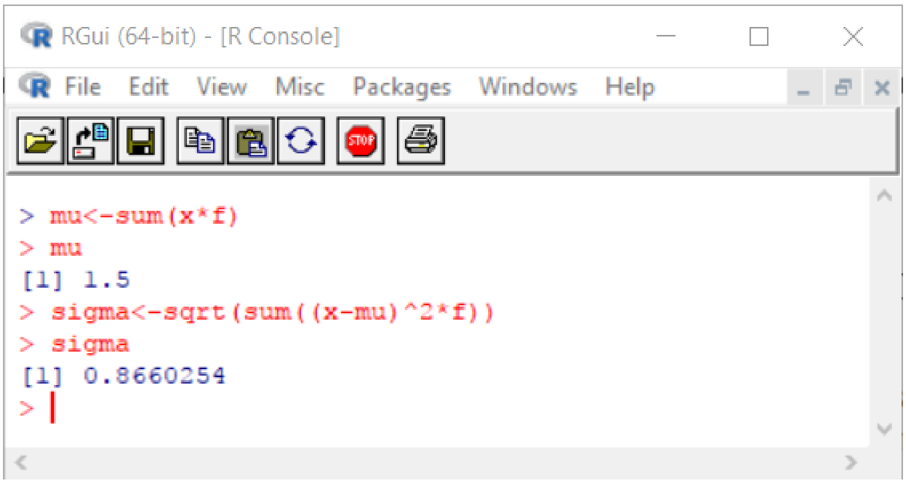
R 3.3.2

Dans la gestion de projets, l'actuariat ou la fiabilité industrielle on doit parfois travailler avec des distributions discrètes! Voyons comment gérer ces dernières dans R.

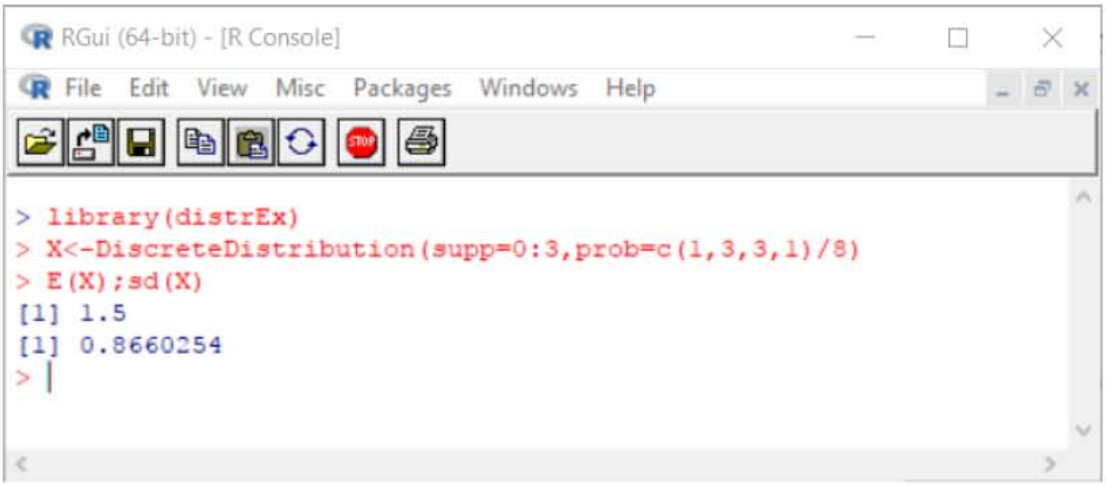
Considérons:



La moyenne et l'écart-type seront trivialement donnés par:



Mais on peut faire plus simple avec la commande **DiscreteDistribution( )** du package **distrEx**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(distrEx)
> X<-DiscreteDistribution(supp=0:3,prob=c(1,3,3,1)/8)
> E(X);sd(X)
[1] 1.5
[1] 0.8660254
>

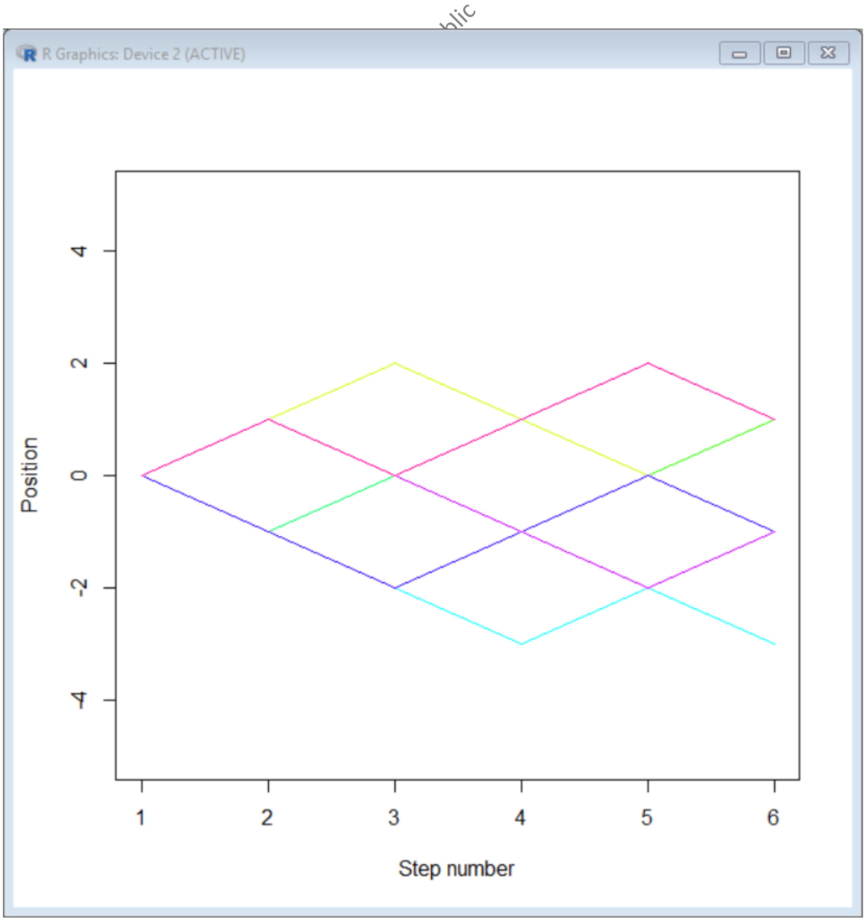
Exemple 158.: Marche aléatoire

R 3.4.2

Nous avons vu dans le cours théorique l'importance de la marche aléatoire pour comprendre nombre de phénomènes physiques. Voyons comment reproduire cela d'abord avec 5 tirages et 10 tracés:

```
> random.walk <- function(nsteps, x0=0) {
+   x <- numeric(nsteps + 1)
+   x[1] <- x0
+   for (i in seq_len(nsteps)) {
+     x[i+1] <- x[i] + sign(rnorm(n=1, mean=0))
+     }
+   x
+ }
> nsteps <- 5
> nrep <- 10
> cols <- rainbow(nrep)
> plot(NA, xlim=c(1, nsteps+1), ylim=c(-nsteps , nsteps ), xlab="Step number", ylab="Position")
> for (i in 1:nrep){
+   lines(random.walk(nsteps), col=cols[i])
+ }
> 
```

Ce qui donne:



Avec 20 tirages et 50 tracés:

```
> random.walk <- function(nsteps, x0=0) {
+   x <- numeric(nsteps + 1)
+   x[1] <- x0
+   for (i in seq_len(nsteps)) {
+     x[i+1] <- x[i] + sign(rnorm(n=1, mean=0))
+     }
+   x
+ }
> nsteps <- 20
> nrep <- 50
> cols <- rainbow(nrep)
> plot(NA, xlim=c(1, nsteps+1), ylim=c(-nsteps , nsteps ), xlab="Step number", ylab="Position")
> for (i in 1:nrep){
+   lines(random.walk(nsteps), col=cols[i])
+ }
>
```

Ce qui donne:

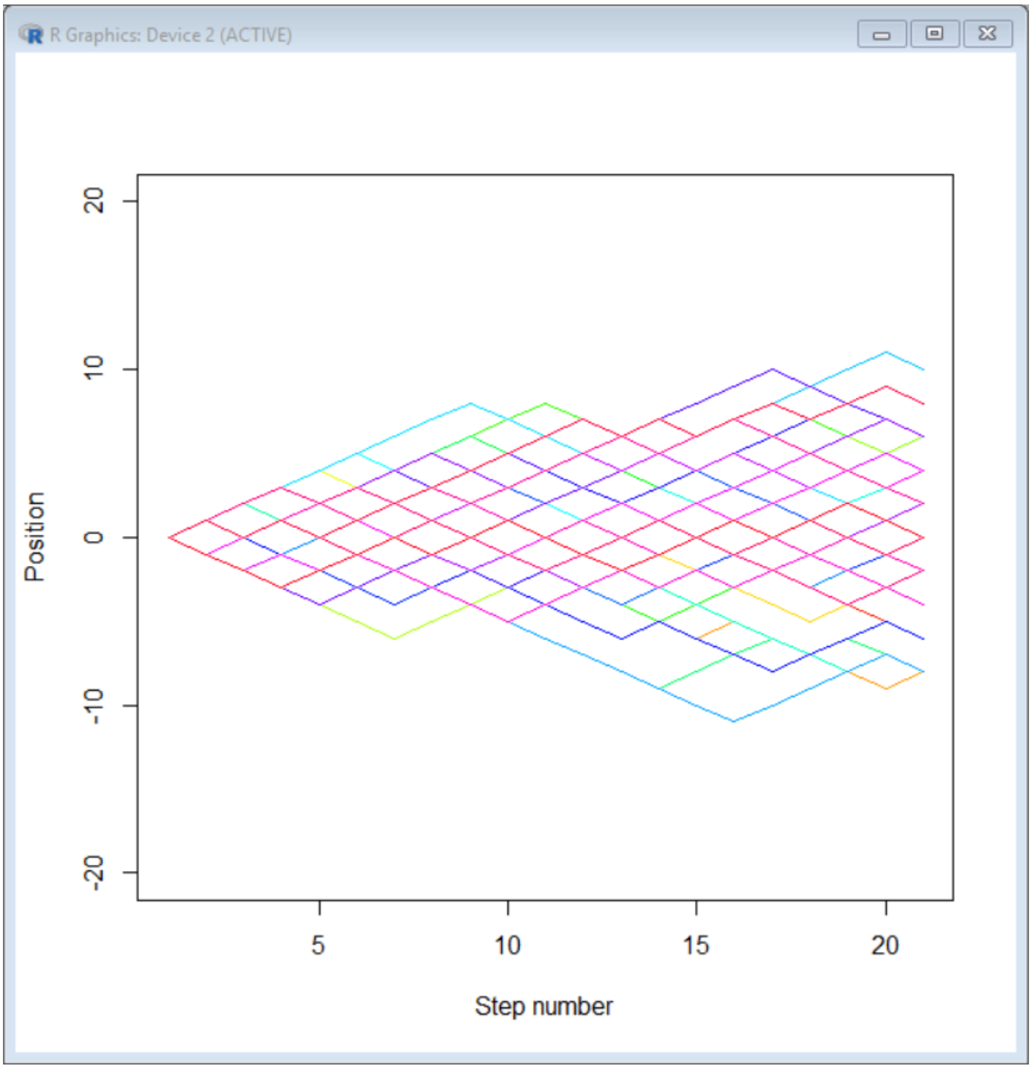
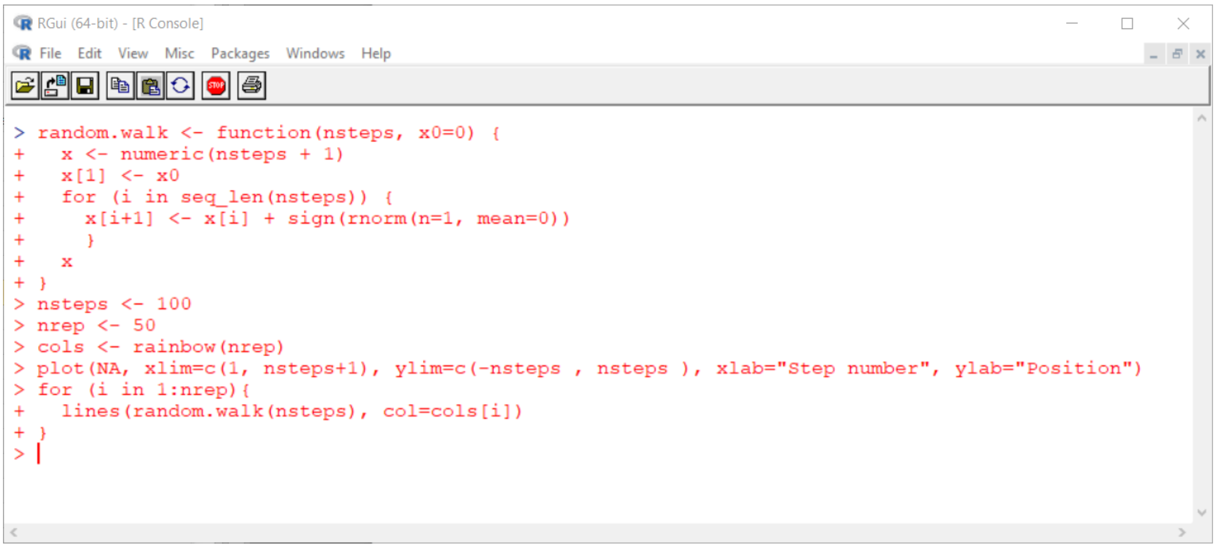
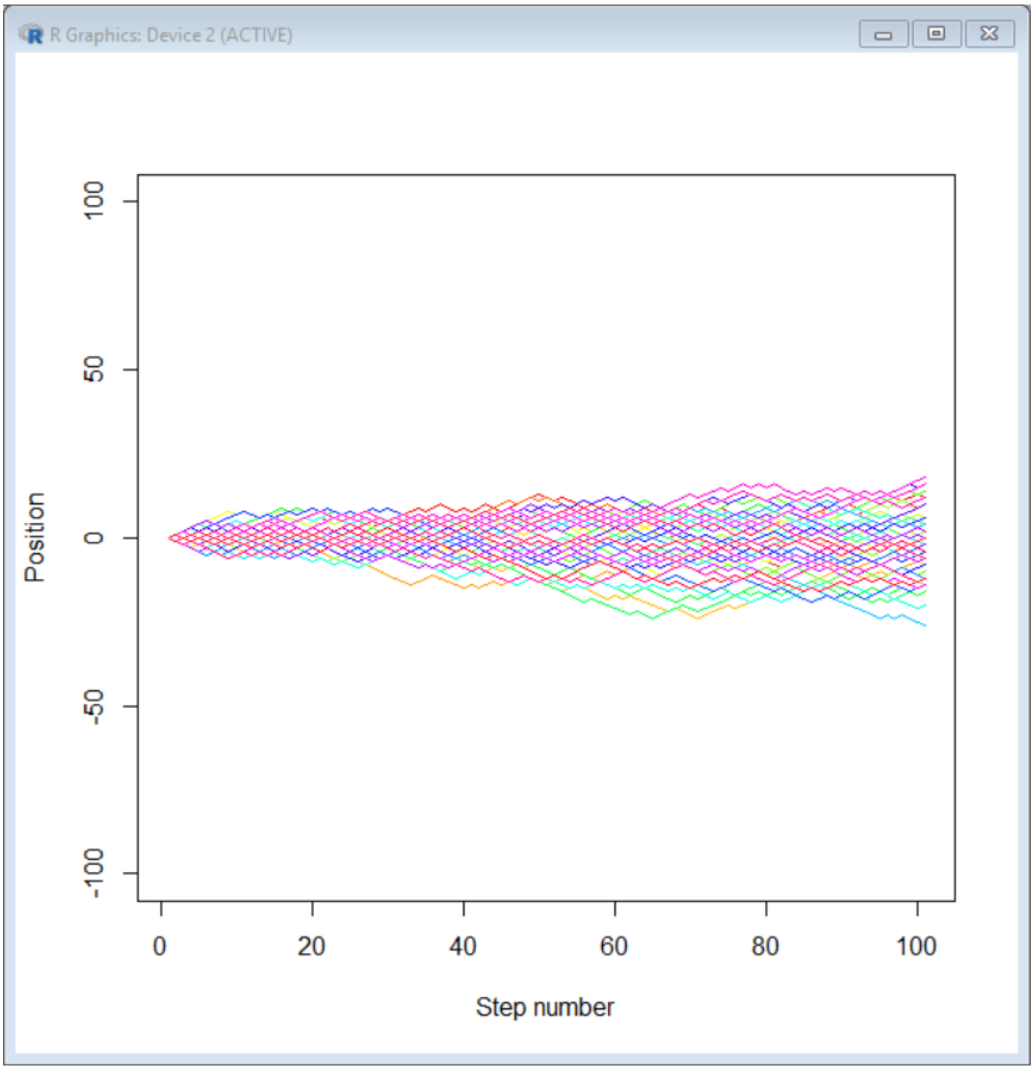


Figure 207 Graph de marche aléatoire à temp discret

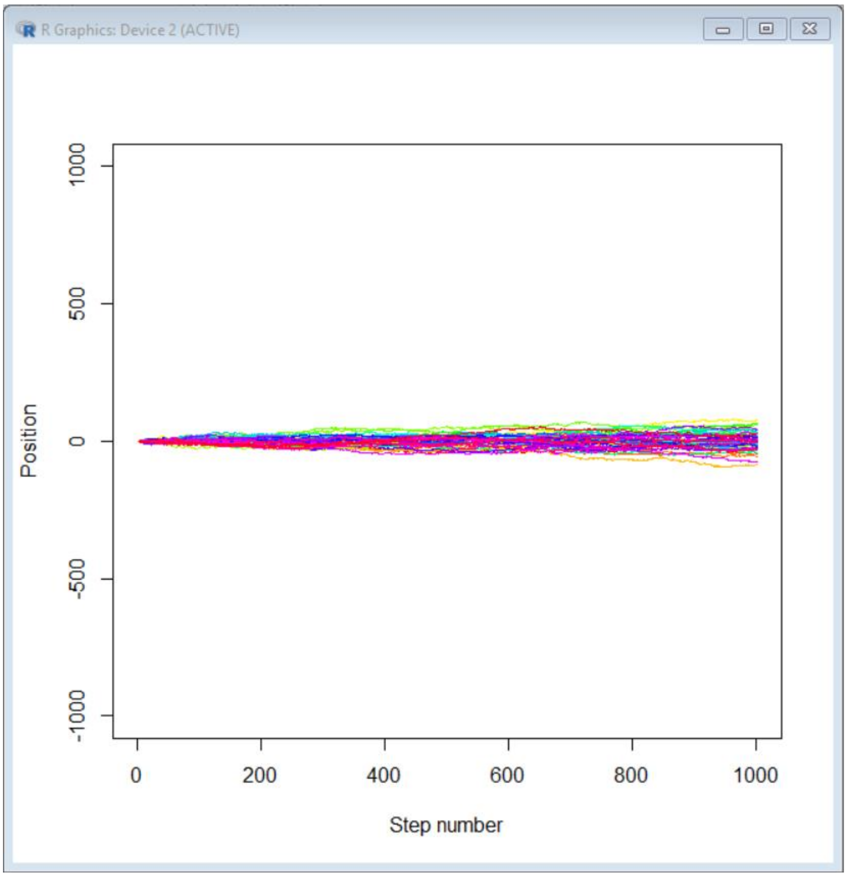
Avec 100 tirages et 50 tracés:



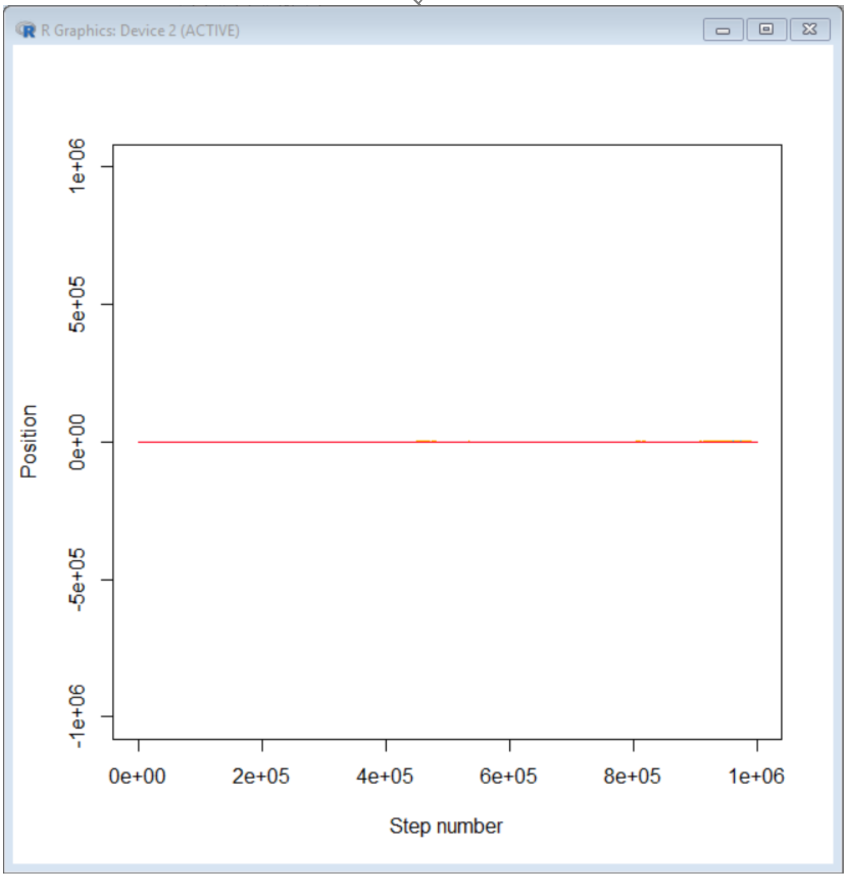
Ce qui donne:



et toujours avec le même code mais avec 1000 tirages et toujours 50 tracés:



Et avec 1'000'000 de tirages et 1'000 tracés:

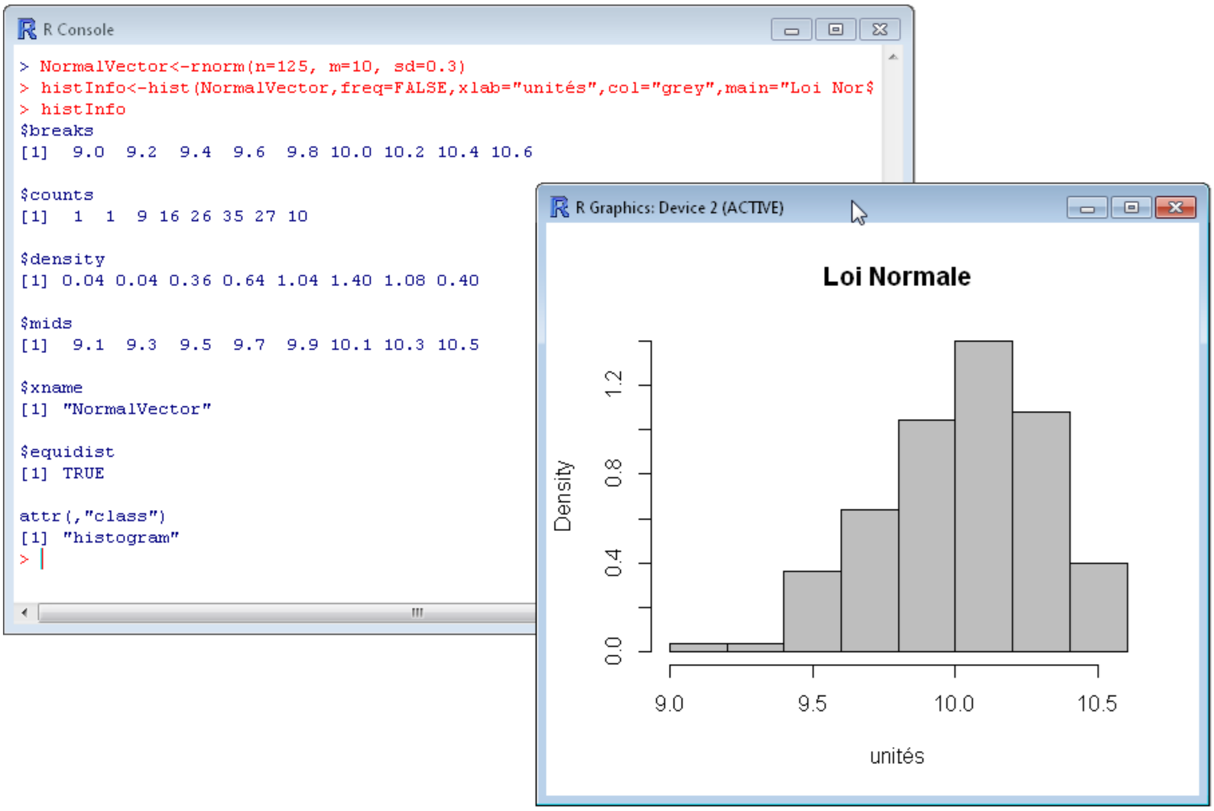


Exemple 159.: Loi Normale

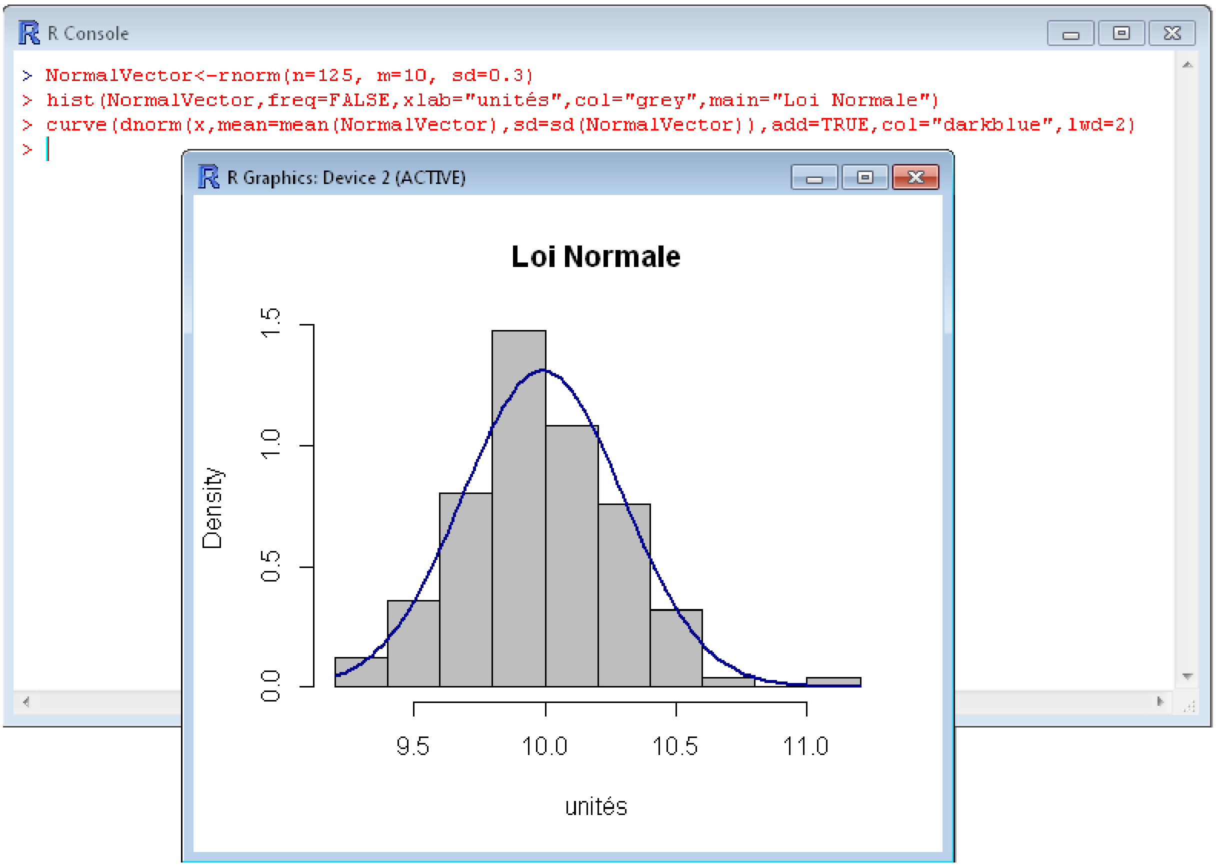
R 3.0.2

Nous allons commencer avec la loi Normale (elle reste la plus utilisée à ce jour) en générant un vecteur Normal et ensuite en affichant son histogramme et des informations y relatives en utilisant les commandes **rnorm( )** et **dnorm( )** .

Commençons donc par un simple histogramme:



et pour ajuster en utilisant la commande **curve( )** :



Ou encore un autre très grand classique:

```
> x<-seq(-5,5,0.01)
> curve(dnorm(x,0,1),xlim=c(-10,10),ylim=c(0,0.5),col="red",lwd=3,ylab="")
> curve(dnorm(x,2,3),col="green",lwd=3,add=T)
> curve(dnorm(x,0,5),col="blue",add=T)
> curve(dnorm(x,-2,0.8),col="orange",add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(mu[1],"=0,",sigma[1],"=1")),
+ expression(paste(mu[2],"=2,",sigma[2],"=3")),
+ expression(paste(mu[3],"=0,",sigma[3],"=5")),
+ expression(paste(mu[4],"=-2,",sigma[4],"=0.8"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution Gaussienne")
>
```

qui donne:

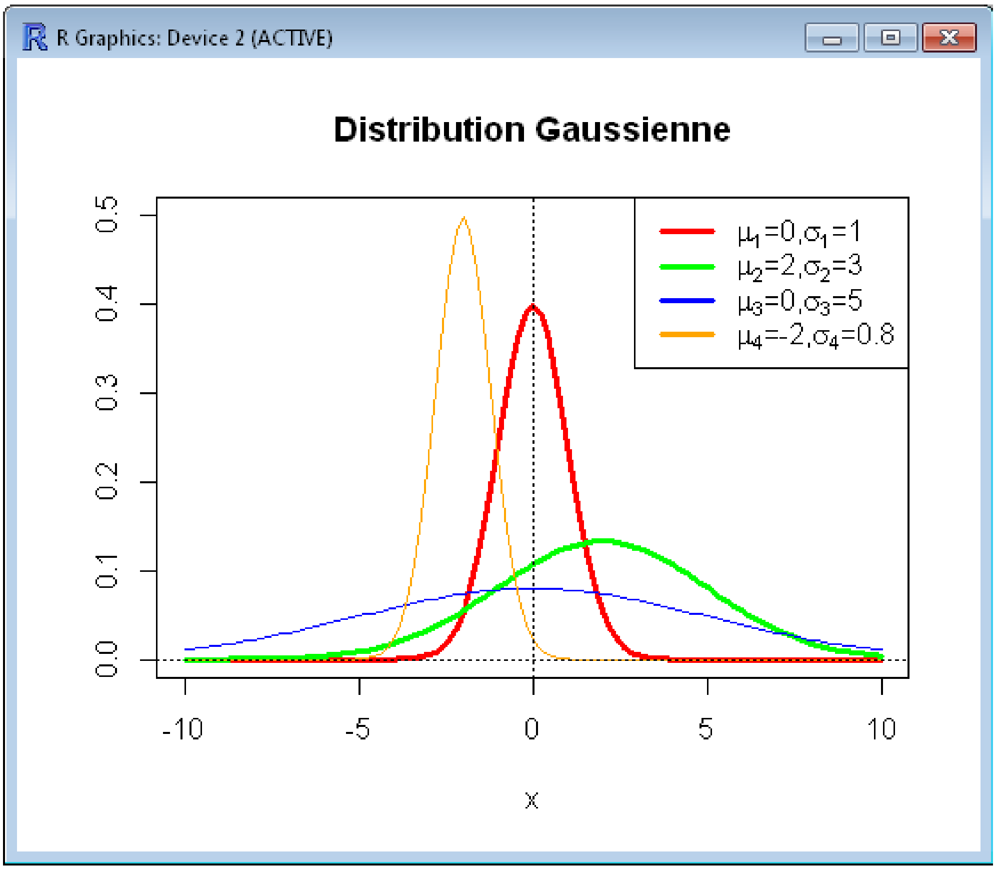
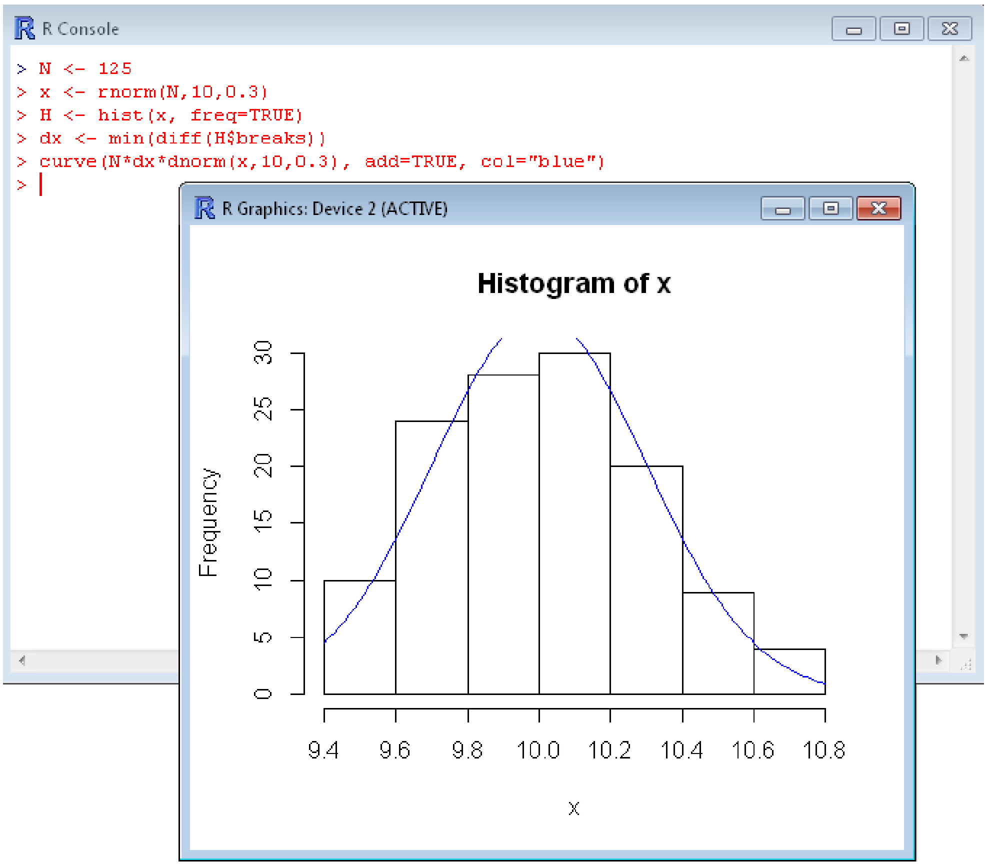


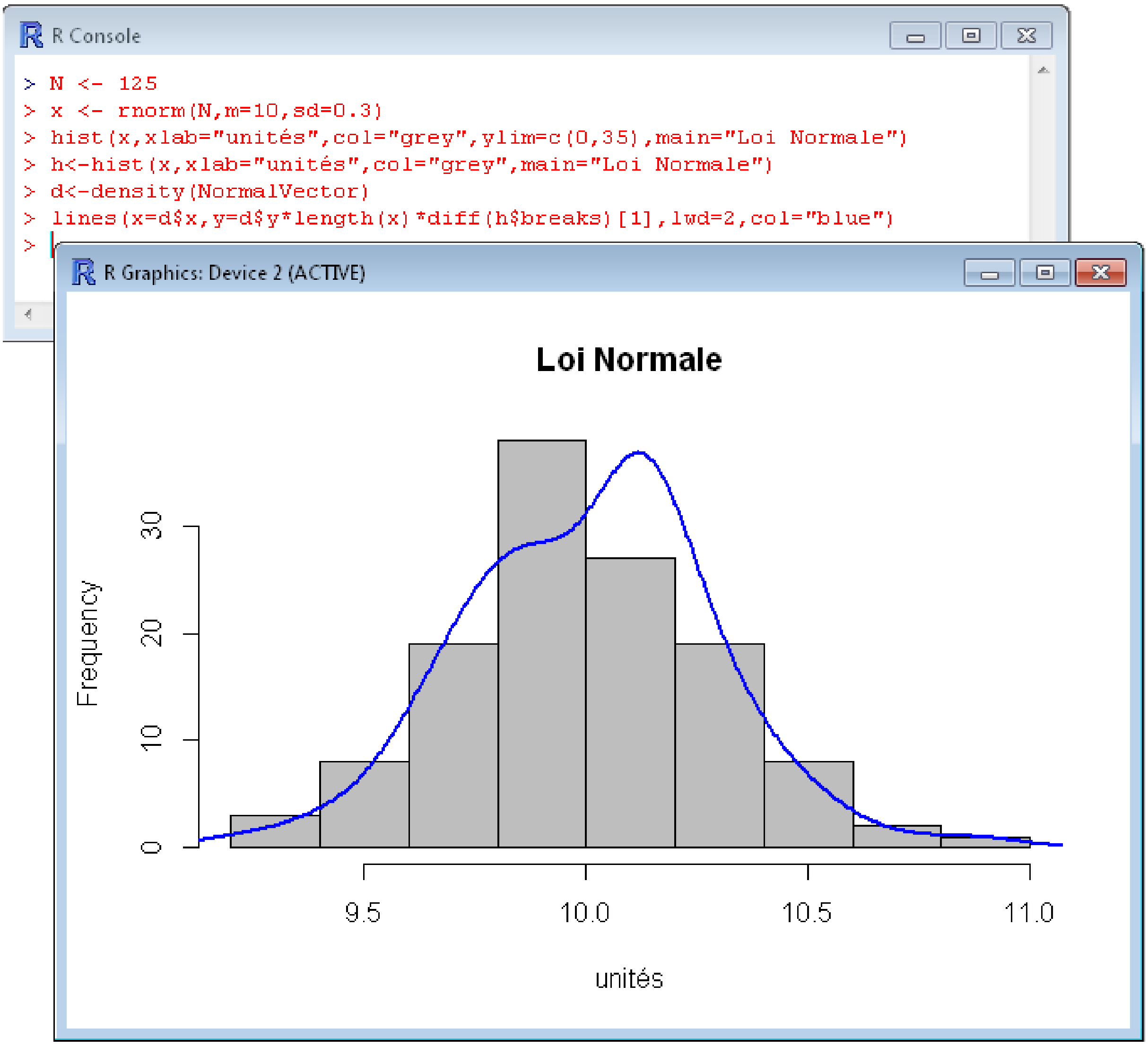
Figure 208 Graph de fonctions de densités multiples

Loi Normale avec fréquences

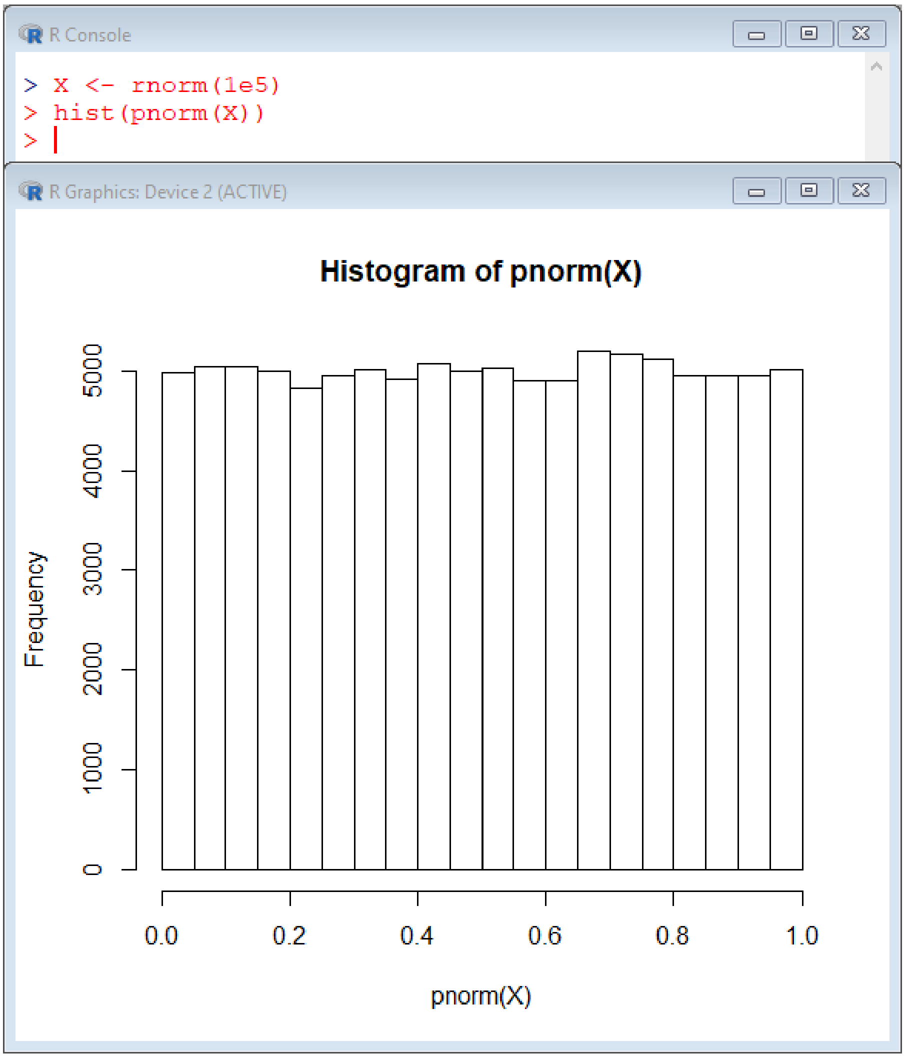
Exceptionnellement et seulement pour la loi Normale (sinon cela commencerait à être ennuyeux si nous le faisons pour toutes les autres lois) nous allons donner la méthode pour avoir comme dans Minitab, le graphique non pas avec les densités mais avec les fréquences:



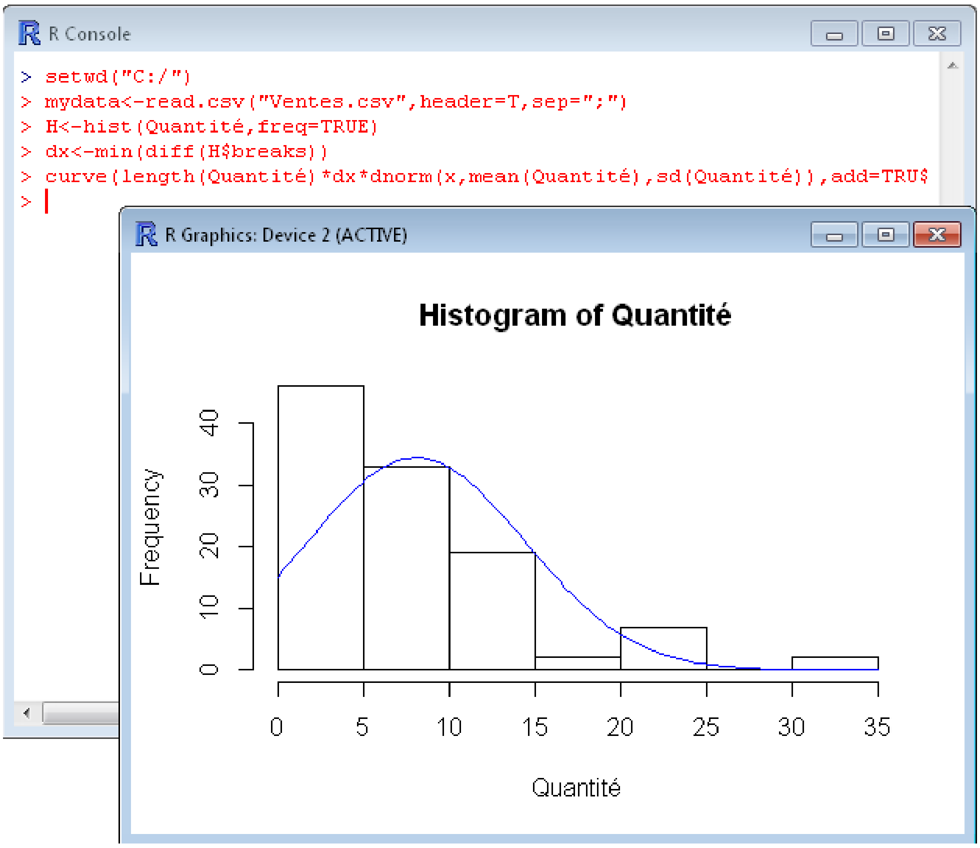
Ou une autre manière de faire utilisant la commande **density( )** :



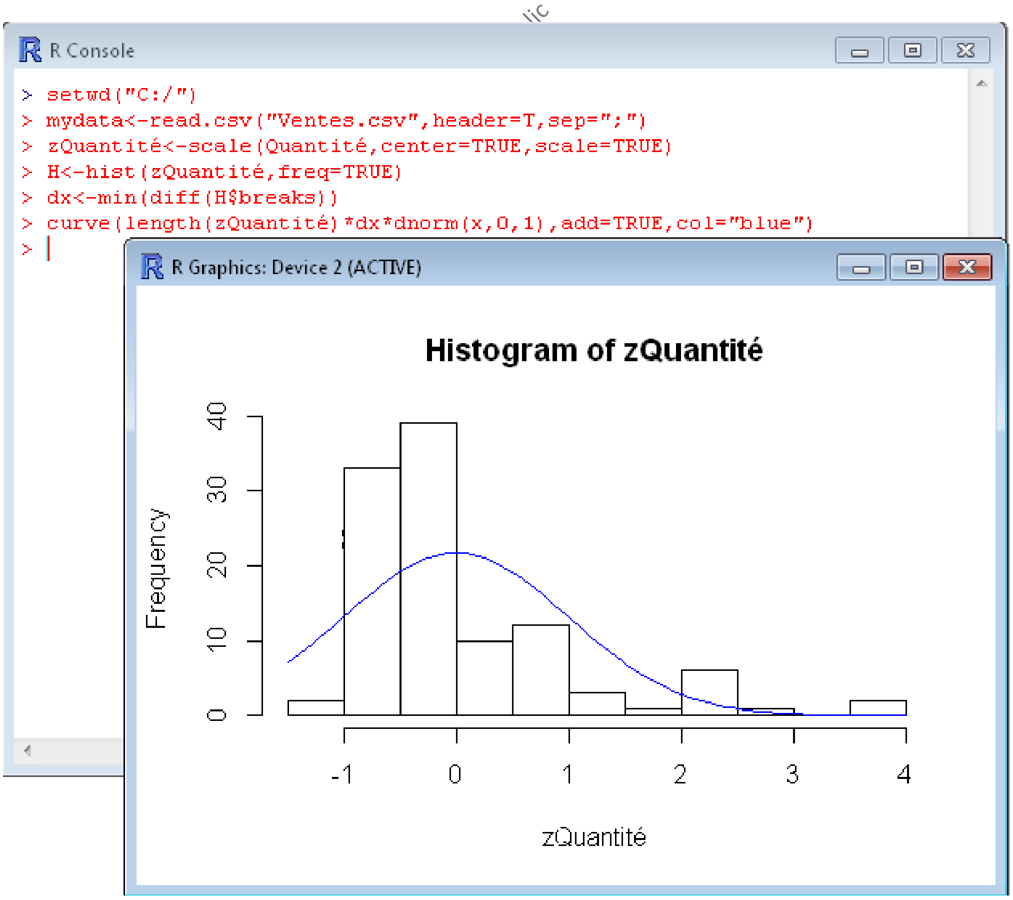
Nous pouvons également par un cas pratique démontrer que les centiles sont uniformément distribués (rappelons que pour beaucoup de gens c'est non-intuitif et que nous avons fait la démonstration mathématique dans le cours théorique):



Il est aussi possible de travailler avec des données que typiquement des non statisticiens vont vouloir ajuster quand même à une loi normale:



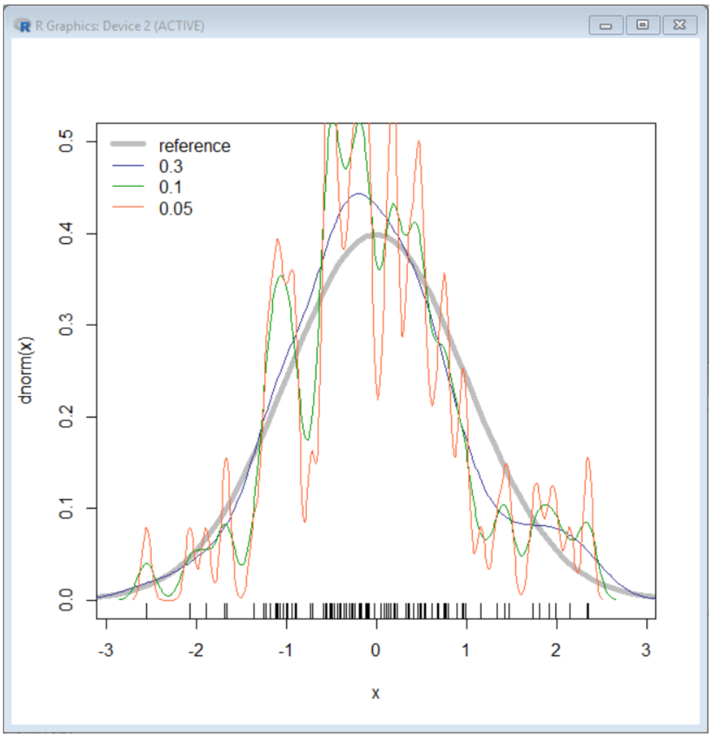
Et nous pouvons centrer réduire les valeurs aussi (très utile en finance):



Enfin, pour ceux qui cherchent à reproduire le graphique que nous avons présenté dans le cours théorique concernant la méthode par densité de noyaux, voici le script:

```
> # "Tight" x-axis (so it appears all horizontal lines span infinitely both left and right
> par(xaxs="i")
>
> # Sequence between -3.1 and 3.1 with 0.1 steps
> x <- seq(-3.1, 3.1, 0.1)
>
> # Plot a perfect bell curve, or a normal distribution probability density function
> plot(x, dnorm(x), "l", ylim=c(0, 0.5), col="grey", lwd=5)
>
> # 100 normally distributed random values with mean 0 and standard deviation 1
> rx <- rnorm(100)
>
> # Add lines that attempt to characterize the distribution probability density for "rx"
> # using different bandwidth "bw" values 0.3, 0.1 and 0.5 (coarse to fine widths)
> lines(density(rx, bw=0.3), col="#333399")
> lines(density(rx, bw=0.1), col="#009900")
> lines(density(rx, bw=0.05), col="#FF6633")
>
> # Ticks at bottom to indicate where the random "rx" values are; add legend
> rug(rx)
> legend("topleft", legend=c("reference", "0.3", "0.1", "0.05"), bty="n",
+        col=c("grey", "#333399", "#009900", "#FF6633"), lwd=c(5, 1, 1, 1))
>
```

Ce qui donne:



Évidemment ce genre de manipulation avec les données que nous avons ci-dessus se passe de commentaires...

Loi Normale avec double axe

Minitab n'est pas capable de plotter un histogramme et une distribution avec un axe respectif pour chacun ainsi qu'avec un échelle personnalisée adaptée à chaque graph. Par contre avec le tableur Microsoft Excel c'est très simple. Pour R je suis arrivé à faire mieux que Minitab en obtenant le double axe mais je ne suis pas arrivé à ce jour à trouver une solution simple (comme dans le tableur Excel) pour changer l'échelle du deuxième axe à droite de 0 à 1... (si quelqu'un à une solution je suis preneur):

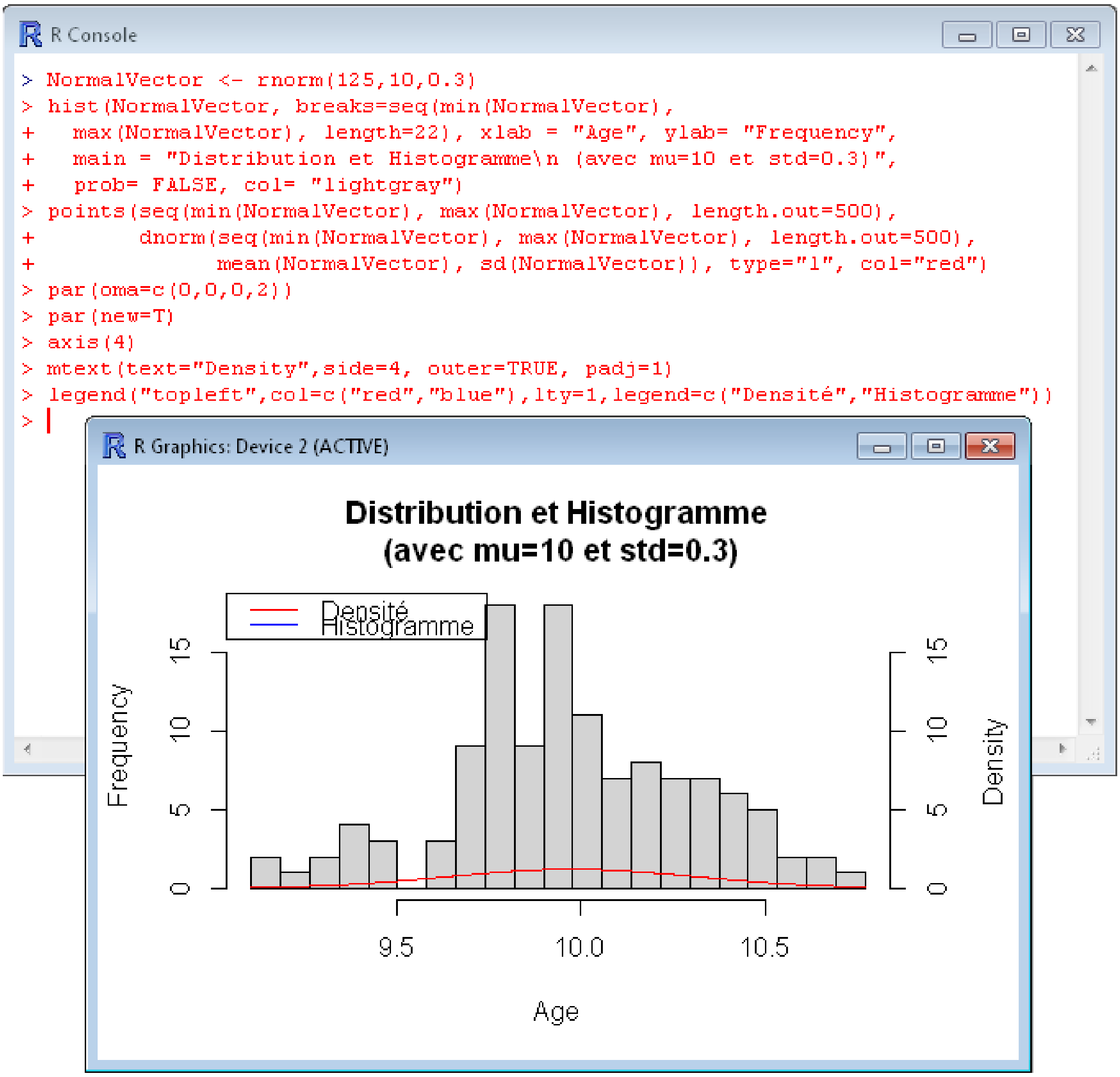


Figure 209 Graph histogramme avec courbe de densité et double axe

Loi Normale avec distribution et répartition des fréquences

C'est un cas souvent demandé dans le cours Statistique avec des tableurs voyons comment faire cela avec R car c'est loin d'être intuitif pour changer...

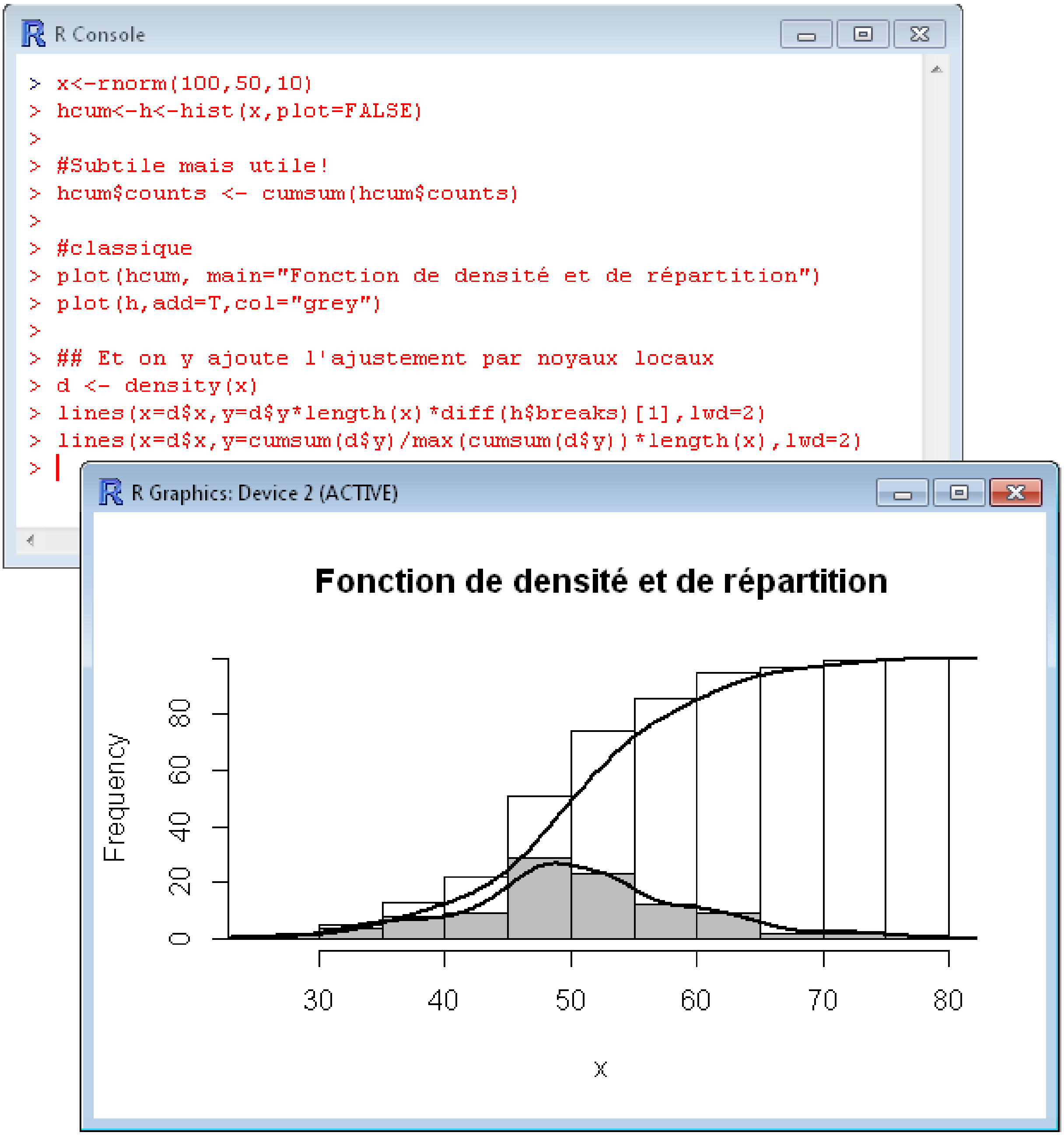


Figure 210 Graph de fonction de densité avec histogramme et fonction de répartition

Convolution de deux lois Normales

Le but ici étant de reproduire la démonstration mathématique faite dans le cours théorique, donc de voir un cas pratique de convolution aussi par la même occasion, et vérifier si on retombe sur le même résultat qu'avec MATLAB:

Pour les détails sur ce sujet, nous renvoyons le lecteur au chapitre sur le traitement du signal à la page 2870.

Loi Normale bivariée

Un cas très important en finance et en maîtrise statistique des processus. Nous allons pour cela utiliser la théorie vue en cours sous forme de fonction R:

```
> mu1 <- 0
> mu2 <- 0.5
> sig1 <- 0.5
> sig2 <- 2
> rho <- 0.5
>
> #limites pour les axes
> xm <- -3
> xp <- 3
> ym <- -3
> yp <- 5
>
> x <- seq(xm, xp, length= as.integer((xp + abs(xm)) * 10))
> y <- seq(ym, yp, length= as.integer((yp + abs(ym)) * 10))
>
> bivariate <- function(x,y){
+ term1 <- 1 / (2 * pi * sig1 * sig2 * sqrt(1 - rho^2))
+ term2 <- (x - mu1)^2 / sig1^2
+ term3 <- -(2 * rho * (x - mu1)*(y - mu2))/(sig1 * sig2)
+ term4 <- (y - mu2)^2 / sig2^2
+ z <- term2 + term3 + term4
+ term5 <- term1 * exp((-z / (2 *(1 - rho^2))))
+ return (term5)
+ }
>
> z <- outer(x,y,bivariate)
>
> persp(x, y, z, main = "Bivariate Normal Distribution",
+ sub = bquote(bold(mu[1])==.(mu1)~", "~sigma[1]==.(sig1)~", "~mu[2]==.(mu2)~
+ ", "~sigma[2]==.(sig2)~", "~rho==.(rho)),
+ col="orchid2", theta = 55, phi = 30, r = 40, d = 0.1, expand = 0.5,
+ ltheta = 90, lphi = 180, shade = 0.4, ticktype = "detailed", nticks=5)
>
```

Ce qui donne:

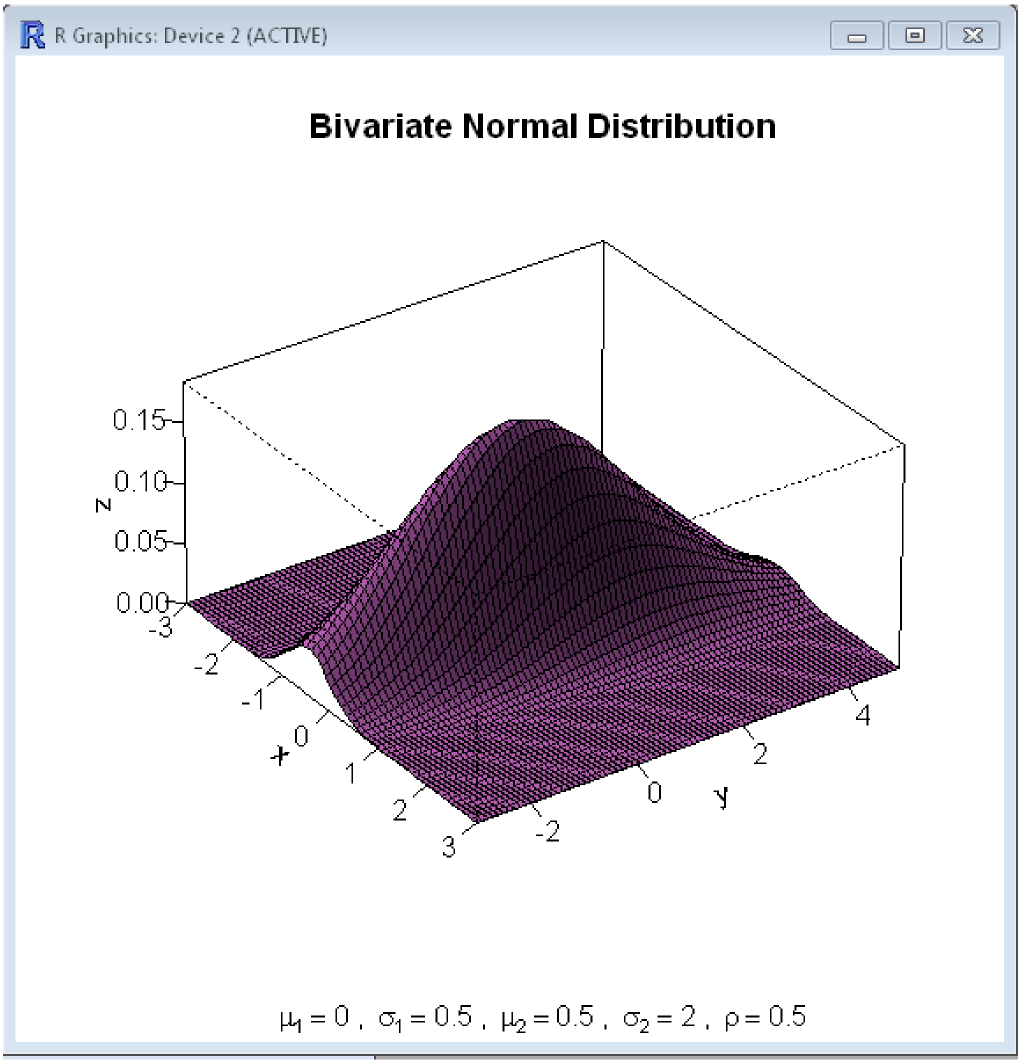
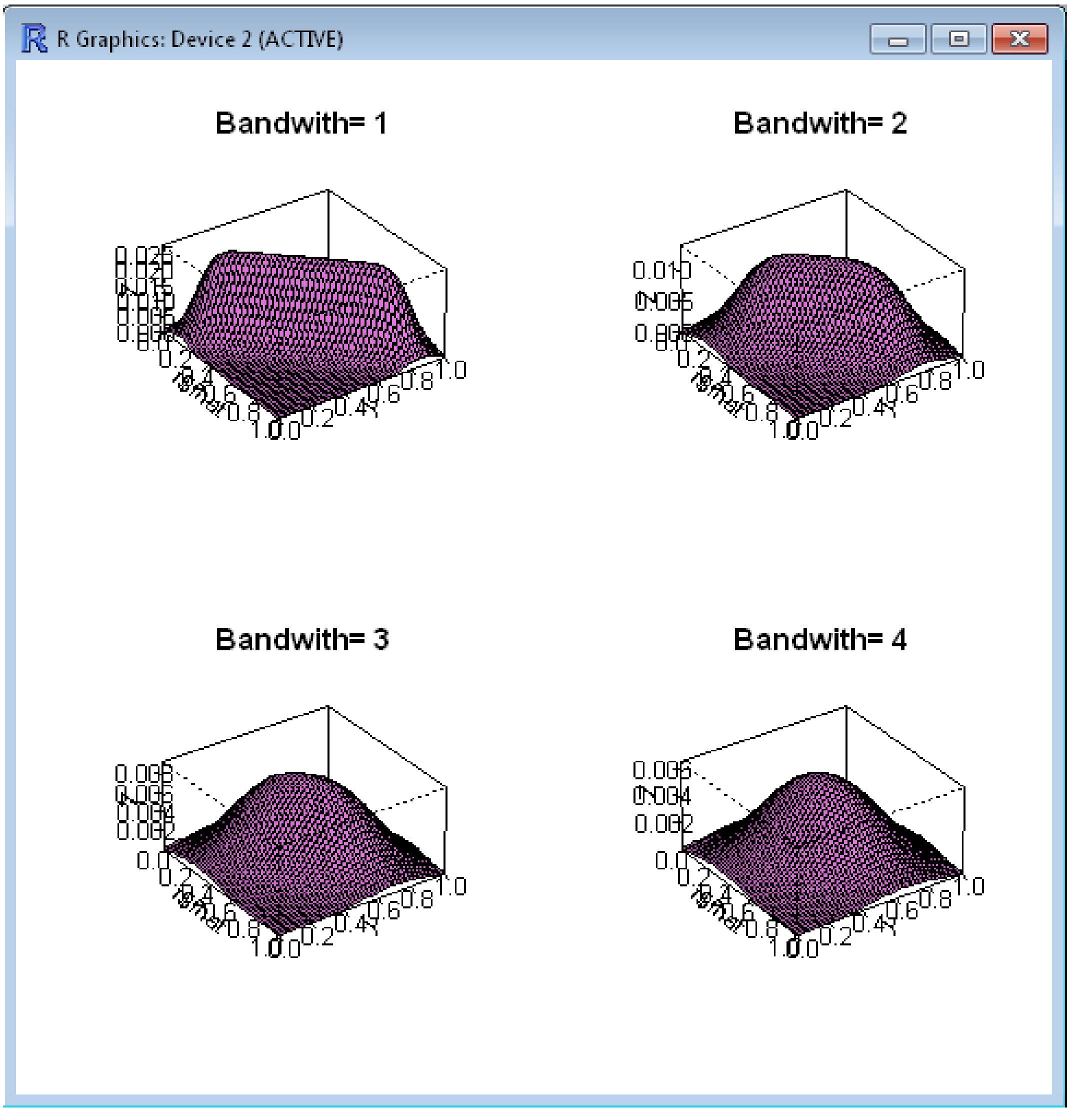


Figure 211 Graph de loi Normale bivariée

Si nous n'avons que des points dont la distribution est inconnue, nous pouvons utiliser la commande **bkde2D( )** du package **KernSmooth** pour chercher la meilleure densité 3D approchée par noyaux locaux:

```
> mu1<-0;mu2<-0;sig1<-0.5;sig2<-2;rho<-0.5
> xm<--5;xp<-5;ym<--5;yp<-5
> x<-seq(xm,xp,length=as.integer((xp+abs(xm))*100))
> y<-seq(ym,yp,length=as.integer((yp+abs(ym))*100))
> d<-cbind(x,y)
> #pour la bandwith elle est donnée pour x et respectivement pour y
> #plus ses valeurs sont grandes, plus la convergence est bonne
> par(mfrow=c(2,2))
> for(i in 1:4){
+ r<-bkde2D(d,bandwidth=c(i,i))
+ persp(r$fhat,col="orchid2",d=0.1,r=40,shade=0.4,phi=30,theta=55,
+ ltheta=90,ticktype="detailed",expand=0.5,main=paste("Bandwith=",i))
+ }
>
```

Ce qui donne:



Ellipse de confiance (Youden Plot)

Dans un autre style en utilisant le package **mnormt** et la commande **dmnorm( )** associée nous avons un cas très classique beaucoup utilisé dans le domaine industriel (contrôle statistique des procédés) dans le cas de côtes bi-dimensionnelles:

```
> r=.5
> library(mnormt)
> S=matrix(c(1,r,r,1),2,2)
> f=function(x,y) dmnorm(cbind(x,y),varcov=S)
> vx=seq(-3,3,length=201)
> vy=seq(-3,3,length=201)
> z=outer(vx,vy,f)
> set.seed(1)
> X=rmnorm(1500,varcov=S)
> xhist <- hist(X[,1], plot=FALSE)
> yhist <- hist(X[,2], plot=FALSE)
> top <- max(c(xhist$density, yhist$density,dnorm(0)))
> nf <- layout(matrix(c(2,0,1,3),2,2,byrow=TRUE), c(3,1), c(1,3), TRUE)
> par(mar=c(3,3,1,1))
> image(vx,vy,z,col=rev(heat.colors(101)))
> contour(vx,vy,z,col="blue",add=TRUE)
> points(X,cex=.2)
> par(mar=c(0,3,1,1))
> barplot(xhist$density, axes=FALSE, ylim=c(0, top), space=0,col="light green")
> lines((density(X[,1])$x-xhist$breaks[1])/diff(xhist$breaks)[1],
+ dnorm(density(X[,1])$x),col="red")
> par(mar=c(3,0,1,1))
> barplot(yhist$density, axes=FALSE, xlim=c(0, top), space=0,
+ horiz=TRUE,col="light green")
> lines(dnorm(density(X[,2])$x),(density(X[,2])$x-yhist$breaks[1])/
+ diff(yhist$breaks)[1],col="red")
> 
```

Ce qui donne:

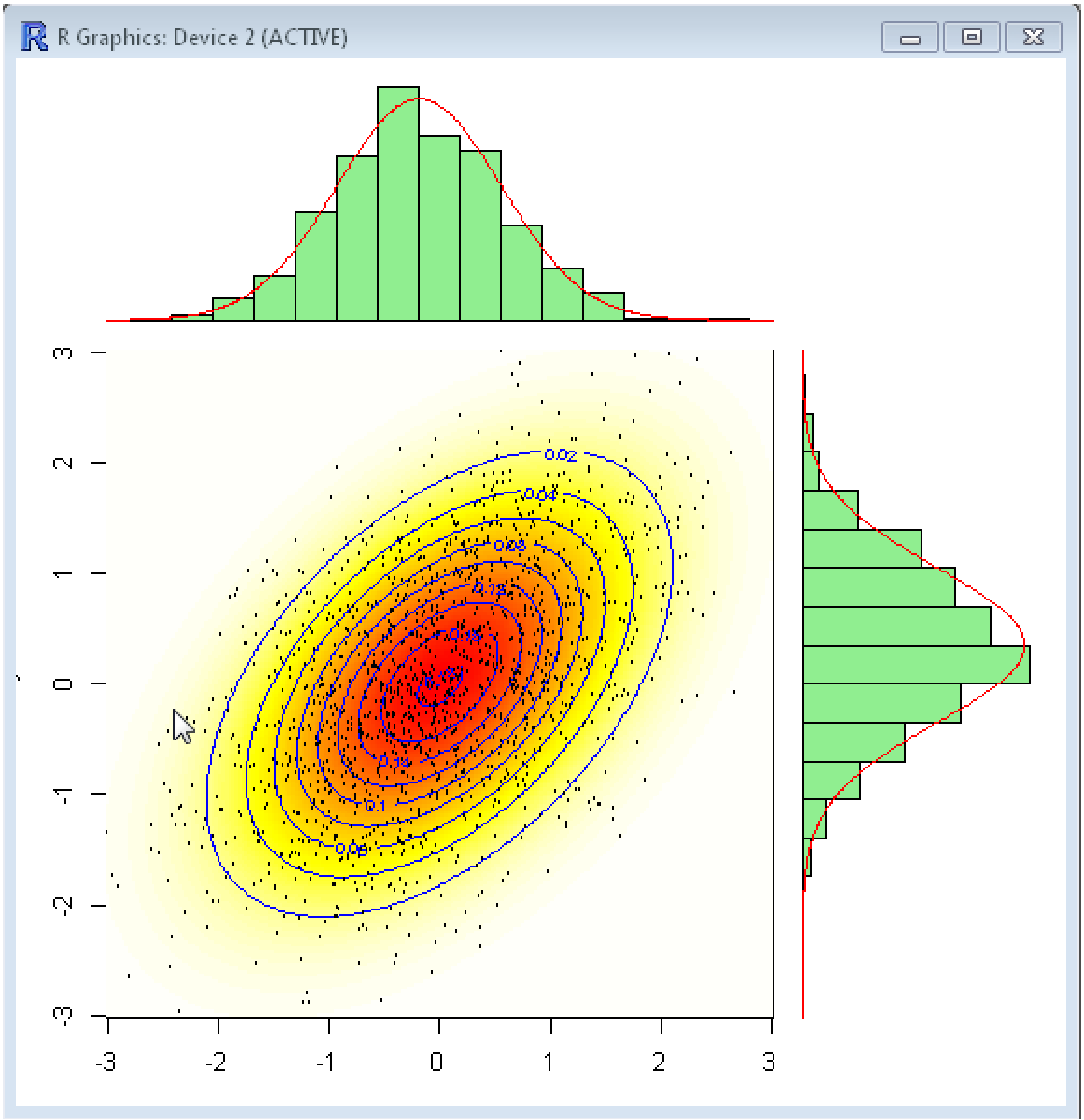
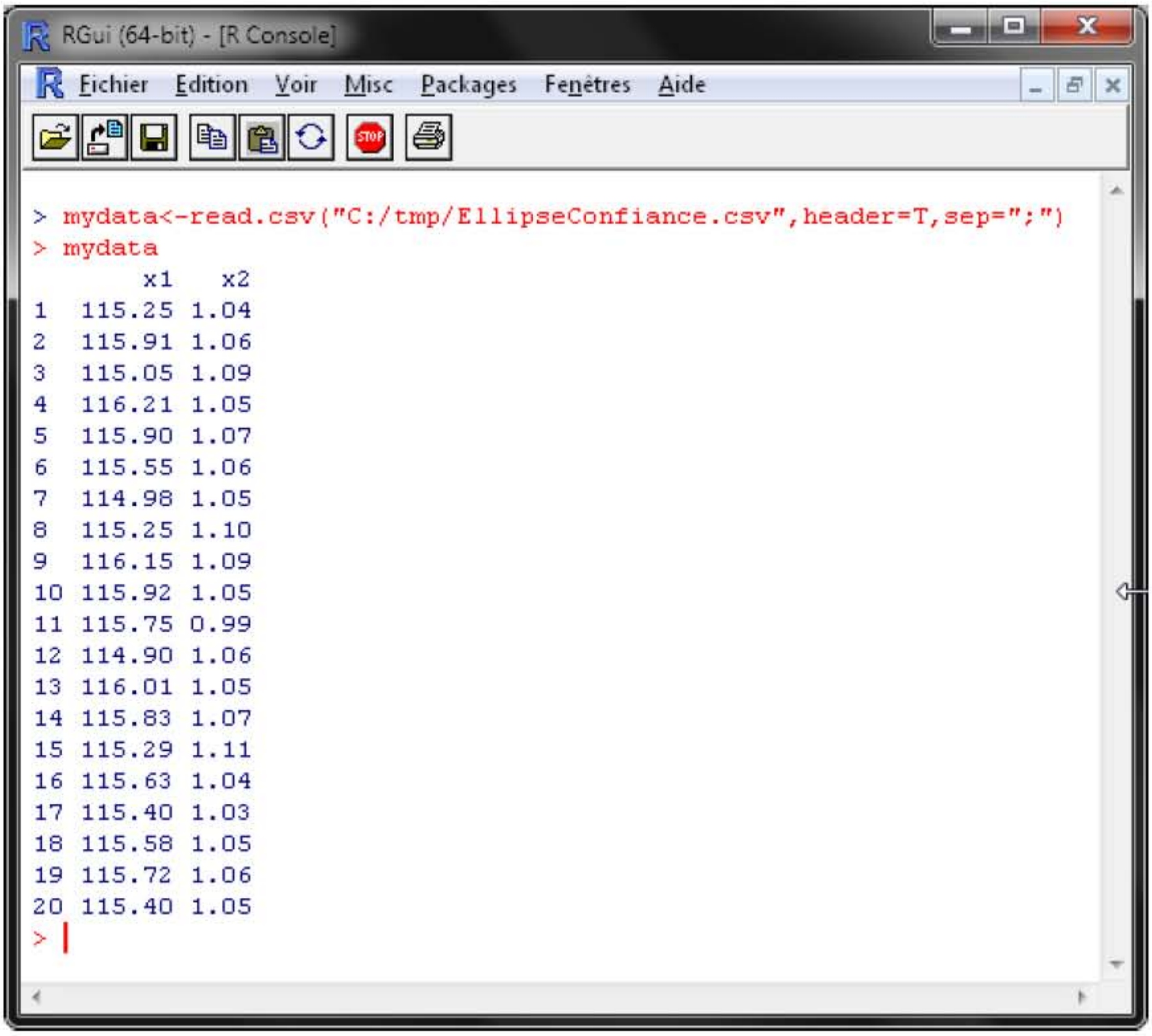


Figure 212 Graph de Youden (fonction de densité bivariée avec histogramme et fonction de densité en marge)

Enfin, si nous avons deux variables aléatoires centrées réduites corrélées entre elles, nous pouvons représenter les ellipses de confiance plus simplement en utilisant les données du cours théorique:



Nous avons alors:

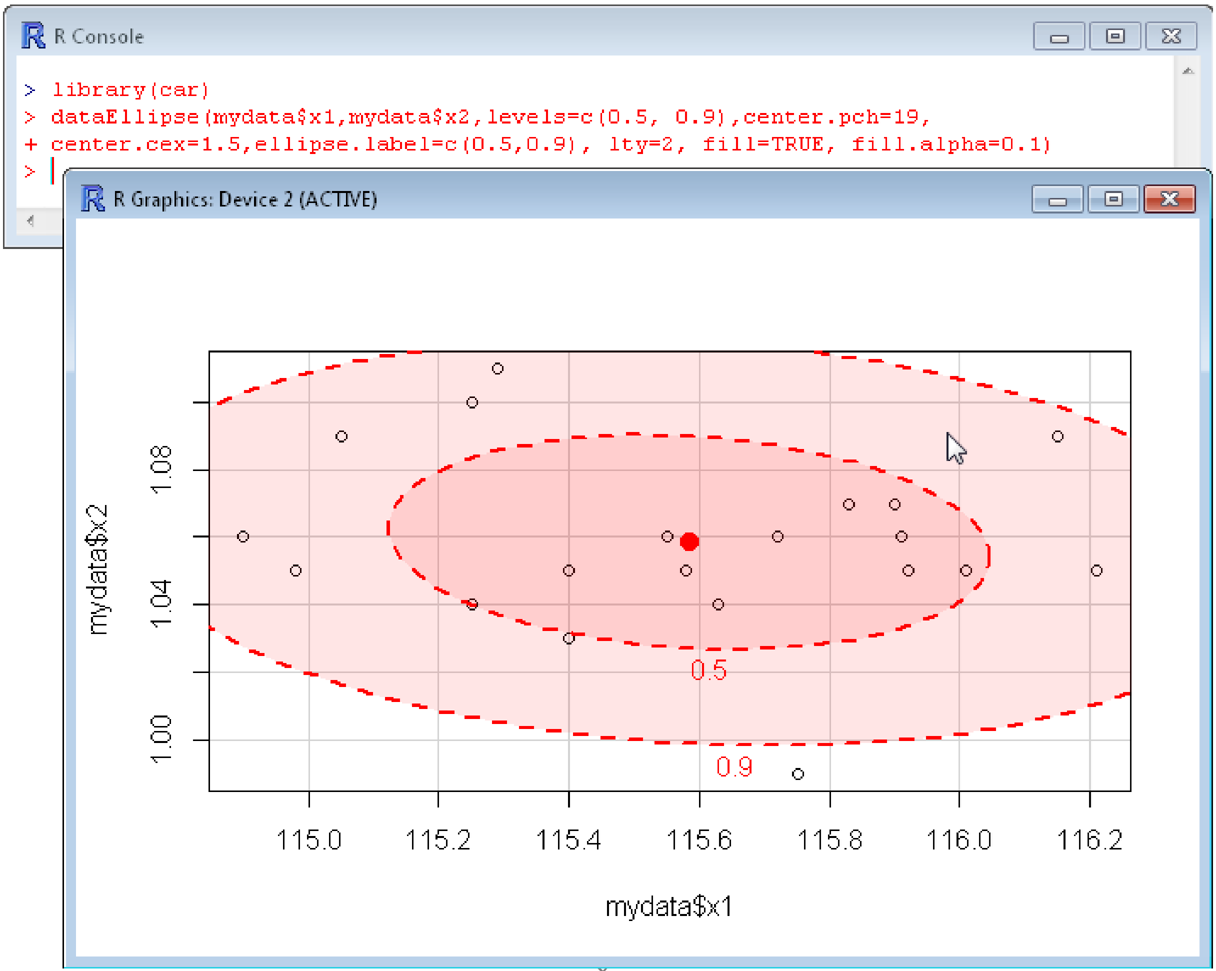


Figure 213 Graph d'ellipses de confiance pour fonctions bivariées

Loi Normale trivariée

Considérons un processus de fabrication sous contrôle statistique de joints dont trois cotes sont mesurées à chaque fois. Diamètre interne, externe et épaisseur. Les trois cotes suivante une loi Normale centrée. Évidemment, ci-dessous nous générons les données mais on peut simplement avoir un fichier *.csv avec trois colonnes (dixit trois vecteurs) avec les valeurs mesurées:

```
R Console
> library(rgl)
> #Plot d'une fonction Normale trivariée avec intervalle
> #de probabilité de 95% d'un processus de fabrication sous contrôle
> #avec espérance de déviation e n% égale 0, variance unitaires et
> #corrélation de 0.8
> if (requireNamespace("MASS")) {
+ Sigma<-matrix(c(10,3,0,3,2,0,0,0,1), 3,3)
+ Mean<-c(0,0,0)
+ x <- MASS::mvrnorm(1000, Mean, Sigma)
+ open3d()
+ #On ne peut pas mettre d'accents dans les légendes et le titre
+ plot3d(x, xlab="Var. Diametre interne %",ylab="Var. Diametre externe%",
+ zlab="Var. Epaisseur%", box=TRUE)
+ plot3d( ellipse3d(Sigma, centre=Mean), col="green", alpha=0.5, add = TRUE)
+ }
>
```


Ce qui donne:

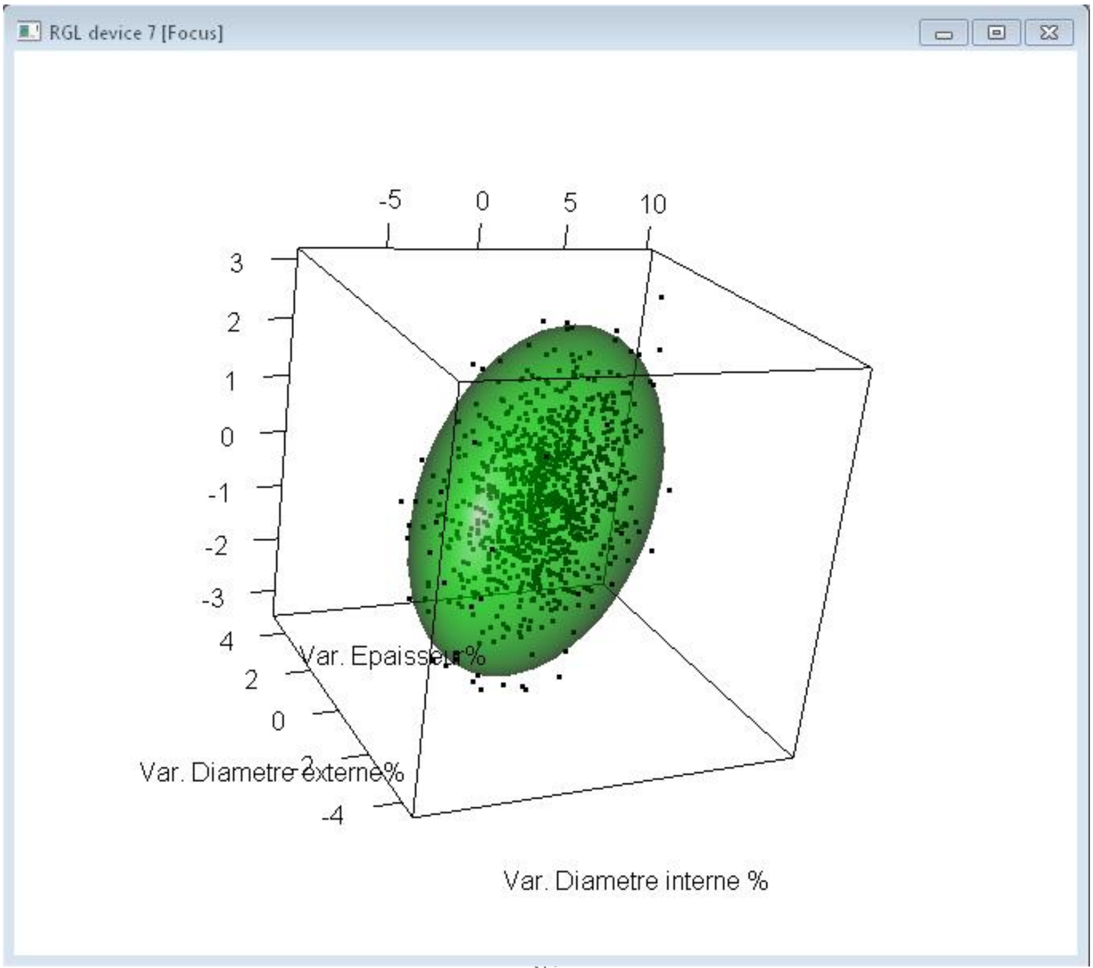


Figure 214 Graph de fonction de densité trivariée

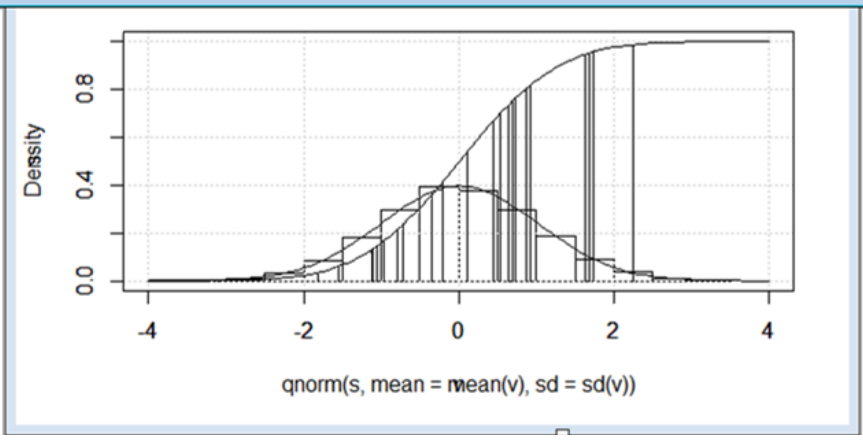
Évidemment c'est une approche purement qualitative!

Exemple 160.: Loi Uniforme (continue)

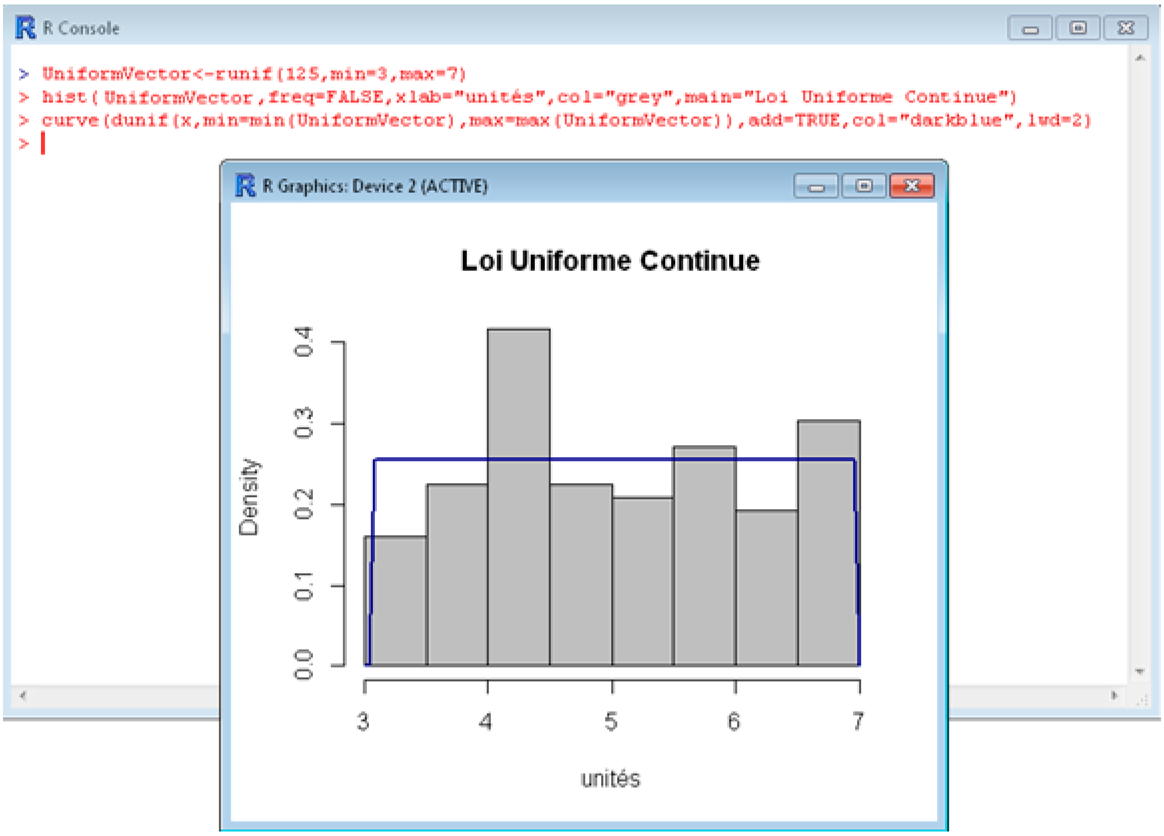
R 3.0.2

Nous allons voir à partir de ce point comment représenter graphiquement les diverses fonctions les plus connues associées à des variables aléatoires d'importance majeure! Cependant avant de commencer par le premier exemple, il est important je pense de considérer le cas suivant qui dans un même graphique (certes un peu dense et manquant de couleurs) rassemble les 4 aspectes possibles d'une variable aléatoire normale (v.a. continue pour rappel):

```
R Console
> v<-rnorm(n=10000,m=0,sd=1)
> xrange<-c(-4,4);yrange<-c(0,1)
> hist(v)
> hist(v,freq=F,ylim=yrange,xlim=xrange)
> grid()
> curve(dnorm(x,mean=mean(v),sd=sd(v)),ylim=yrange,xlim=xrange,add=T) #densité
> curve(pnorm(x,mean=mean(v),sd=sd(v)),ylim=yrange,xlim=xrange,add=T) #distribution
> size<-25
> s<-sample(pnorm(v,mean=mean(v),sd=sd(v)),size)
> par(new=TRUE)
> plot(qnorm(s,mean = mean(v), sd = sd(v)),s,ylim=yrange,xlim=xrange,type="h")
> |
```



Pour la loi uniforme (continue) nous aurons en utilisant les commandes **runif( )** et **dunif( )** :

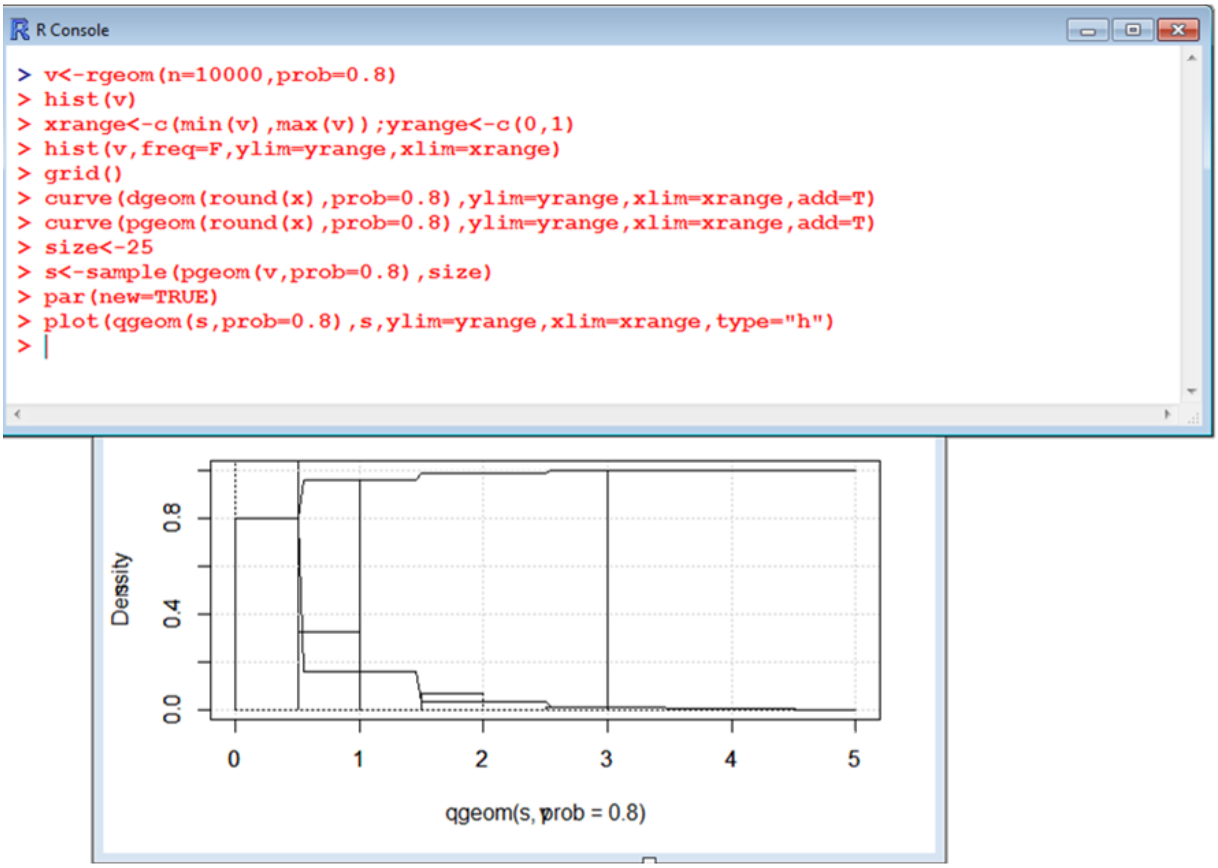


et là contrairement au support sur Minitab et SPSS nous allons faire les autres distributions classiques vu dans le cours théorique de niveau BAC puisque les commandes ne sont pas directement visibles.

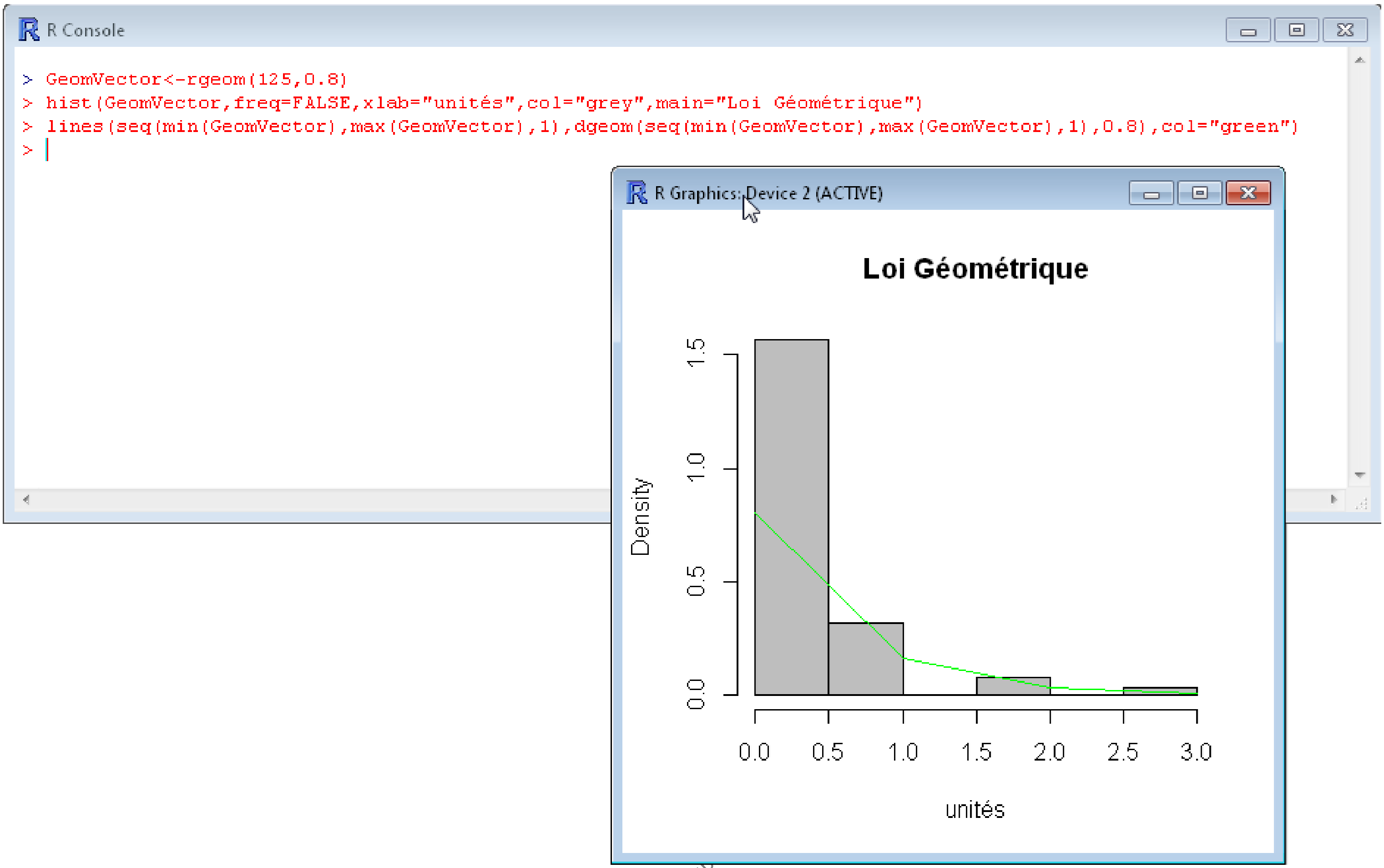
Exemple 161.: Loi Géométrique

R 3.0.2

Voici notre première v.a. discrète. Là aussi avant de poursuivre considérez le graphique suivant qui en un seul tenant rassemble les 4 aspect d'une variable aléatoire et remarquez surtout les arrondis (via l'utilisation de la fonction **round( )**) pour gérer le fait que le support concerne les valeurs entières:

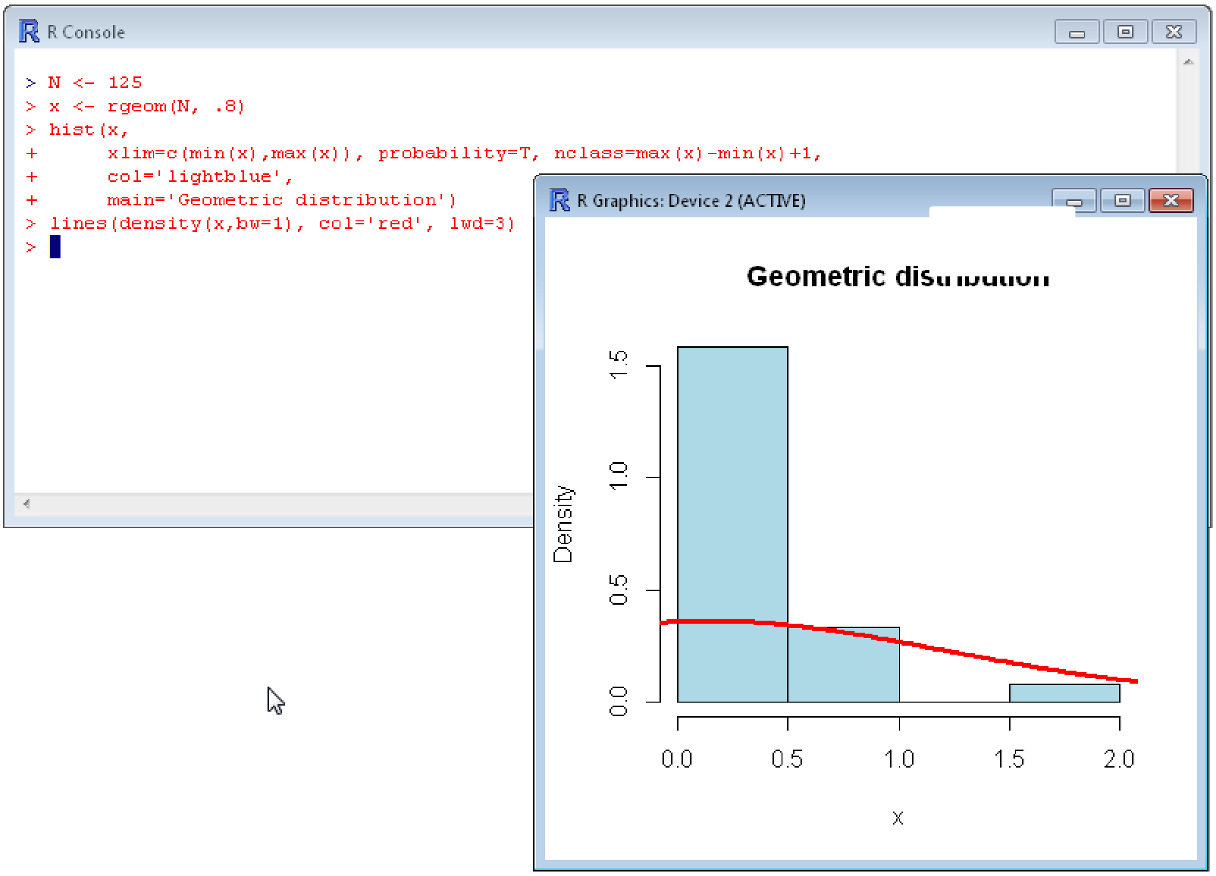


Nous avons vu dans le cours théorique que la loi géométrique (loi discrète pour rappel) était en réalité un simple cas particulier de la loi binomiale négative. Voyons comment tracer aussi cette dernière en utilisant les commandes **rgeom( )** et **dgeom( )** :



Nous pouvons voir qu'il y a priori un souci ici car une densité supérieure à l'unité cela n'est normalement pas possible et le regroupement des barres (intervalles) n'est pas des plus adéquats...

Il existe encore une fois une autre manière d'avoir un plus joli résultat (mais en réalité un peu incorrecte puisque cela peut donner l'impression que la fonction de densité est continue) et qui utilise la commande **lines( )** et **density( )** cette dernière étant une méthode d'estimation de fonction de densité par noyaux et par défaut utilisant une gaussienne:



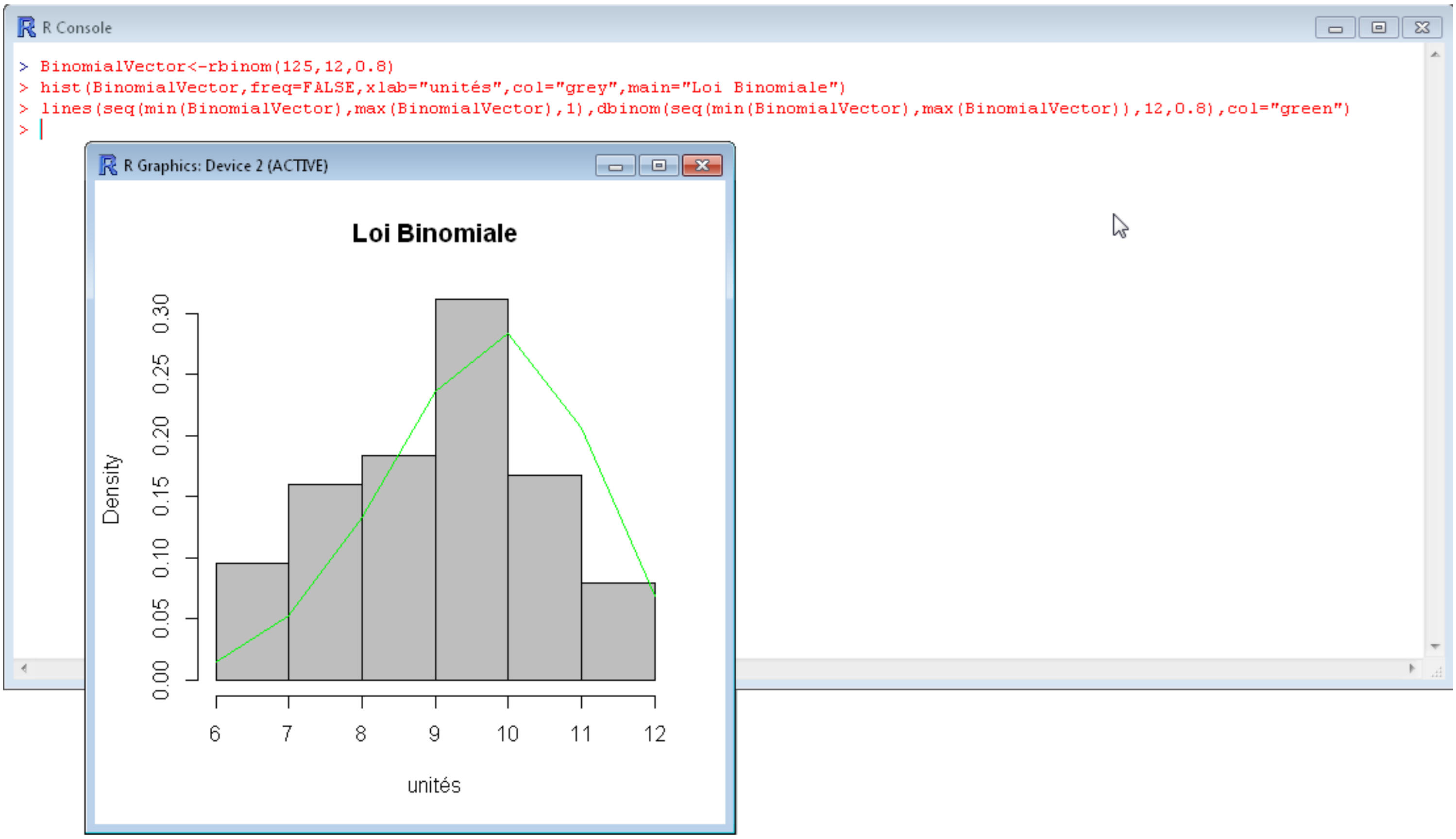
Mais là on voit que la ligne de densité devient vraiment n'importe quoi mais au moins la densité de l'histogramme est juste! Donc bref il y a un compromis à trouver (sinon on augmente le nombre de simulation de 125 à 10'000 par exemple...).

Puisque la loi binomiale tend vers une loi de Poisson quand le nombre d'essais devient infini et que la probabilité devient infiniment petite... il nous faut donc l'étudier!

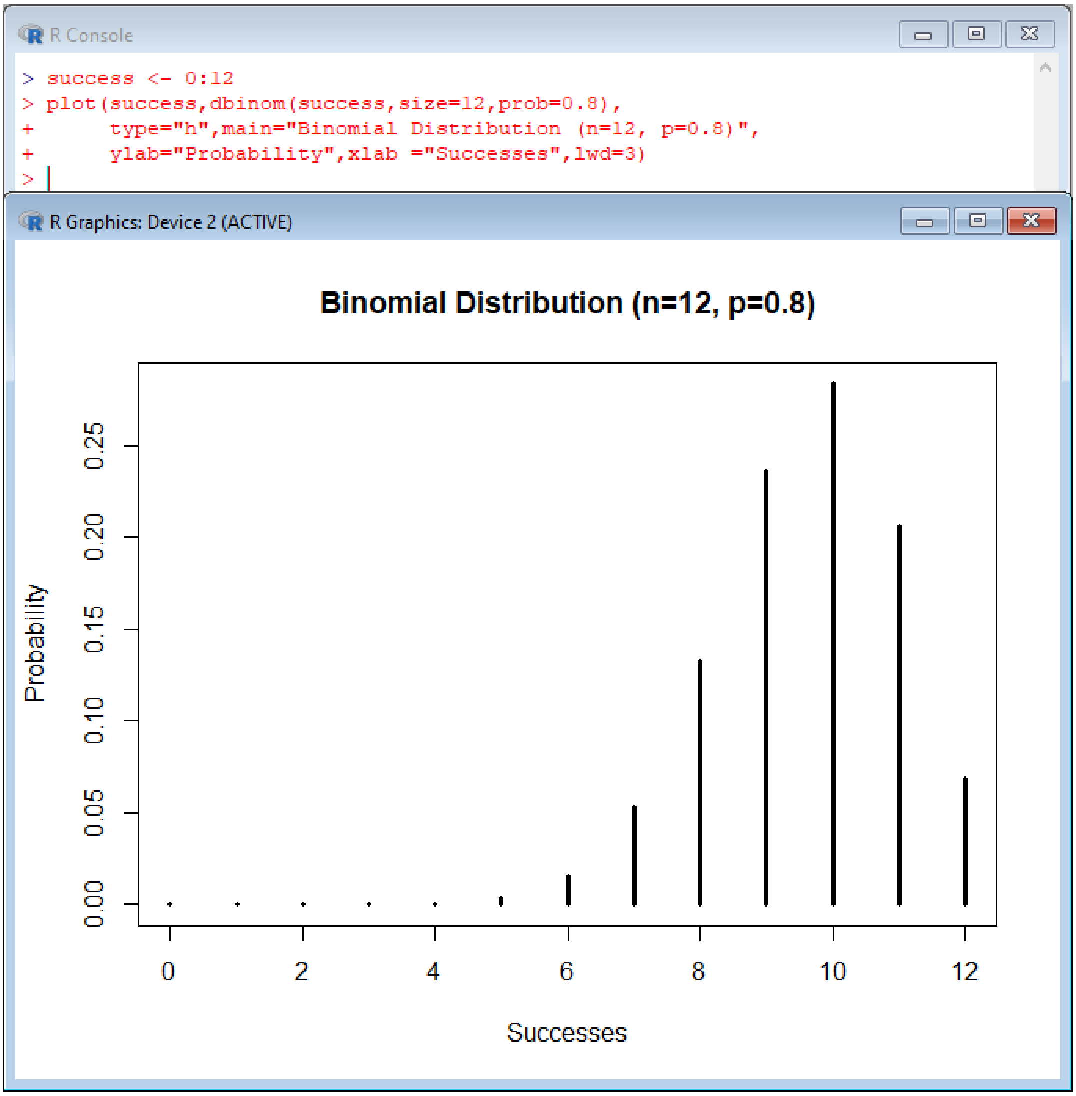
Exemple 162.: Loi Binomiale

R 3.0.2

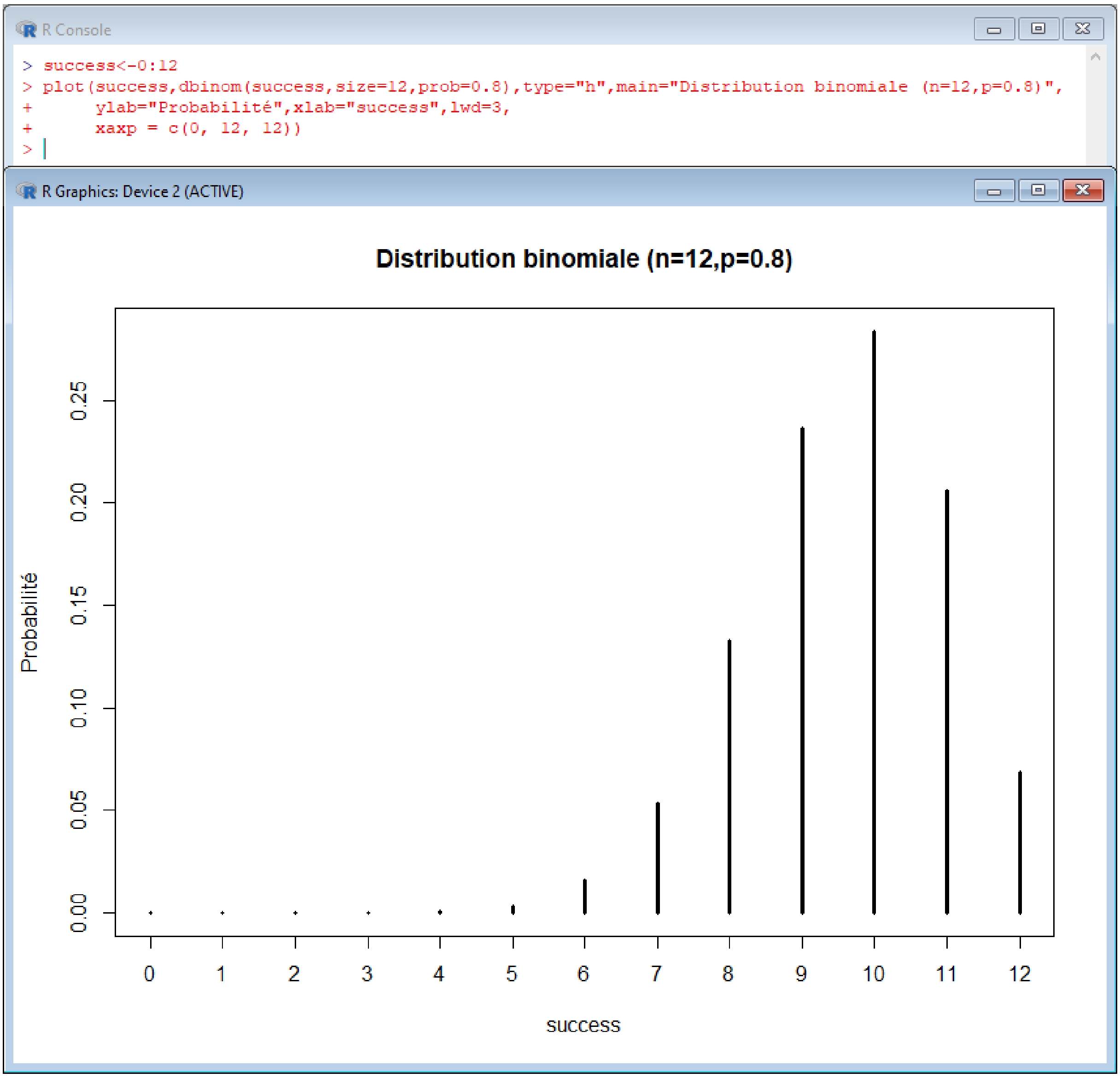
Pour la loi binomiale (loi discrète pour rappel) je n'ai à ce jour pas trouvé mieux que ce qu'il y a ci-dessous en utilisant les commandes **rbinom( )** et **dbinom( )** (c'est un peu particulier car c'est une loi bien évidemment discrète):



Dans la pratique on préfère représenter les lois discrètes par des diagrammes en bâtons (stick charts), voyons comment reproduire le diagramme ci-dessus adéquatement pour le coup:

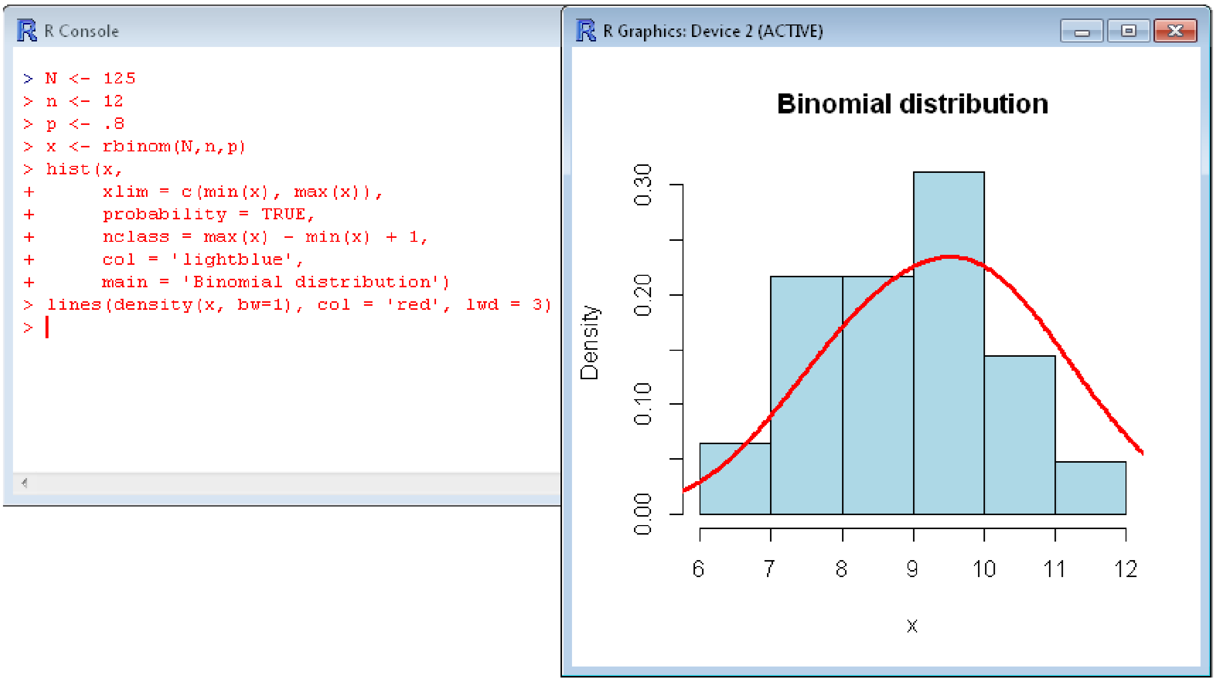


Nous pouvons aussi forcer l'affichage de tous les pas de l'axe des abscisses:



Malheureusement nous n'avons pas trouvé à ce jour comment contrôler la distance entre les barres…

Il existe une autre manière d'avoir un plus joli résultat (mais en réalité un peu incorrecte puisque cela peut donner l'impression que la fonction de densité est continue) et qui utilise la commande **lines( )** et **density( )** cette dernière étant une méthode d'estimation de fonction de densité par noyaux et par défaut utilisant une gaussienne:

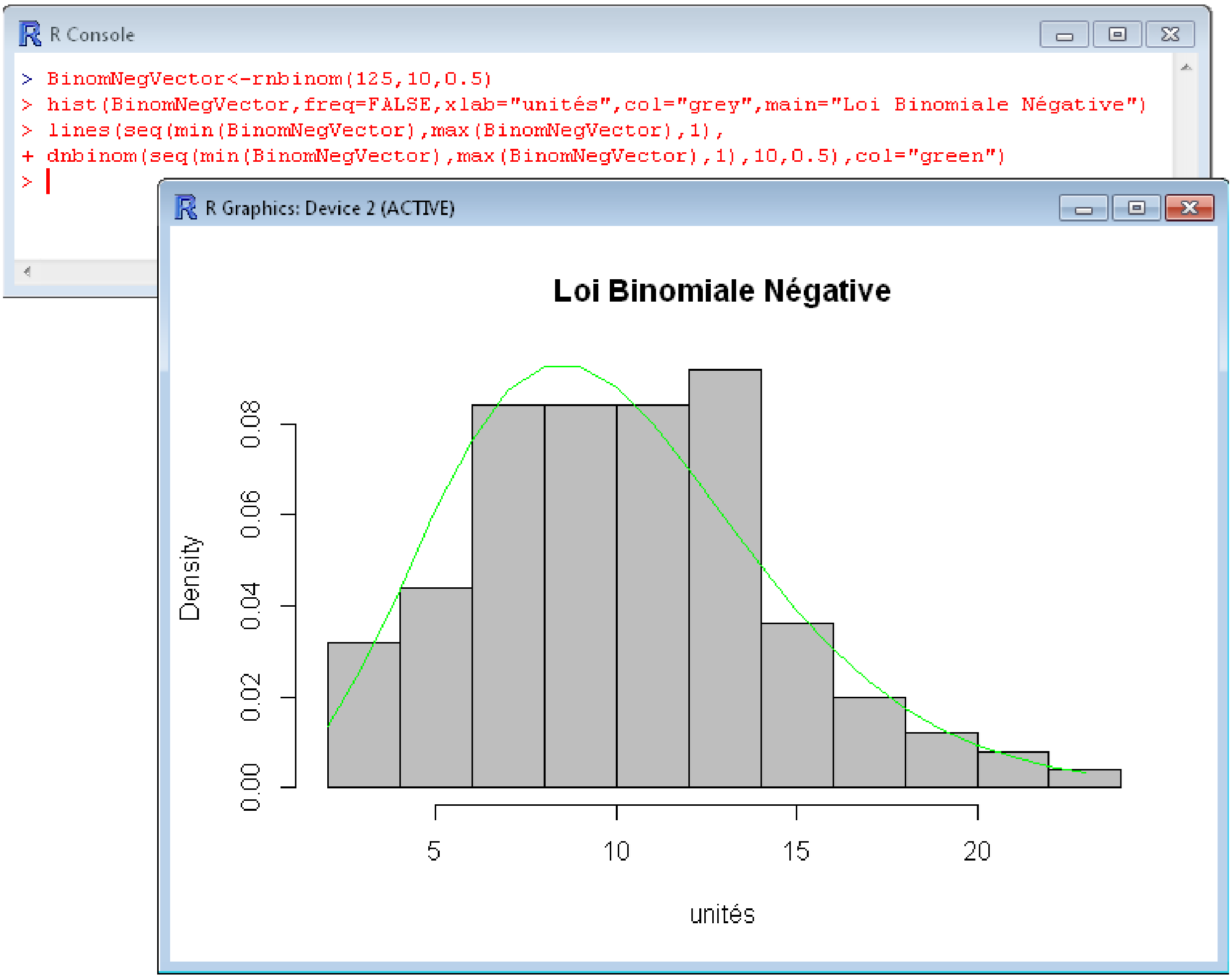


Mais là on voit que la ligne de densité n'est pas vraiment juste non plus mais au moins la densité de l'histogramme est juste! Donc bref il y a un compromis à trouver (sinon on augmente le nombre de simulation de 125 à 10'000 par exemple...).

Exemple 163.: Loi Binomiale négative

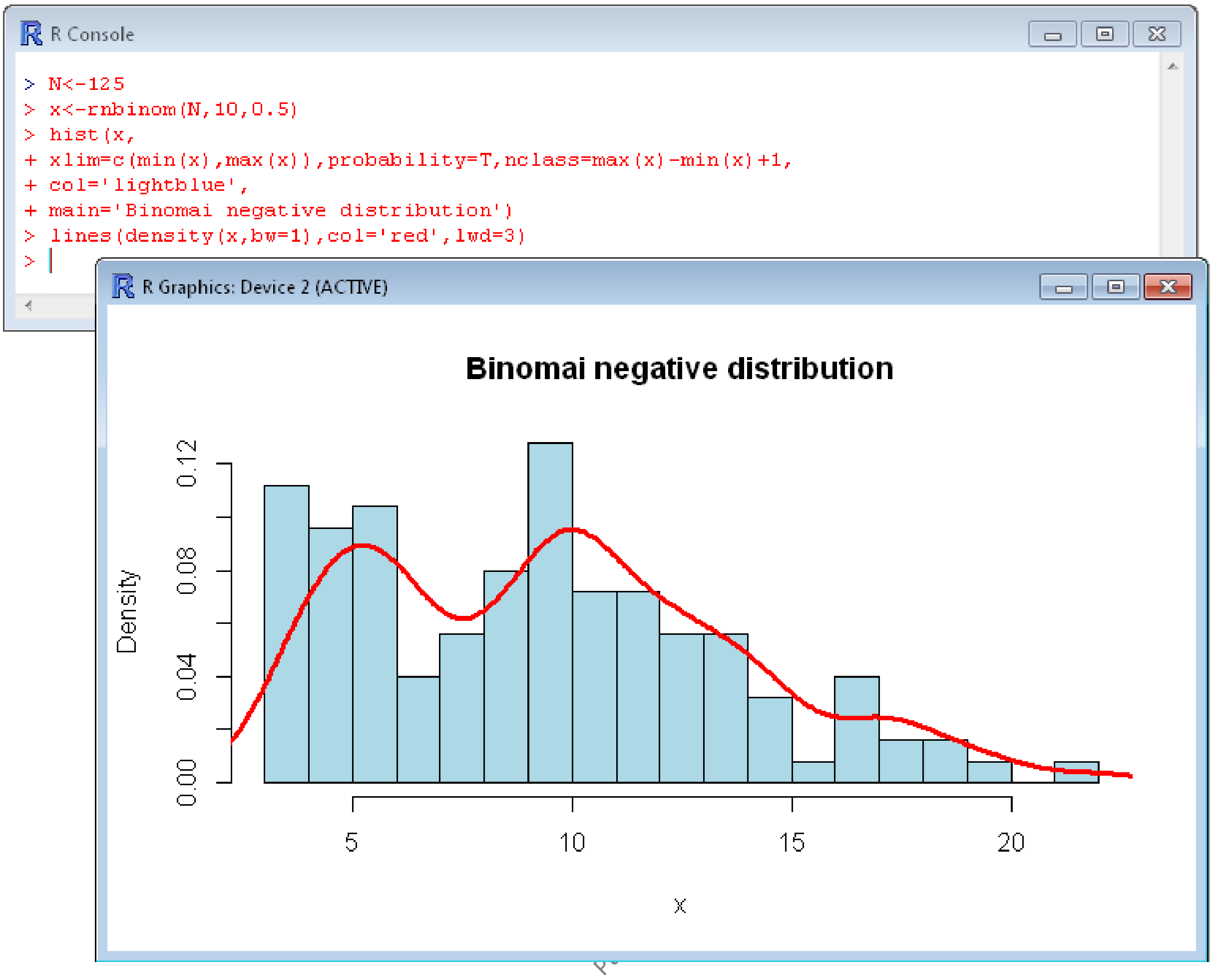
R 3.0.2

Nous avons vu dans le cours théorique que la loi géométrique était en réalité un simple cas particulier de la loi binomiale négative (loi discrète). Voyons comment tracer aussi cette dernière en utilisant les commandes **rnbinom( )** et **dnbinom( )** :



Nous pouvons voir qu'il y a priori un souci ici car une densité supérieure à l'unité cela n'est normalement pas possible et le regroupement des barres (intervalles) n'est pas des plus adéquats...

Il existe encore une fois une autre manière d'avoir un plus joli résultat (mais en réalité un peu incorrecte puisque cela peut donner l'impression que la fonction de densité est continue) et qui utilise la commande **lines( )** et **density( )** cette dernière étant une méthode d'estimation de fonction de densité par noyaux et par défaut utilisant une gaussienne:



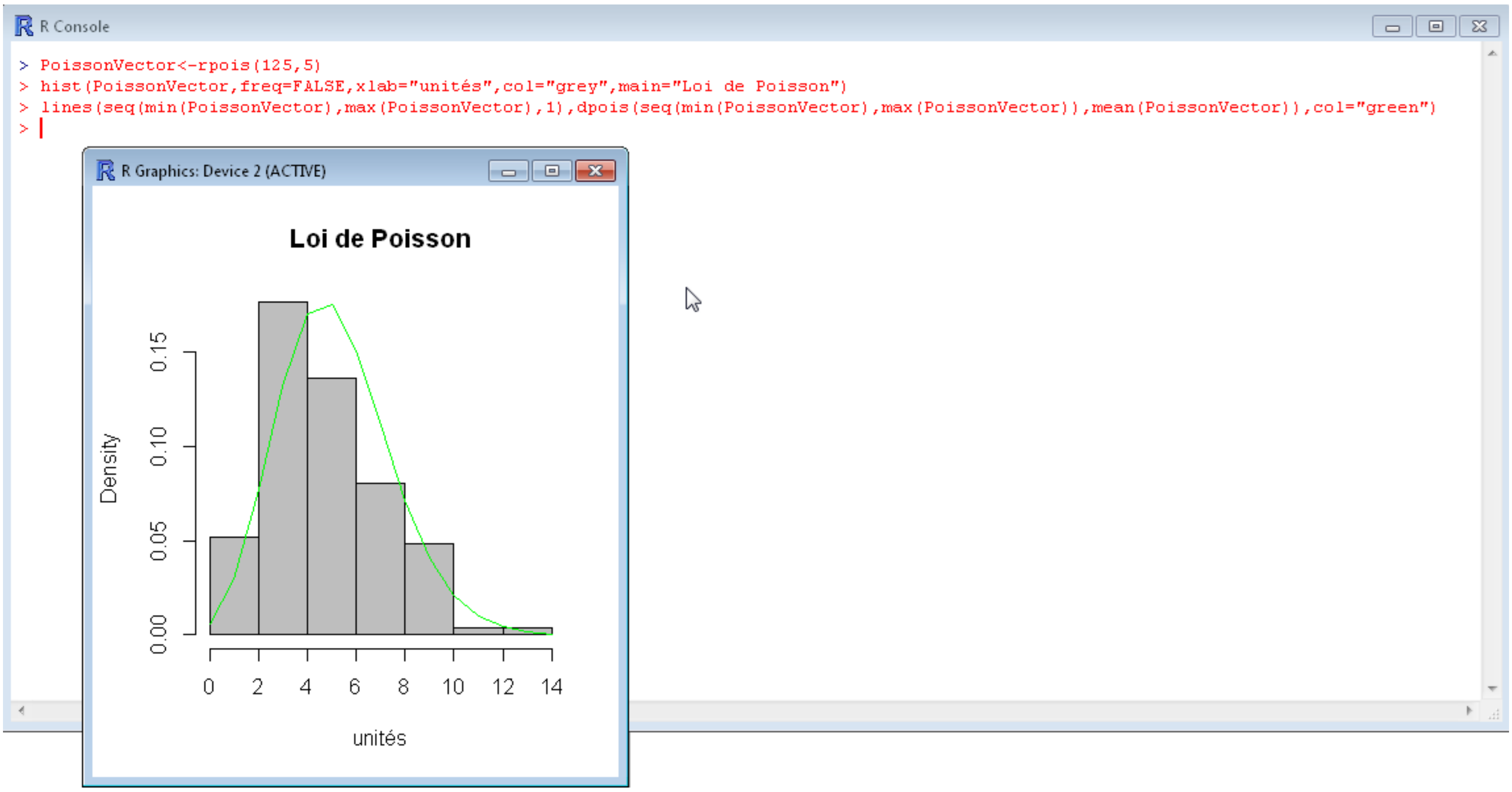
Mais là on voit que la ligne de densité devient vraiment n'importe quoi mais au moins la densité de l'histogramme est juste! Donc bref il y a un compromis à trouver (sinon on augmente le nombre de simulation de 125 à 10'000 par exemple...).

Puisque la loi binomiale tend vers une loi de Poisson quand le nombre d'essais devient infini et que la probabilité devient infiniment petite... il nous faut donc l'étudier!

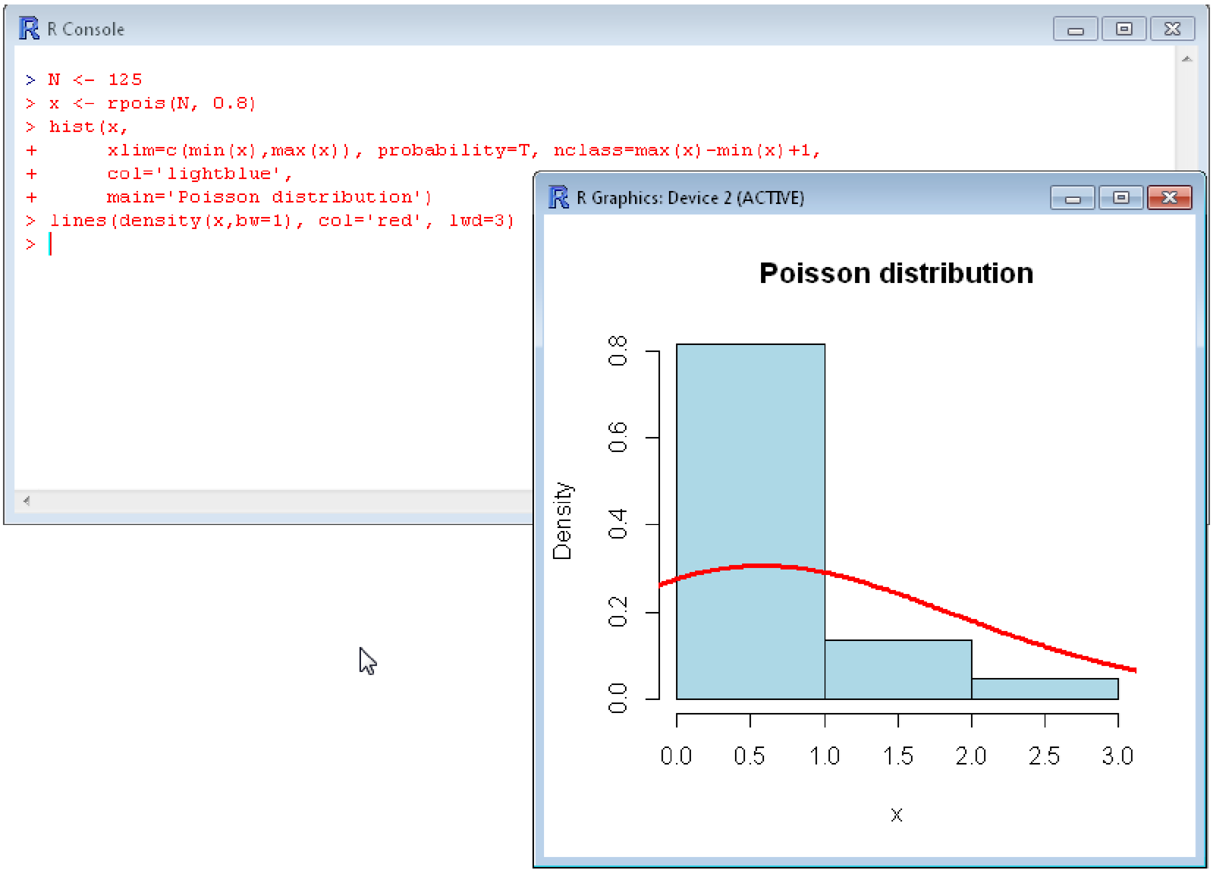
Exemple 164.: Loi de Poisson

R 3.0.2

Pour la loi Poisson (loi discrète pour rappel) je n'ai à ce jour pas trouvé mieux que ce qu'il y a ci-dessous et utilisant les commandes **rpois( )** et **dpois( )** (c'est un peu particulier car c'est une loi bien évidemment discrète):



Là aussi il existe encore une fois une autre manière d'avoir un plus joli résultat (mais en réalité un peu incorrecte puisque cela peut donner l'impression que la fonction de densité est continue) et qui utilise la commande **lines( )** et **density( )** cette dernière étant une méthode d'estimation de fonction de densité par noyaux et par défaut utilisant une gaussienne:



Mais là on voit que la ligne de densité devient vraiment n'importe quoi mais au moins la densité de l'histogramme est juste! Donc bref il y a un compromis à trouver (sinon on augmente le nombre de simulation de 125 à 10'000 par exemple...).

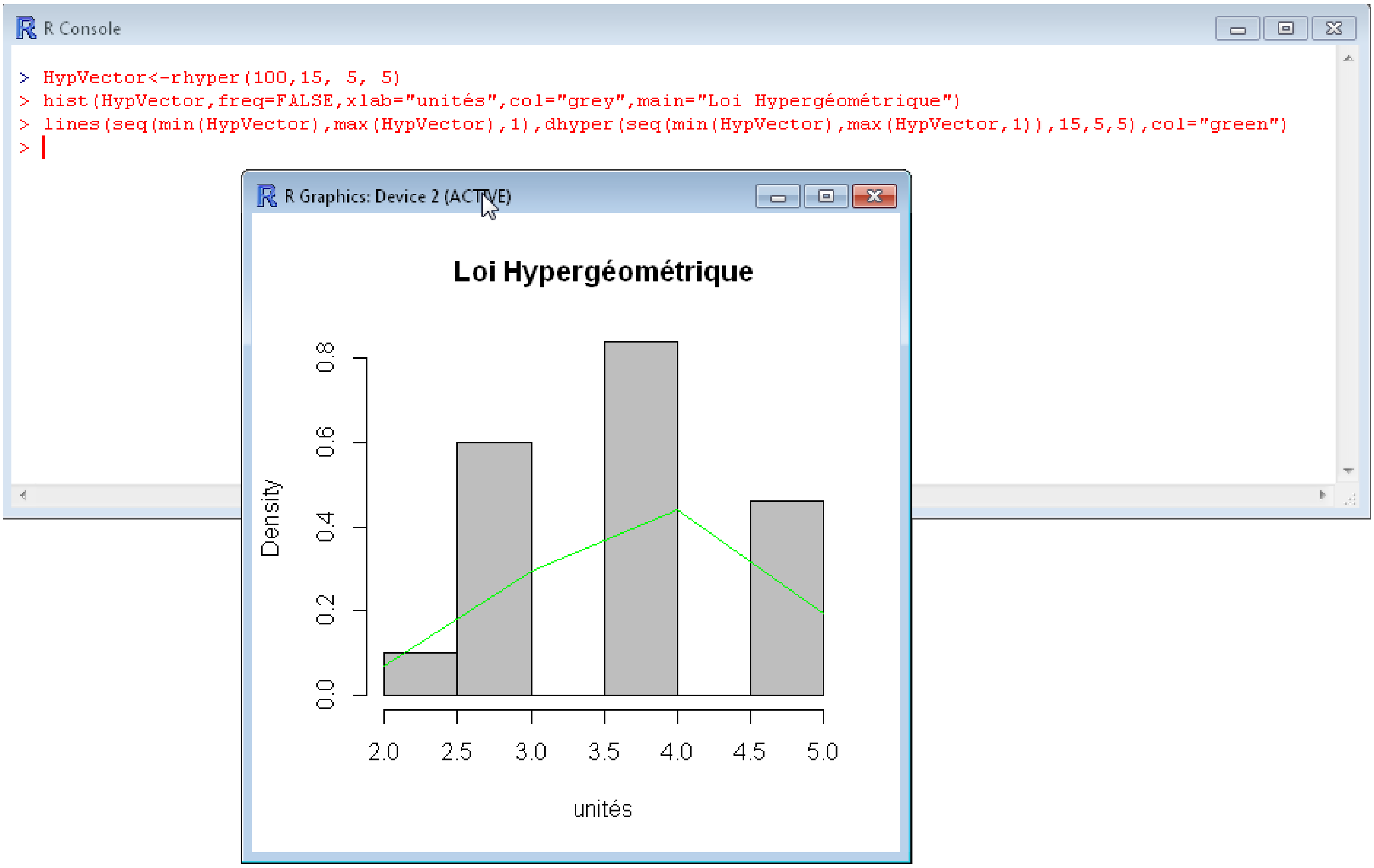
La distribution de Poisson tend vers une loi Normale dont l'écart-type est égale à l'espérance et que l'espérance (in extenso l'écart-type) deviennent très grand. Mais nous avons déjà étudié la loi Normale comme tout premier exemple, nous n'allons donc pas y revenir.

Nous avons démontré aussi dans le cours théorique que la loi hypergéométrique était une généralisation de la loi binomiale (et donc par extension une généralisation de la loi géométrique). Voyons comment l'obtenir.

Exemple 165.: Loi Hypergéométrique

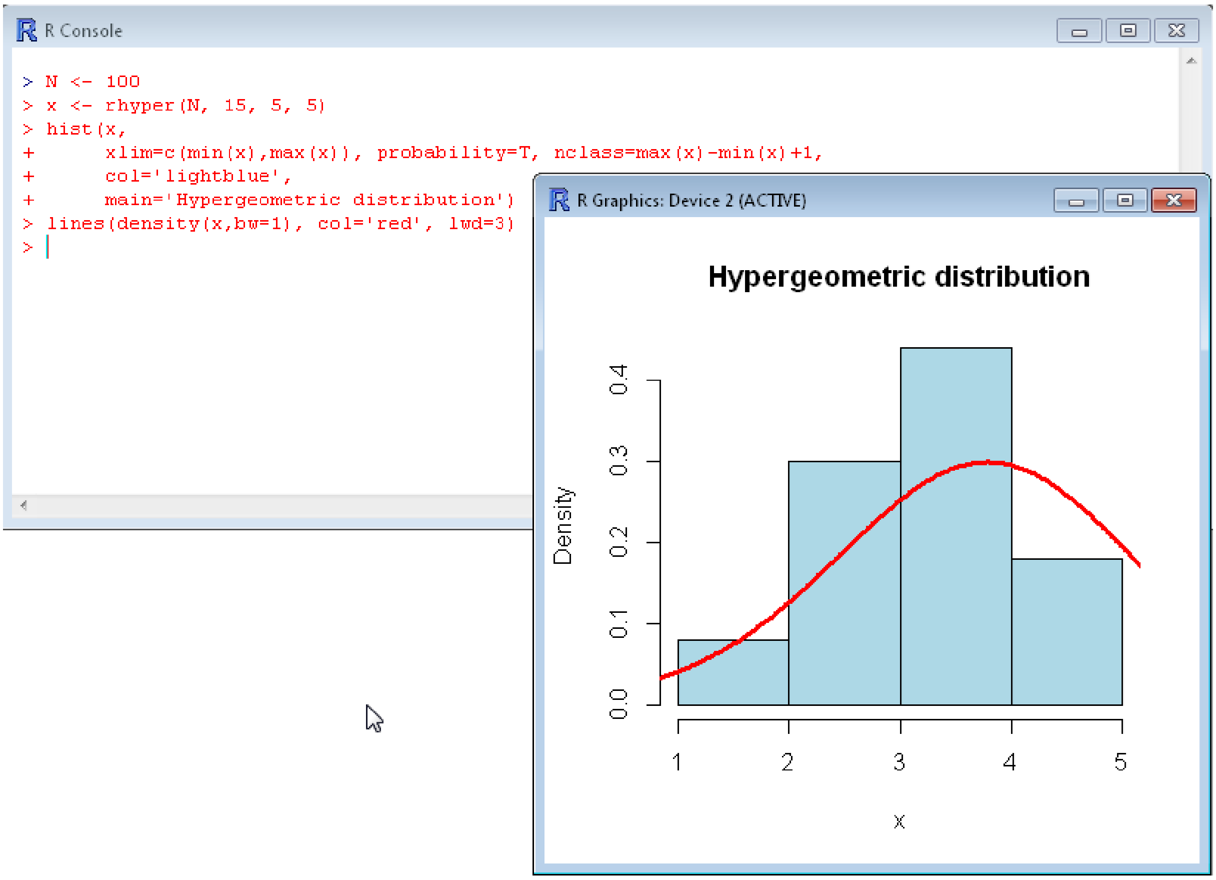
R 3.0.2

Donc pour la loi géométrique (loi discrète pour rappel) en se basant toujours sur la même approche que les cas précédents mais avec les commandes **rhyper( )** et **dhyper( )** , nous avons:



Nous pouvons voir encore une fois, qu'il y a encore une fois un souci avec le regroupement des barres (intervalles) qui n'est pas des plus adéquats....

Il existe (encore une fois...) une autre manière d'avoir un plus joli résultat (mais en réalité un peu incorrecte puisque cela peut donner l'impression que la fonction de densité est continue) et qui utilise la commande **lines( )** et **density( )** cette dernière étant une méthode d'estimation de fonction de densité par noyaux et par défaut utilisant une gaussienne:



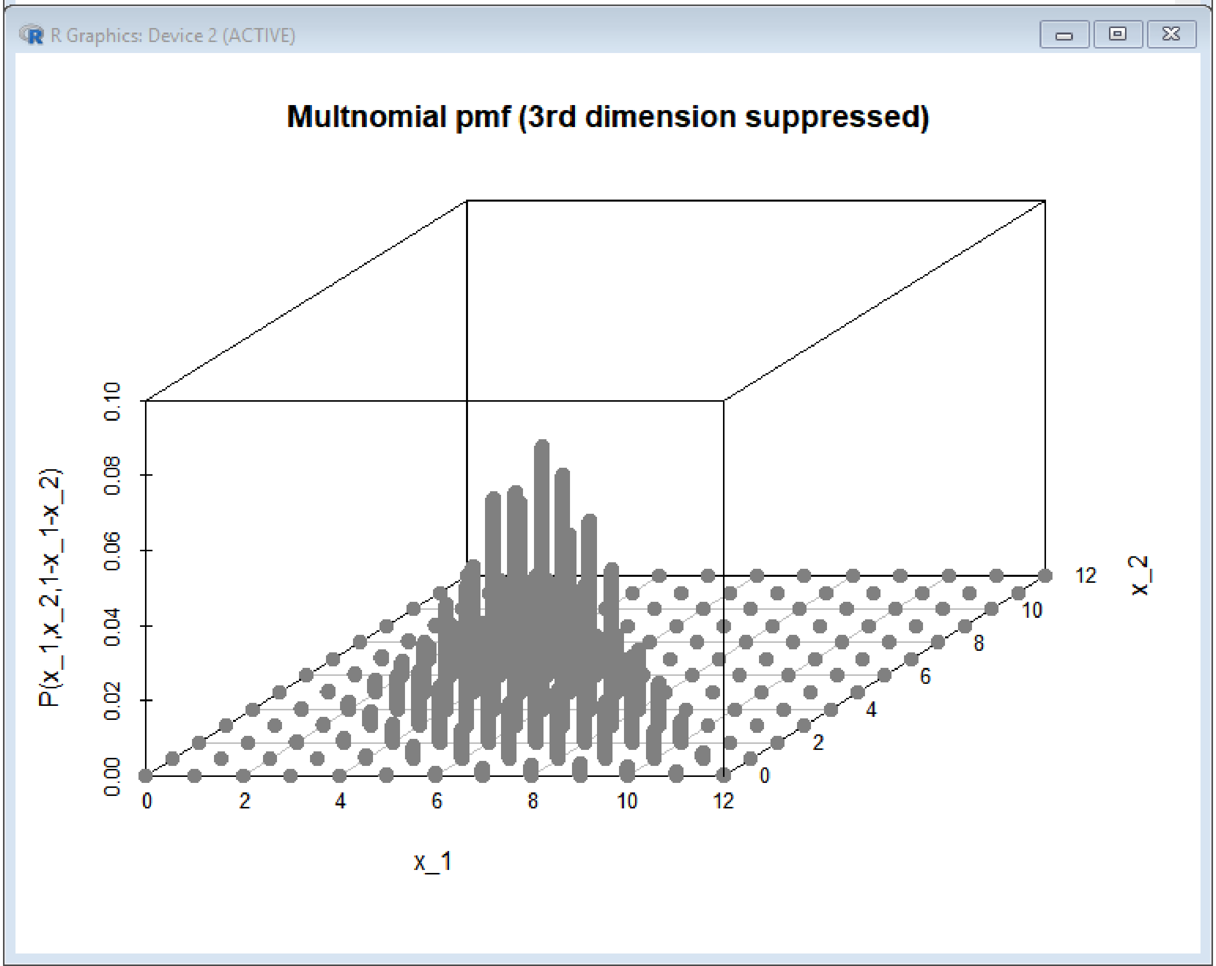
Bref c'est beaucoup mieux quand même!

Exemple 166.: Loi Multinomiale

R 3.0.2

Nous savons que la loi multinomiale est importante dans plusieurs domaines comme nous l'avons vu dans le cours théorique (test de McNemar et régression logistique multinomiale). Voyons comme faire un graphique de cette dernière en utilisant la fonction **dmultinom( )** ainsi que la fonction **scatterplot3d( )** du package **scatterplot3d** pour l'aspect visuel. Vous allez voir que ce n'est pas trivial:

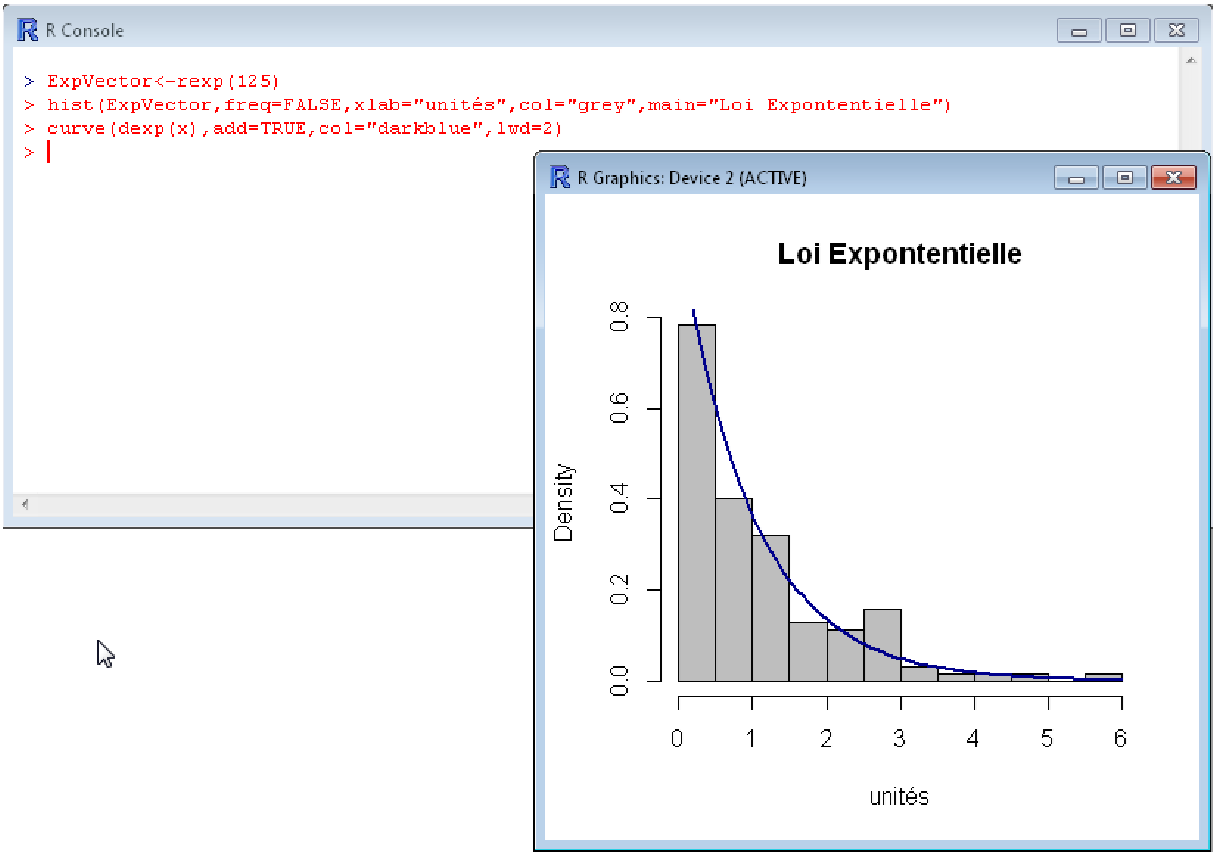
```
R Console
> library("scatterplot3d")
> R = expand.grid(0:12, 0:12)
> R[, 3] = 12 - R[, 1] - R[, 2]
> names(R) = c("x1", "x2", "x3")
> for (i in 1:nrow(R)) {
+     v = R[i, 1:3]
+     if (sum(v) == 12 & all(v >= 0)) {
+         R$p[i] = dmultinom(v, prob = c(0.5, 0.3, 0.2))
+     } else R$p[i] = 0
+ }
> scatterplot3d(x = R$x1, y = R$x2, z = R$p, type = "h",
+     lwd = 10, pch = " ", xlab = "x_1", ylab = "x_2",
+     zlab = "P(x_1,x_2,1-x_1-x_2)", color = grey(1/2))
> title("Multnomial pmf (3rd dimension suppressed)")
>
```



Exemple 167.: Loi Exponentielle

R 3.0.2

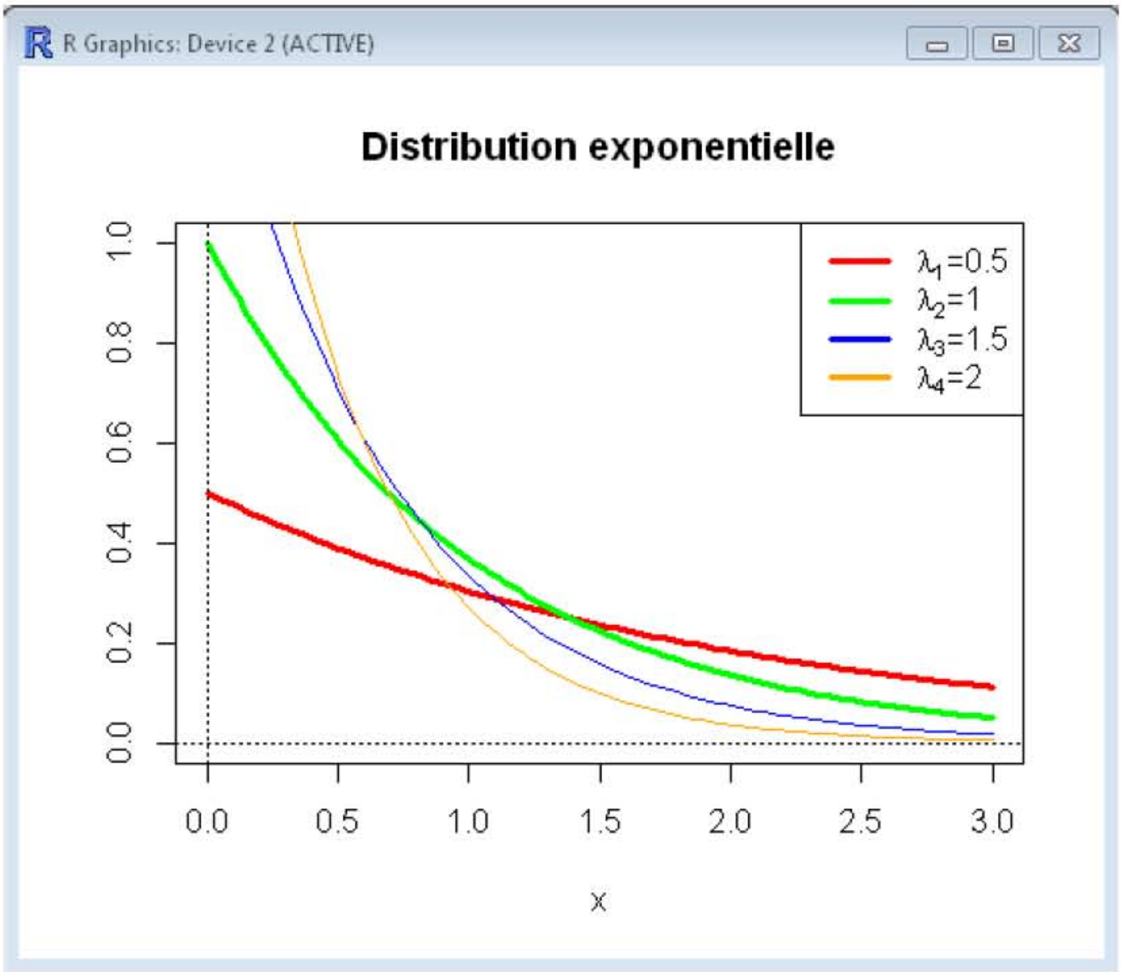
La loi exponentielle (loi continue pour rappel) découle de plusieurs manières de la loi de Poisson comme nous l'avons vu dans le cours théorique. Voyons donc comme l'obtenir directement et proprement en utilisant les commandes **rexp( )** et **dexp( )** :



ou encore le grand classique:

```
R Console
> x<-seq(0,5,0.01)
> curve(dexp(x,0.5),xlim=c(0,3),ylim=c(0,1),col="red",lwd=3,ylab="")
> curve(dexp(x,1),col="green",lwd=3,add=T)
> curve(dexp(x,1.5),col="blue",add=T)
> curve(dexp(x,2),col="orange",add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(lambda[1],"=0.5")),
+ expression(paste(lambda[2],"=1")),
+ expression(paste(lambda[3],"=1.5")),
+ expression(paste(lambda[4],"=2"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution exponentielle")
>
```

qui donne:


R Graphics: Device 2 (ACTIVE)
Distribution exponentielle
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
λ1=0.5
λ2=1
λ3=1.5
λ4=2

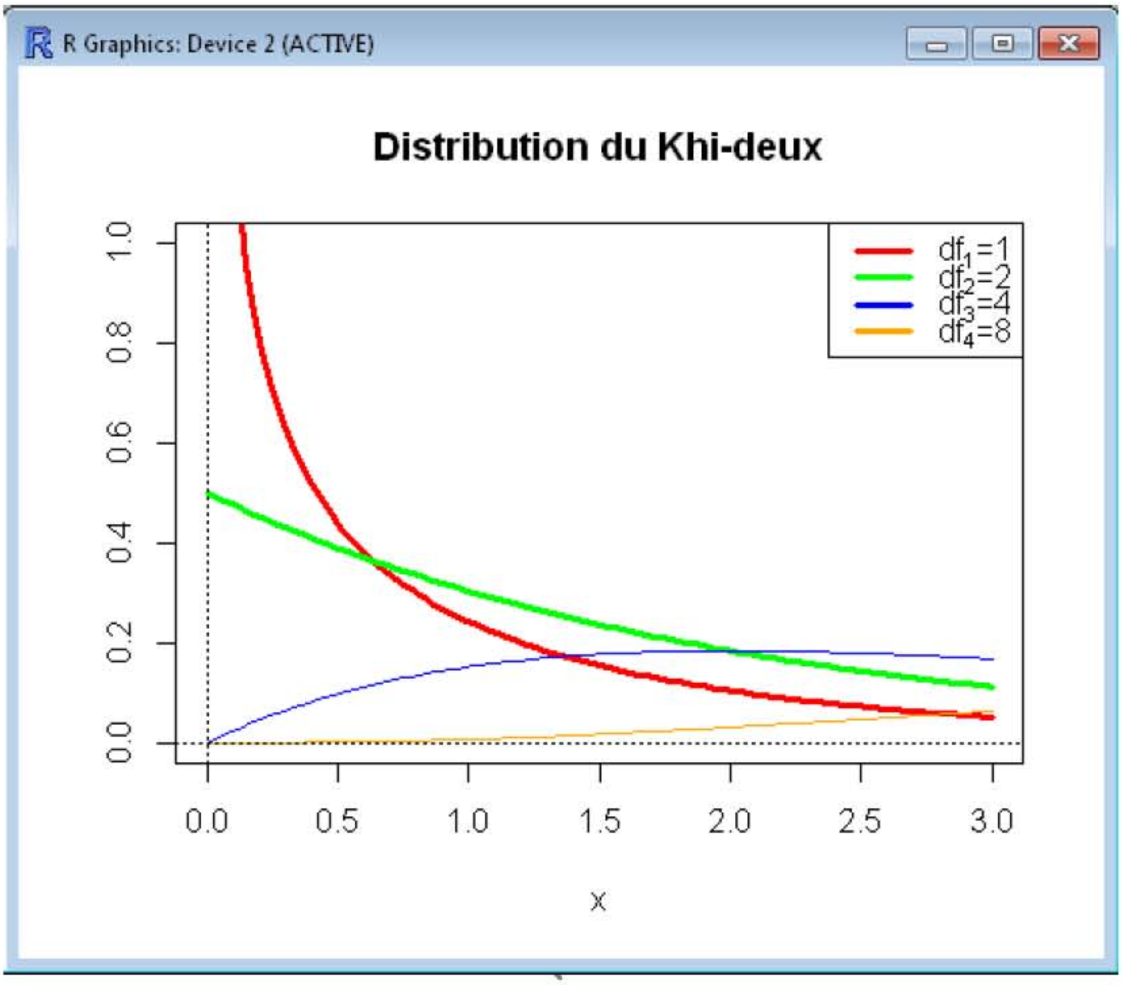
Exemple 168.: Loi du khi-2

R 3.0.2

La loi du khi-2 (loi continue pour rappel) découle de la loi Gamma comme nous l'avons vu dans le cours théorique. Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dchisq( )** et on arrête de faire les histogrammes car on a compris le principe!:

```
R Console
> x<-seq(0,5,0.01)
> curve(dchisq(x,1),xlim=c(0,3),ylim=c(0,1),col="red",lwd=3,ylab="")
> curve(dchisq(x,2),col="green",lwd=3,add=T)
> curve(dchisq(x,4),col="blue",add=T)
> curve(dchisq(x,8),col="orange",add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(df[1],"=1")),
+ expression(paste(df[2],"=2")),
+ expression(paste(df[3],"=4")),
+ expression(paste(df[4],"=8"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution du Khi-deux")
>
```

qui donne:


R Graphics: Device 2 (ACTIVE)
Distribution du Khi-deux
df1=1
df2=2
df3=4
df4=8
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

Exemple 169.: Loi de Student

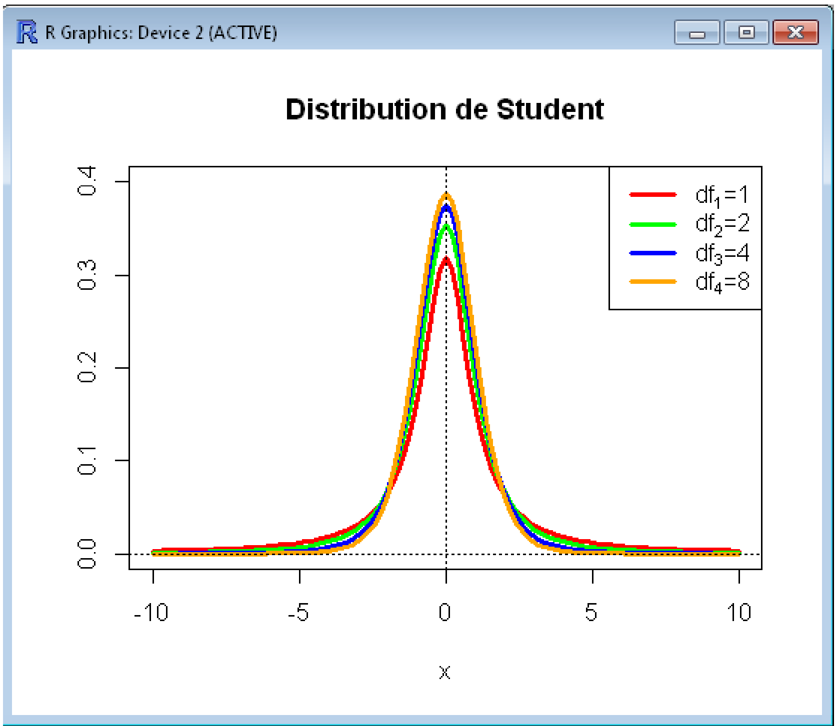
R 3.0.2

La loi de Student (loi continue pour rappel) découle comme nous l'avons démontré dans le cours théorique du rapport entre une variable aléatoire normalement distribuée et la racine carrée d'un variable aléatoire distribuée selon une loi du khi-2. Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dt( )** :

R Console

```
> x<-seq(-3,3,0.01)
> curve(dt(x,1),xlim=c(-10,10),ylim=c(0,0.4),col="red",lwd=3,ylab="")
> curve(dt(x,2),col="green",lwd=3,add=T)
> curve(dt(x,4),col="blue",lwd=3,add=T)
> curve(dt(x,8),col="orange",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(df[1],"=1")),
+ expression(paste(df[2],"=2")),
+ expression(paste(df[3],"=4")),
+ expression(paste(df[4],"=8"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution de Student")
> 
```

Ce qui nous donne:



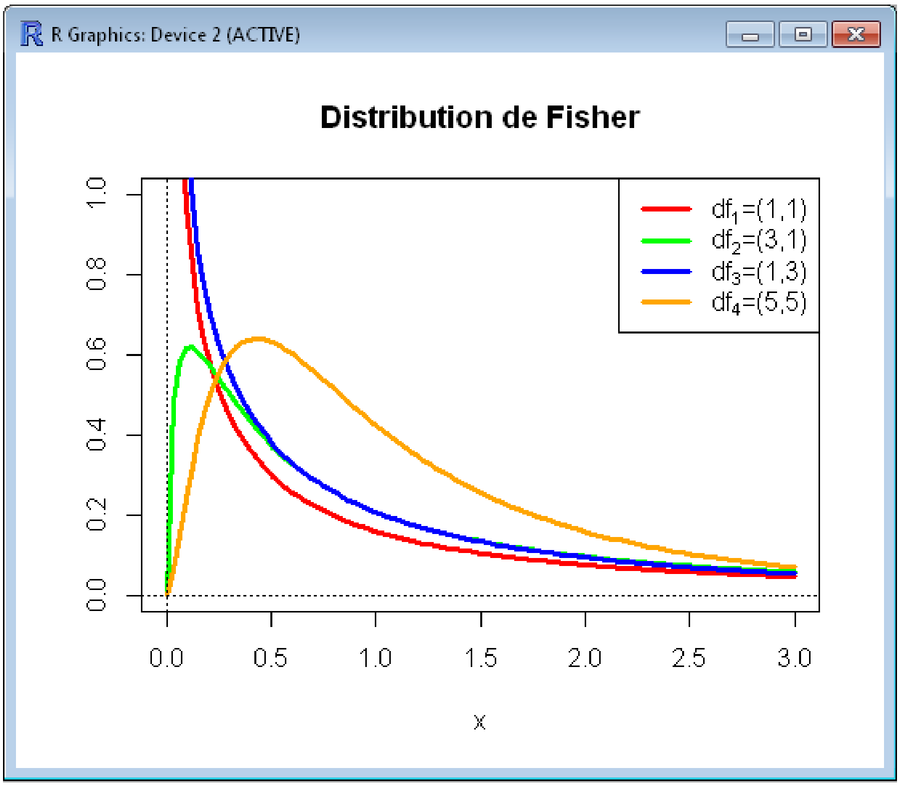
Exemple 170.: Loi de Fisher

R 3.0.2

La loi de Fisher (loi continue pour rappel) découle comme nous l'avons démontré dans le cours théorique du rapport entre deux variables aléatoires indépendantes distribuée selon une loi du khi-2. Voyons donc comme l'obtenir directement et proprement en utilisant la commande **df( )** :

```
R Console
> x<-seq(0,2,0.01)
> curve(df(x,1,1),xlim=c(0,3),ylim=c(0,1),col="red",lwd=3,ylab="")
> curve(df(x,3,1),col="green",lwd=3,add=T)
> curve(df(x,1,3),col="blue",lwd=3,add=T)
> curve(df(x,5,5),col="orange",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(df[1],"=(1,1)")),
+ expression(paste(df[2],"=(3,1)")),
+ expression(paste(df[3],"=(1,3)")),
+ expression(paste(df[4],"=(5,5)"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution de Fisher")
> 
```

Ce qui nous donne:



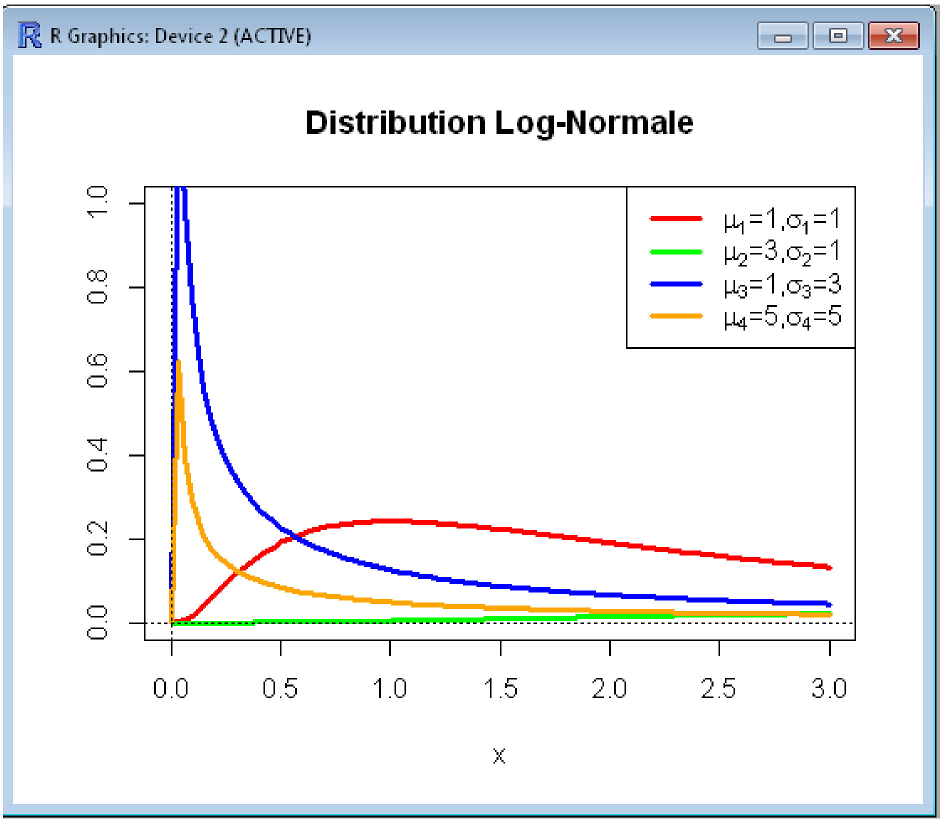
Exemple 171.: Loi Log-Normale

R 3.0.2

La loi Log-Normale (loi continue pour rappel) découle comme nous l'avons démontré dans le cours théorique du logarithme d'une variable aléatoire Normale (cas très courant en finance!). Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dlnorm( )** :

```
> x<-seq(0,2,0.01)
> curve(dlnorm(x,1,1),xlim=c(0,3),ylim=c(0,1),col="red",lwd=3,ylab="")
> curve(dlnorm(x,3,1),col="green",lwd=3,add=T)
> curve(dlnorm(x,1,3),col="blue",lwd=3,add=T)
> curve(dlnorm(x,5,5),col="orange",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(mu[1],"=1,",sigma[1],"=1")),
+ expression(paste(mu[2],"=3,",sigma[2],"=1")),
+ expression(paste(mu[3],"=1,",sigma[3],"=3")),
+ expression(paste(mu[4],"=5,",sigma[4],"=5"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution Log-Normale")
>
```


ce qui donne:



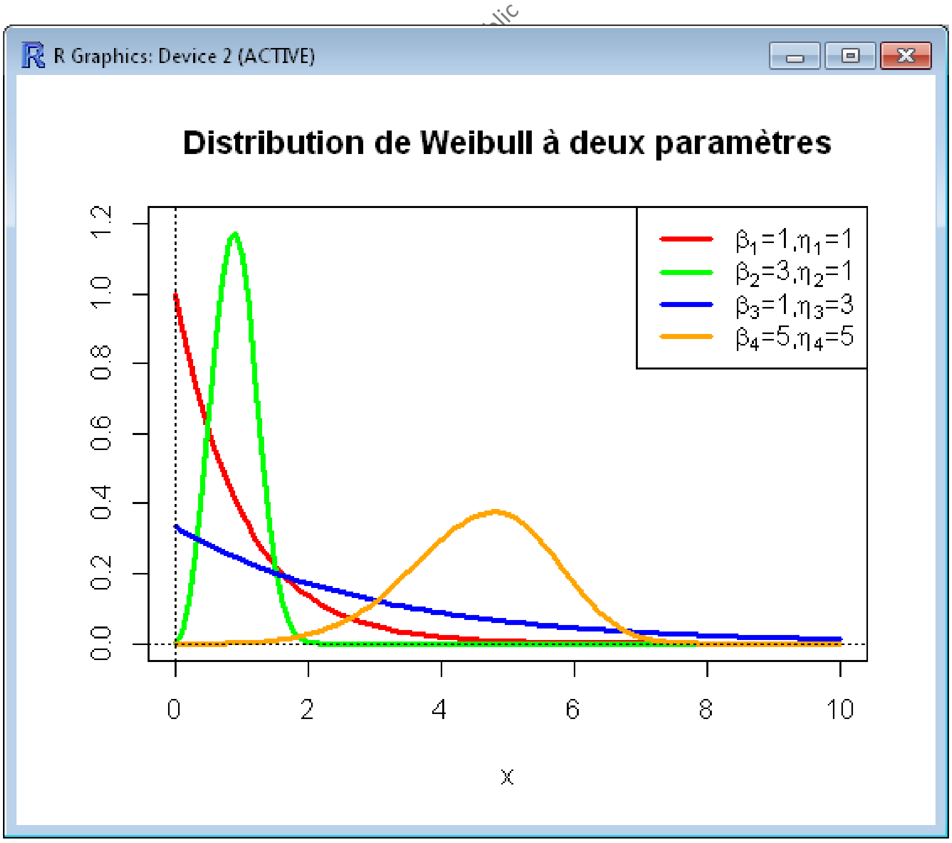
Exemple 172.: Loi de Weibull

R 3.0.2

La loi de Weibull (loi continue pour rappel) serait donc empirique comme nous l'avons mentionné dans le cours théorique (cas très courant en ingénierie!). Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dweibull( )** :

```
> x<-seq(0,2,0.01)
> curve(dweibull(x,shape=1,scale=1),xlim=c(0,10),ylim=c(0,1.2),col="red",lwd=3,ylab="")
> curve(dweibull(x,shape=3,scale=1),col="green",lwd=3,add=T)
> curve(dweibull(x,shape=1,scale=3),col="blue",lwd=3,add=T)
> curve(dweibull(x,shape=5,scale=5),col="orange",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(beta[1],"=1,",eta[1],"=1")),
+ expression(paste(beta[2],"=3,",eta[2],"=1")),
+ expression(paste(beta[3],"=1,",eta[3],"=3")),
+ expression(paste(beta[4],"=5,",eta[4],"=5"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution de Weibull à deux paramètres")
>
```

Ce qui nous donne:



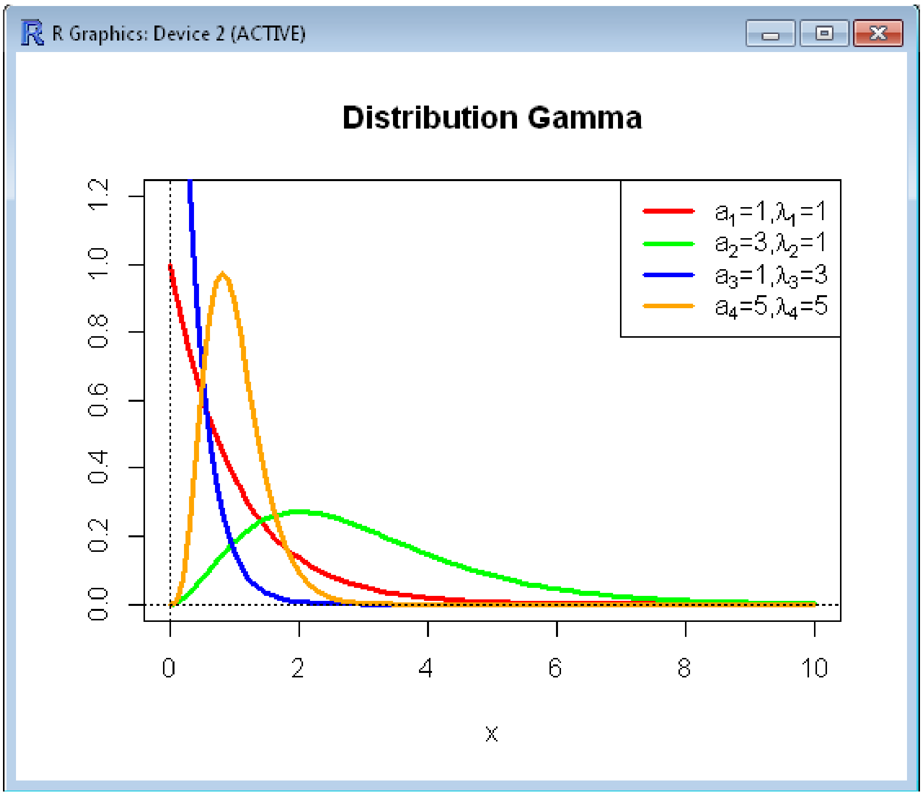
Exemple 173.: Loi Gamma

R 3.0.2

La loi Gamma (loi continue pour rappel) serait donc empirique comme nous l'avons mentionné dans le cours théorique (cas très courant dans les théories statistiques en général!). Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dgamma( )** :

```
> x<-seq(0,5,0.01)
> curve(dgamma(x,shape=1,rate=1),xlim=c(0,10),ylim=c(0,1.2),col="red",lwd=3,ylab="")
> curve(dgamma(x,shape=3,rate=1),col="green",lwd=3,add=T)
> curve(dgamma(x,shape=1,rate=3),col="blue",lwd=3,add=T)
> curve(dgamma(x,shape=5,rate=5),col="orange",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(a[1],"=1,",lambda[1],"=1")),
+ expression(paste(a[2],"=3,",lambda[2],"=1")),
+ expression(paste(a[3],"=1,",lambda[3],"=3")),
+ expression(paste(a[4],"=5,",lambda[4],"=5"))),
+ lwd=3,col=c("red","green","blue","orange"))
> title(main="Distribution Gamma")
>
```

Ce qui nous donne:



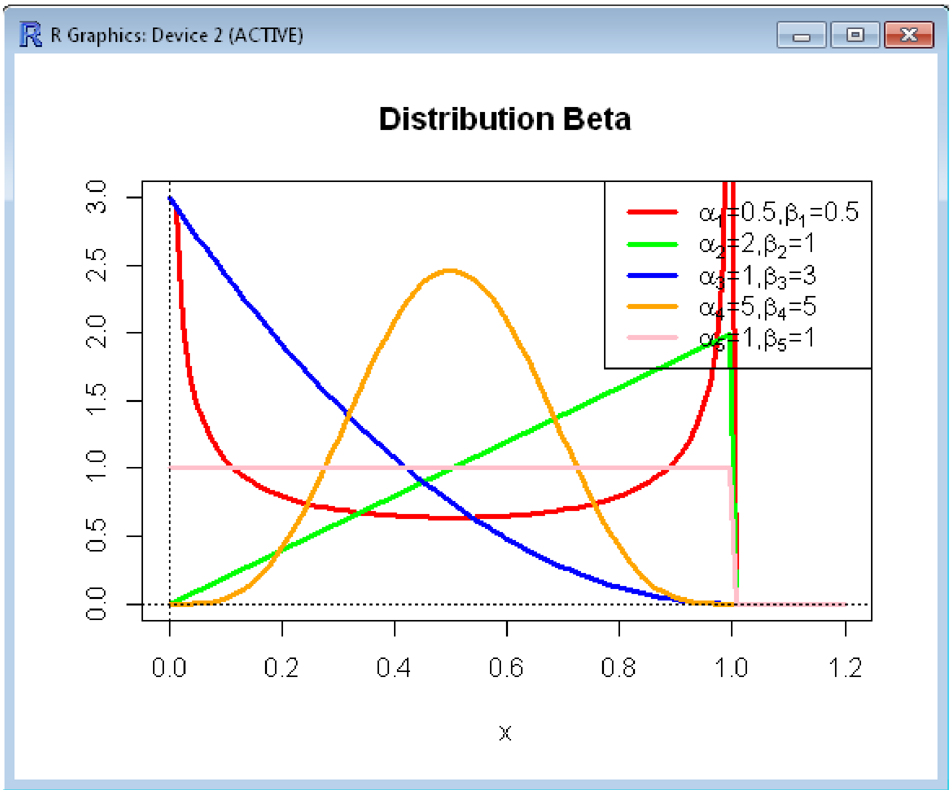
Exemple 174.: Loi Beta

R 3.0.2

La loi Beta (loi continue pour rappel) serait donc à peu près empirique comme nous l'avons mentionné dans le cours théorique (cas très courant dans la gestion de projets et certains modèles d'analyse bayésienne!). Voyons donc comme l'obtenir directement et proprement en utilisant la commande **dbeta( )** :

```
> x<-seq(0,5,0.01)
> curve(dbeta(x,shape1=0.5,shape2=0.5),xlim=c(0,1.2),ylim=c(0,3),col="red",lwd=3,ylab="")
> curve(dbeta(x,shape1=2,shape2=1),col="green",lwd=3,add=T)
> curve(dbeta(x,shape1=1,shape2=3),col="blue",lwd=3,add=T)
> curve(dbeta(x,shape1=5,shape2=5),col="orange",lwd=3,add=T)
> curve(dbeta(x,shape1=1,shape2=1),col="pink",lwd=3,add=T)
> abline(h=0,lty=3);abline(v=0,lty=3)
> legend(par("usr")[2],par("usr")[4],xjust=1,
+ c(
+ expression(paste(alpha[1],"=0.5,",beta[1],"=0.5")),
+ expression(paste(alpha[2],"=2,",beta[2],"=1")),
+ expression(paste(alpha[3],"=1,",beta[3],"=3")),
+ expression(paste(alpha[4],"=5,",beta[4],"=5")),
+ expression(paste(alpha[5],"=1,",beta[5],"=1"))),
+ lwd=3,col=c("red","green","blue","orange","pink"))
> title(main="Distribution Beta")
>
```


Ce qui nous donne:



Maintenant voyons un cas intéressant…

Rappelons que dans le cours théorique, nous avons démontré pour le test A/B Bayésien que:

$$\Pr(p_B > p_A) = \sum_{i=0}^{\alpha_B - 1} \frac{B(\alpha_A + i, \beta_B + \beta_A)}{(\beta_B + i) B(1 + i, \beta_B) B(\alpha_A, \beta_A)}$$

Voyons cependant, par curiosité, d'abord la comparaison non-bayésienne avec **prop.test( )**:

```
> prop.test(n=c(1000,1000), x=c(100,78), correct=F)

        2-sample test for equality of proportions without continuity correction

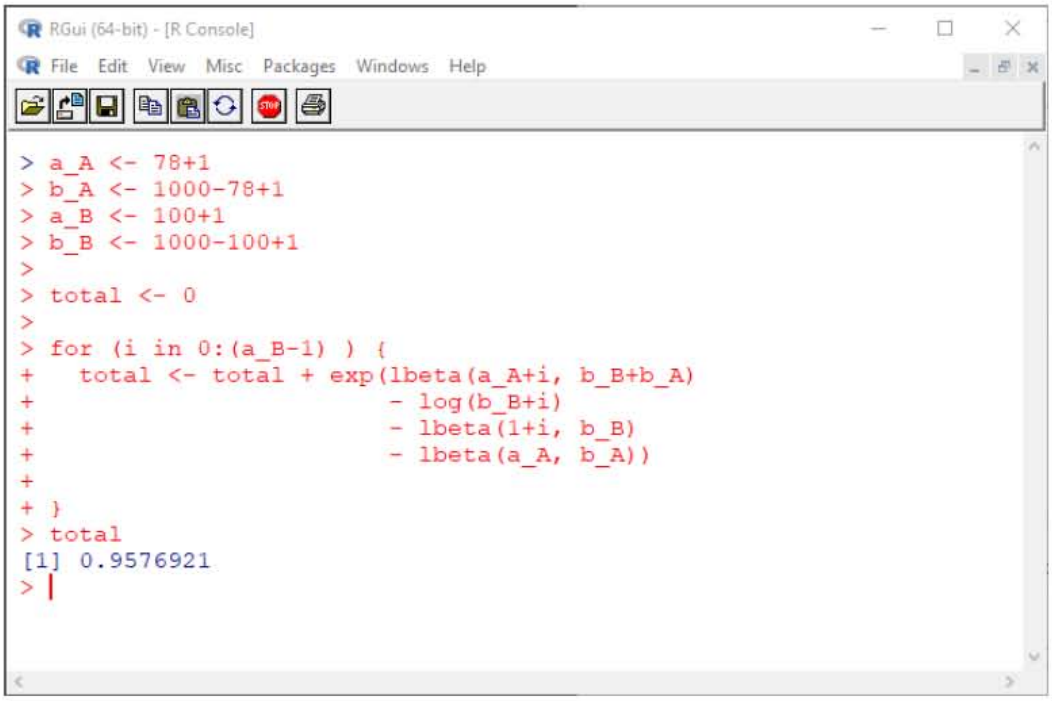
data:  c(100, 78) out of c(1000, 1000)
X-squared = 2.9847, df = 1, p-value = 0.08405
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.0029398  0.0469398
sample estimates:
prop 1 prop 2
 0.100  0.078

>
```

Maintenant écrivons explicitement la somme du cas bayésien comme l'intuition nous dirait de le faire:


```
> # succès+1 de A
> a_control <- 78+1
> # échecs+1 de A
> b_control <- 1000-78+1
> # succès+1 de B
> a_test <- 100+1
> # échecs+1 de b
> b_test <- 1000-100+1
>
> Prob_B_est_meilleur <- 0
> for (i in 0:(a_test-1) ) {
+   Prob_B_est_meilleur <- Prob_B_est_meilleur+beta(a_control+i,b_control+b_test) /
+                          (b_test+i)*beta(1+i,b_test)*beta(a_control,b_control)
+
+ }
>
> round(Prob_B_est_meilleur, 4)
[1] 0
>
```

Il y clairement un problème, et celui-ci vient en réalité de la fonction **beta( )** qui donne très vite des Infinis. Il vaut alors mieux travailler avec la fonction log-beta **lbeta( )** comme ci-dessous:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> a_A <- 78+1
> b_A <- 1000-78+1
> a_B <- 100+1
> b_B <- 1000-100+1
>
> total <- 0
>
> for (i in 0:(a_B-1)) {
+ total <- total + exp(lbeta(a_A+i, b_B+b_A)
+ - log(b_B+i)
+ - lbeta(1+i, b_B)
+ - lbeta(a_A, b_A))
+
+ }
> total
[1] 0.9576921
>

Exemple 175.: Loi Triangulaire

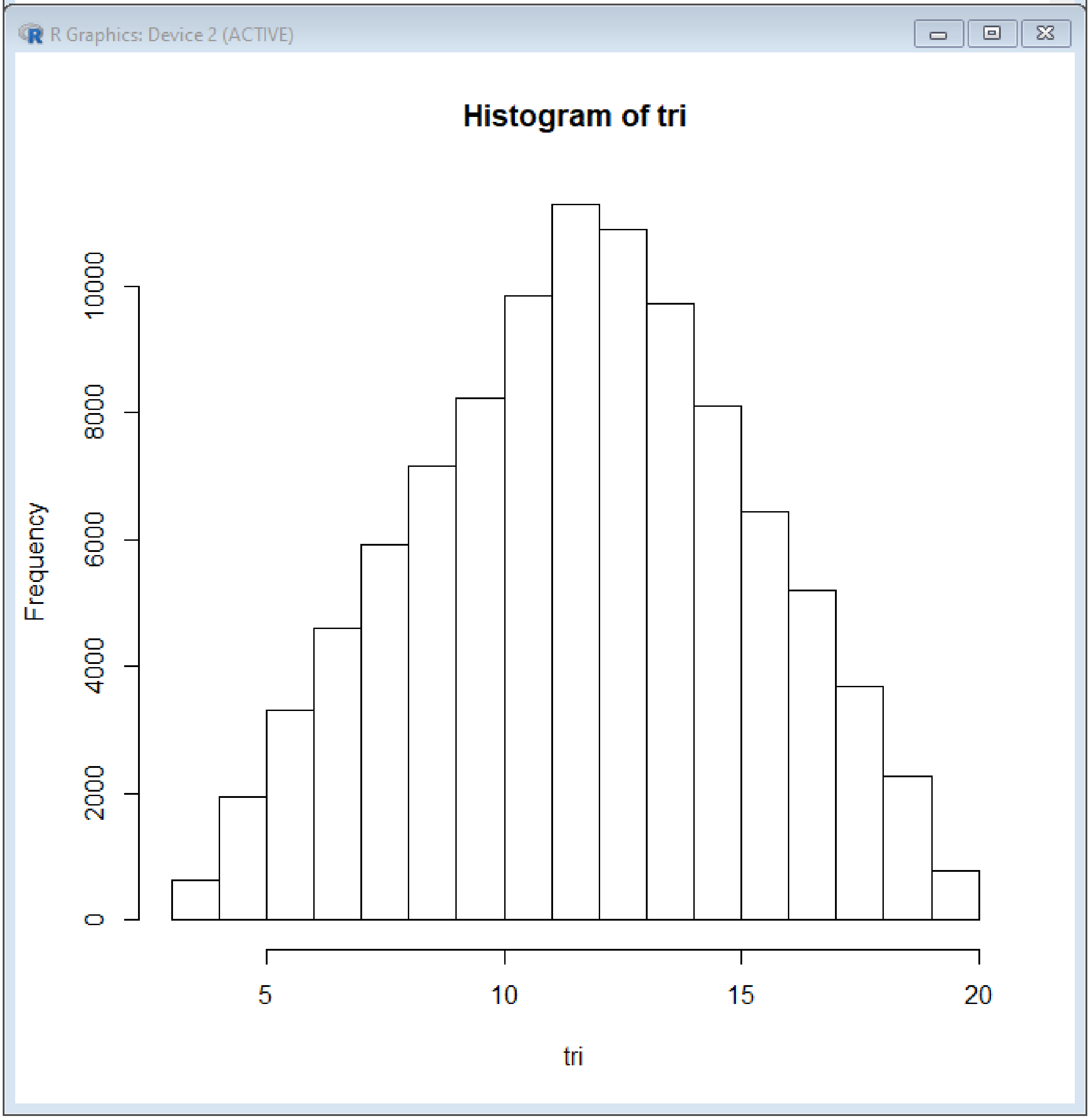
R 3.6.3

La loi triangulaire est très connue par les managers qui débutent en simulation de scénarios en entreprise (c'est souvent la progression intellectuelle qui suit l'estimation ponctuelle et précède la méthode utilisant la loi bêta).

Voyons donc comment créer cette fameuse variable aléatiore à l'aide du **EnvStats** et de ses fonctions **dtri( )**, **ptri( )**, **qtri( )** et surtout de **rtri( )**.

R Console

```
> library("EnvStats")
> tri<-rtri(n=100000, min=3, max=20, mode=12)
> hist(tri)
>
```



Exemple 176.: Petite compilation pour comparaisons

R 3.0.2

Voici si jamais une petite commande pour rassembler quelques fonctions de distribution (ensuite à vous de généraliser en fonction de vos besoins):

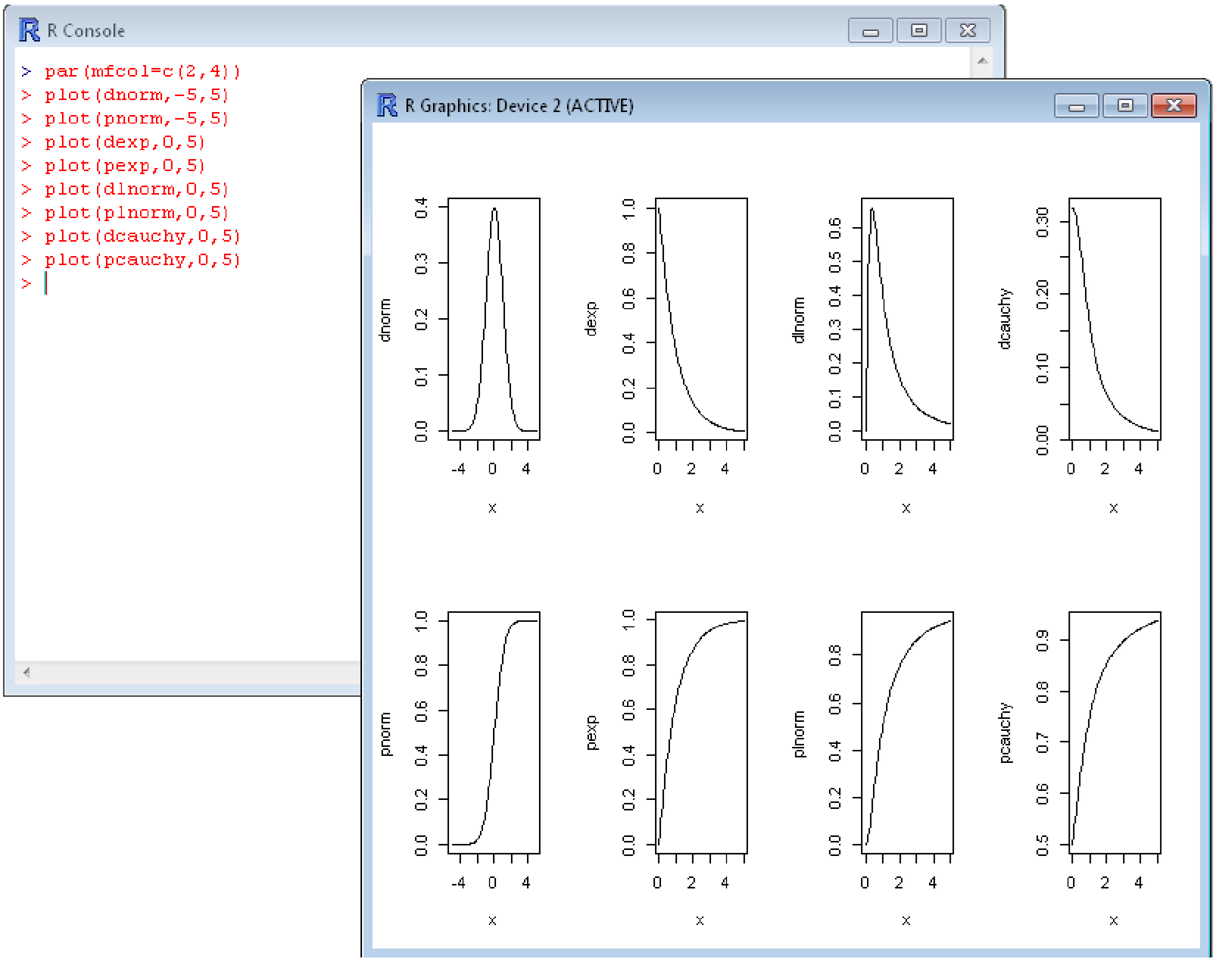
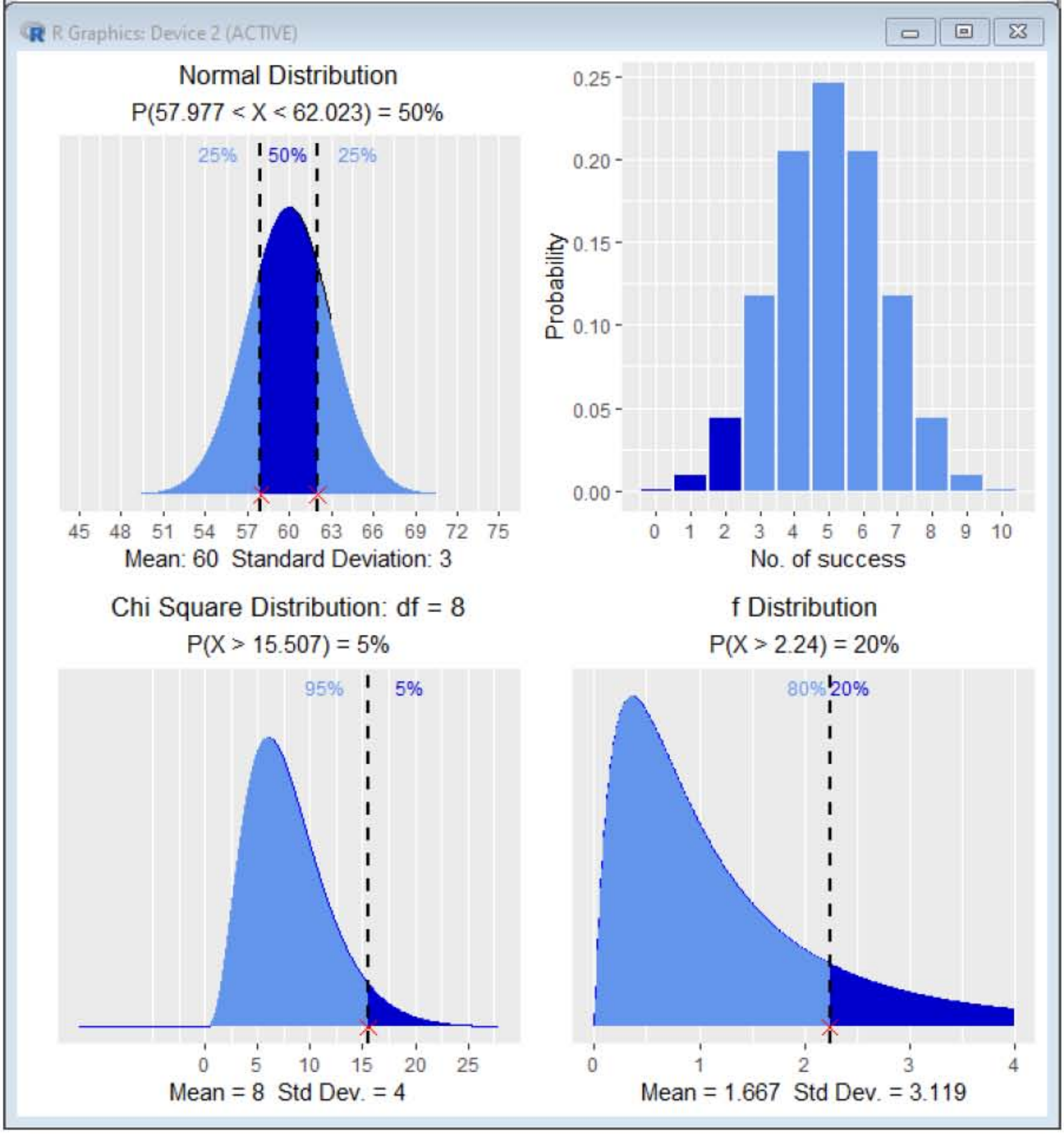


Figure 215 Graph d'une galerie de fonctions de répartition et densité

Notons qu'il existe un package sympa, nommé **vistributions**, pour enseigner les 4 lois les plus fondamentales:

R Console

```
> library("vistributions")
> library("ggpubr")
> plot1<-vdist_normal_perc(probs=0.5,mean= 60, sd=3, type="both")
> plot2<-vdist_binom_perc(n=10, p=0.5, 0.05)
> plot3<-vdist_chisquare_perc(probs=0.05, df=8, type="upper")
> plot4<-vdist_f_perc(probs=0.20, num_df=4, den_df=5, type="upper")
> #les quatre graphes dans une unique fenêtre
> ggarrange(plot1,plot2,plot3,plot4,ncol = 2, nrow = 2)
>
```



Exemple 177.: Diagramme quantile-quantile Normal (q-q plot)

R 3.0.2

La commande **qqnorm( )** permet de comparer des données à la distribution Normale empirique correspondante:

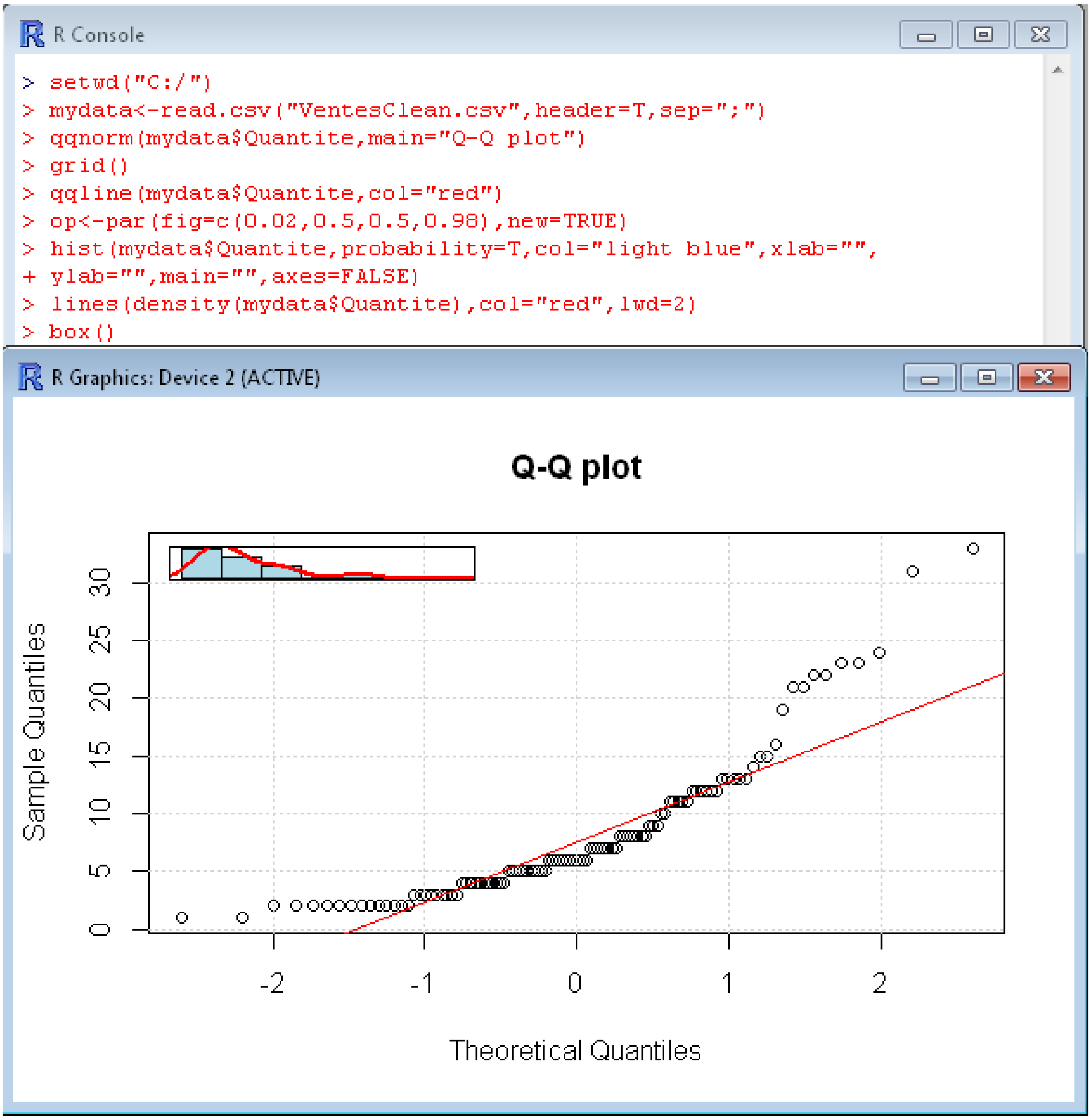


Figure 216 Graph quantile-quantile avec histogramme et fonction de densité

Ou avec les mêmes données que dans le cours Minitab:

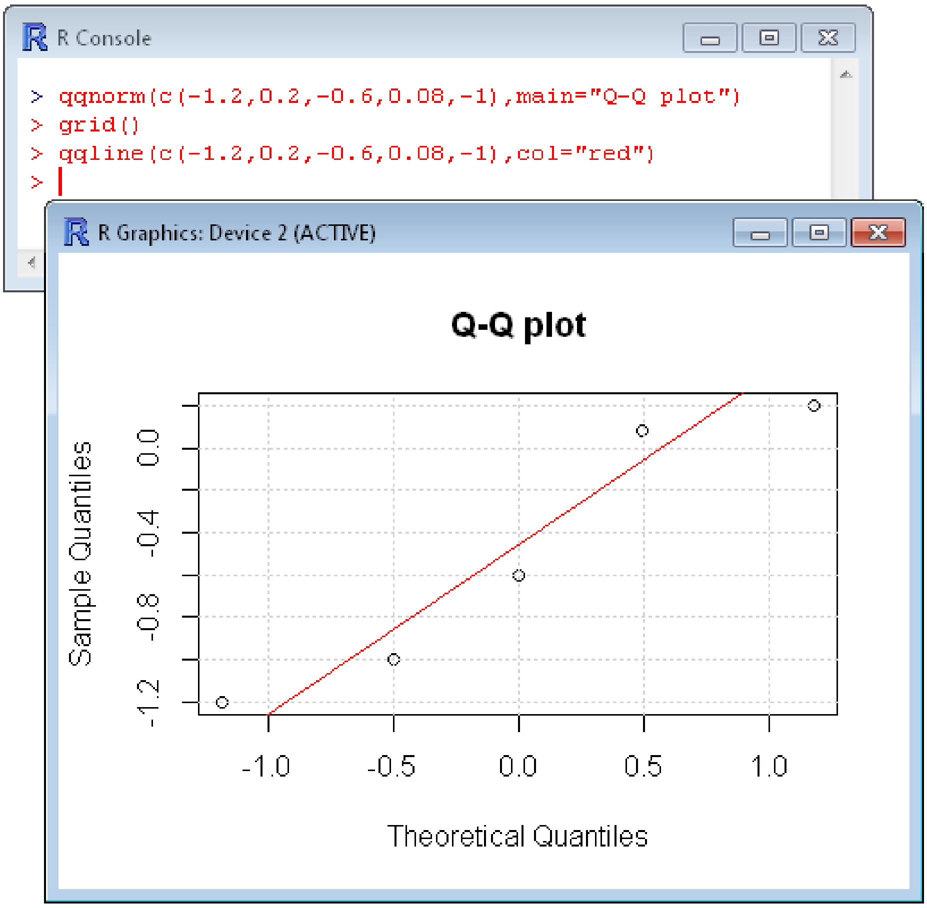
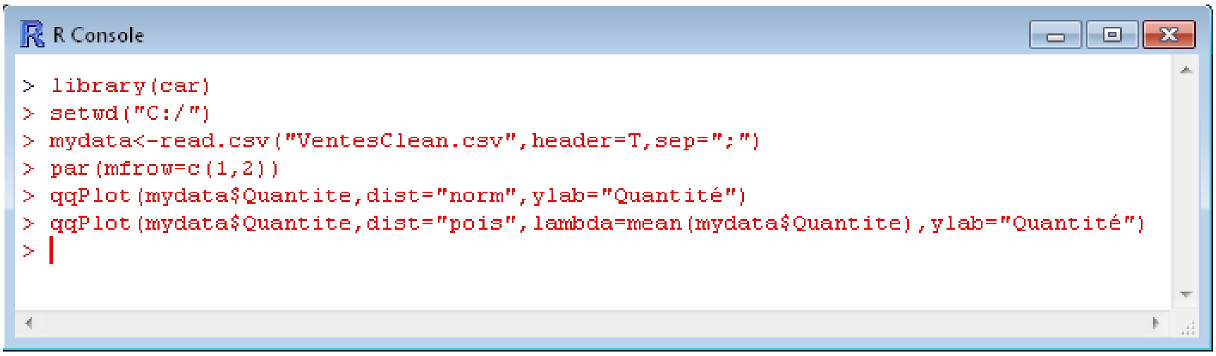


Figure 217 Graph q-q plot simple

On peut faire mieux avec le package **car** et la commande **qqPlot( )** non seulement au niveau de la qualité du visuel mais aussi au niveau du choix des fonctions d'ajustement:



Ce qui donne:

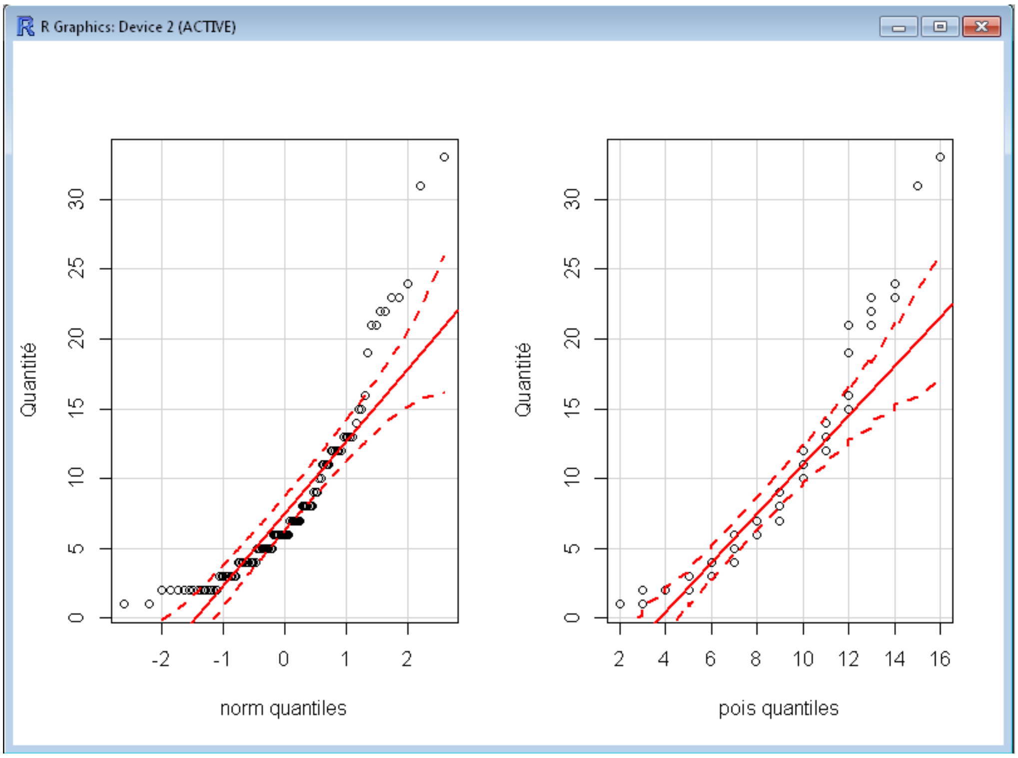


Figure 218 Graph q-q plot avec intervalles de confiance

On peut aussi mélanger deux graphiques classiques entre eux:

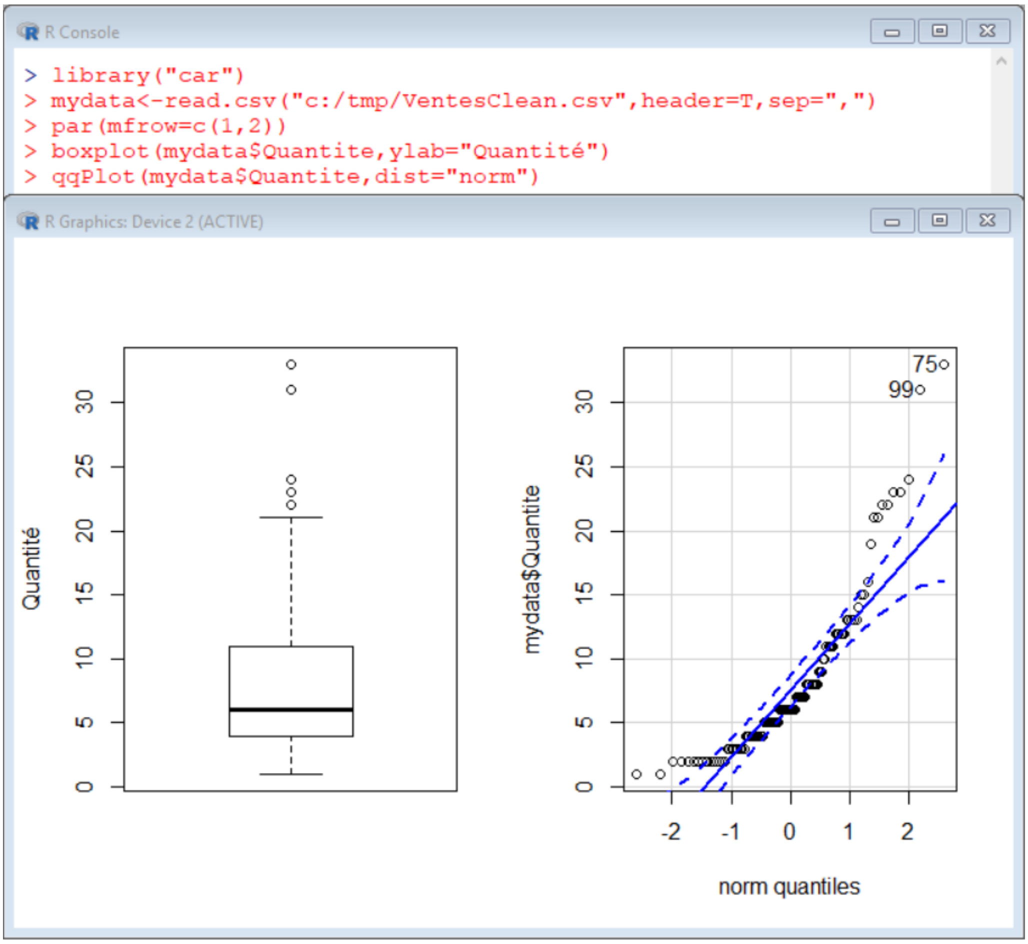
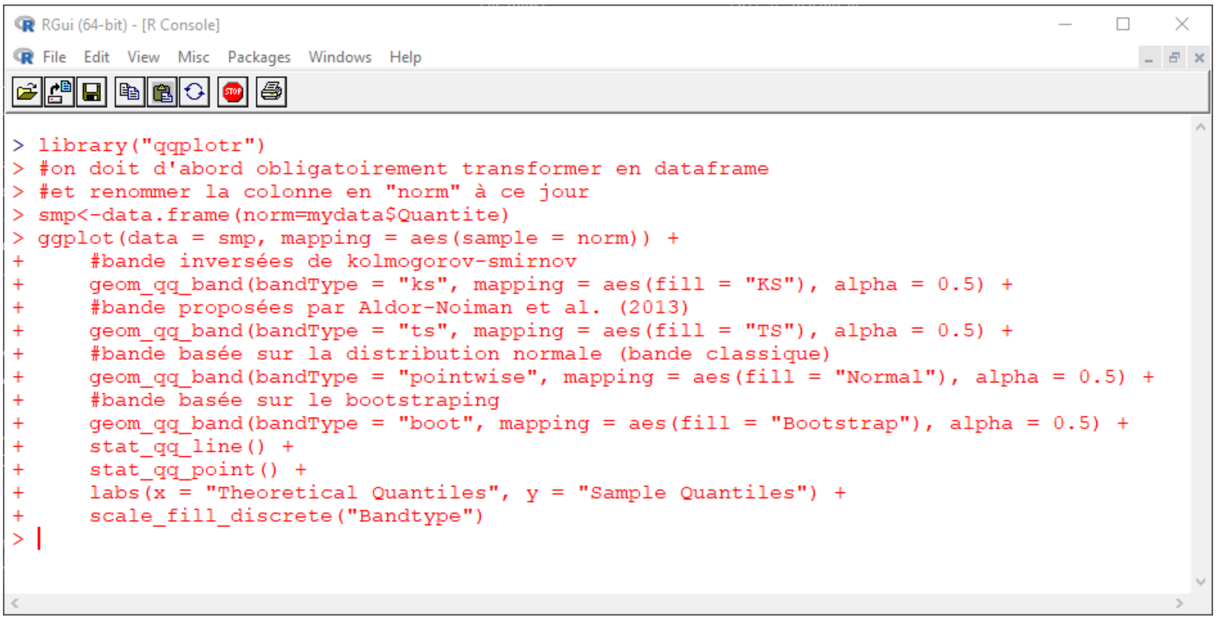
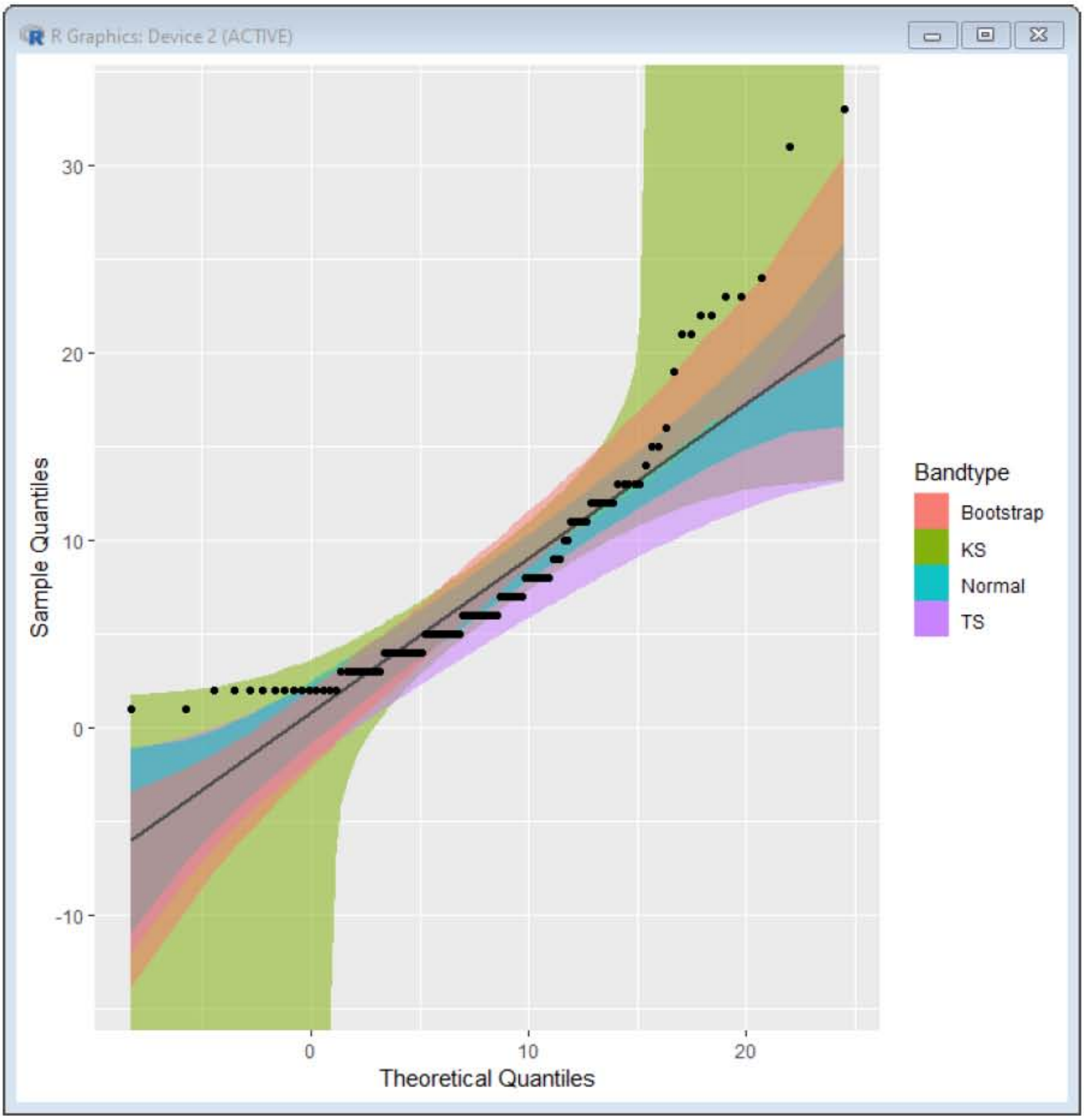


Figure 219 Graph q-q plot avec Box-Whisker

On peut aller encore plus loin avec le package **qqplotr** et 4 bandes différentes:



Ce qui donnera graphiquement:


R Graphics: Device 2 (ACTIVE)
Sample Quantiles
Theoretical Quantiles
30
20
10
0
-10
0
10
20
Bandtype
Bootstrap
KS
Normal
TS

Exemple 178.: Loi faibles des grandes nombres vs loi forte des grands nombres

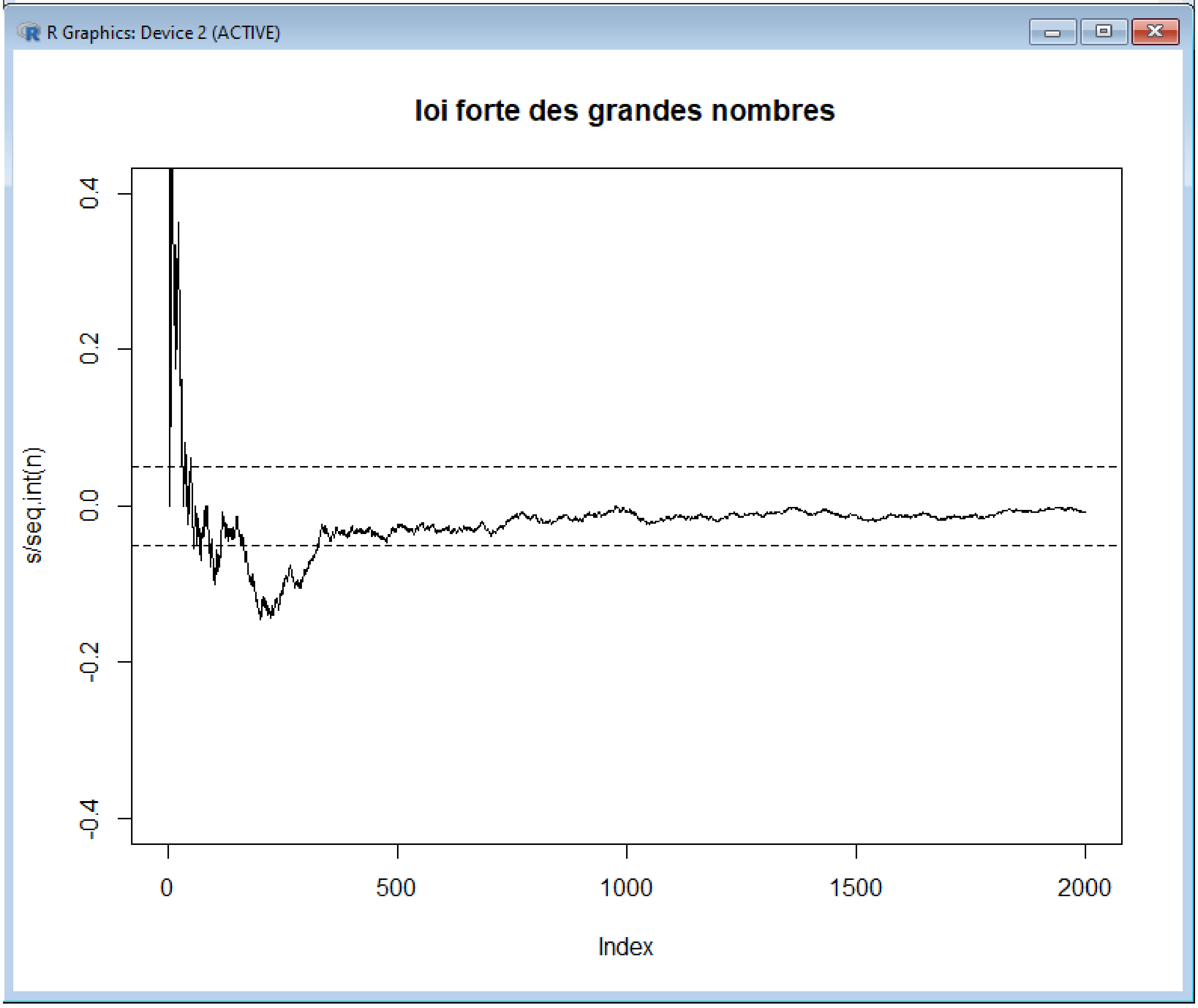
R 4.1.1

Dans ce qui suit, nous parlons d'une marche aléatoire simple, $X_i = \pm 1$ avec une probabilité égale, et nous calculons des moyennes glissantes:

$$\frac{S_n}{n} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad n = 1, 2, \ldots$$

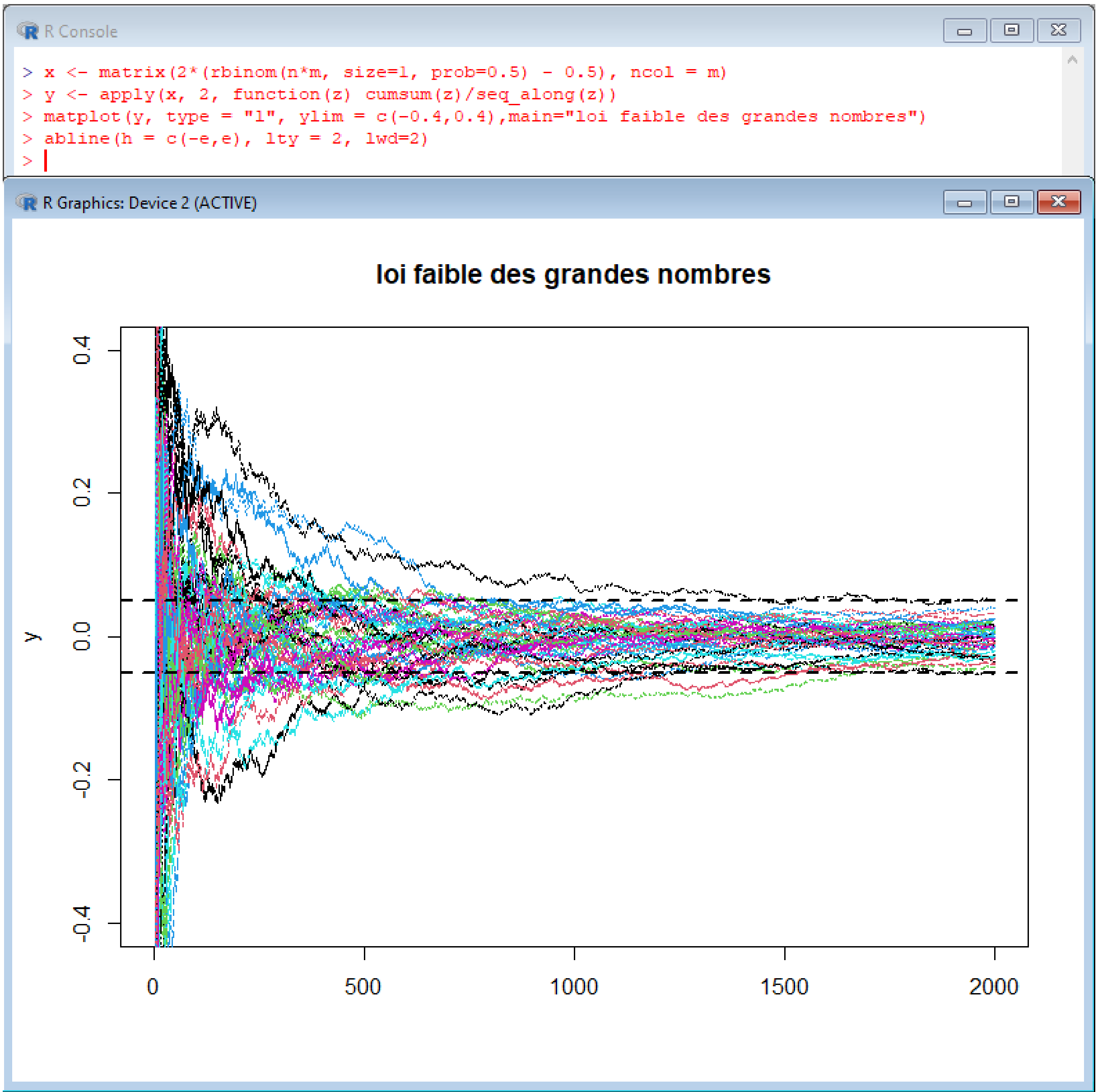
Ce qui sous forme de langage et graphique R sera donné par:

```
> n <- 2000;  m <- 50; e <- 0.05
> s <- cumsum(2*(rbinom(n, size=1, prob=0.5) - 0.5))
> plot(s/seq.int(n), type = "l", ylim = c(-0.4, 0.4),main="loi forte des grandes nombres")
> abline(h = c(-e,e), lty = 2)
>
```



La loi forte des grands nombres (convergence presque sûrement) dit que nous pouvons être sûrs à 100 % que cette courbe qui s'étend vers la droite finira par, à un moment donné, tomber entièrement dans les bandes pour toujours par la suite (vers la droite)!

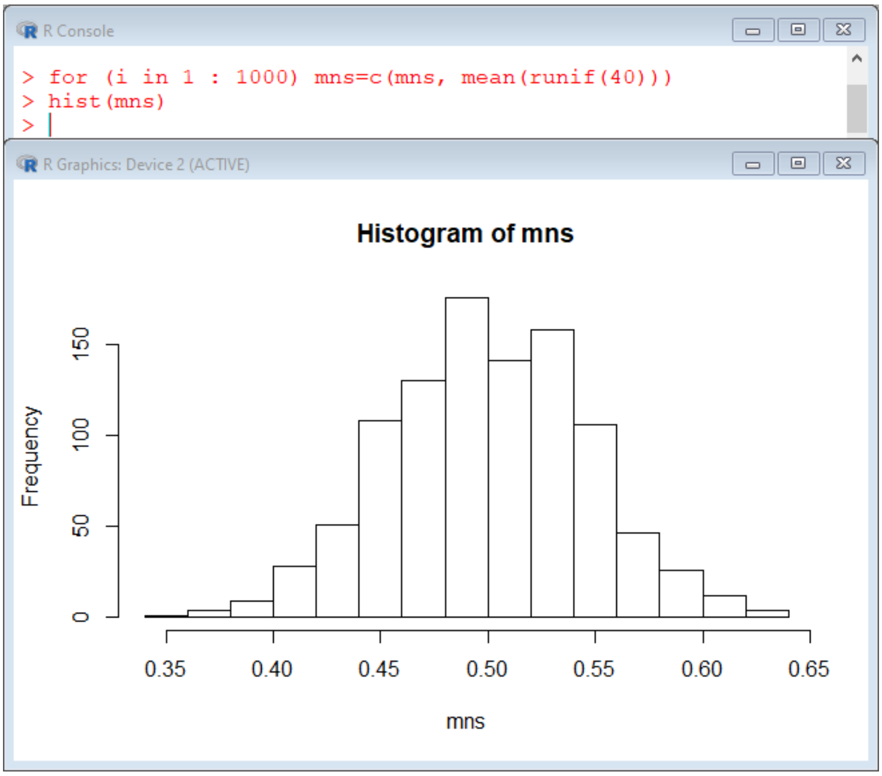
Alors que la loi faible des grands nombres (convergence en probabilité) indique qu'une grande partie des chemins d'échantillonnage seront dans les bandes du côté droit, au temps *n*. Nous ne pouvons jamais être sûrs qu'une courbe particulière sera à l'intérieur à un moment donné, mais à regarder la masse de nouilles dans la bande dans le graphique ci-dessous nous montre que ce serait un pari assez sûr. La loi faible des grands nombres indique également que nous pouvons rendre la proportion de nouilles à l'intérieur aussi proche de 1 que nous le souhaitons en rendant *n* suffisammen grand.



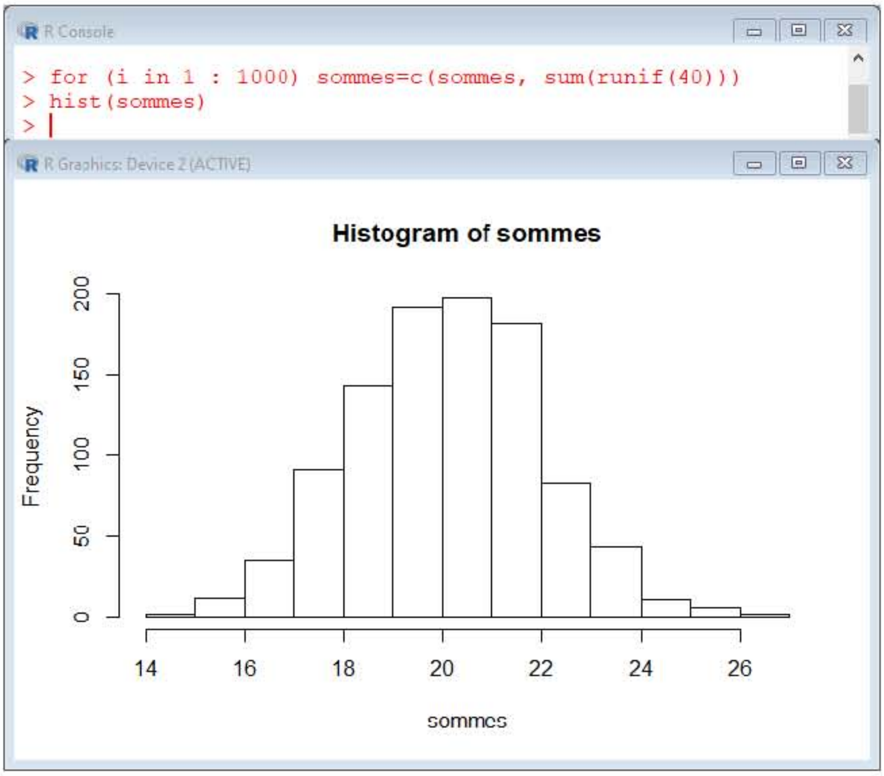
Exemple 179.: Théorème central limite

R 3.6.2

Bon voici les deux des trois exemples types du théorème central limite (le code lui-même suffit comme explication):



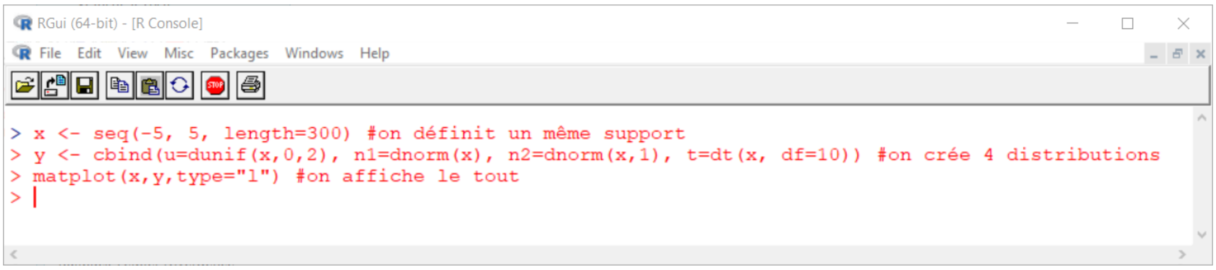
et évidemment on peut faire le même exemple avec la somme plutôt que la moyenne:


R Console
> for (i in 1 : 1000) sommes=c(sommes, sum(runif(40)))
> hist(sommes)
>
R Graphics: Device 2 (ACTIVE)
Histogram of sommes
Frequency
0
50
100
150
200
14
16
18
20
22
24
26
sommes

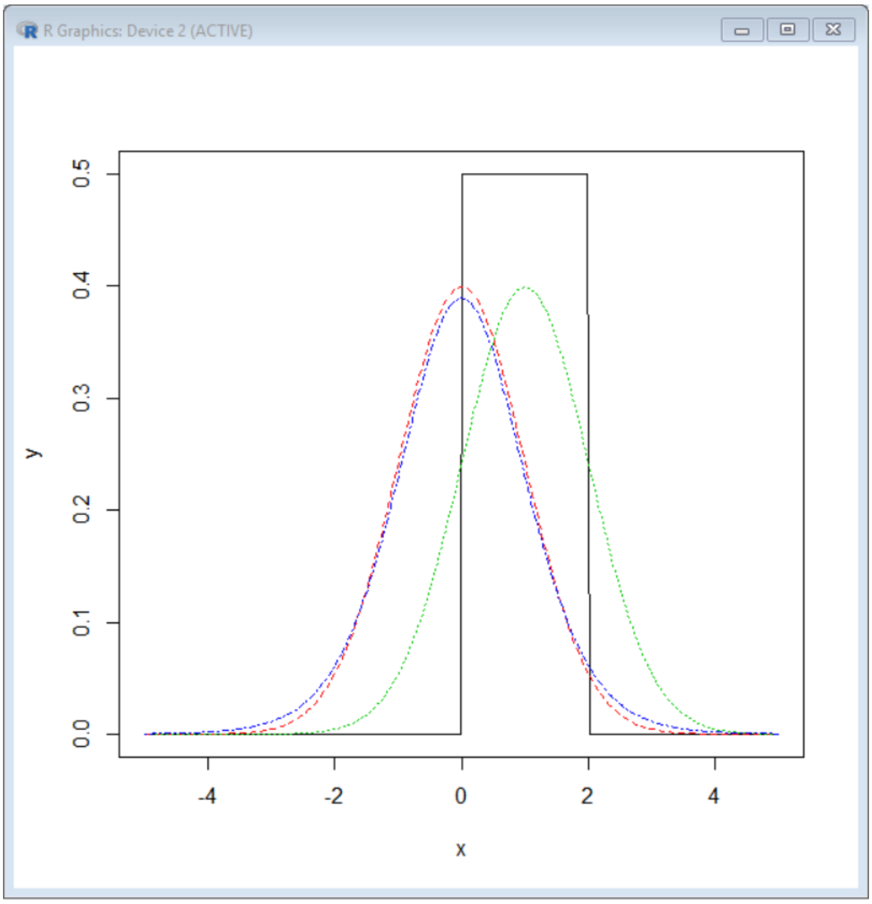
Exemple 180.: Divergence de Kullback-Leibler et distance de Wasserstein

R 3.5.2

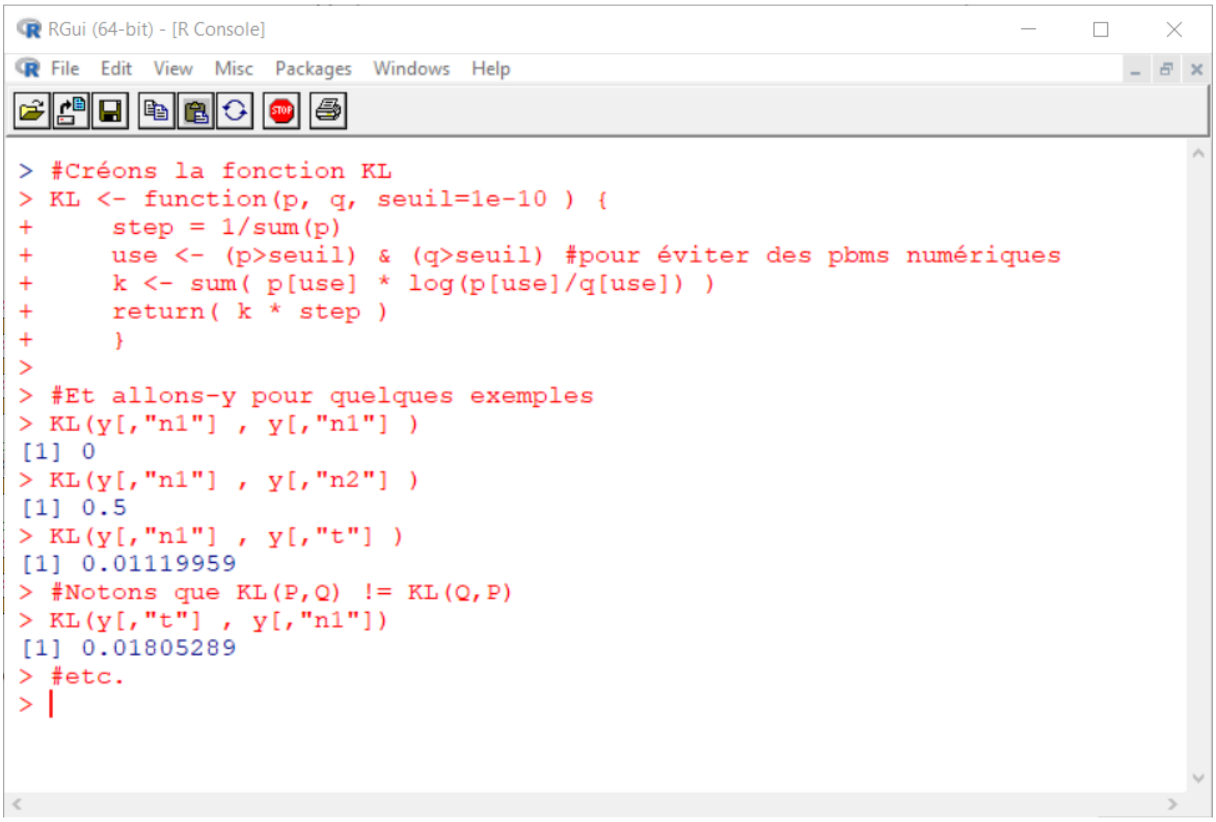
Comme promis, voyons une application numérique du calcul de la divergence de Kullback-Leibler et de la distance de Wasserstein car non seulement c'est important de savoir les calculer, mais c'est aussi un exercice pour faire un peu de script avec R:



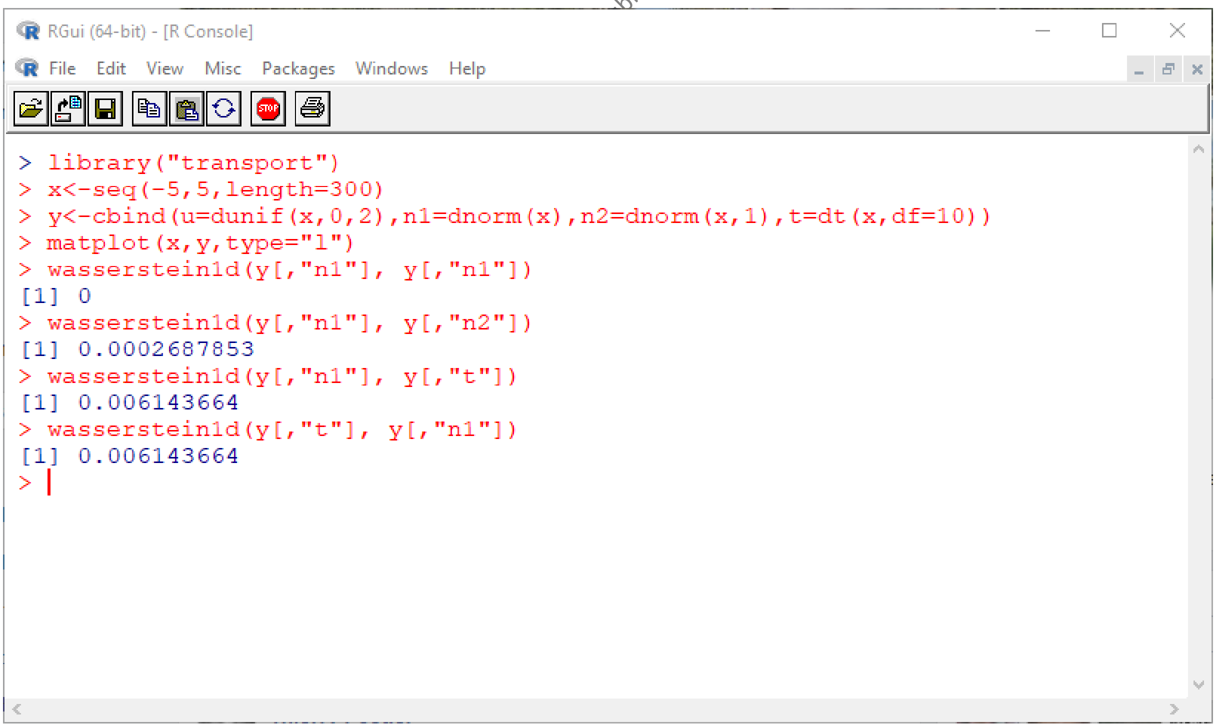
Donc le graphique fait à l'aide de la fonction **matplot( )** donne:



Et maintenant les applications numériques donnent pour la divergence de Kullback-Leibler (qui pour rappel, ne satisfait pas les propriétés d'une distance!):



Et voyons maintenant la distance de Wasserstein avec la fonction **wasserstein1d( )** du package **transport**:



Si vous avez tenu jusqu'ici en lisant tous les chapitres précédents, vous êtes à même maintenant (en théorie) à comprendre la cheatsheet suivante sur les bases de R:

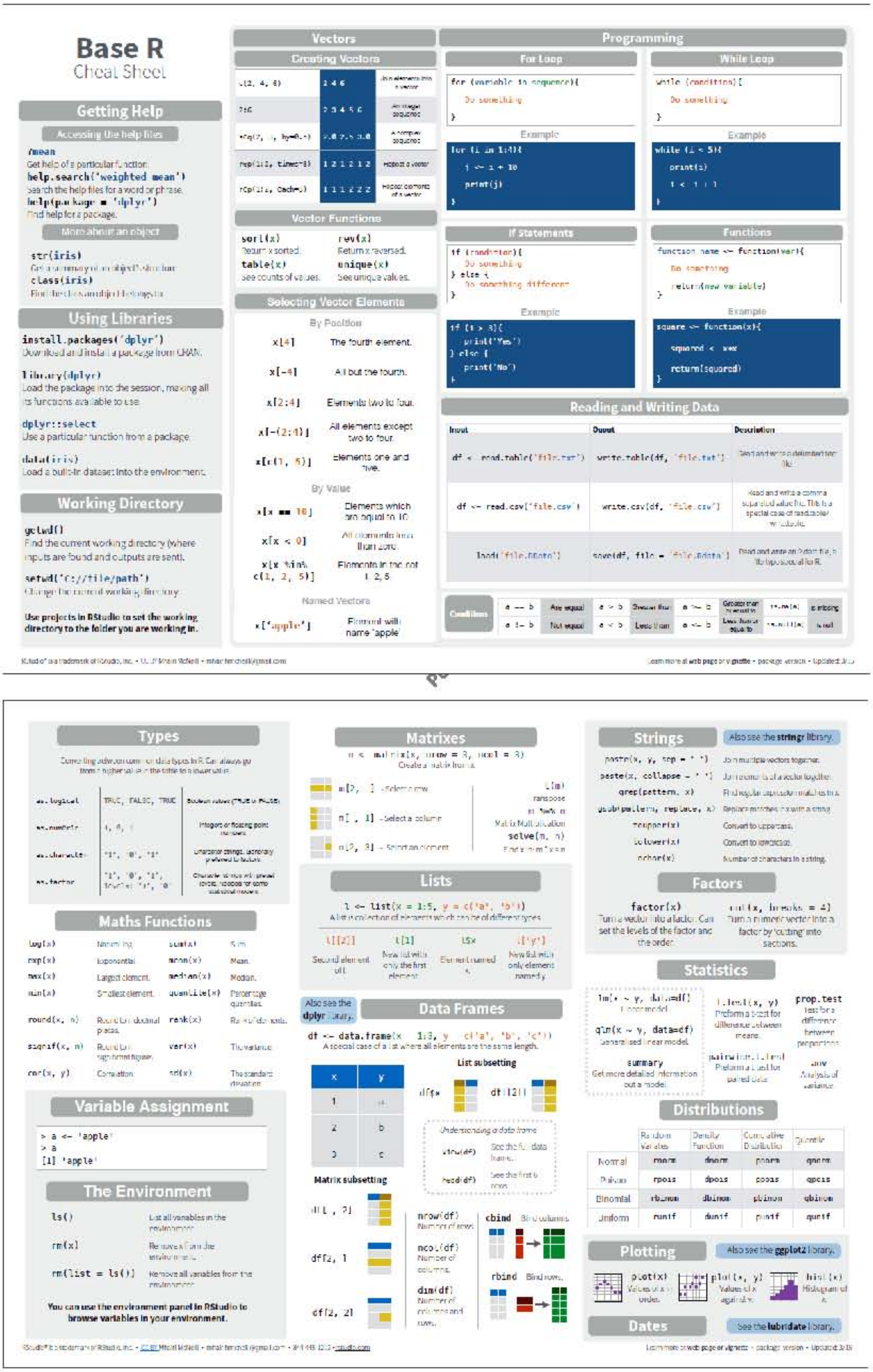


Figure 220 Cheatsheet Base R

13. Statistiques paramétriques

Nous allons donc voir dans ce chapitre si nous retrouvons les résultats calculés à la main dans le cours théorique pour voir si les résultats correspondent aux démonstrations mathématiques détaillées et aussi vérifier s'il y a correspondance avec Minitab, SPSS et MS Excel (puisque nous savons que l'implémentation des algorithmes et l'approche calculatoire peut différer!).

Pour que les choses soient claires, nous ne reviendrons pas dans ce chapitre ni dans celui des Statistiques non paramétriques sur le problème de la *p*-value et de ses utilisations fallacieuses que nous avons longuement développement lors des séances théoriques.

Ce chapitre contient actuellement seulement:

32 tests paramétriques sur un peu plus de 416 connus

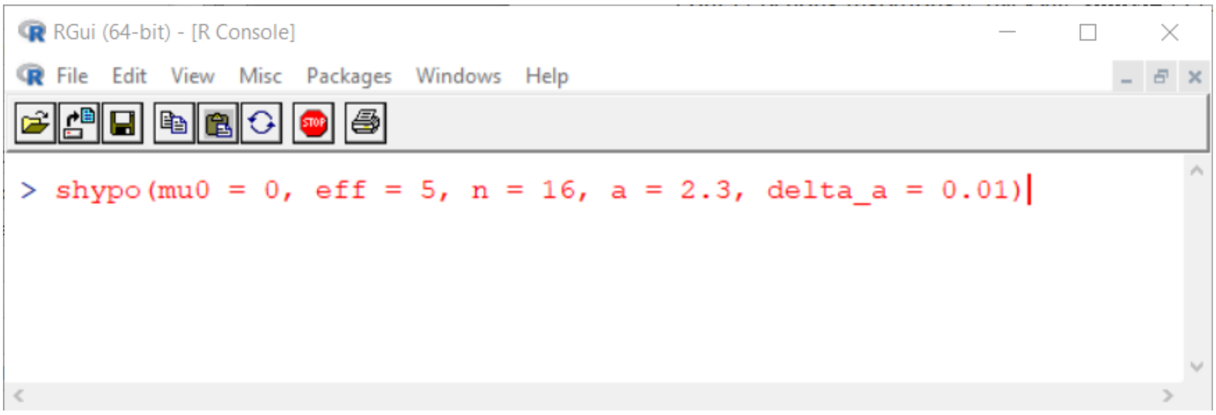
Comme nous le savons sur un total de plus de 800 tests paramétriques et non-paramétriques!!!

Remarque: Nous verrons à la fin de ce chapitre qu'il existe un package qui construit automatiquement les phrases de conclusions et d'analyses (en anglais) pour les tests statistiques les plus simples (voir page 1442).

Pédagogie

Commençons avec un package qui sera très utile aux professeurs (et probablement aussi aux étudiants/praticiens des statistiques) pour enseigner et faire comprendre de façon assez intuitive et illustrée le risque de première et deuxième étape.

Pour cela nous installons le package **smovie** et exécutons la commande suivante:



Ce qui aura pour effet d'ouvrir la fenêtre suivante:

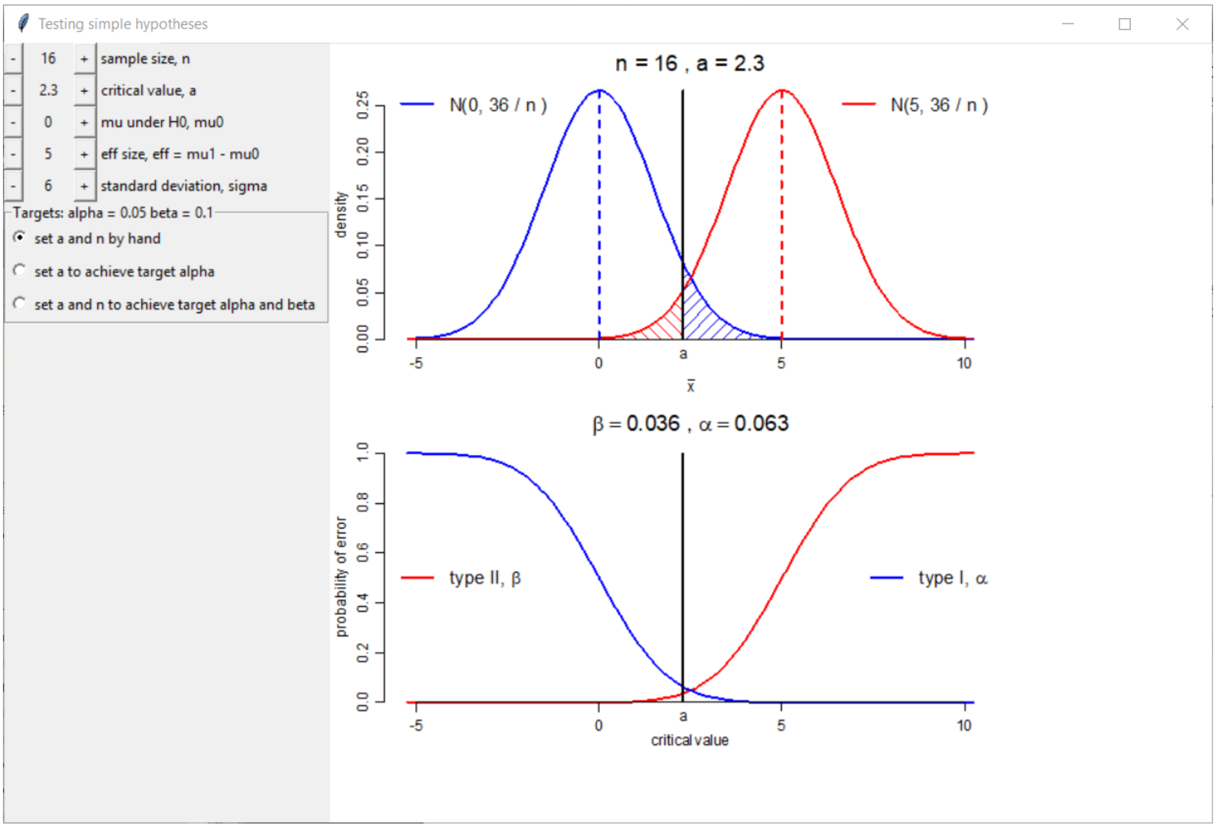


Figure 221 Graph avec erreur de type I et II

Il ne reste ensuite plus qu'à jouer avec les boutons pour enseigner ou se sensibiliser au sujet des erreurs de type I et II de façon ludique et interactive!

Exemple 181.: Tests de Normalité (ou d'adéquations en général)

Nous savons que la grande majorité des tests statistiques analytiques (ceux pour lesquels une formule close existe) nécessitent l'hypothèse de normalité. Il est logique de commencer par ce type de vérification avant de poursuivre dans une quelconque analyse.

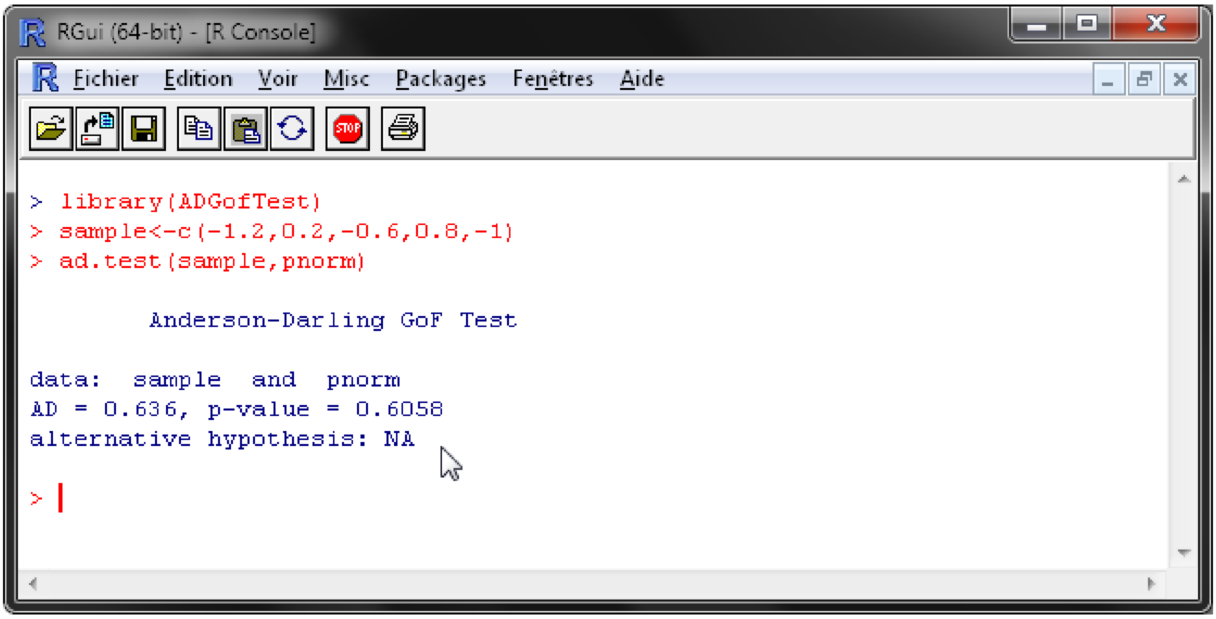
Remarque: Nous ne reviendrons pas dans ce cours/livre R du "pseudo-débat" qu'il y a chez certains praticiens quant à l'usage des tests de normalité/adéquation et le nom ("test de normalité") qui leur a été attribué.

Test d'adéquation d'Anderson-Darling (ie Agostino-Stephens)

R 3.0.2

Nous souhaitons ici simplement vérifier que nous obtenons les mêmes résultats que ceux obtenus dans Microsoft Excel avec la méthode Monte Carlo (et in extenso vérifier les résultats obtenus lors de l'étude théorique du test de d'Anderson-Darling) et Minitab.

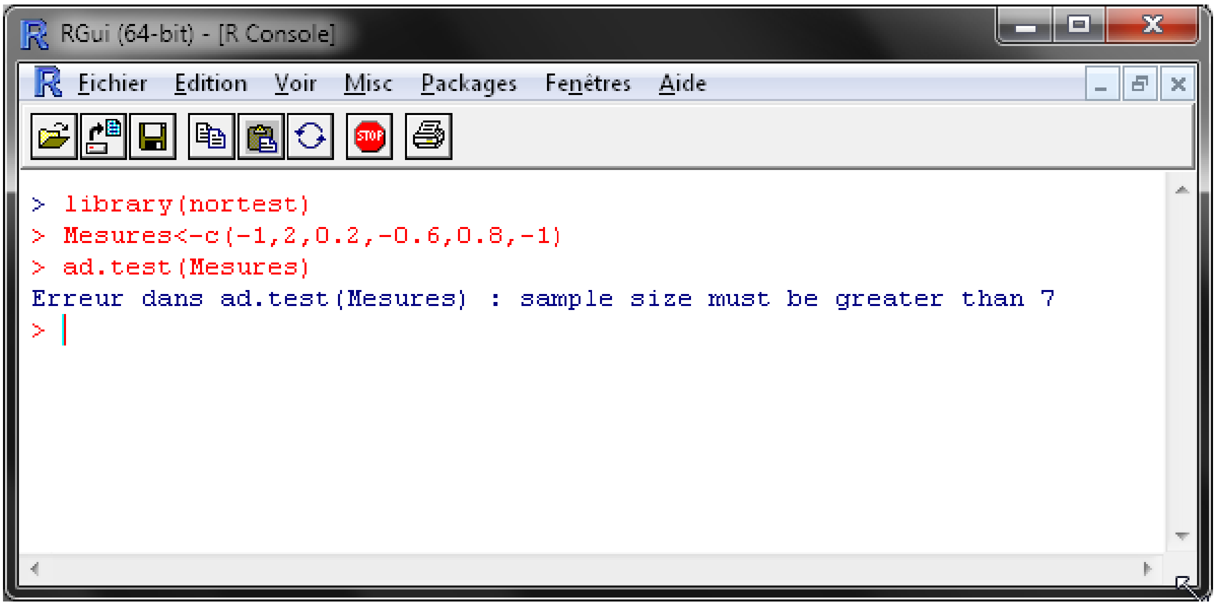
En utilisant la commande **ad.test( )** et avec les mêmes données brutes que dans le cours théorique, nous avons en utilisant le package **ADGofTest**:



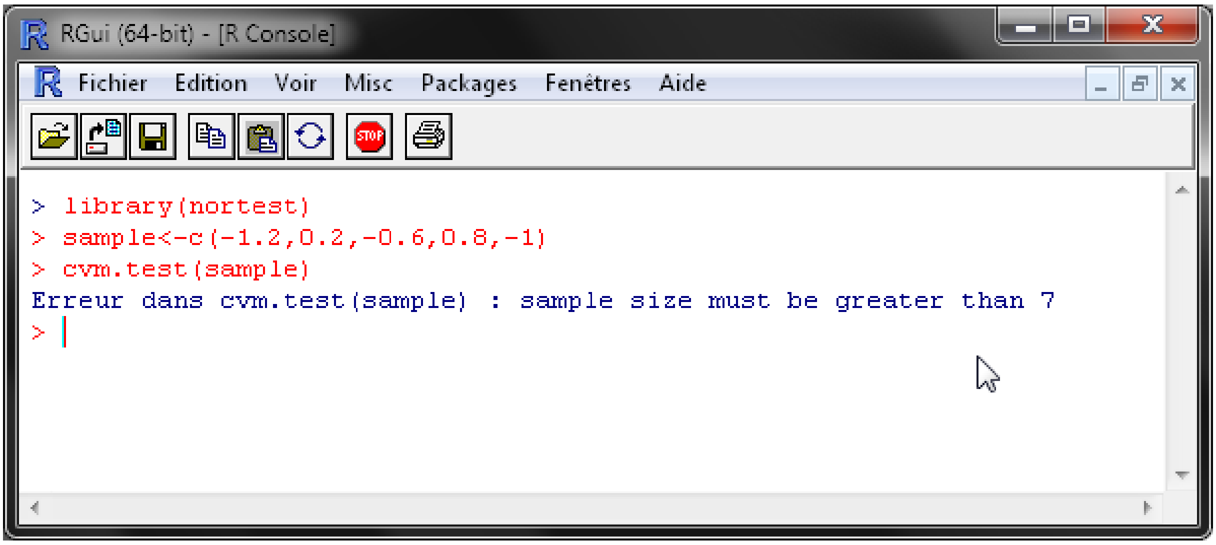
Donc ici nous voyons un énorme avantage par rapport à Minitab déjà: nous pouvons choisir la distribution selon nos désirs!

Nous voyons cependant une différence significative entre R et Minitab. Mais nous en connaissons l'origine puisque nous avons fortement critiqué certains aspects du test d'AD pendant le cours théorique (cependant la conclusion reste la même).

Il existe un autre package spécialisé dans les tests de Normalité et qui est **nortest** mais bon voilà il souffre d'un léger "problème" comme en atteste la capture d'écran ci-dessous (car il suite la recommandation des articles d'origine des auteurs):



Accessoirement, rappelons que nous avons vu dans le cours théorique que le test d'ajustement de Cramer-Von Mises est un cas particulier du test d'Anderson-Darling (il n'y a simplement pas de dénominateur). Ce test existe aussi sous la commande **cvm.test( )** dans le package **nortest** mais il souffre du même défaut. Effectivement:



Sinon on peut aussi s'amuser à comparer avec plein de distribution:

```
> ad.test(rnorm(6),pnorm) # teste une distribution normale

        Anderson-Darling GoF Test

data:  rnorm(6)  and  pnorm
AD = 0.41755, p-value = 0.8256
alternative hypothesis: NA

> ad.test(rpois(6,4),ppois,4) # teste une distribution de poisson

        Anderson-Darling GoF Test

data:  rpois(6, 4)  and  ppois
AD = 2.5732, p-value = 0.04734
alternative hypothesis: NA

> ad.test(runif(6,1,10),punif,1,10) # teste une distribution uniforme

        Anderson-Darling GoF Test

data:  runif(6, 1, 10)  and  punif
AD = 0.17466, p-value = 0.9972
alternative hypothesis: NA

> ad.test(rexp(6,2),pexp,2) # test une distribution exponentielle

        Anderson-Darling GoF Test

data:  rexp(6, 2)  and  pexp
AD = 2.2392, p-value = 0.07027
alternative hypothesis: NA

> ad.test(rlnorm(6),plnorm) # test une distribution log-normale

        Anderson-Darling GoF Test

data:  rlnorm(6)  and  plnorm
AD = 2.9487, p-value = 0.03068
alternative hypothesis: NA
```

Il est intéressant d'observer que la fonction **ad.test( )** accepte des fonctions de distribution discrètes en entrée…

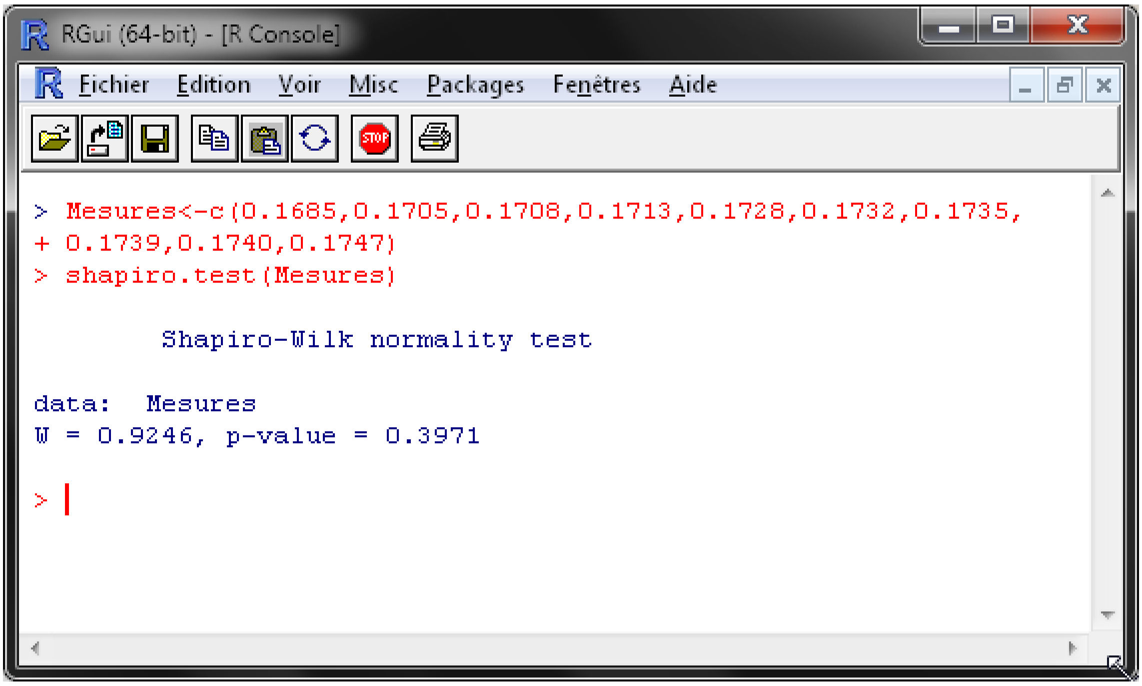
Test de Normalité de Shapiro-Wilk

R 3.0.2

Nous souhaitons ici vérifier que nous obtenons les mêmes résultats que ceux obtenus dans Microsoft Excel avec la méthode Monte Carlo (et in extenso vérifier les résultats obtenus lors de l'étude théorique du test de Ryan-Joiner) et Minitab.

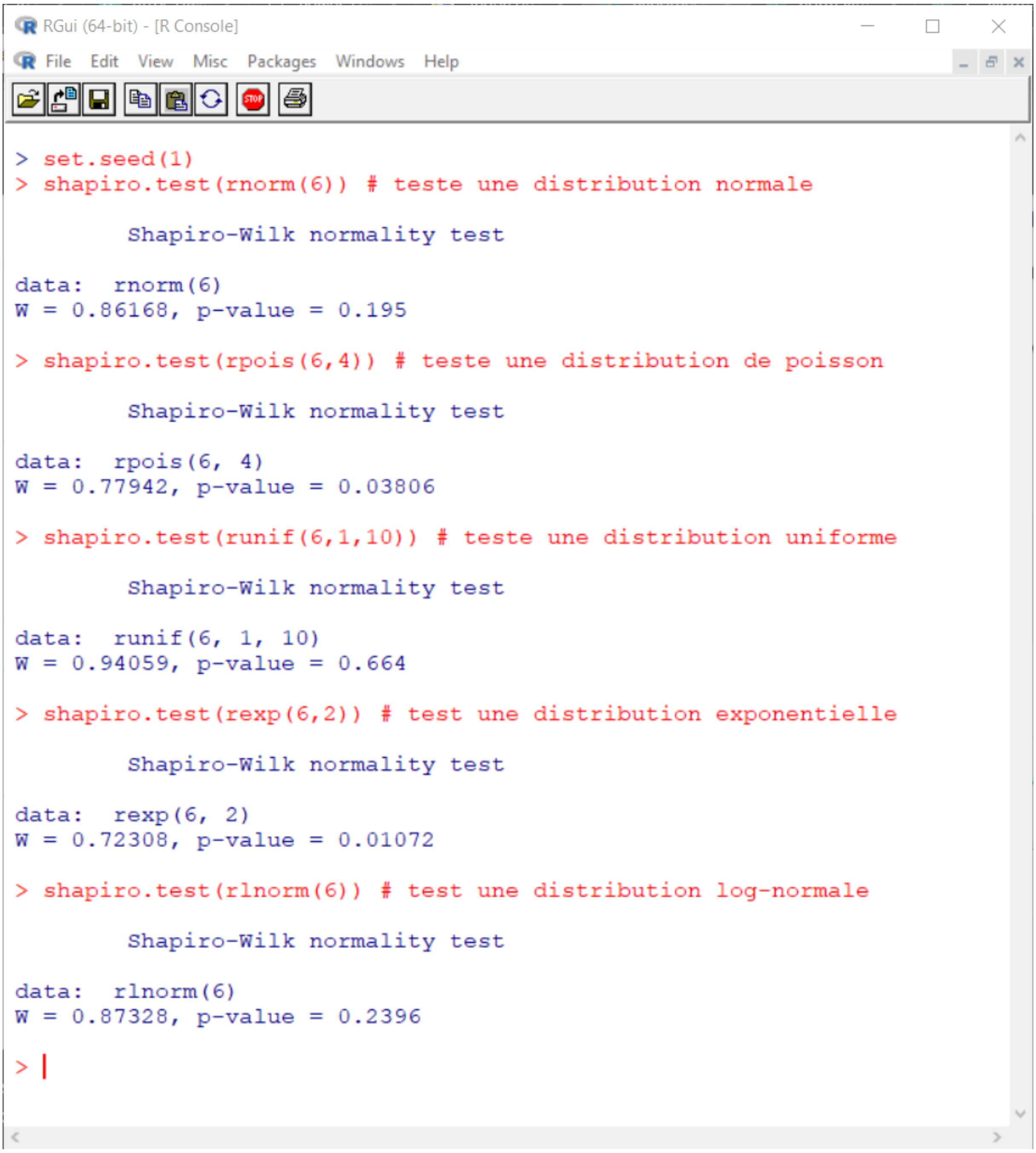
Rappelons que l'avantage de ce test non paramétrique de normalité (donc basé sur les rangs) est sa simplicité mais qu'il n'est pas contre pas adapté lorsque que trop de valeurs identiques se répètent.

En utilisant la commande **shapiro.test( )** et avec les mêmes données brutes que dans le cours théorique, nous avons:



La sortie est complément différente de Minitab au niveau esthétique (c'est bien dommage!) mais la valeur *W* et la *p*-value sont conformes en ordre de grandeur (voir mon livre Minitab pour les valeurs numériques exacte à comparer).

Sinon on peut aussi s'amuser a voir le comportement de ce type de tests:



Là encore il est intéressant d'observer que la fonction **shapiro.test( )** accepte des fonctions de distribution discrètes en entrée…

Puissance du test d'Anderson-Darling ou Shapiro-Wilk (ou autre)

R 3.5.1

Une question et un besoin fréquent dans la pratique est d'estimer la puissance d'un test de normalité et indirectement la taille de l'échantillon.

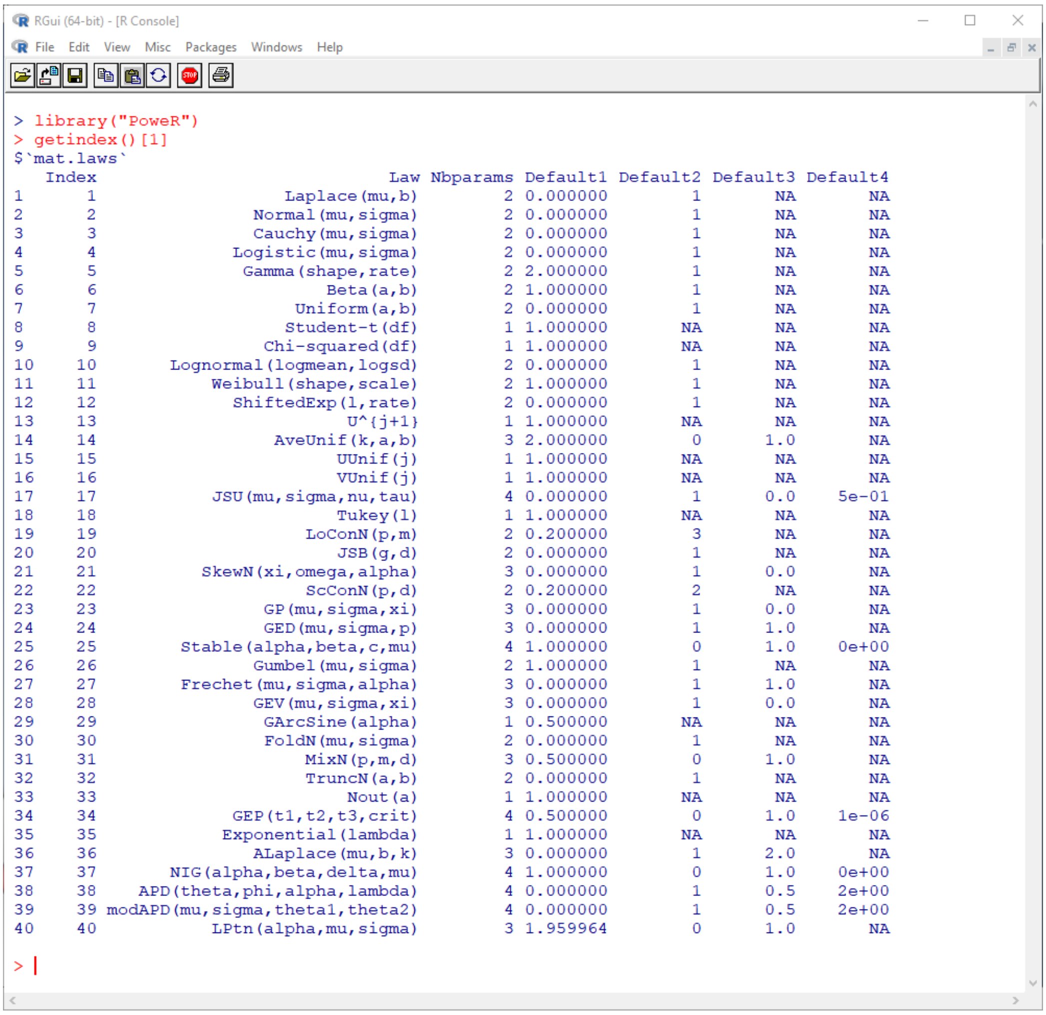
Avant de voir comment faire cela, nous laisserons le soin au lecteur de se documenter. De nombreuses études (dont le code n'est pratiquement jamais donné) semblent montrer que:

1. Les tests-T, la régression linéaire, et l'ANOVA sont relativement robustes voire… très robustes à la non-normalité (mais il faut vérifier toutefois dans la pratique quand on manipule des données de sujets qui sont relatifs à la santé public).

2. Le test de Shapiro-Wilk est plus puissant que le test d'Anderson-Darling (ou même de Cramèr-von-Mises ou de Kolmogorov-Smirnov)

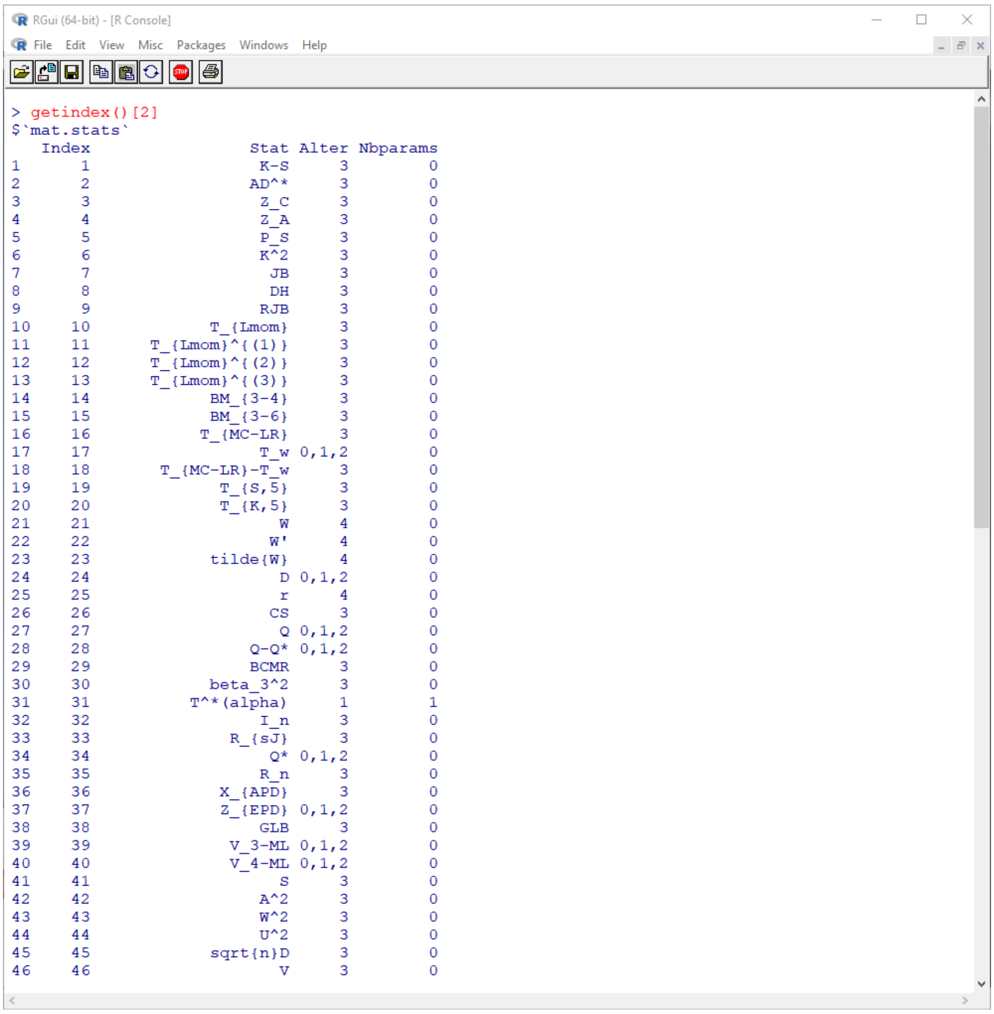
À ce jour, et a priori, on peut calculer la puissance de ce type de tests qu'avec les techniques de Monte-Carlo.

Pour cela nous allons utiliser le package **PoweR** et ses nombreuses fonctions, mais en particulier sa fonction **powcomp.fast( )**. Mais pour la comprendre ainsi que le package en général, il faut d'abord voir la fonction **get.index( )** et ses deux listes:



```
> library("PoweR")
> getindex()[1]
$`mat.laws`
   Index                              Law Nbparams Default1 Default2 Default3 Default4
1      1                     Laplace(mu,b)        2 0.000000        1       NA       NA
2      2                  Normal(mu,sigma)        2 0.000000        1       NA       NA
3      3                  Cauchy(mu,sigma)        2 0.000000        1       NA       NA
4      4                Logistic(mu,sigma)        2 0.000000        1       NA       NA
5      5                Gamma(shape,rate)        2 2.000000        1       NA       NA
6      6                         Beta(a,b)        2 1.000000        1       NA       NA
7      7                      Uniform(a,b)        2 0.000000        1       NA       NA
8      8                     Student-t(df)        1 1.000000       NA       NA       NA
9      9                   Chi-squared(df)        1 1.000000       NA       NA       NA
10    10          Lognormal(logmean,logsd)        2 0.000000        1       NA       NA
11    11              Weibull(shape,scale)        2 1.000000        1       NA       NA
12    12                ShiftedExp(l,rate)        2 0.000000        1       NA       NA
13    13                          U^{j+1}        1 1.000000       NA       NA       NA
14    14                   AveUnif(k,a,b)        3 2.000000        0      1.0       NA
15    15                          UUnif(j)        1 1.000000       NA       NA       NA
16    16                          VUnif(j)        1 1.000000       NA       NA       NA
17    17            JSU(mu,sigma,nu,tau)        4 0.000000        1      0.0    5e-01
18    18                          Tukey(l)        1 1.000000       NA       NA       NA
19    19                       LoConN(p,m)        2 0.200000        3       NA       NA
20    20                          JSB(g,d)        2 0.000000        1       NA       NA
21    21           SkewN(xi,omega,alpha)        3 0.000000        1      0.0       NA
22    22                       ScConN(p,d)        2 0.200000        2       NA       NA
23    23                  GP(mu,sigma,xi)        3 0.000000        1      0.0       NA
24    24                  GED(mu,sigma,p)        3 0.000000        1      1.0       NA
25    25          Stable(alpha,beta,c,mu)        4 1.000000        0      1.0    0e+00
26    26                Gumbel(mu,sigma)        2 1.000000        1       NA       NA
27    27         Frechet(mu,sigma,alpha)        3 0.000000        1      1.0       NA
28    28                 GEV(mu,sigma,xi)        3 0.000000        1      0.0       NA
29    29                  GArcSine(alpha)        1 0.500000       NA       NA       NA
30    30                  FoldN(mu,sigma)        2 0.000000        1       NA       NA
31    31                       MixN(p,m,d)        3 0.500000        0      1.0       NA
32    32                      TruncN(a,b)        2 0.000000        1       NA       NA
33    33                          Nout(a)        1 1.000000       NA       NA       NA
34    34              GEP(t1,t2,t3,crit)        4 0.500000        0      1.0    1e-06
35    35              Exponential(lambda)        1 1.000000       NA       NA       NA
36    36                  ALaplace(mu,b,k)        3 0.000000        1      2.0       NA
37    37         NIG(alpha,beta,delta,mu)        4 1.000000        0      1.0    0e+00
38    38       APD(theta,phi,alpha,lambda)        4 0.000000        1      0.5    2e+00
39    39 modAPD(mu,sigma,theta1,theta2)        4 0.000000        1      0.5    2e+00
40    40            LPtn(alpha,mu,sigma)        3 1.959964        0      1.0       NA

> 
```

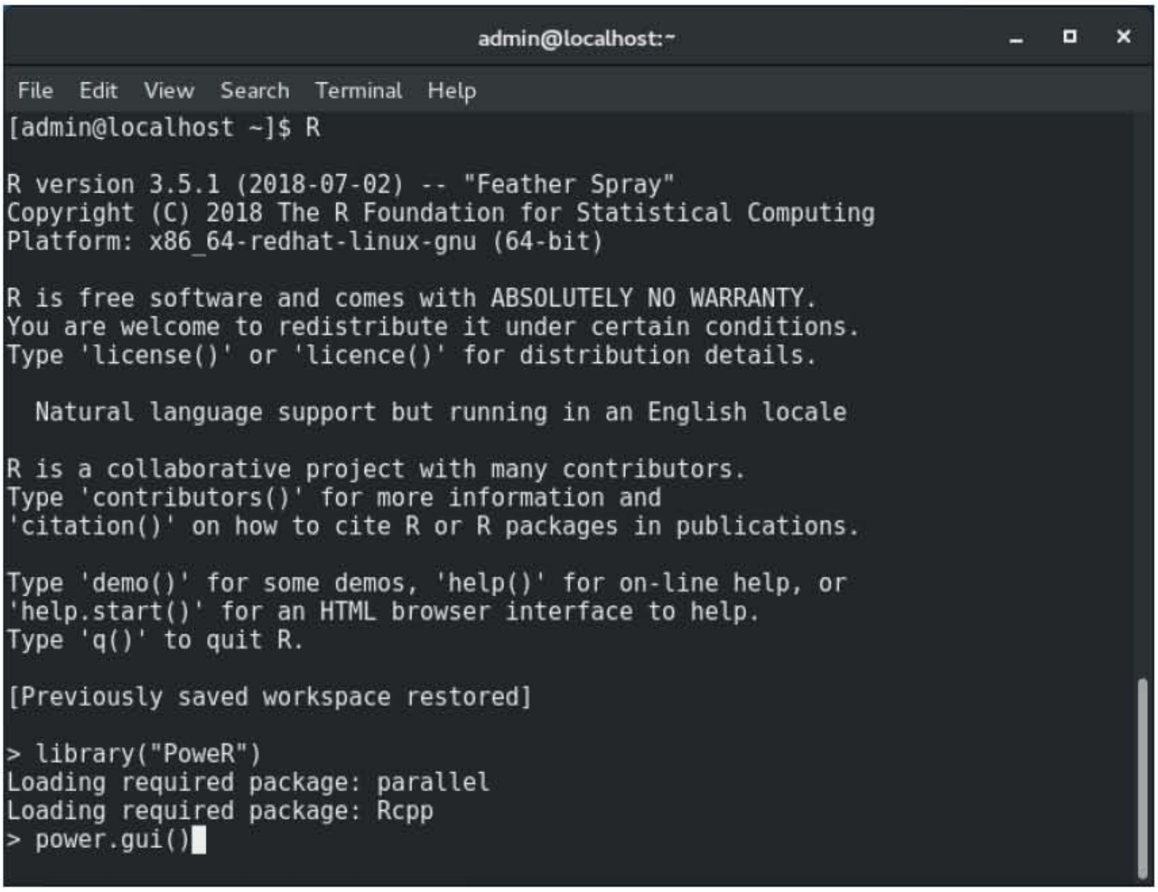


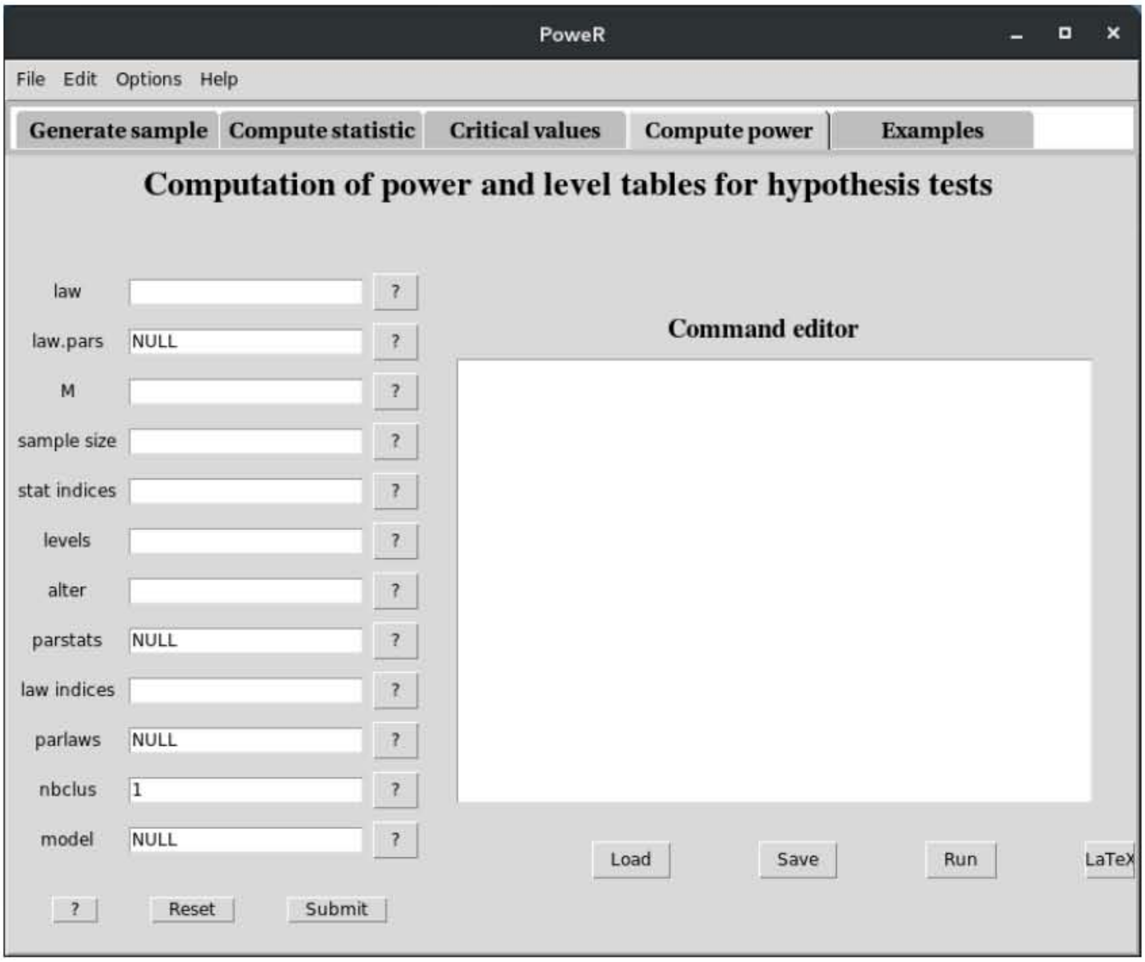
```
> getindex()[2]
$`mat.stats`
   Index                Stat Alter Nbparams
1      1                 K-S     3        0
2      2                AD^*     3        0
3      3                 Z_C     3        0
4      4                 Z_A     3        0
5      5                 P_S     3        0
6      6                 K^2     3        0
7      7                  JB     3        0
8      8                  DH     3        0
9      9                 RJB     3        0
10    10             T_{Lmom}     3        0
11    11       T_{Lmom}^{(1)}     3        0
12    12       T_{Lmom}^{(2)}     3        0
13    13       T_{Lmom}^{(3)}     3        0
14    14             BM_{3-4}     3        0
15    15             BM_{3-6}     3        0
16    16            T_{MC-LR}     3        0
17    17                 T_w 0,1,2        0
18    18        T_{MC-LR}-T_w     3        0
19    19              T_{S,5}     3        0
20    20              T_{K,5}     3        0
21    21                   W     4        0
22    22                  W'     4        0
23    23            tilde{W}     4        0
24    24                   D 0,1,2        0
25    25                   r     4        0
26    26                  CS     3        0
27    27                   Q 0,1,2        0
28    28                Q-Q* 0,1,2        0
29    29                BCMR     3        0
30    30            beta_3^2     3        0
31    31          T^*(alpha)     1        1
32    32                 I_n     3        0
33    33              R_{sJ}     3        0
34    34                  Q* 0,1,2        0
35    35                 R_n     3        0
36    36             X_{APD}     3        0
37    37             Z_{EPD} 0,1,2        0
38    38                 GLB     3        0
39    39              V_3-ML 0,1,2        0
40    40              V_4-ML 0,1,2        0
41    41                   S     3        0
42    42                 A^2     3        0
43    43                 W^2     3        0
44    44                 U^2     3        0
45    45            sqrt{n}D     3        0
46    46                   V     3        0
```

Au cas où cette image ne serait pas compréhensible voici un extrait de l'article de l'auteur d'origine:

	Test	Reference
1	Lilliefors $K-S$	Lilliefors (1967)
2	Anderson-Darling AD^*	D'Agostino and Stephens (1986)
3	Zhang-Wu Z_C	Zhang and Wu (2005)
4	Zhang-Wu Z_A	Zhang and Wu (2005)
5	Glen-Leemis-Barr P_S	Glen, Leemis, and Barr (2001)
6	D'Agostino-Pearson K^2	D'Agostino and Pearson (1973, 1974)
7	Jarque-Bera JB	Jarque and Bera (1987)
8	Doornik-Hansen DH	Doornik and Hansen (2008)
9	Gel-Gastwirth RJB	Gel and Gastwirth (2008)
10	Hosking T_{Lmom}	Hosking (1990)
11	Hosking $T_{Lmom}^{(1)}$	Hosking (1990)
12	Hosking $T_{Lmom}^{(2)}$	Hosking (1990)
13	Hosking $T_{Lmom}^{(3)}$	Hosking (1990)
14	Bontemps-Meddahi BM_{3-4}	Bontemps and Meddahi (2005)
15	Bontemps-Meddahi BM_{3-6}	Bontemps and Meddahi (2005)
16	Brys-Hubert-Struyf T_{MC-LR}	Brys, Hubert, and Struyf (2008)
17	Bonett-Seier T_w	Bonett and Seier (2002)
18	Combination of T_{MC-LR} & T_w	Brys *et al.* (2008), Bonett and Seier (2002)
19	Cabana-Cabana $T_{S,l}$	Cabaña and Cabaña (1994)
20	Cabana-Cabana $T_{K,l}$	Cabaña and Cabaña (1994)
21	Shapiro-Wilk W	Shapiro and Wilk (1965)
22	Shapiro-Francia W'	Shapiro and Francia (1972)
23	modified Shapiro-Wilk $\tilde{W}$	Rahman and Govindarajulu (1997)
24	D'Agostino D	D'Agostino (1971)
25	Filliben r	Filliben (1975)
26	Chen-Shapiro CS	Chen and Shapiro (1995)
27	Zhang Q	Zhang (1999)
28	Zhang $Q-Q^*$	Zhang (1999)
29	Barrio-Cuesta Albertos-Matran-Rodriguez $BCMR$	del Barrio *et al.* (1999)
30	Coin β_3^2	Coin (2008)
31	Epps-Pulley $T^*(\alpha)$	Epps and Pulley (1983)
32	Martinez-Iglewicz I_n	Martinez and Iglewicz (1981)
33	Gel-Miao-Gastwirth R_{sJ}	Gel, Miao, and Gastwirth (2007)
34	Zhang Q^*	Zhang (1999)
35	Desgagné-Lafaye de Micheaux-Leblanc P_1	Desgagné *et al.* (2009)
36	Desgagné-Lafaye de Micheaux-Leblanc P_2	Desgagné *et al.* (2009)
37	Desgagné-Lafaye de Micheaux-Leblanc S_2	Desgagné and Lafaye de Micheaux (2016)
38	Desgagné-Lafaye de Micheaux-Leblanc K_2	Desgagné and Lafaye de Micheaux (2016)
39	Desgagné-Lafaye de Micheaux-Leblanc X_{APD}	Desgagné and Lafaye de Micheaux (2016)
40	Desgagné-Lafaye de Micheaux-Leblanc	Desgagné *et al.* (2013)
41	Spiegelhalter S	Spiegelhalter (1977)

La suite est un peu cauchemardesque pour les utilisateurs de R sous Microsoft Windows. Dans l'idéal il faudrait utiliser Linux où ce package propose l'assistant que voilà via la fonction **power.gui( )**:


admin@localhost:~
File Edit View Search Terminal Help
[admin@localhost ~]$ R
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
> library("PoweR")
Loading required package: parallel
Loading required package: Rcpp
> power.gui()


PoweR
File Edit Options Help
Generate sample
Compute statistic
Critical values
Compute power
Examples
Computation of power and level tables for hypothesis tests
law
law.pars NULL
M
sample size
stat indices
levels
alter
parstats NULL
law indices
parlaws NULL
nbclus 1
model NULL
Command editor
Load
Save
Run
LaTeX
?
Reset
Submit

Si on utilise cet assistant, cela nous donne le code suivant une fois copié/collé dans l'éditeur R de Microsoft Windows:

```
> library("PoweR")
> #la loi de l'hypothese nulle: N(0,1)
> law.index <- 2
> #le nombre de simulation de Monte Carlo
> M <- 10^5
> #la taille des échantillons
> vectn <- c(10, 15, 20, 35, 50, 75, 100)
> #le seuil de confiance
> levels <- 0.05
> #les types de tests: Kolmogor-Smirnov, Anderson-Darling, Jarque-Bera, Shapiro-Wilk
> stat.indices <- c(1, 2, 7, 21)
> #quelques lois prises come tests
> #Laplace, Normal, Beta, Uniforme,Student,Chi-2,Log-Normal,Weibull
> law.indices <- c(1,2,6,7,8,9,10,11)
> #le type d'hypothèses alternatives (bilatéral, bilatéral sans val. extrêmes)
> alter <- create.alter(stat.indices,c(3,3,3,4,3))
> critval <- many.crit(law.index, stat.indices, M, vectn, levels, alter)
> table <- powcomp.fast(law.indices, stat.indices, vectn, M, levels,
+ critval = critval, alter)
>
```

Soit la longue table suivante:

```
> table
                         law   n level     K-S    AD^*     JB       W
1           Laplace(mu=0,b=1)  10  0.05  13.917  16.138  17.852  15.513
2                              15  0.05  17.900  21.667  24.451  20.829
3                              20  0.05  22.228  27.287  30.348  26.116
4                              35  0.05  32.980  41.670  44.278  39.732
5                              50  0.05  43.441  54.583  55.470  51.896
6                              75  0.05  57.671  70.646  69.774  67.735
7                             100  0.05  69.816  82.411  79.778  79.412
8      Normal(mu=0,sigma=1)    10  0.05   4.956   5.178   4.984   5.190
9                              15  0.05   4.877   5.005   4.993   5.003
10                             20  0.05   4.950   4.952   5.137   5.076
11                             35  0.05   4.841   4.928   4.951   4.927
12                             50  0.05   5.027   5.026   5.108   5.019
13                             75  0.05   4.852   4.818   5.130   4.919
14                            100  0.05   4.895   4.887   5.212   4.996
15              Beta(a=1,b=1)  10  0.05   6.102   7.826   1.560   8.350
16                             15  0.05   7.765  11.722   0.612  12.765
17                             20  0.05  10.019  17.065   0.293  20.212
18                             35  0.05  17.281  36.945   0.078  48.340
19                             50  0.05  26.188  58.154   0.947  75.215
20                             75  0.05  42.290  82.825  29.553  95.818
21                            100  0.05  58.604  94.896  75.507  99.636
22           Uniform(a=0,b=1)  10  0.05   6.189   8.047   1.622   8.501
23                             15  0.05   7.784  11.760   0.622  12.928
24                             20  0.05   9.975  17.057   0.259  20.177
25                             35  0.05  17.288  36.924   0.089  48.333
26                             50  0.05  26.116  58.002   0.941  75.076
27                             75  0.05  41.983  82.751  29.336  95.885
28                            100  0.05  58.451  95.058  75.455  99.635
29             Student-t(df=1) 10  0.05  57.764  61.481  58.862  59.587
30                             15  0.05  74.093  78.297  76.029  76.424
31                             20  0.05  84.487  87.949  86.174  86.574
32                             35  0.05  96.771  98.118  97.397  97.678
33                             50  0.05  99.383  99.745  99.536  99.635
34                             75  0.05  99.970  99.988  99.976  99.984
35                            100  0.05 100.000 100.000  99.998 100.000
```

La suite…:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

36           Chi-squared(df=1)  10  0.05  53.399  70.042  52.638  73.972
37                              15  0.05  74.919  89.670  71.646  92.615
38                              20  0.05  88.633  97.080  84.719  98.466
39                              35  0.05  99.485  99.973  98.635  99.992
40                              50  0.05  99.989  99.999  99.945 100.000
41                              75  0.05 100.000 100.000 100.000 100.000
42                             100  0.05 100.000 100.000 100.000 100.000
43 Lognormal(logmean=0,logsd=1)  10  0.05  45.873  58.100  50.257  61.095
44                              15  0.05  65.161  79.235  69.607  82.683
45                              20  0.05  79.148  90.598  82.752  93.259
46                              35  0.05  96.262  99.347  97.634  99.736
47                              50  0.05  99.557  99.971  99.810  99.993
48                              75  0.05  99.990  99.999  99.999 100.000
49                             100  0.05 100.000 100.000 100.000 100.000
50      Weibull(shape=1,scale=1)  10  0.05  29.799  41.406  33.578  44.877
51                              15  0.05  44.534  61.885  49.424  67.764
52                              20  0.05  58.559  77.615  63.145  83.721
53                              35  0.05  85.237  96.615  88.282  98.647
54                              50  0.05  96.201  99.678  97.685  99.956
55                              75  0.05  99.768  99.998  99.939 100.000
56                             100  0.05  99.994 100.000  99.998 100.000
                n level    K-S   AD^*     JB      W
Average power  10  0.05 27.250 33.527 27.669 34.636
2              15  0.05 37.129 44.905 37.173 46.376
3              20  0.05 44.750 52.450 44.103 54.200
4              35  0.05 56.268 64.315 53.918 67.173
5              50  0.05 61.988 71.895 57.430 75.849
6              75  0.05 68.316 80.128 66.713 83.043
7             100  0.05 73.970 84.656 79.493 85.460
              n level    K-S  AD^*     JB     W
Average gap  10  0.05  7.915 1.637  7.496 0.529
2            15  0.05  9.934 2.158  9.890 0.687
3            20  0.05 10.159 2.458 10.805 0.708
4            35  0.05 11.531 3.484 13.881 0.626
5            50  0.05 14.333 4.426 18.890 0.472
6            75  0.05 15.118 3.305 16.720 0.391
7           100  0.05 11.892 1.205  6.368 0.402
            n level    K-S   AD^*     JB     W
Worst gap  10  0.05 20.573  3.930 21.334 2.339
2          15  0.05 23.230  5.879 20.969 3.622
3          20  0.05 25.162  6.106 20.576 4.232
4          35  0.05 31.059 11.409 48.262 4.546
5          50  0.05 49.027 17.074 74.268 3.574
6          75  0.05 53.902 13.134 66.549 2.911
7         100  0.05 41.184  4.740 24.180 2.999
> |
```

On peut même demander à ce package d'avoir cette table en LaTeX:

RGui (64-bit) - [R Console]

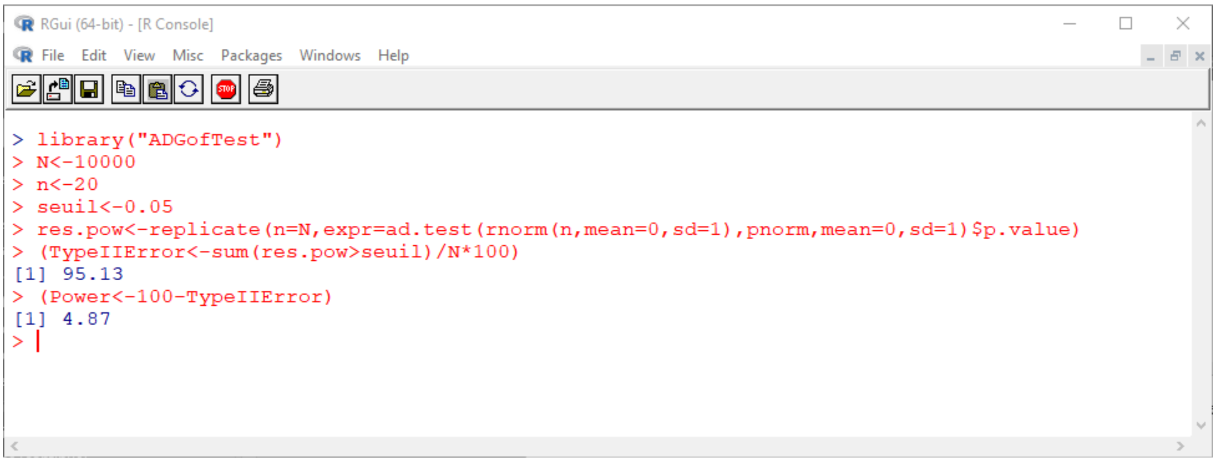
File Edit View Misc Packages Windows Help

```
\begin{table}[ht]
\caption[]{Power of $K-S$, $AD^*$, $JB$  and  $W$ tests}
\begin{center}
\begin{tabular}{l r r r r r r}
\hline
& & & \multicolumn{4}{c}{\textbf{Goodness-of-fit tests}} \\
\cline{4-7} \\ [-1.5ex]
\textbf{Alternative} & $n$ & $\alpha$ & $K-S$ & $AD^*$ & $JB$ & $W$ \\
\hline \\ [-1.5ex]
Laplace(mu=0,b=1) & 10 & 0.05 & 14.068 & 15.63 & 17.609 & 15.152 \\
 & 15 & 0.05 & 18.637 & 21.885 & 24.355 & 21.144 \\
 & 20 & 0.05 & 22.33 & 27.185 & 29.902 & 25.834 \\
 & 35 & 0.05 & 33.217 & 41.933 & 44.267 & 40.225 \\
 & 50 & 0.05 & 43.204 & 54.62 & 55.647 & 52.322 \\
 & 75 & 0.05 & 59.057 & 71.434 & 70.014 & 68.678 \\
 & 100 & 0.05 & 71.07 & 82.95 & 80.149 & 80.056 \\
Normal(mu=0,sigma=1) & 10 & 0.05 & 5.002 & 5.013 & 4.902 & 4.918 \\
 & 15 & 0.05 & 5.138 & 4.961 & 4.981 & 5.005 \\
 & 20 & 0.05 & 4.943 & 4.927 & 4.909 & 4.919 \\
 & 35 & 0.05 & 5.074 & 5.009 & 4.979 & 5.143 \\
 & 50 & 0.05 & 5.067 & 4.961 & 5.006 & 5.071 \\
 & 75 & 0.05 & 5.137 & 5.059 & 5.041 & 5.286 \\
 & 100 & 0.05 & 5.258 & 5.203 & 5.013 & 5.166 \\
Beta(a=1,b=1) & 10 & 0.05 & 6.314 & 7.976 & 1.679 & 8.379 \\
 & 15 & 0.05 & 8.093 & 11.852 & 0.592 & 13.1 \\
 & 20 & 0.05 & 10.11 & 16.829 & 0.263 & 19.641 \\
 & 35 & 0.05 & 17.474 & 37.264 & 0.074 & 49.073 \\
 & 50 & 0.05 & 26.125 & 58.165 & 0.901 & 75.606 \\
 & 75 & 0.05 & 43.224 & 83.552 & 28.719 & 96.216 \\
 & 100 & 0.05 & 59.46 & 95.158 & 74.804 & 99.656 \\
Uniform(a=0,b=1) & 10 & 0.05 & 6.397 & 7.888 & 1.672 & 8.3 \\
 & 15 & 0.05 & 8.127 & 11.817 & 0.615 & 12.968 \\
 & 20 & 0.05 & 10.012 & 16.886 & 0.313 & 19.692 \\
 & 35 & 0.05 & 17.447 & 37.124 & 0.088 & 48.951 \\
 & 50 & 0.05 & 25.858 & 57.782 & 0.907 & 75.326 \\
 & 75 & 0.05 & 43.197 & 83.297 & 28.665 & 96.021 \\
 & 100 & 0.05 & 59.358 & 95.047 & 74.505 & 99.636 \\
Student-t(df=1) & 10 & 0.05 & 57.516 & 61.024 & 58.511 & 59.073 \\
 & 15 & 0.05 & 74.566 & 78.377 & 75.946 & 76.319 \\
 & 20 & 0.05 & 84.395 & 87.999 & 86.099 & 86.545 \\
 & 35 & 0.05 & 96.798 & 98.128 & 97.409 & 97.727 \\
 & 50 & 0.05 & 99.357 & 99.703 & 99.568 & 99.635 \\
 & 75 & 0.05 & 99.968 & 99.991 & 99.979 & 99.985 \\
 & 100 & 0.05 & 99.997 & 100 & 99.999 & 100 \\
Chi-squared(df=1) & 10 & 0.05 & 53.421 & 69.674 & 52.129 & 73.46 \\
 & 15 & 0.05 & 75.269 & 89.492 & 70.982 & 92.562 \\
```

On peut facilement "à la main" retomber sur les valeurs calculées par ce package. Par exemple reproduisons la valeur suivante:

Normal(mu=0,sigma=1)	10	0.05	4.956	5.178	4.984	5.190
	15	0.05	4.877	5.005	4.993	5.003
	20	0.05	4.950	4.952	5.137	5.076
	35	0.05	4.841	4.928	4.951	4.927
	50	0.05	5.027	5.026	5.108	5.019
	75	0.05	4.852	4.818	5.130	4.919
	100	0.05	4.895	4.887	5.212	4.996

Cela donne alors:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("ADGofTest")
> N<-10000
> n<-20
> seuil<-0.05
> res.pow<-replicate(n=N,expr=ad.test(rnorm(n,mean=0,sd=1),pnorm,mean=0,sd=1)$p.value)
> (TypeIIError<-sum(res.pow>seuil)/N*100)
[1] 95.13
> (Power<-100-TypeIIError)
[1] 4.87
>

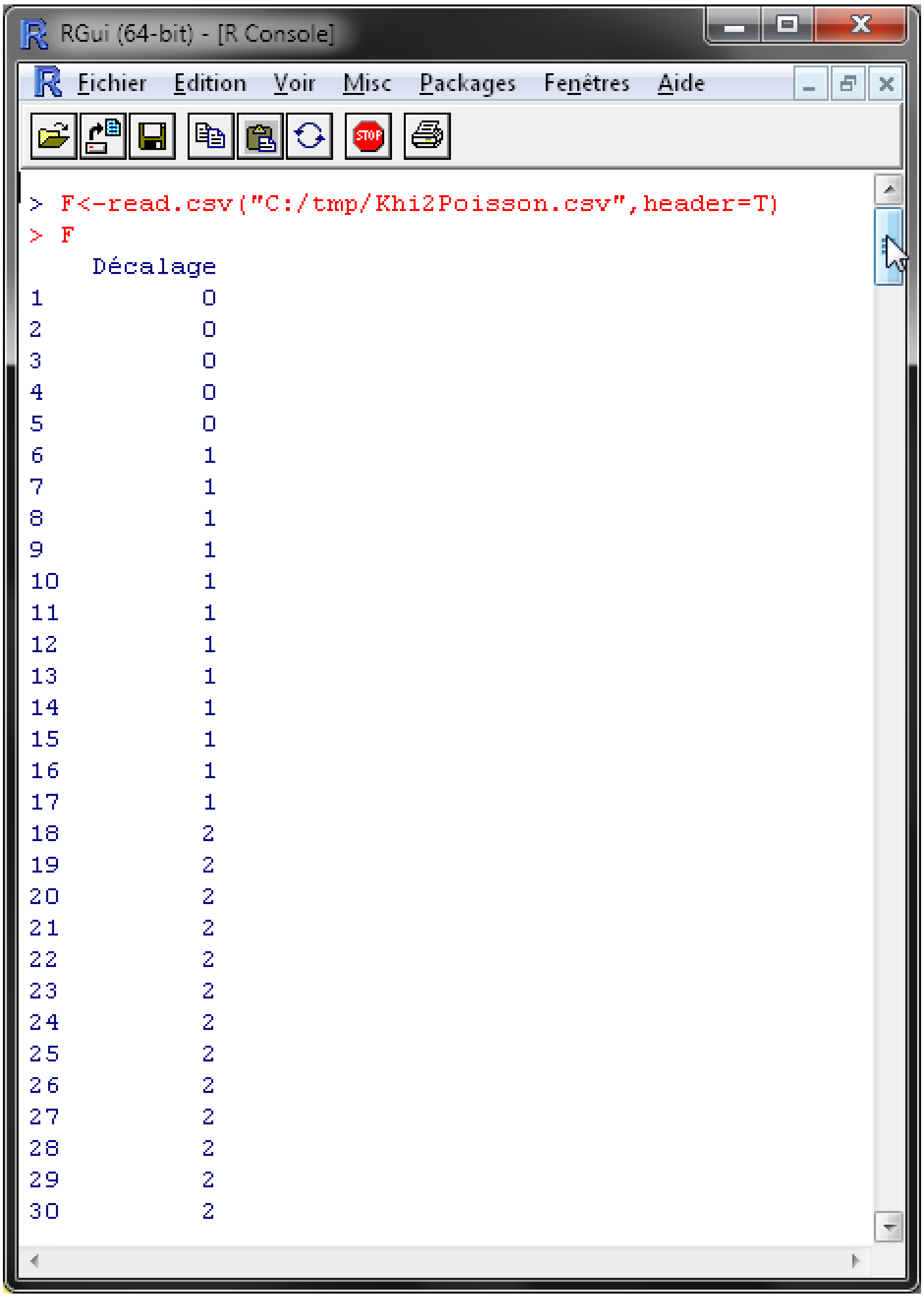
Test d'adéquation d'une loi de Poisson (ou binomiale) par la méthode de minimisation du Khi-2

R 3.2.0

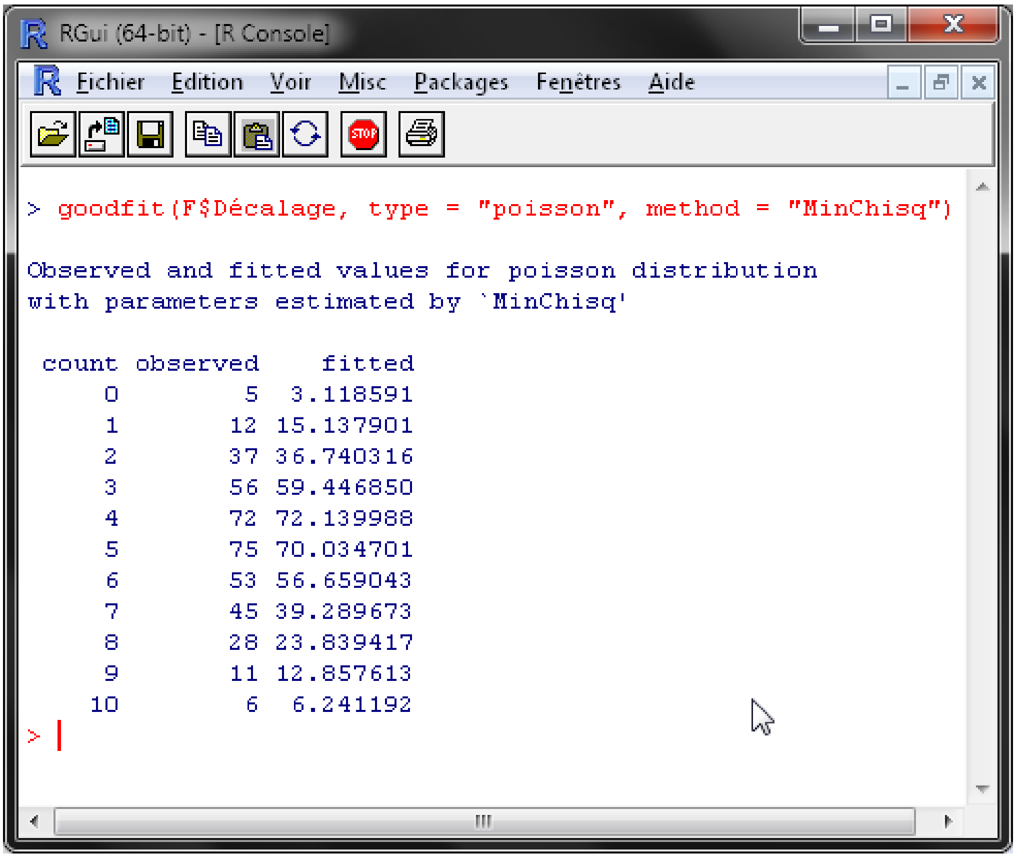
Nous voulons à nouveau vérifier les calculs effectués à la main dans le cours théorique (avec l'aide de MS Excel) et comparer aussi les résultats avec Minitab.

Donc d'abord, R a comme Minitab pas la possibilité à ce jour d'imposer l'espérance de loi de Poisson donc il va en calculer l'estimateur.

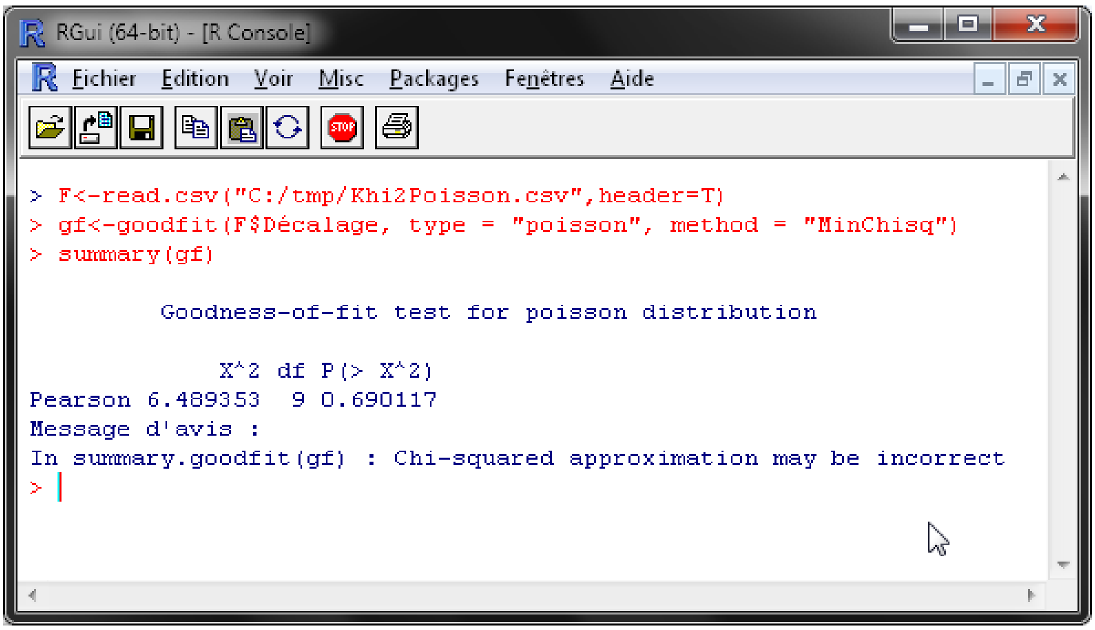
Donc d'abord nous importons le jeu de données histoire de pas l'écrire à la main:



Ensuite, nous installons le package **vcd** et utilisons la fonction **goodfit( )** de ce dernier (fonction qui ne prend à ce jour que trois distributions discrètes possibles en entrée) pour dans un premier temps avoir un tableau de fréquences observées et théoriques:



Bon c'est bien joli même si c'est conforme aux calculs faits à la main, mais allons au point qui nous intéresse:



Nous avons bien les mêmes résultats qu'avec Minitab et nous sommes très proches du calcul effectué "à la main".

Test d'adéquation d'une loi de Poisson (ou binomiale) par la méthode du maximum de vraisemblance

R 3.5.1

Nous avons vu dans le cours théorique une autre manière de faire un ajustement de certaines lois ou modèles appelées "test d'ajustement par la méthode du maximum de vraisemblance". Nous avons également discuté du débat qu'il y a entre cette méthode et la précédente…

Même si nous n'avons pas fait les calculs à la main dans le cours théorique par fénéantise, voyons toutefois comment exécuter un test de ce type en utilisant à nouveau le package **vcd** et la fonction **goodfit( )**:

```
> library("vcd")
> F<-read.csv("c:/tmp/Khi2Poisson.csv",header=T)
> gf<-goodfit(F$Décalage,type="poisson",method="ML")
> gf

Observed and fitted values for poisson distribution
with parameters estimated by `ML'

 count observed    fitted pearson residual
     0        5  3.186632       1.01582759
     1       12 15.399399      -0.86626397
     2       37 37.208798      -0.03422968
     3       56 59.937172      -0.50855301
     4       72 72.411595      -0.04836894
     5       75 69.985807       0.59937145
     6       53 56.367735      -0.44856220
     7       45 38.913869       0.97563890
     8       28 23.506409       0.92683070
     9       11 12.621636      -0.45645237
    10        6  6.099405      -1.37924584
> summary(gf)

         Goodness-of-fit test for poisson distribution

                      X^2 df  P(> X^2)
Likelihood Ratio 13.17236  9 0.1549612
> |
```

A comparer avec la méthode précédente…

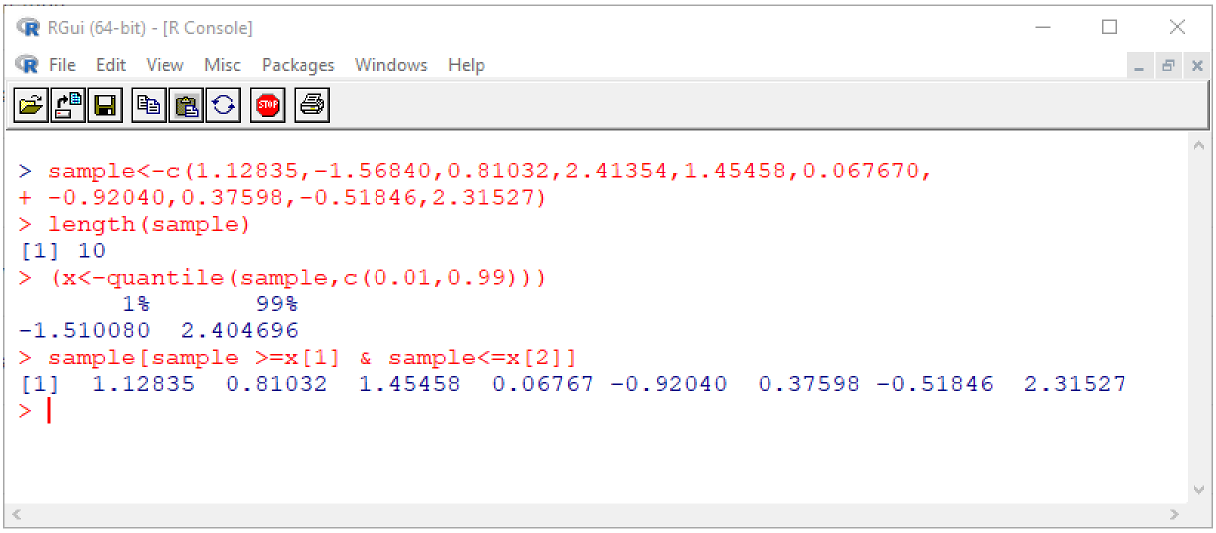
Exemple 182.: Tests des valeurs aberrantes

Nous savons que les tests de normalité peuvent échouer pour de nombreuses raisons (distribution non-normale, mesures expérimentales inadéquates, etc.) mais une qui est assez facile à identifier et résoudre est la présence (et donc l'élimination) de valeurs aberrantes.

Valeurs aberrantes avec les centiles

R 3.6.1

On peut éliminer de manière brute les valeurs aberrantes selon des critères de centiles simples et ce sans packages comme l'illustre l'exemple suivant:



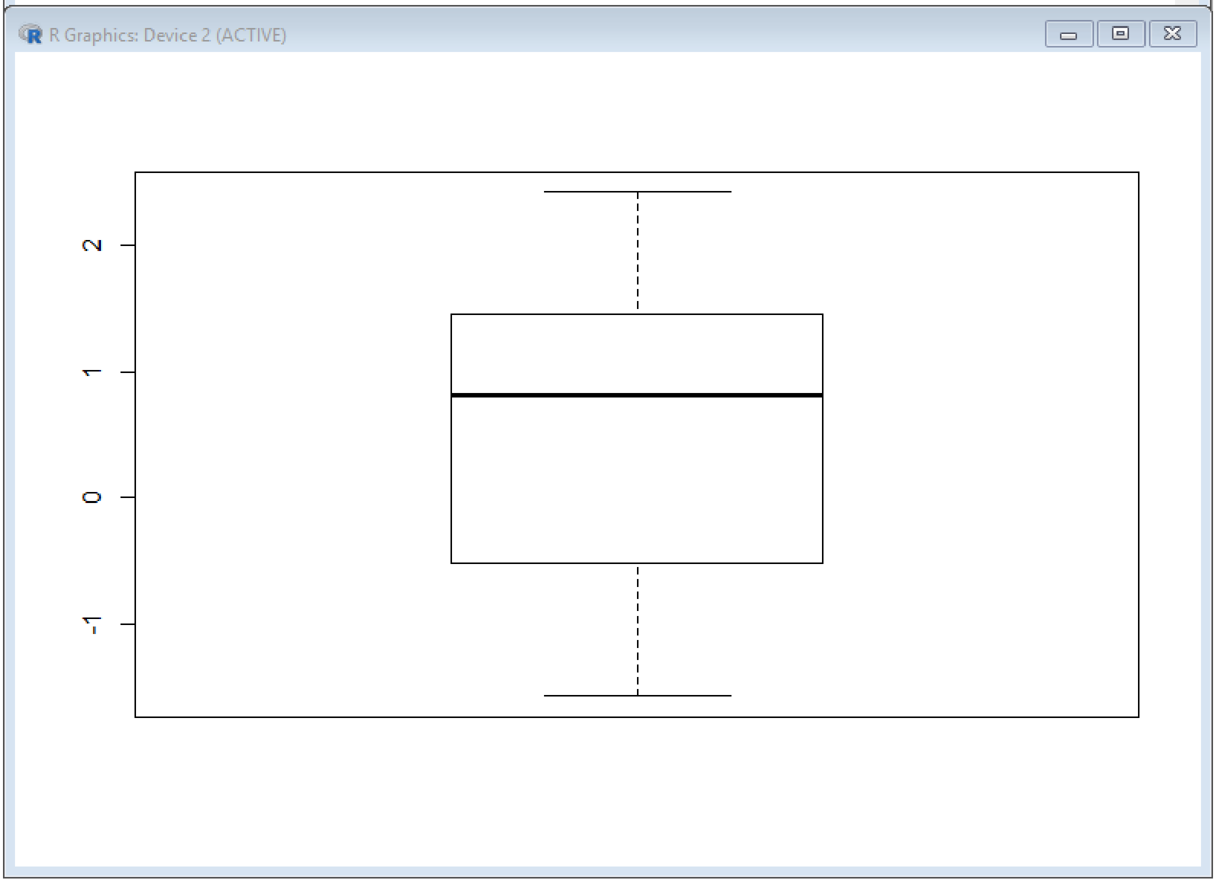
Valeurs aberrantes avec les centiles des Box-Plot

R 3.5.1

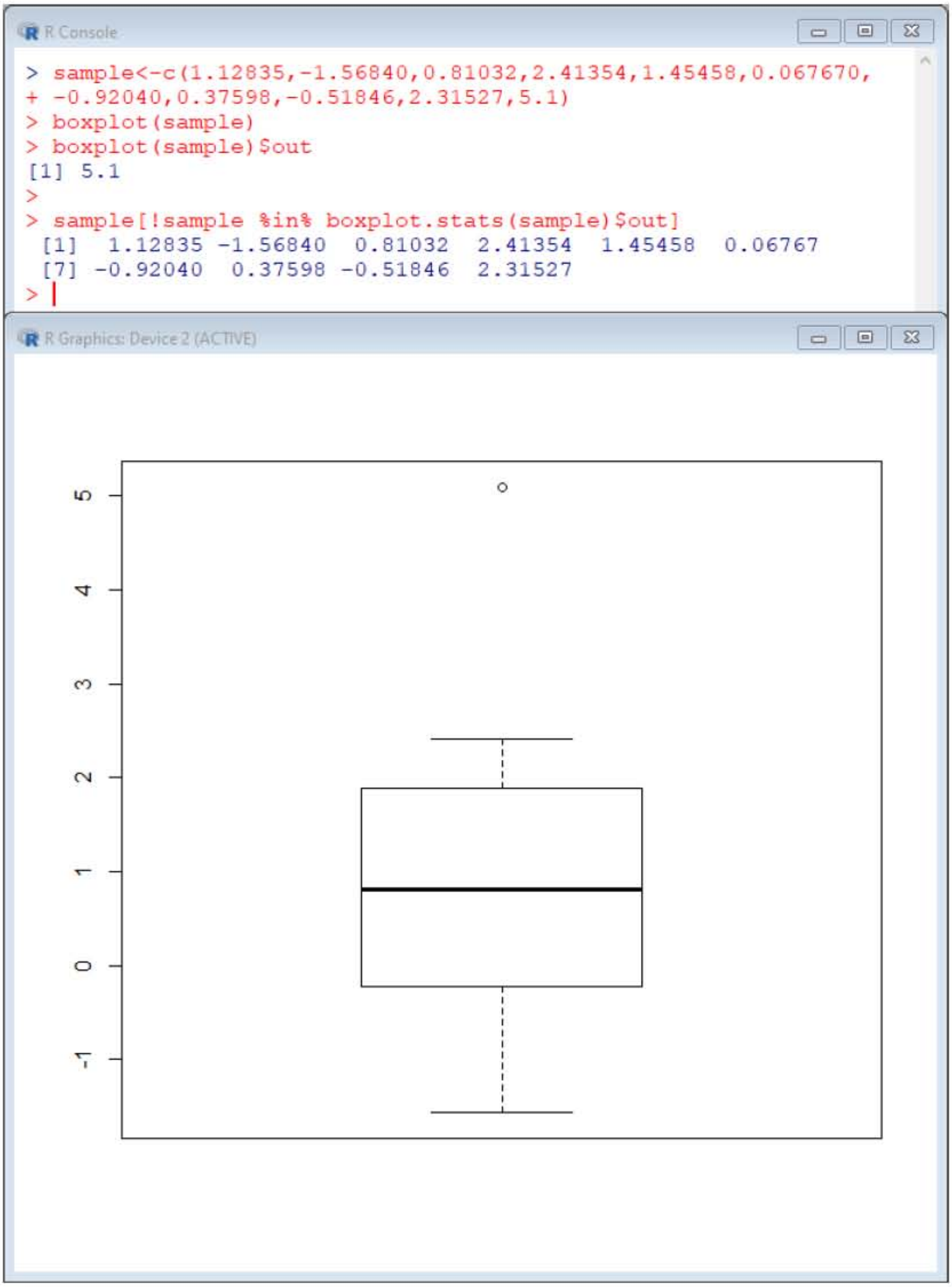
Pas grand-chose à dire ici puisqu'il s'agit d'un outil principalement visuel et basé sur les centiles! Cependant à noter… Les box-plots utilisent tous des indicateurs (médiane, quartiles, centiles), qui sont des indicateurs non-paramétriques (!). Si nous avons placé ce sujet ici, c'est juste parce que cela semble pédagogiquement plus convenable (sinon quoi elle serait dans le chapitre sur les techniques non-paramétriques!)..

Donc voyons à nouveau la fonction **boxplot( )** à l'œuvre:

```
> sample<-c(1.12835,-1.56840,0.81032,2.41354,1.45458,0.067670
+ -0.92040,0.37598,-0.51846,2.31527)
> boxplot(sample)
> boxplot(sample)$out
numeric(0)
> #donc aucun outlier dans cet échantillon!
> 
```



On peut rajouter une valeur aberrante et ensuite voir comment l'enlever:


R Console
> sample<-c(1.12835,-1.56840,0.81032,2.41354,1.45458,0.067670,
+ -0.92040,0.37598,-0.51846,2.31527,5.1)
> boxplot(sample)
> boxplot(sample)$out
[1] 5.1
>
> sample[!sample %in% boxplot.stats(sample)$out]
[1] 1.12835 -1.56840 0.81032 2.41354 1.45458 0.06767
[7] -0.92040 0.37598 -0.51846 2.31527
>
R Graphics: Device 2 (ACTIVE)
5
4
3
2
1
0
-1

Valeurs aberrantes avec les centiles des bagplots (cas 2D)

R 3.5.1

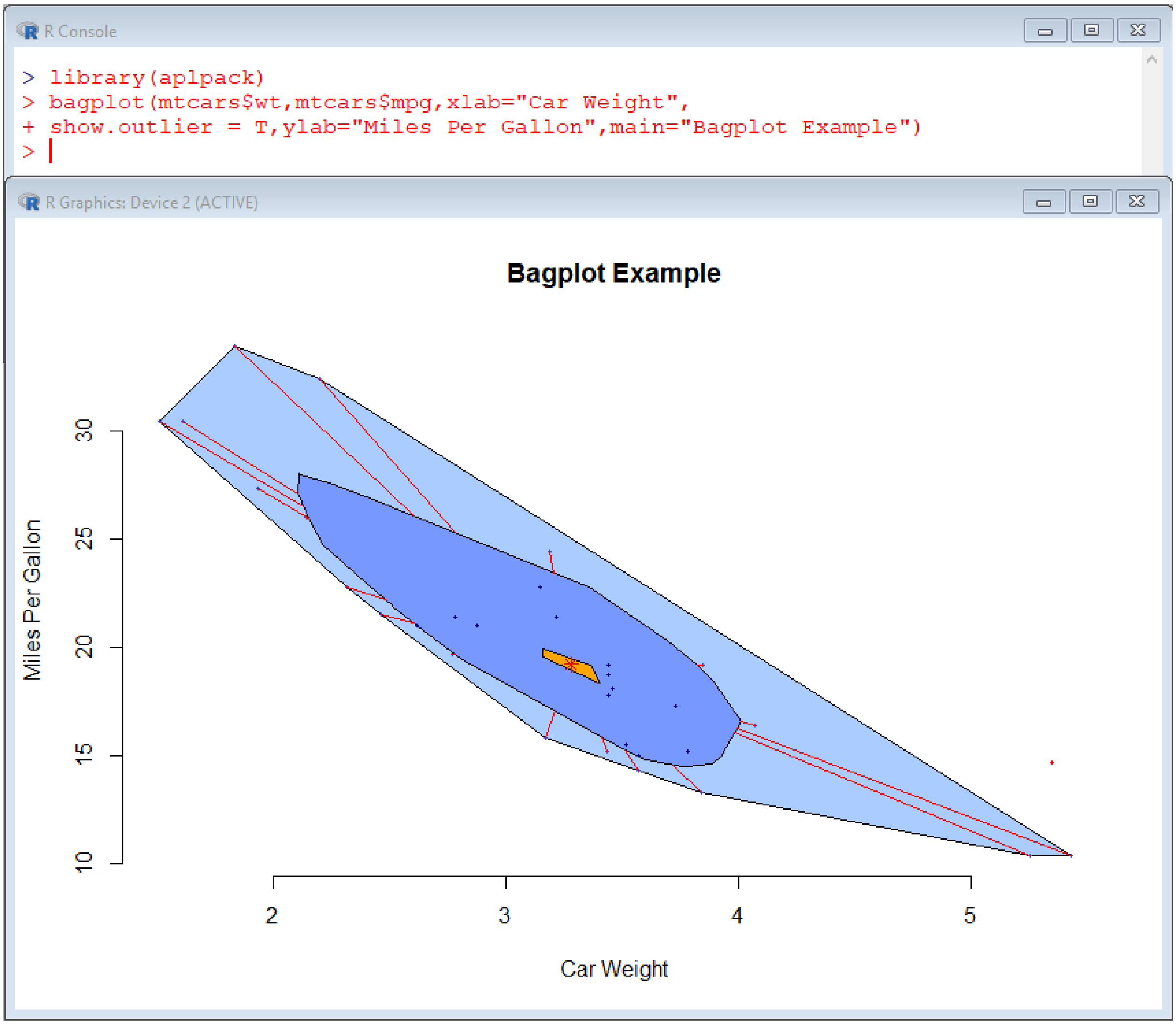
Bon là c'est pour info car:

1. Je n'ai jamais vu de mes yeux qui ce soit l'utiliser (ce qui ne veut pas dire que c'est inutile!)

2. Je n'ai aucune idée de comment ça marche et je n'ai pas envie de le savoir (faut donc pas nous poser de questions là-dessus)

Il semblerait que cette technique utilise elle aussi que des indicateurs non-paramétriques (!), donc elle est aussi présente dans ce chapitre que pour des raisons pédagogiques (sinon quoi elle serait dans le chapitre sur les techniques non-paramétriques!).

Donc voyons à la fonction **bagplot( )** du package **aplpack** à l'œuvre:

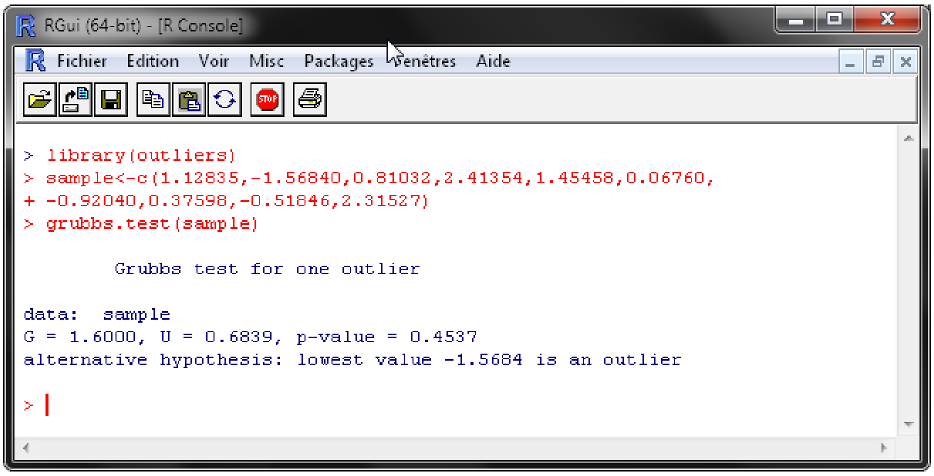


Test de Grubbs (test des valeurs aberrantes de Grubbs)

R 3.0.2

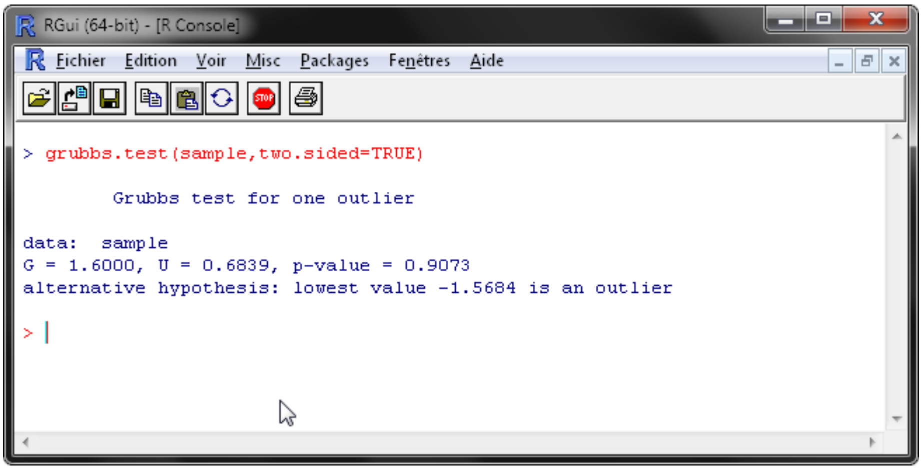
Nous allons ici vérifier validité du test de Grubbs qui est implémenté dans R que nous avons vu dans le cours théorique et dont nous avons appris à calculer les valeurs critiques à la main pour n'importe quelle distribution. Nous allons aussi comparer le résultat à ce que renvoie Minitab.

Nous utilisons alors la commande **grubbs.test( )** du package **outliers**:

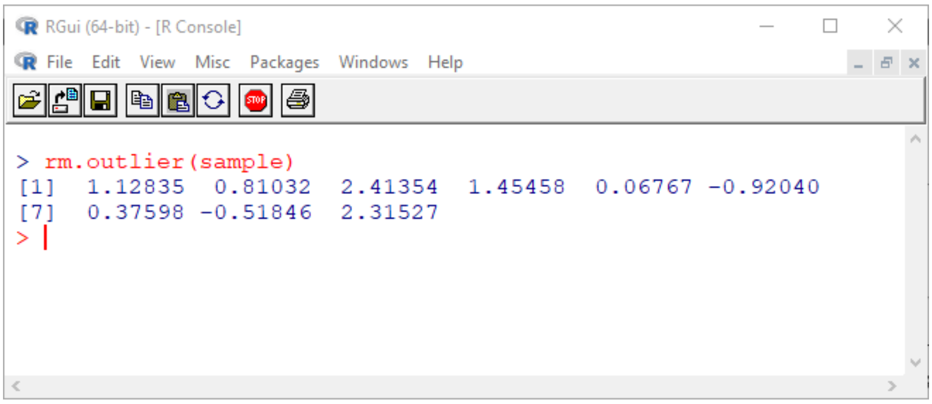


Nous retrouvons donc la même valeur de G que celles calculées à la main par Monte-Carlo dans Microsoft Excel et que dans Minitab (à partir de la version 17 pour ce dernier). Par contre pour les p-value nous en sommes à des kilomètres (il y a 50% de différence).

Au fait c'est un piège car il faut ajouter un paramètre pour retrouver les valeurs que nous connaissons (la fonction faisant un test unilatéral par défaut):



Toujours avec le package **outlier** et sa fonction **rm.outlier( )** on peut supprimer les outliers:

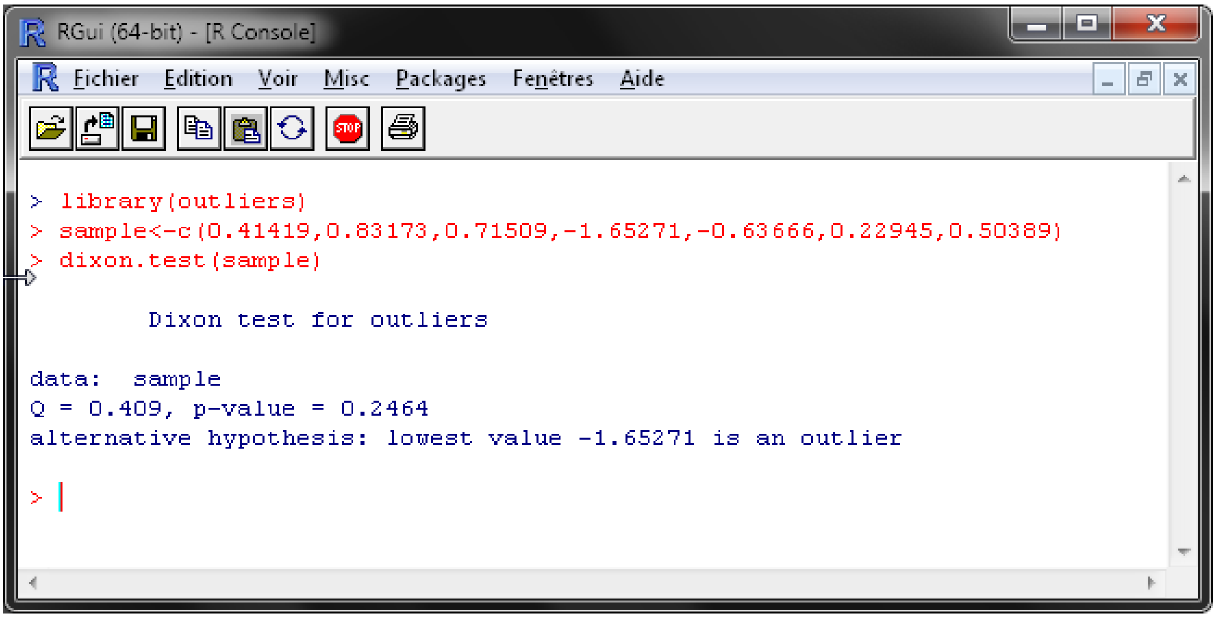

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> rm.outlier(sample)
[1] 1.12835 0.81032 2.41354 1.45458 0.06767 -0.92040
[7] 0.37598 -0.51846 2.31527
>

Test de Dixon (test des valeurs aberrantes/outliers de Dixon)

R 3.0.2

Nous allons ici vérifier validité du test de Dixon qui est implémenté dans R que nous avons vu dans le cours théorique et dont nous avons appris à calculer les valeurs critiques à la main pour n'importe quelle distribution. Nous allons aussi comparer le résultat à ce que renvoie Minitab.

Nous utilisons alors la commande **dixon.test( )** du package **outliers** qui par défaut fait un test en bilatéral (contrairement à la fonction grubbs.test):



Nous retrouvons donc exactement les mêmes valeurs que celles calculées à la main par Monte-Carlo dans Microsoft Excel et que dans Minitab (à partir de la version 17 pour ce dernier).

Le point qui est dommage cependant c'est que contrairement à Minitab, nous ne pouvons pas choisir le niveau de confiance (du moins à ce jour).

Test α-Poisson des valeurs aberrantes/outliers dans une table de contingence

R 3.0.2

Dans de nombreuses situations du management (des "affaires" ou du "business" si vous préférez…), ou des sciences sociales ainsi que des statistiques gouvernementales, il arrive de devoir jongler avec des tables de contingences de dimensions quelconques et de devoir y identifier des valeurs aberrantes.

Nous n'avons pas étudié ce sujet dans le cours théorique car il est principalement "computationnel" (ie: il nécessite des calculs intensifs). Voyons donc comment effectuer un test dans R en utilisant le package **alphaOutlier** et sa fonction **aout.conttab( )** :

```
> library("alphaOutlier")
> HairEyeColor[,,1]
       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8
> aout.conttab(data = HairEyeColor[,,1], param = "ML", alpha = 0.01, show.estimates = TRUE)
   data is.outlier     param
1    32       TRUE 19.670251
2    53      FALSE 50.229391
3    10      FALSE 11.942652
4     3       TRUE 16.157706
5    11      FALSE 20.272401
6    50      FALSE 51.767025
7    10      FALSE 12.308244
8    30       TRUE 16.652330
9    10      FALSE  9.433692
10   25      FALSE 24.089606
11    7      FALSE  5.727599
12    5      FALSE  7.749104
13    3      FALSE  6.623656
14   15      FALSE 16.913978
15    7      FALSE  4.021505
16    8      FALSE  5.440860
>
```

Comme nous pouvons le constater, le résultat est (comme souvent) contre intuitif pour la majorité des lecteurs et la majorité des cerveaux.

Exemple 183.: Transformations de variables

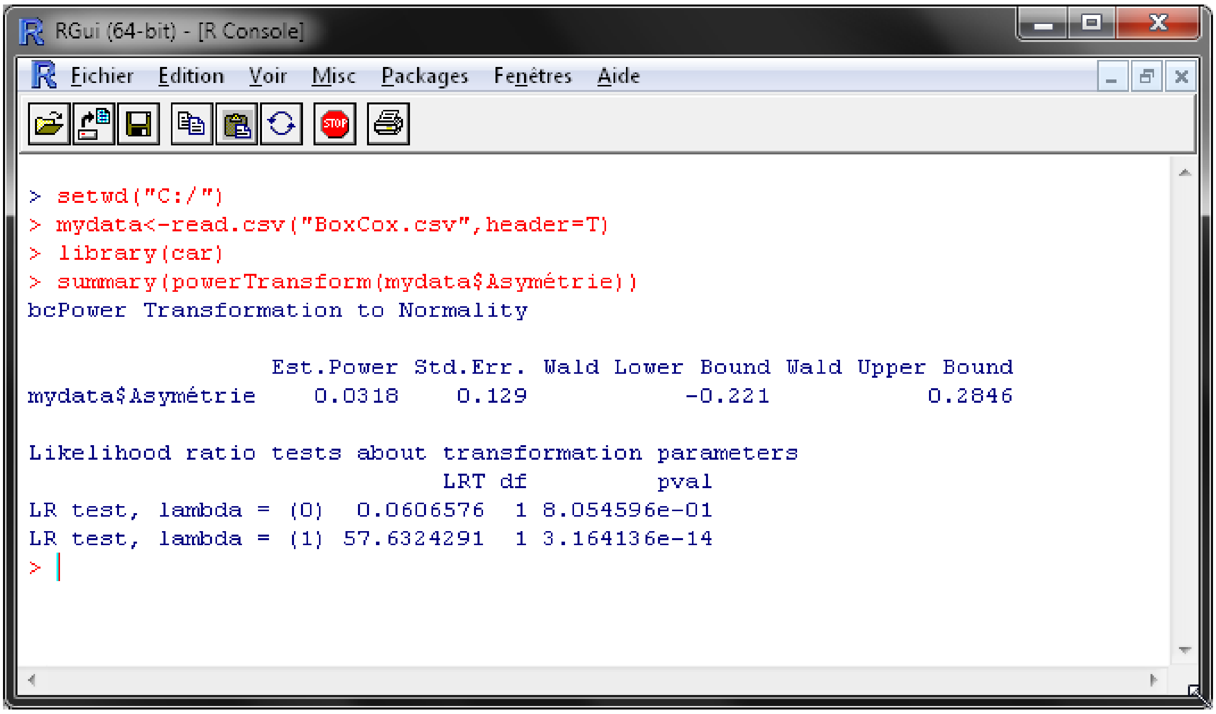
R 3.0.2

Dans le cours théorique nous avons mentionné les transformations empiriques de Box-Cox et Yale-Johnson qui permettent de travailler toutefois sur des tests statistiques importants nécessitant la normalité et nous avons mis en pratique ces derniers dans Minitab. Voyons comment faire la même chose avec R!

Transformation de Box-Cox

R 3.0.2

Ici nous allons juste contrôler que nous retrouvons le même résultat de transformation (empirique) de Box-Cox que dans le cours Minitab. Nous utilisons pour cela un fichier contenant 125 mesures **toutes positives définies** dans une seule colonne et utilisons la commande **powerTransform( )** du package **car**:



Donc cette commande suggère qu'on peut normaliser cette variable en la mettant à la puissance 0.0318. Nous retrouvons donc la même valeur que dans Minitab à la différence que R est beaucoup plus précis et donne la p-value de l'utilité de la transformation.

Nous pouvons ensuite utiliser notre connaissance des graphiques pour observer le résultat qualitativement:

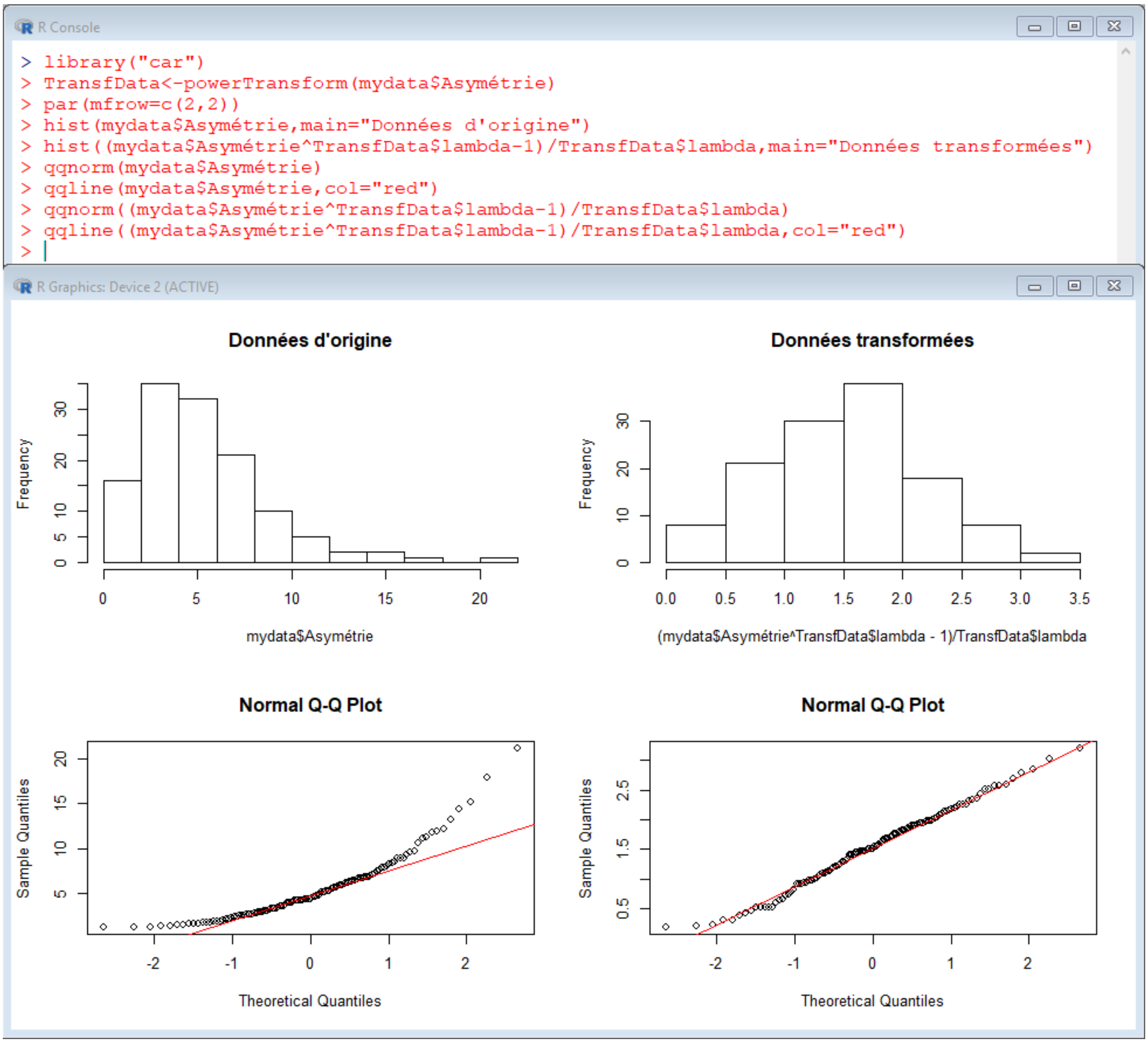


Figure 222 Graph q-q plot de données transformées (avant/après)

Nous pouvons aussi faire un test d'Anderson-Darling avant/après:

```
> library("car")
> ad.test(mydata$Asymétrie)

        Anderson-Darling normality test

data:  mydata$Asymétrie
A = 3.6822, p-value = 3.068e-09

> ad.test((mydata$Asymétrie^TransfData$lambda-1)/TransfData$lambda)

        Anderson-Darling normality test

data:  (mydata$Asymétrie^TransfData$lambda - 1)/TransfData$lambda
A = 0.24891, p-value = 0.7432

> |
```

Sinon, il y a une manière plus directe pour transformer un vecteur de données avec la transformation de Box-Cox avec le package **caret** et la fonction **preProcess( )** comme nous le verrons lors de notre introduction au Machine Learning à la page 2050.

Transformations de Johnson (Yeo-Johnson)

R 3.0.2

Bon le but ici ne sera pas de débattre de la justesse de ces transformations (ou de leur origine scientifique) qui ont donc pour rappel de "normaliser"[21] des données. Ceci a aussi déjà été fait en cours et de plus, il suffit de Googler un peu pour se faire sa propre opinion sur cette technique "d'ingénierie statistique".

Les transformations proposées sont les mêmes que pour Minitab mais avec une notation différente pour les coefficients. Donc si jamais voici un extrait du package:

> The Johnson system (Johnson 1949) is a very flexible system for describing statistical distributions. It is defined by
>
> $$z = \gamma + \delta \log f(u), u = (x - \xi)/\lambda$$
>
> and where $f()$ has four possible forms:
>
> SL: $f(u) = u$ the log normal
> SU: $f(u) = u + \sqrt{1 + u^2}$ an unbounded distribution
> SB: $f(u) = u/(1 - u)$ a bounded distribution
> SN: $\exp(u)$ the normal

Nous reprenons donc les mêmes données que pour la transformation de Box-Cox et utilisons la commande **RE.Johnson( )** du package **Johnson**:

[21] Dans le sens: faire que la distribution ressemble le plus possible à une loi Normale

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> mydata<-read.csv("c:/tmp/Johnson.csv",header=T,sep=";")
> library("Johnson")
> print(RE.Johnson(mydata$Asymétrie))
[[1]]
[1] "Johnson Transformation"

$`function`
[1] "SB"

$p
[1] 0.8855616

$transformed
  [1] -0.7795899655  0.3197706016  0.2003421347 -1.4407310753 -0.6891497014  0.5550079913
  [7]  0.3970087488  0.2161383659 -0.6537811064 -0.4795806295 -0.0008394425  1.1302044129
 [13]  0.8575438925  1.7466184278  0.4250469439 -0.8108308706 -0.1140903650  0.7647341607
 [19] -0.0486705064 -0.0439025255  1.4382462632 -1.1619016606 -0.8916019580 -0.1173707327
 [25]  0.6615474569 -0.3412953906  2.7546531446  2.3594738795 -0.7930629138 -0.3496759653
 [31] -0.8690820276  0.2145449210  0.2052739267 -1.7361173211  0.4084217666  0.6125645420
 [37]  0.8545844307 -0.5384528488 -0.8950878559  1.3403005864 -0.1320183929  0.3598749975
 [43]  0.4999277983 -1.3169314841  0.0435761495 -1.4193751787 -0.0801577289  0.4789831252
 [49] -1.0136207556 -1.5121654346 -0.0521961416 -0.8483603486  0.5168224468  0.7043974856
 [55] -0.1896353643 -0.3713609943 -1.4345850541  0.3966950441 -1.5616883657  1.4513751813
 [61] -0.1273580535  0.3112613794 -0.1261861679 -0.1446430428 -1.2475139543 -0.6436482619
 [67] -0.4533525513 -0.6076663955 -1.6090900283  0.5841830635 -0.5860191084  0.5102992942
 [73] -0.0640142724  0.3279131215  1.1771959097 -0.3154176023 -0.8017294121 -0.2105508140
 [79]  1.0466233303  0.4517959561 -0.5836483174  0.0803706608  0.1673266682  0.5786257541
 [85]  0.9372266089  1.5957405342 -1.0829250366  0.2968309572  0.8020192767 -0.4882885190
 [91]  0.6302309352 -1.7396482829  0.9821745486 -0.2056599490  0.8839882491  1.1844287263
 [97] -0.1180064026 -1.8270301620  0.5808743008 -1.8967358321 -1.4237879044 -0.8950878559
[103]  2.0178972910 -1.8762046865  1.5561705022  0.5364007290  1.5365571161 -1.4398876043
[109] -1.2591538414  1.9140759796 -0.6397120901  0.0119762011 -0.2073010310  0.0973701731
[115]  0.1966559999  0.6846486787 -0.4736109994 -0.4273854931 -1.1349880626  0.9510728483
[121]  0.6308070191  1.0513683823  1.0418679387  0.0302584869 -0.7373594274

$f.gamma
[1] 2.955939

$f.lambda
[1] 45.651

$f.epsilon
[1] 0.02256813

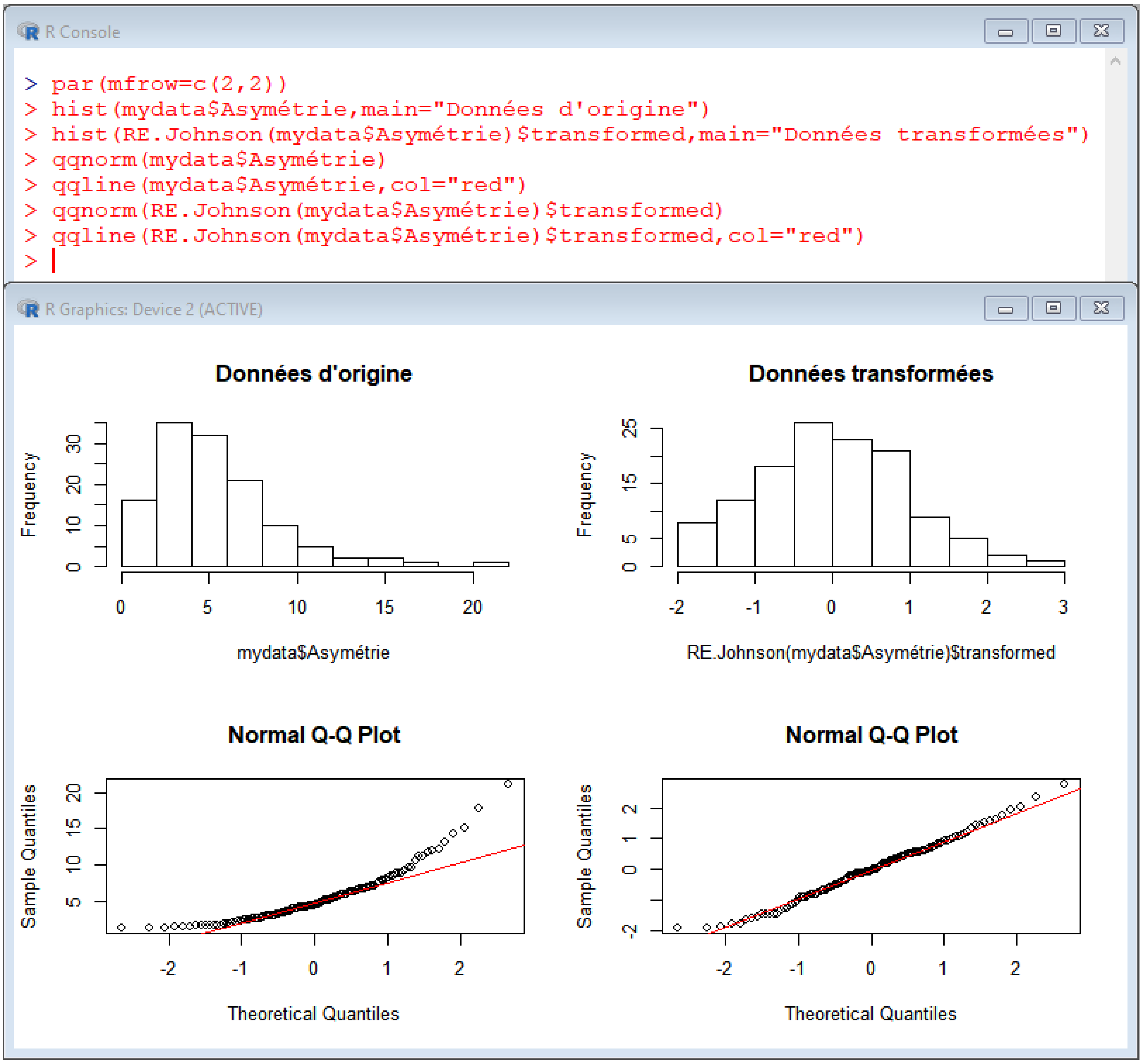
$f.eta
[1] 1.337693

> |
```

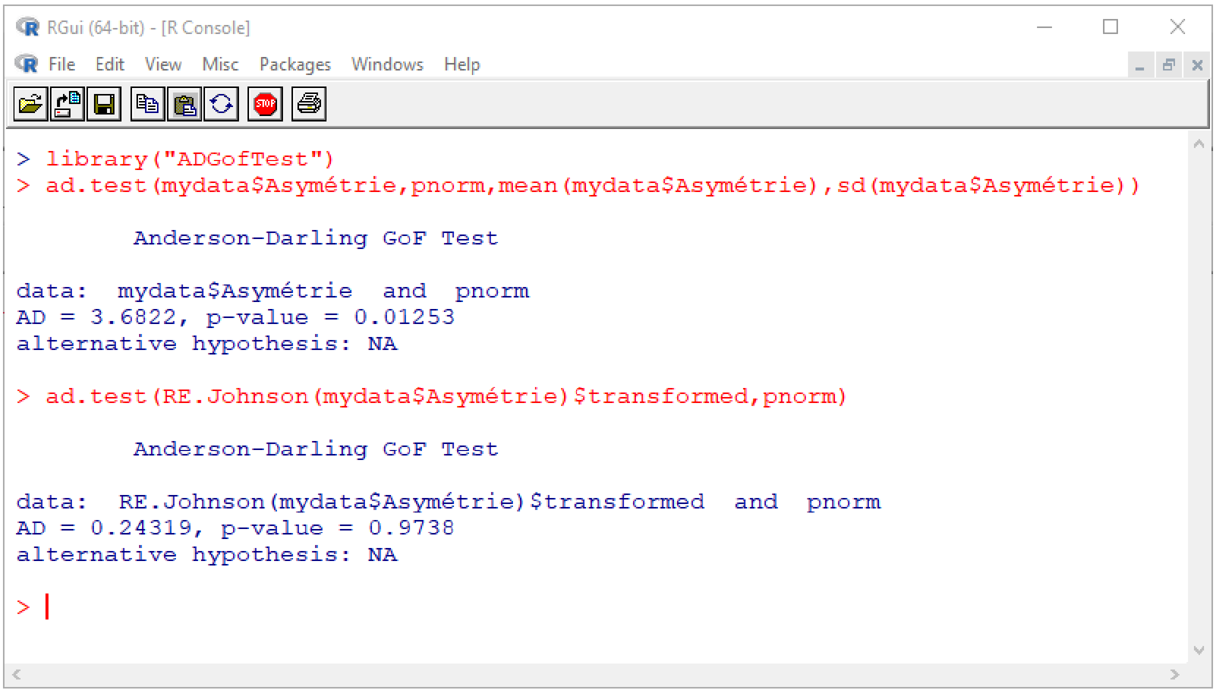
Comparons avec la sortie de Minitab 15:

Valeur de P pour un meilleur ajustement : 0.885562
Valeur de Z pour un meilleur ajustement : 0.48
Meilleur type de transformation : SB
Fonction de transformation égale
2.95594 + 1.33769 * Ln((X - 0.0225681) / (45.6736 - X))

Et nous pouvons faire le même type d'analyse graphique et de normalité qu'avec celle vue lors de notre étude de la transformation de Box-Cox:



Nous pouvons aussi faire un test d'Anderson-Darling avant/après:



Sinon, il y a aussi une manière plus directe pour transformer un vecteur de données avec la transformation de Johnson avec le package **caret** et la fonction **preProcess( )** comme nous le verrons lors de notre introduction au Machine Learning à la page 2050.

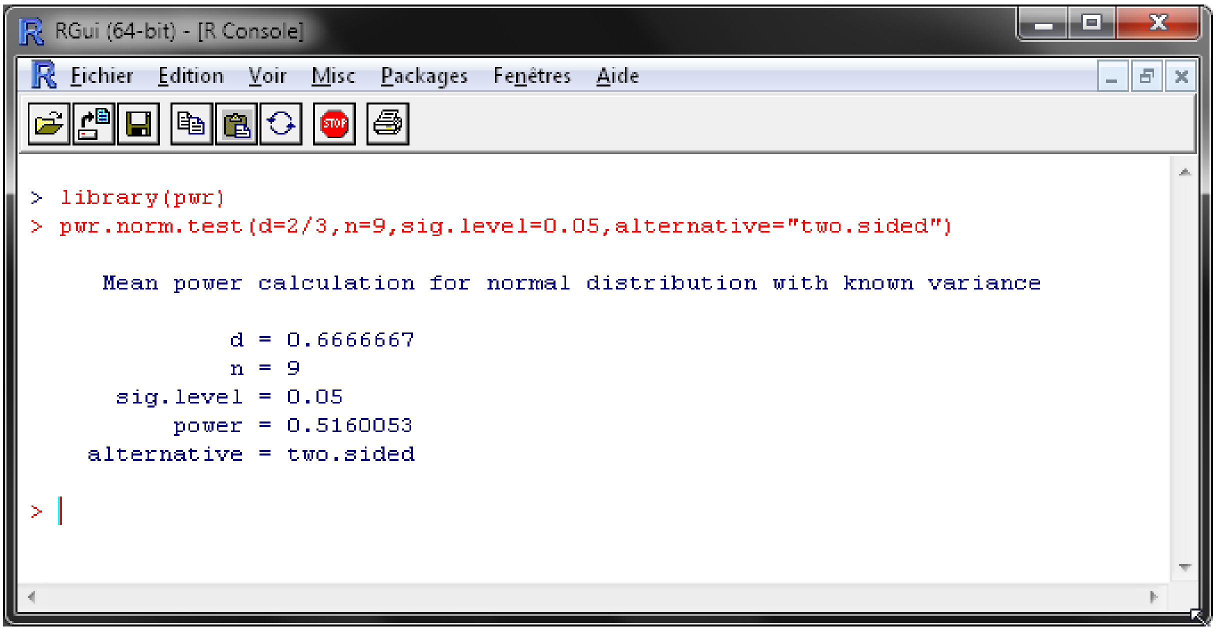
Exemple 184.: Puissance d'un test Z à 1 échantillon en bilatéral

R 3.0.2

Calculer la puissance d'un test a priori ou a posteriori est important dans la démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Remarque: Le package que nous allons utiliser contient de nombreux calculs de puissance et de taille d'échantillon, mais au même titre que pour le cours Minitab, nous nous limiterons seulement à ce que nous avons démontré mathématiquement dans le cours théorique.

Pour cela, nous allons utiliser la commande **pwr.norm.test( )** du package **pwr**:



Donc à part que la syntaxe de la commande est piège puisque:

$$pwr.norm.test\left(\frac{\delta}{\sigma}, n, \alpha, H_2\right)$$

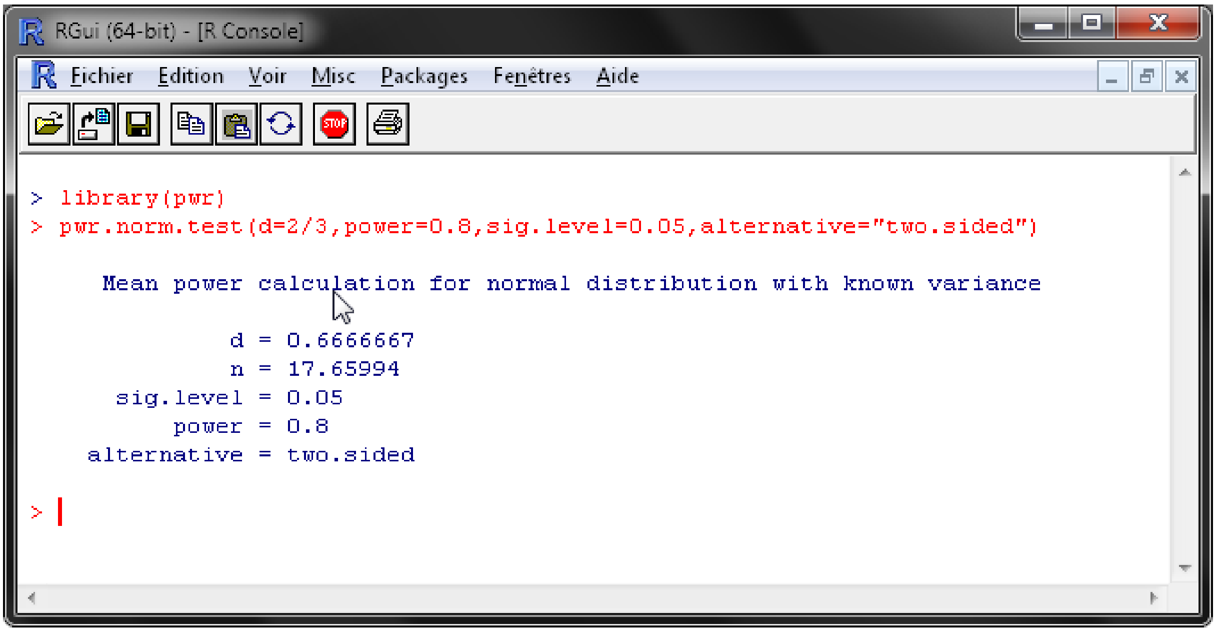
Nous retrouvons bien les mêmes résultats qu'avec Minitab (et donc les mêmes conclusions) simplement affiche automatiquement la courbe de puissance ce qui est bien pratique.

Exemple 185.: Taille d'échantillon (effectif) d'un test Z en bilatéral

R 3.0.2

Calculer la taille nécessaire d'un échantillon a priori pour avoir une certaine puissance du test est aussi important dans la démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Pour cela, nous allons encore une fois utiliser la commande **pwr.norm.test( )** du package **pwr**:



Donc à part que la syntaxe de la commande est piège puisque:

$$pwr.norm.test\left(\frac{\delta}{\sigma}, P, \alpha, H_2\right)$$

Nous retrouvons bien les mêmes résultats qu'avec Minitab (et donc les mêmes conclusions) simplement affiche automatiquement la courbe de puissance ce qui est bien pratique. Nous pouvons cependant la construire nous même rapidement avec R:

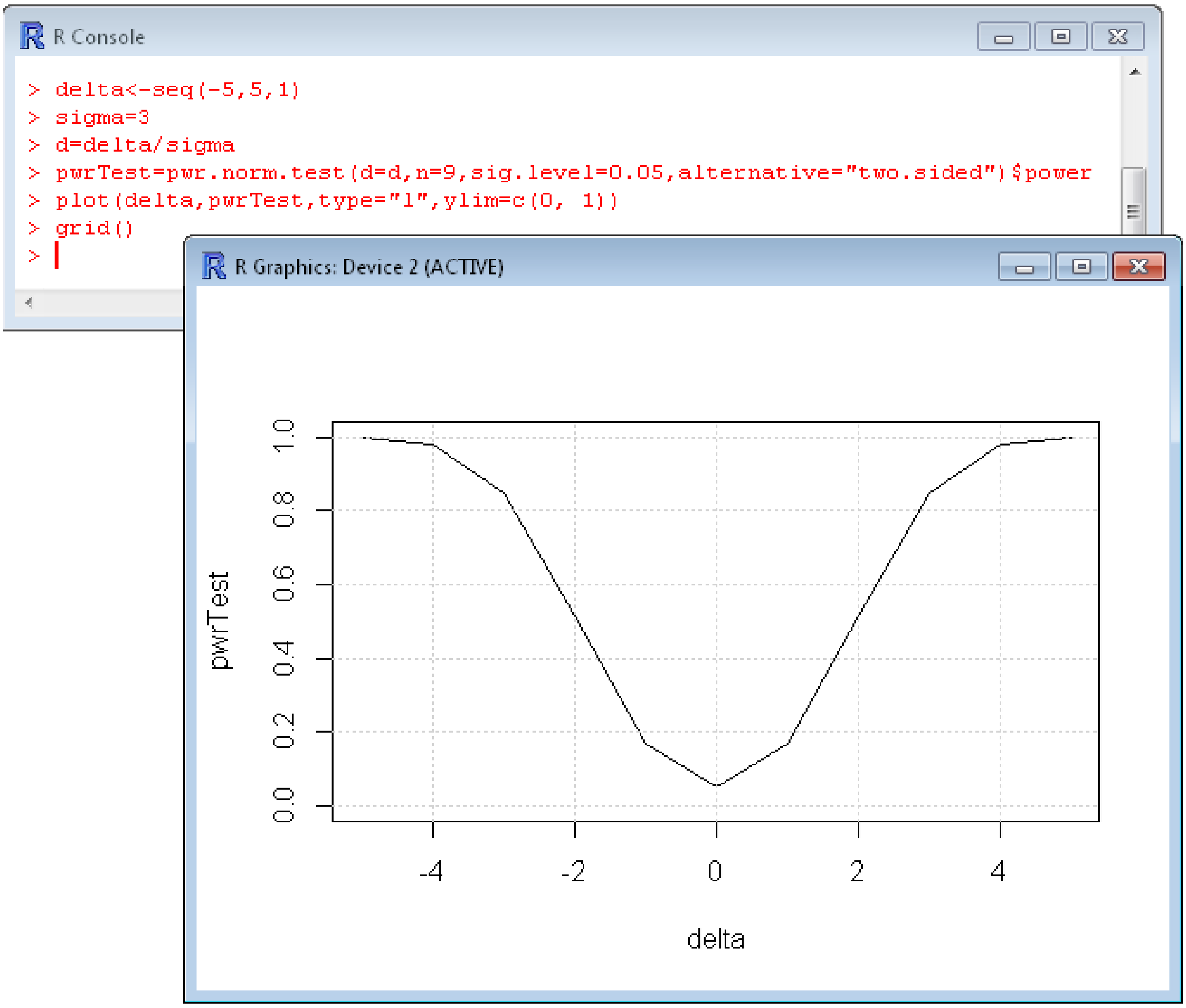


Figure 223 Graph de puissance d'un test

Bingo!

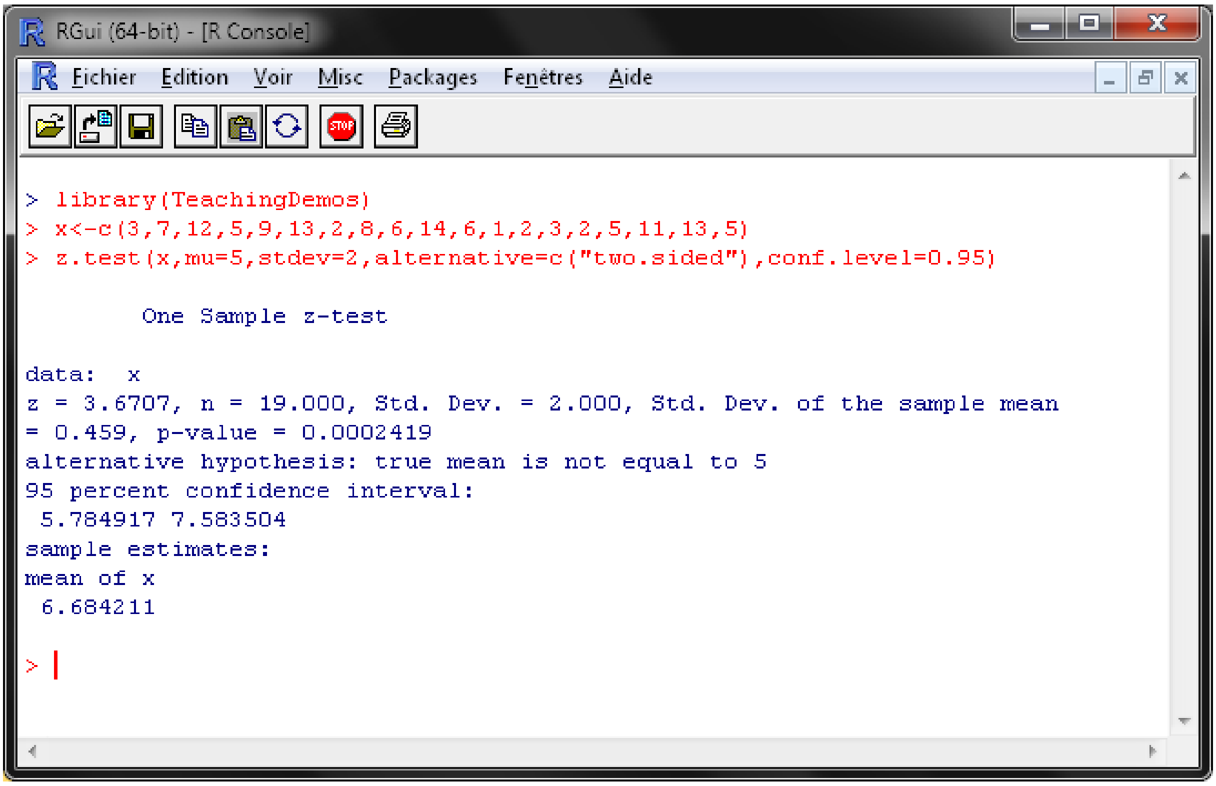
Exemple 186.: Intervalle de confiance de la moyenne (test Z à un 1 échantillon)

R 3.0.2

Curieusement le test *Z* n'est pas intégré par défaut de R. Certes il est extrêmement simple de récrire la fonction correspondante diront certains (et c'est vrai) mais pour moi ce n'est pas un argument car dans ce cas on peut aussi prendre le temps de récrire tous les tests statistiques car ils sont tous simples! Donc soit on a un outil qui nous fait gagner du temps systématiquement pour tout, soit pour rien mais pas l'entre deux! Pour cela, par exemple, je préfère de loin Minitab.

Mais le test *Z* existe quand même dans le package **TeachingDemos** mais seulement dans le cas où les données mesurées sont connues (et non pas juste le résumé).

Voyons cela avec le même exemple que dans le cours Minitab, SPSS et Microsoft Excel en utilisant la commande **z.test( )** :



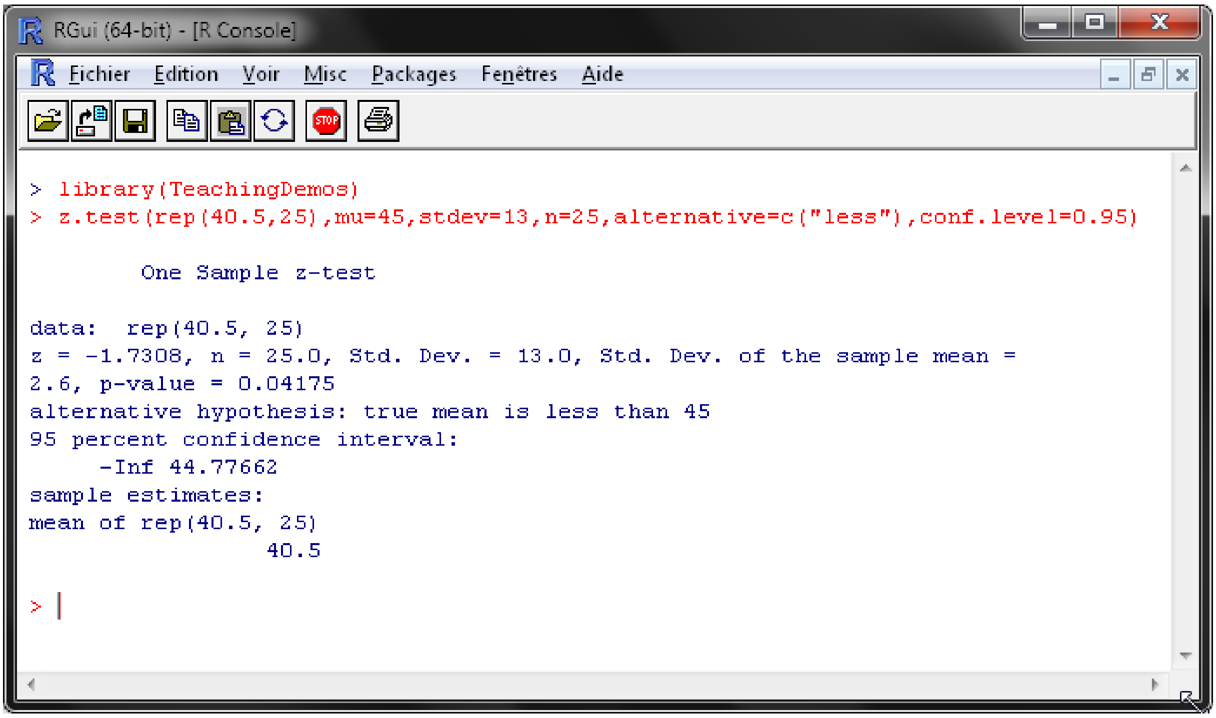
Nous retrouvons donc les valeurs de Minitab mais en plus précis et nous voyons in extenso que les résultats avec Microsoft Excel étaient peu précis.

Exemple 187.: Test *Z* de différence de la moyenne (test Z à un 1 échantillon) en unilatéral gauche

R 3.0.2

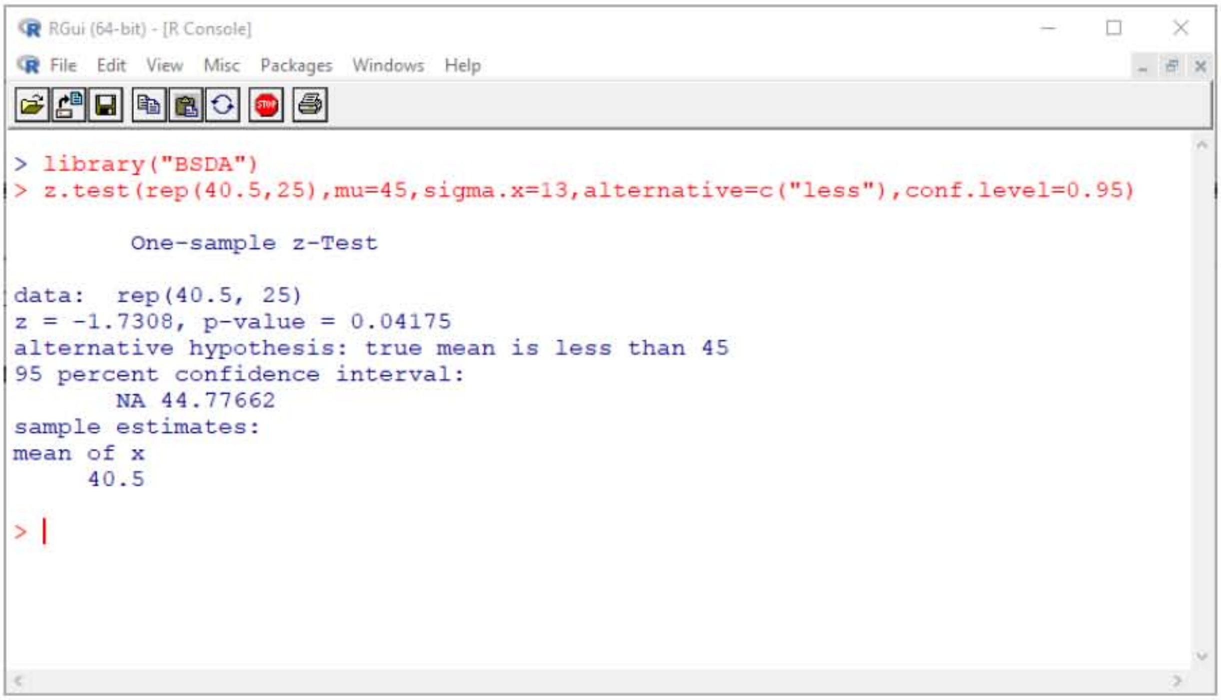
Il est connu dans un État que les enfants d'un certain âge ont un poids de 45 kilogrammes et un écart-type de 13 kilogrammes (espérance et écart-type de la population). Un plainte est posée par des parents d'élèves comme quoi les enfants d'une école sont sous-alimentés. Pour cela les parents d'élèves s'appuient sur le fait que 25 enfants du même âge ont un poids moyen de 40.5 kilogrammes.

Vérifions si ce sous-poids est significativement inférieur comme dans le cours théorique et voyons si nous obtenons le même résultat qu'avec Microsoft Excel et Minitab:



Nous pouvons donc bien constater que nous obtenons les mêmes résultats que ceux calculés à la main et que dans des logiciels comme Microsoft Excel et Minitab.

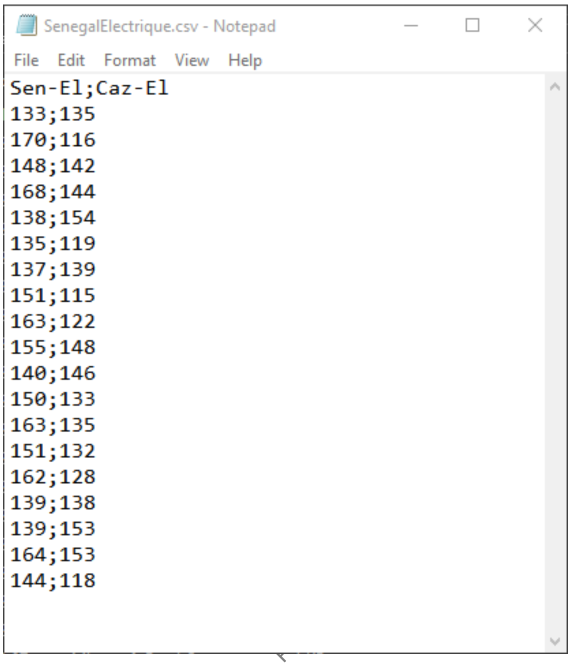
Sinon avec le package **BSDA** (Basic Statistics and Data Analysis) nous avons:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("BSDA")
> z.test(rep(40.5,25),mu=45,sigma.x=13,alternative=c("less"),conf.level=0.95)
One-sample z-Test
data: rep(40.5, 25)
z = -1.7308, p-value = 0.04175
alternative hypothesis: true mean is less than 45
95 percent confidence interval:
NA 44.77662
sample estimates:
mean of x
40.5

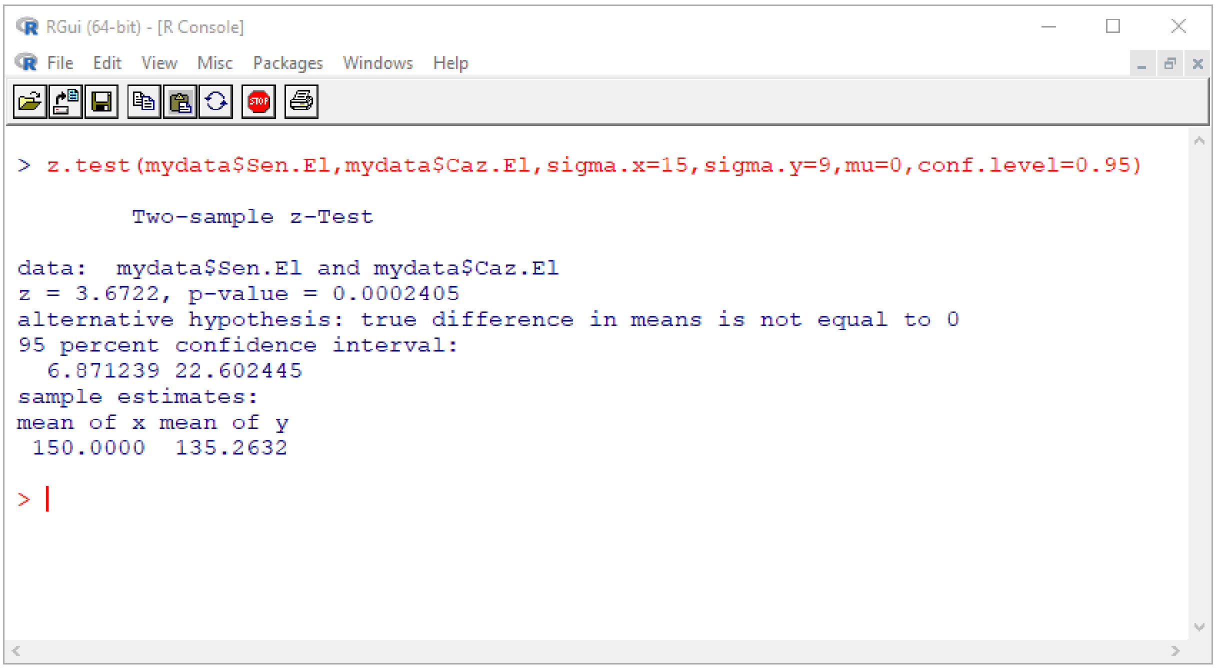
Exemple 188.: Test *Z* de différence de la moyenne à deux échantillons en bilatéral

R 3.0.2

Pour cet exemple nous allons utiliser le même package **BSDA** que précédemment mais avec un jeu de donnée différent:



Une fois importé, le test Z sur ces données nous donne:



A comparer avec le résultat obtenu dans Microsoft Excel (puisque dans Minitab il faut une macro à ce jour pour faire un test Z à deux échantillons):

	A	B	C
1	Test de la différence significative minimale (z-Test)		
2			
3		*Sen-El*	*Caz-El*
4	Moyenne	150	135.2631579
5	Variances (connues)	225	81
6	Observations	19	19
7	Différence hypothétique des moyennes	0	
8	z	3.672150885	
9	P(Z<=z) unilatéral	0.000120259	
10	Valeur critique de z (unilatéral)	1.644853627	
11	P(Z<=z) bilatéral	0.000240518	
12	Valeur critique de z (bilatéral)	1.959963985	

Exemple 189.: Test Z par Monte-Carlo

R 3.0.2

Le but de cet exercice est plus pédagogique qu'autre chose. Imaginons effectivement que vous connaissez les estimateurs de la moyenne et l'écart-type d'une loi Normale et la taille de l'échantillon que vous allez mesurer.

La question pourrait être alors simplement la suivante: si je tirais au hasard des individus d'une telle population (simulable par du Monte Carlo en théorie), quelle est la probabilité cumulée (ou 1 – la *p* value) que ma mesure dépasse une certaine valeur seuil.

Nous avons alors dans le cas particulier unilatéral la possibilité d'utiliser la fonction **as.randtest( )** du package **ade4**. Ce qui donne:

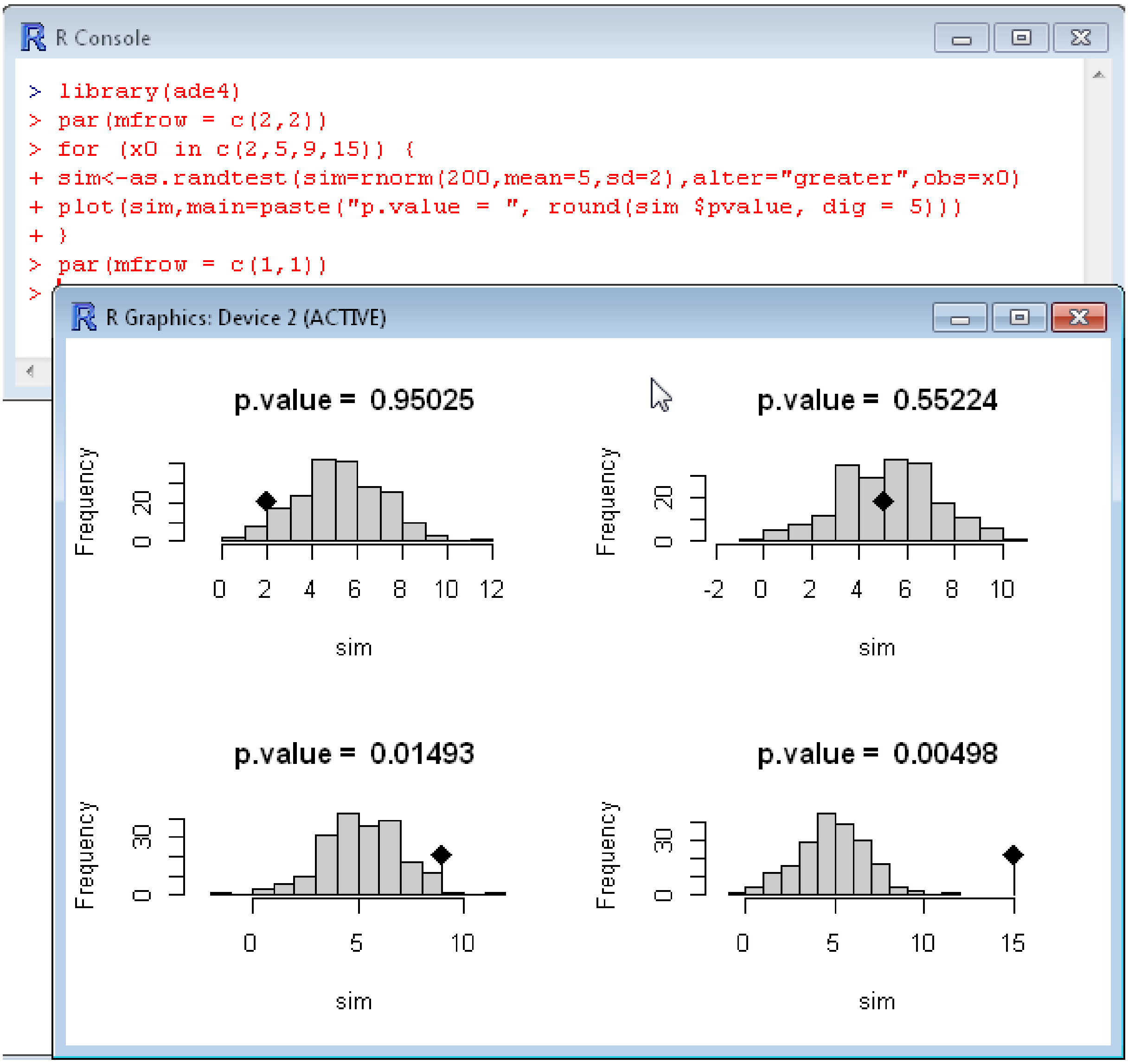


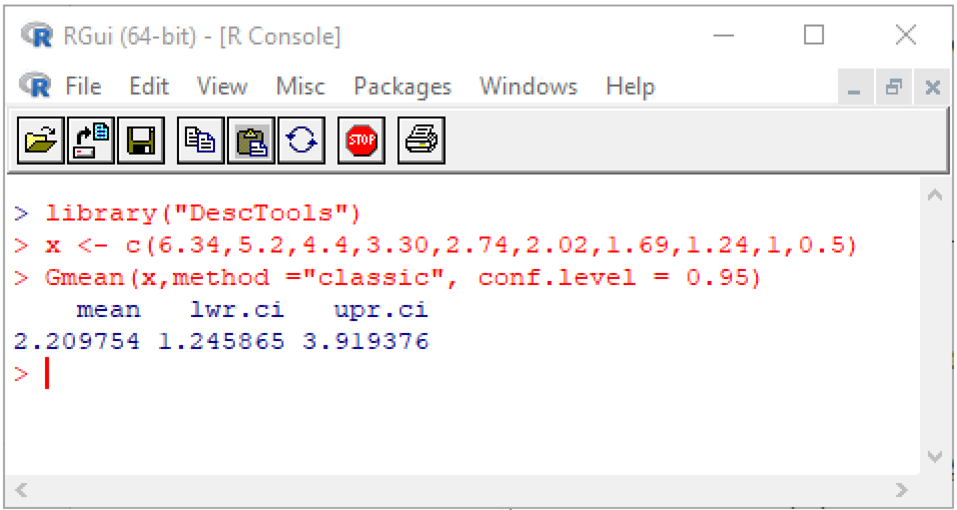
Figure 224 Graph d'histogrammes avec p-value par simulation Monte Carlo

Exemple 190.: Intervalle de confiance de Student de la moyenne géométrique

R 3.4.1

Bon nous allons ici reproduire le même exemple que dans le cours théorique calculé à main et dans Microsoft Excel et nous allons voir si nous retombons sur les mêmes résultats.

Pour calculer l'intervalle de confiance de la moyenne géométrique, nous allons utiliser le package **DescTools** et sa fonction **Gmean( )**:



On ne retrouve donc pas le même résultat que celui calculé à la main et dans Microsoft Excel. La raison est simplement que le package DescTools utilise la méthode de calcul de Stata qui a priori est fausse:

https://www.stata.com/manuals13/rameans.pdf

On ne retrouve pas non plus le même résultat que cette vidéo YouTube:

https://www.youtube.com/watch?v=XsH33QpDQac

car en réalité la personne qui a enregistrée cette vidéo utilise le Z alors que c'est le T de Student qu'il faut utiliser pour l'intervalle et c'est que fait DescTools (et de toute façon comme précisé plus haut ce dernier fait le calcul faux aussi…)!

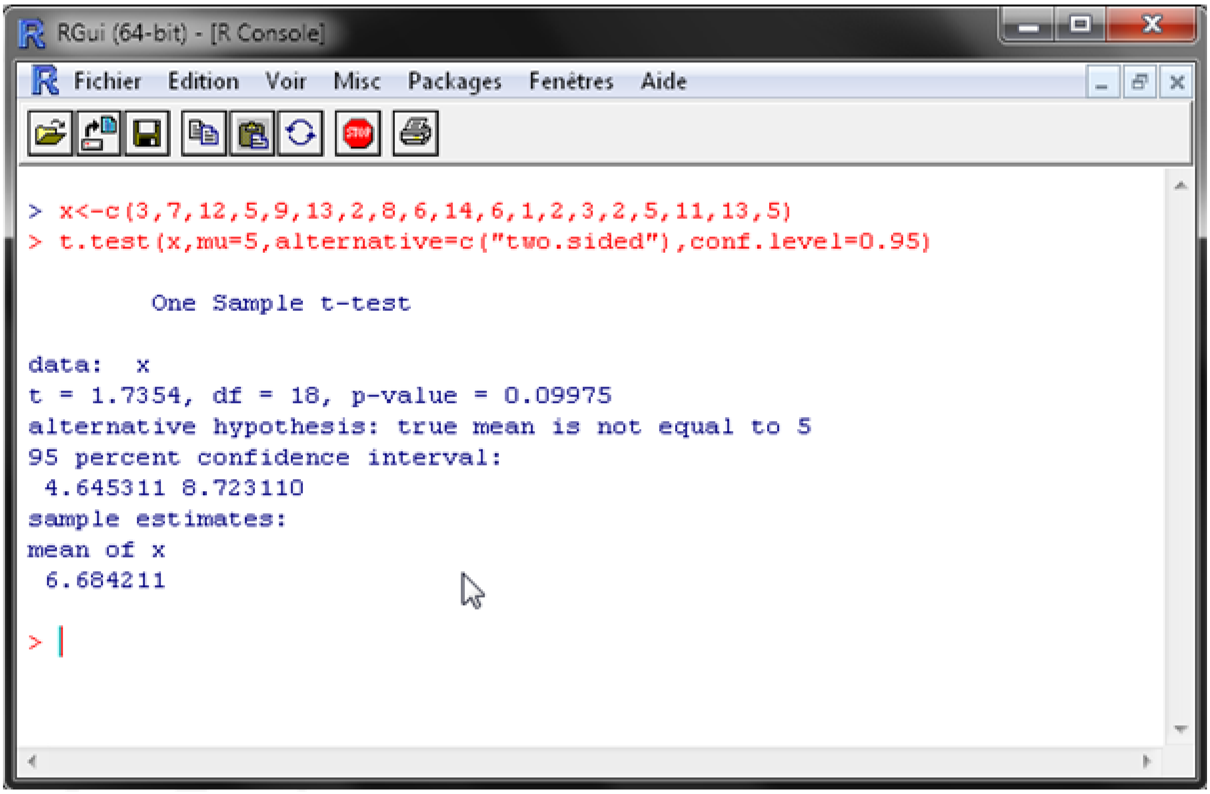
Exemple 191.: Intervalle de confiance de la moyenne (test t à un 1 échantillon)

R 3.0.2

Là encore nous allons vérifier la conformité des résultats avec toujours les mêmes logiciels et les calculs effectués à la main suite à la démonstration mathématique faite dans le cours théorique.

Le test de Student à 1 échantillon est lui intégré par défaut dans R mais là encore seulement si les données brutes sont disponibles (encore une fois c'est dommage mais bon...).

Nous utilisons donc la commande **t.test( )** :



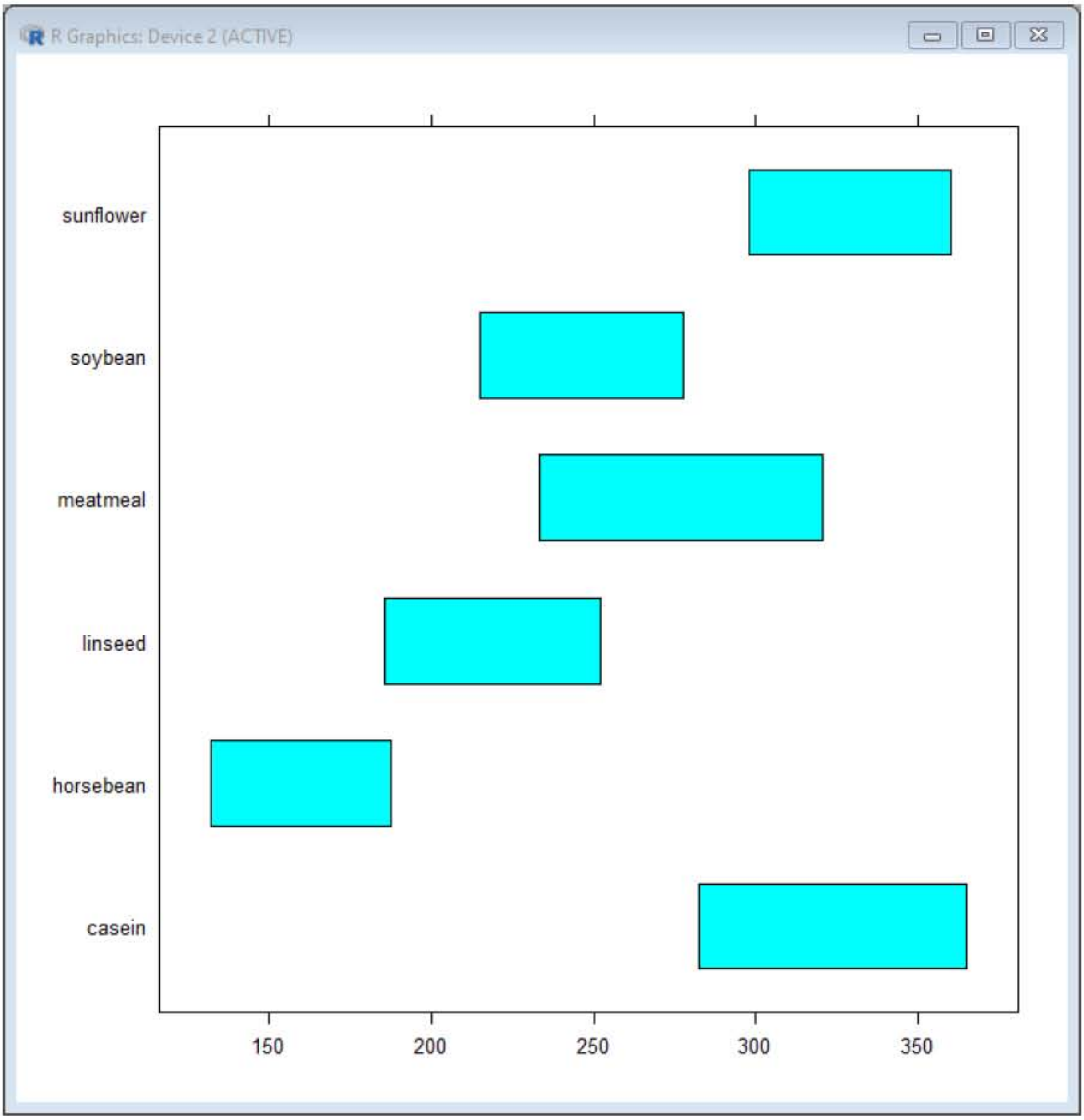
et là nous sommes en conformité avec Minitab (mais ce dernier ne donne pas la p-value) et Microsoft Excel(avec qui on retrouvait toutes les valeurs).

Nous retrouvons donc les bons résultats avec les mêmes conclusions.

On peut avoir de multiples intervalles de confiance à l'aide du package **Rmisc** et de sa fonction **group.CI( )**:

```
> library("Rmisc")
> head(chickwts,10)
   weight      feed
1     179 horsebean
2     160 horsebean
3     136 horsebean
4     227 horsebean
5     217 horsebean
6     168 horsebean
7     108 horsebean
8     124 horsebean
9     143 horsebean
10    140 horsebean
> #intervalle de  confiance de la moyenne par groupe basée sur
> #l'intervalle de Student
> group.CI(weight~feed,data=chickwts,ci=0.95)
       feed weight.upper weight.mean weight.lower
1    casein     364.5226    323.5833     282.6440
2 horsebean     187.8313    160.2000     132.5687
3   linseed     251.9390    218.7500     185.5610
4  meatmeal     320.5099    276.9091     233.3083
5   soybean     277.6818    246.4286     215.1754
6 sunflower     359.9458    328.9167     297.8875
>
> require("latticeExtra")
> with(group.CI(weight~feed,chickwts),
+  segplot(feed~weight.lower+weight.upper,center=weight.mean)
+ )
>
```

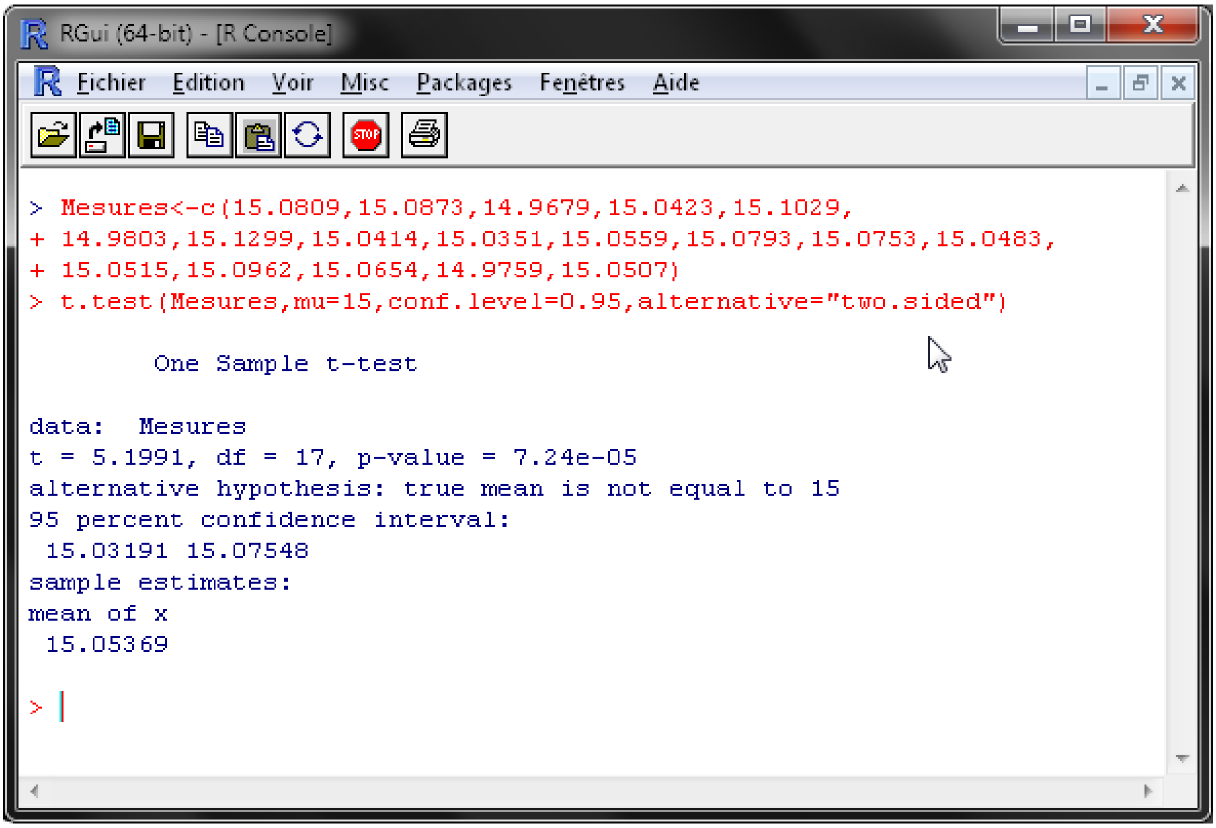
Avec le graphique correspondant:


R Graphics: Device 2 (ACTIVE)
sunflower
soybean
meatmeal
linseed
horsebean
casein
150
200
250
300
350

Exemple 192.: Test t-Student bilatéral d'un échantillon

R 3.0.2

Comme d'habitude nous allons contrôler que nous obtenons les mêmes résultats que ceux calculés à la main ou avec Microsoft Excel et Minitab après la démonstration mathématique du test dans le cours théorique. Nous utiliserons les mêmes données de mesures et la commande **t.test( )** :



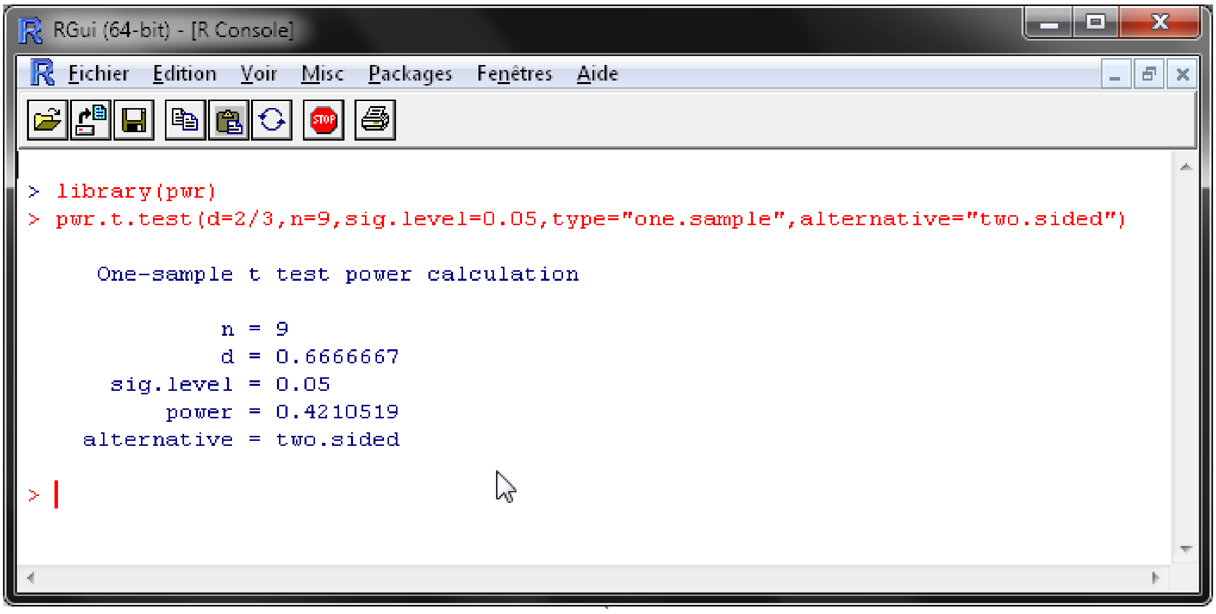
Nous retrouvons les mêmes valeurs avec les mêmes conclusions!

Exemple 193.: Puissance d'un test t à 1 échantillon en bilatéral

R 3.0.2

Calculer la puissance d'un test a priori ou a posteriori est important dans la démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Pour cela, nous allons utiliser la commande **pwr.t.test( )** du package **pwr**:



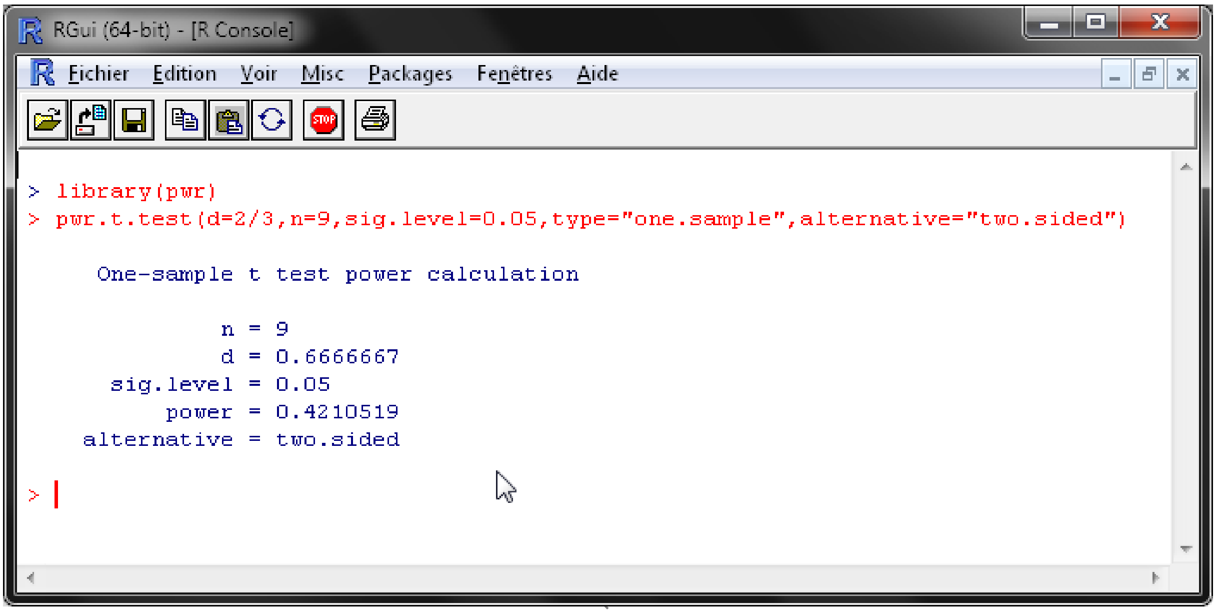
Nous retrouvons donc bien les mêmes valeurs que dans Minitab et presque les mêmes que celles calculées à la main dans le cours théorique.

Exemple 194.: Résolution du test t à 1 échantillon en bilatéral

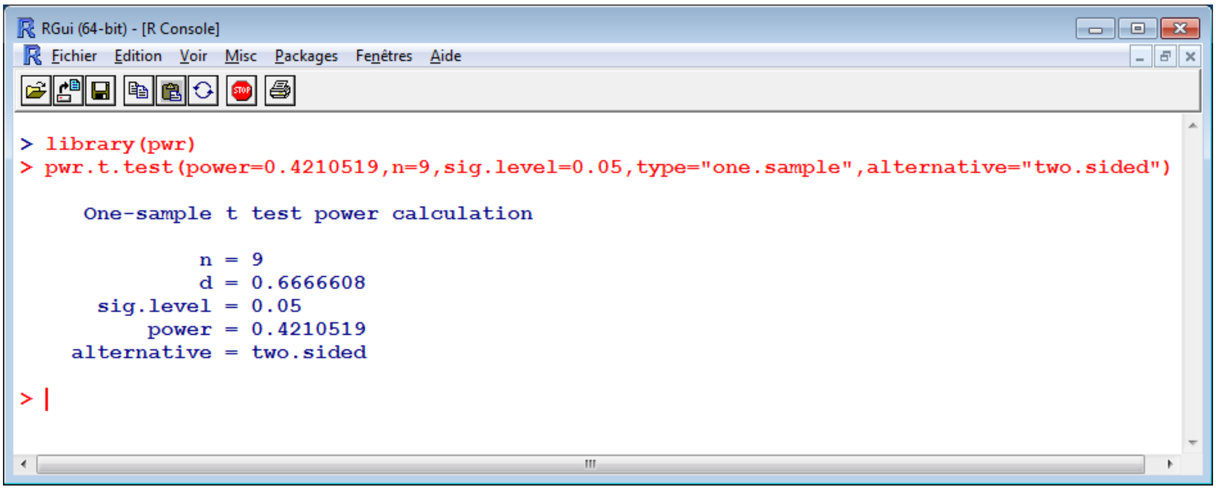
R 3.0.2

N'ayant pas trouvé de fonctions dans quelque package que ce soit pour la résolution du test *Z*, nous allons donc passer directement au *t* et nous sauterons le calcul de la résolution du test *p* n'ayant pas trouvé de package intégrant aussi la fonction.

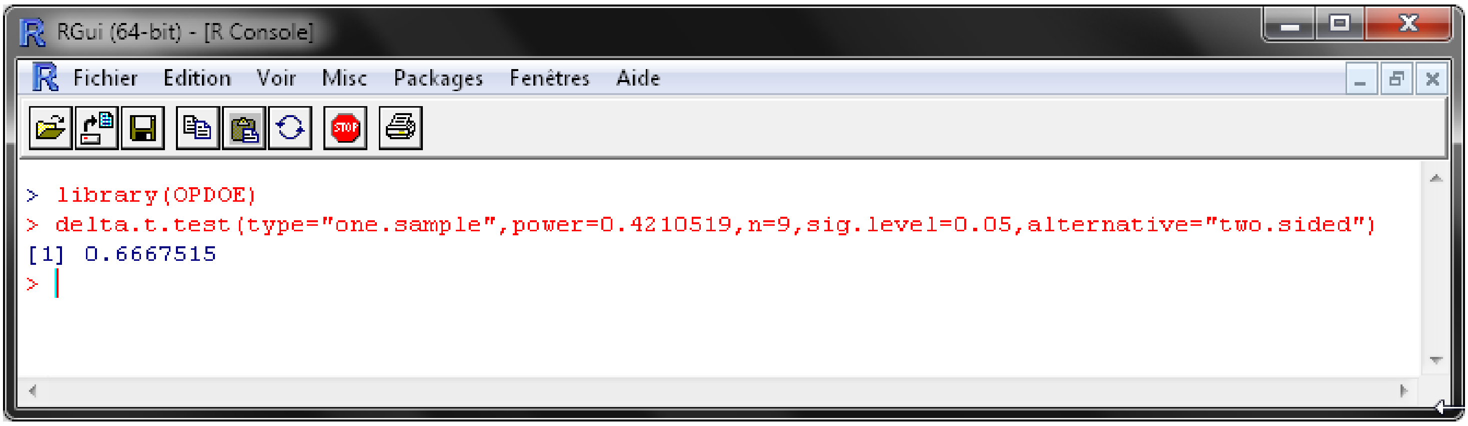
Donc rappelons d'abord que nous avons obtenu ci-dessus:



Nous avons alors simplement en suivant la même ligne directrice:



Eh bien en utilisant le package **OPDOE** et sa fonction **delta.t.test**( ) nous pouvons en quelque sorte rebrousser chemin:



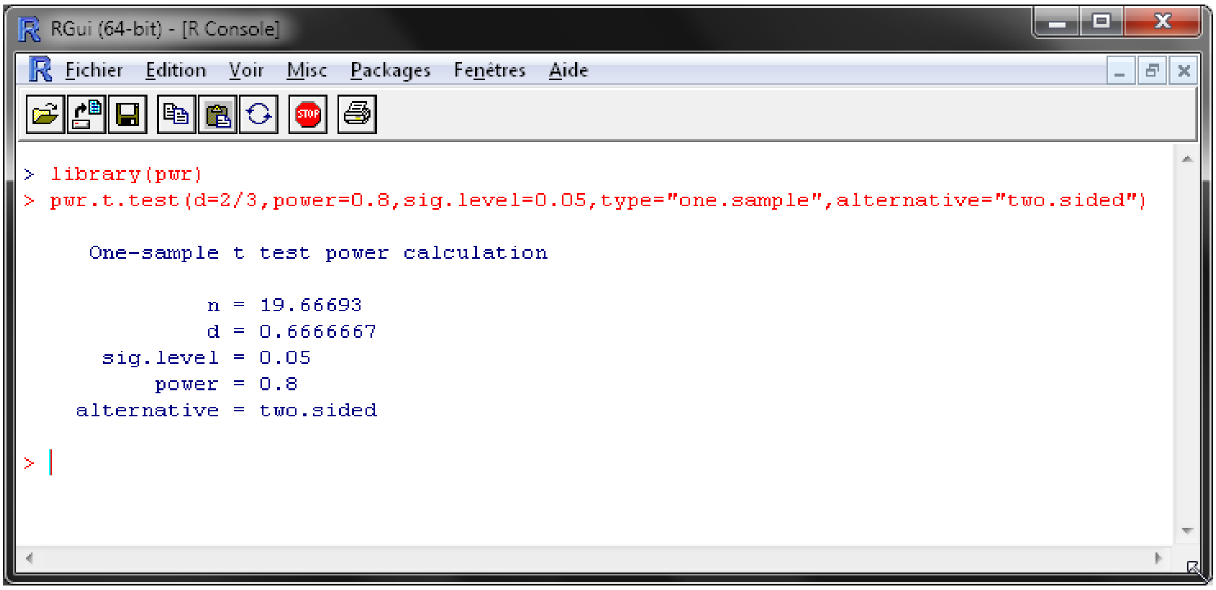
Nous retrouvons bien la différence de 2/3 (à une petite erreur d'arrondi près).

Exemple 195.: Taille d'échantillon (effectif) d'un test t à 1 échantillon en bilatéral

R 3.0.2

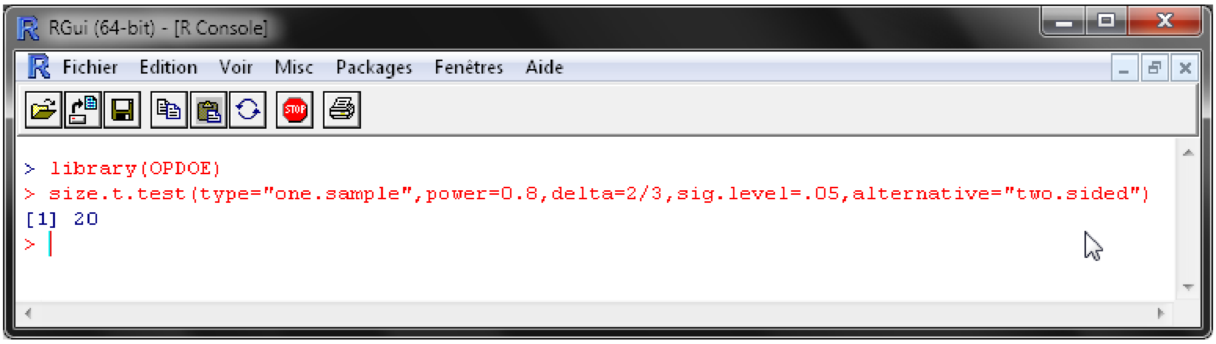
Calculer la taille nécessaire d'un échantillon a priori pour avoir une certaine puissance du test est aussi important dans la démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Pour cela, nous allons encore une fois utiliser la commande **pwr.t.test( )** du package **pwr**:



Nous retrouvons donc bien les mêmes valeurs que dans Minitab et que celles calculées à la main dans le cours théorique au dixième près.

Il est intéressant de comparer avec le package **OPDOE** et sa fonction **size.t.test**( ) :

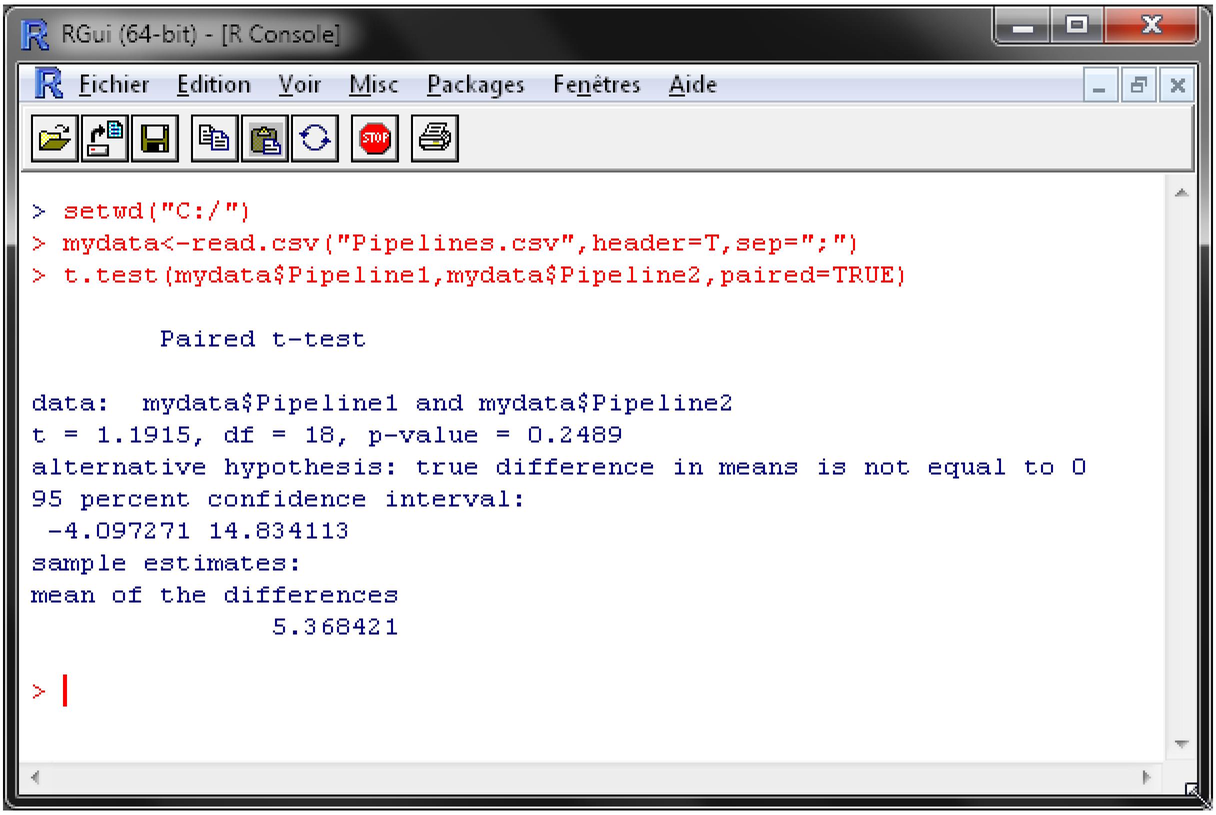


Exemple 196.: Test t-Student pour données appariées (deux échantillons)

R 3.0.2

Là encore nous allons vérifier la conformité des résultats avec toujours les mêmes logiciels et les calculs effectués à la main suite à la démonstration mathématique faite dans le cours théorique.

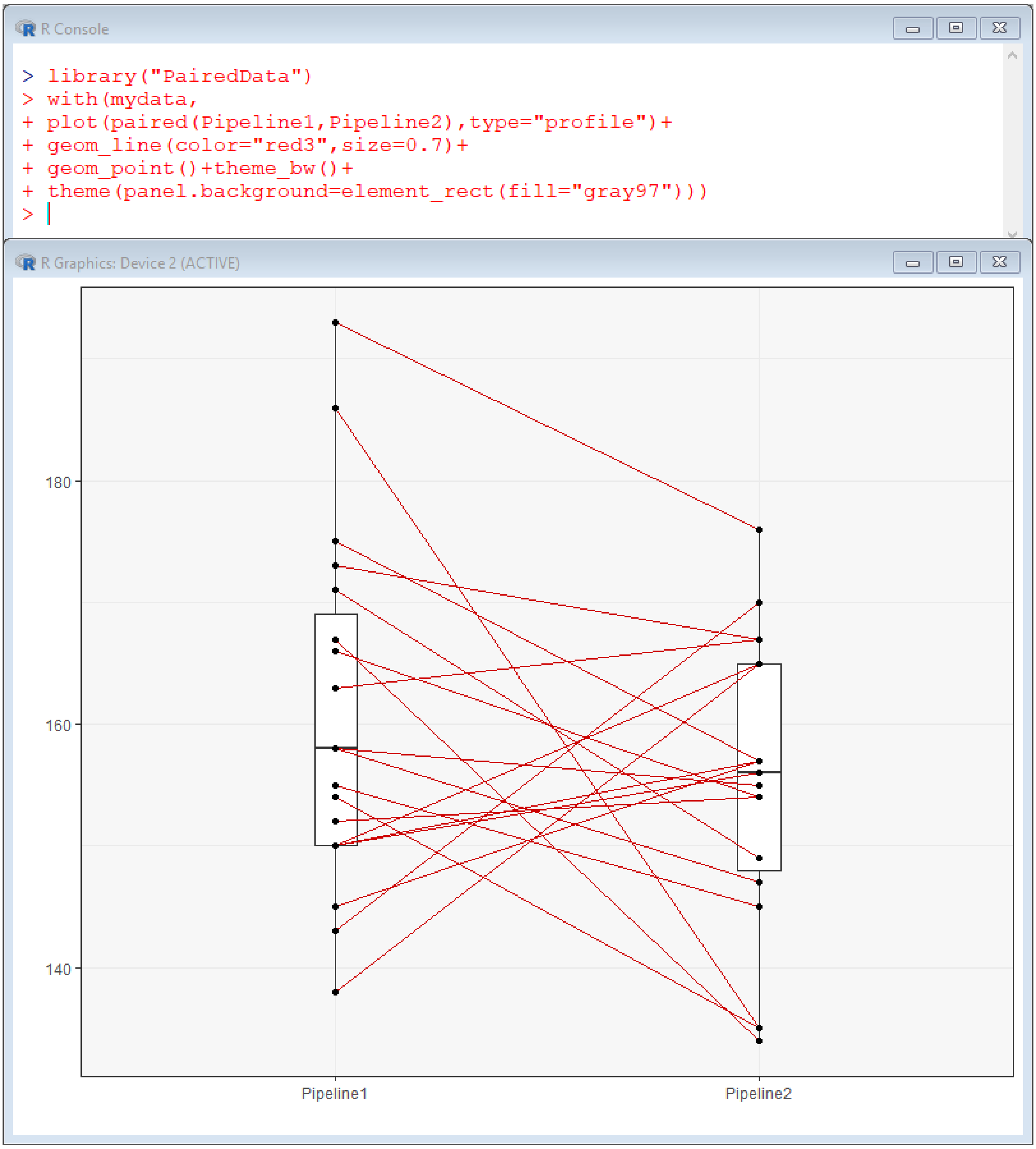
Nous utilisons donc la commande **t.test( )** :



Les résultats sont donc conformes à la différence que la *p*-value est extrêmement précise dans R...

Nous retrouvons donc les bons résultats avec les mêmes conclusions.

Nous pouvons faire un type de graphique très adapté aux données appariées à l'aide du package **PairedDate**:

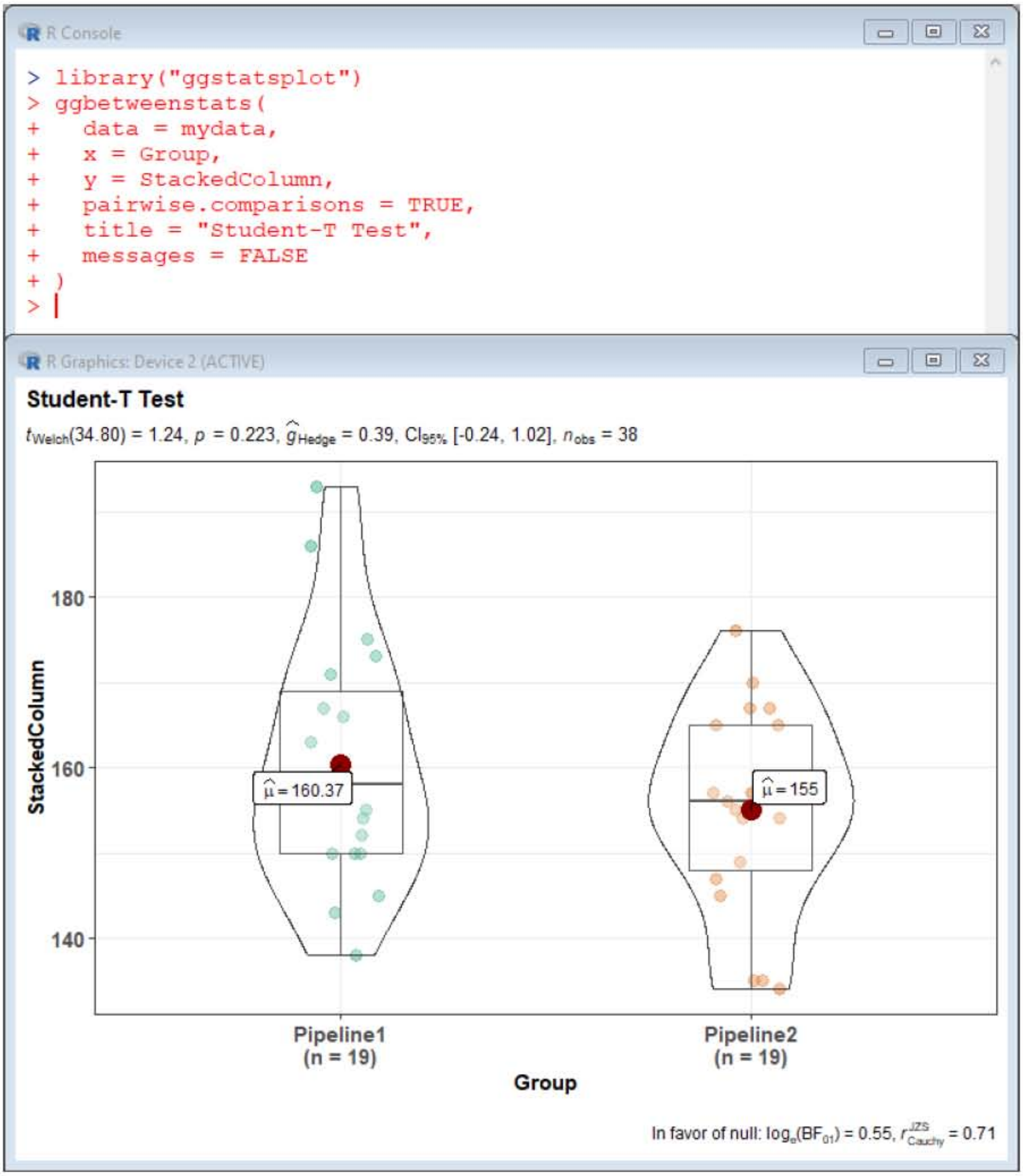


Nous pouvons faire mieux avec le package **ggstatsplot** que nous recommandons fortement au lecteur de parcourir[22]. Pour cela nous arrangeons un peu les données au préalable:

[22] https://github.com/IndrajeetPatil/ggstatsplot/blob/master/README.md

```
> mydata<-read.csv("c:/tmp/Pipelines.csv",header=T,sep=";")
> library("gtools")
> FirstSet<-data.frame(StackedColumn=mydata[,1],Group="Pipeline1")
> SecondSet<-data.frame(StackedColumn=mydata[,2],Group="Pipeline2")
> (mydata<-smartbind(FirstSet,SecondSet))
     StackedColumn     Group
1:1            163 Pipeline1
1:2            150 Pipeline1
1:3            171 Pipeline1
1:4            155 Pipeline1
1:5            186 Pipeline1
1:6            145 Pipeline1
1:7            154 Pipeline1
1:8            173 Pipeline1
1:9            152 Pipeline1
1:10           150 Pipeline1
1:11           143 Pipeline1
1:12           138 Pipeline1
1:13           166 Pipeline1
1:14           193 Pipeline1
1:15           158 Pipeline1
1:16           175 Pipeline1
1:17           167 Pipeline1
1:18           150 Pipeline1
1:19           158 Pipeline1
2:1            167 Pipeline2
2:2            157 Pipeline2
2:3            149 Pipeline2
2:4            145 Pipeline2
2:5            135 Pipeline2
2:6            157 Pipeline2
2:7            135 Pipeline2
2:8            167 Pipeline2
2:9            154 Pipeline2
2:10           165 Pipeline2
2:11           170 Pipeline2
2:12           165 Pipeline2
2:13           154 Pipeline2
2:14           176 Pipeline2
2:15           155 Pipeline2
2:16           157 Pipeline2
2:17           134 Pipeline2
2:18           156 Pipeline2
2:19           147 Pipeline2
>
```

Et ensuite allons-y pour le graphique:

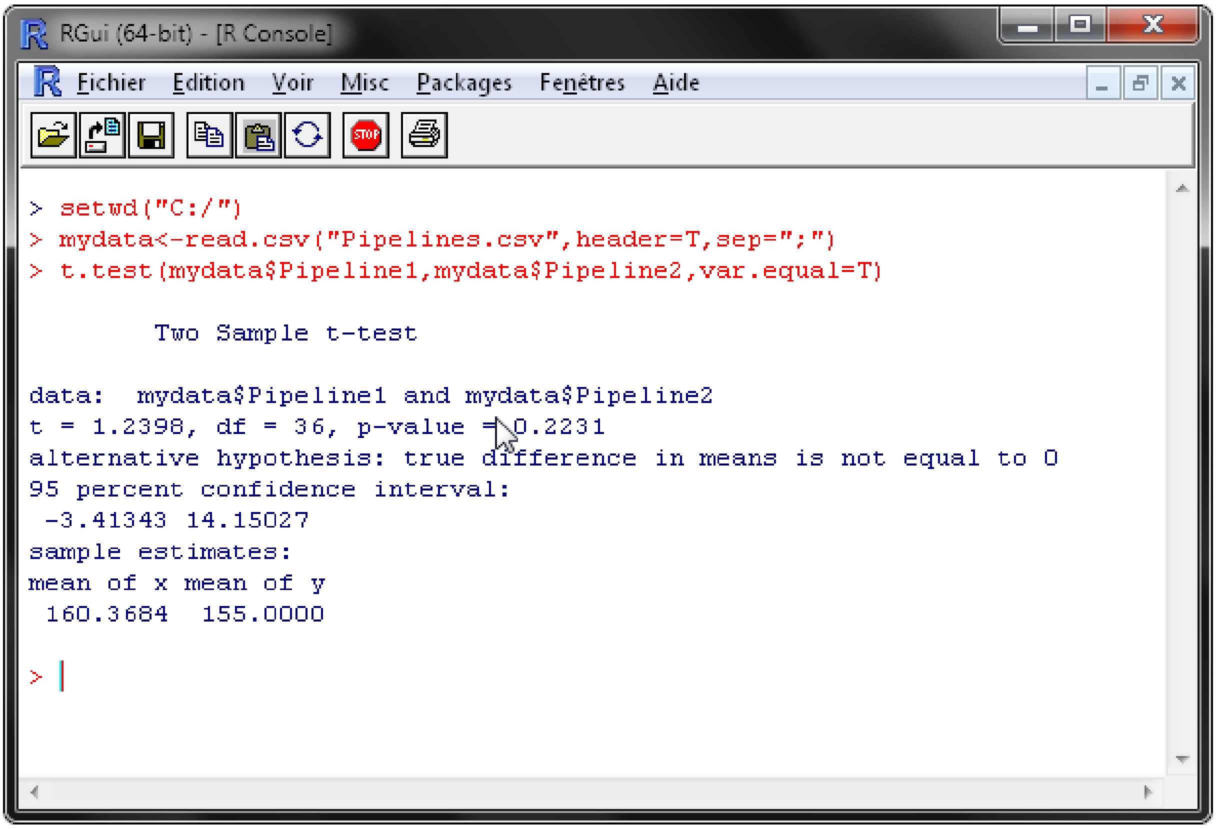

R Console
> library("ggstatsplot")
> ggbetweenstats(
+ data = mydata,
+ x = Group,
+ y = StackedColumn,
+ pairwise.comparisons = TRUE,
+ title = "Student-T Test",
+ messages = FALSE
+)
R Graphics: Device 2 (ACTIVE)
Student-T Test
t_Welch(34.80) = 1.24, p = 0.223, ĝ_Hedge = 0.39, CI95% [-0.24, 1.02], n_obs = 38
180
160
140
StackedColumn
μ̂ = 160.37
μ̂ = 155
Pipeline1
(n = 19)
Pipeline2
(n = 19)
Group
In favor of null: log_e(BF01) = 0.55, r^JZS_Cauchy = 0.71

Exemple 197.: Test t-Student homoscédastique bilatéral d'égalité de la moyenne (deux échantillons)

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons donc la commande **t.test( )** :



Nous retrouvons donc les mêmes valeurs que Minitab et MS Excel. À la différence que Minitab donne des informations que R ne donne pas et réciproquement (alors qu'avec Microsoft Excel on pouvait bien évidemment avoir toutes les informations).

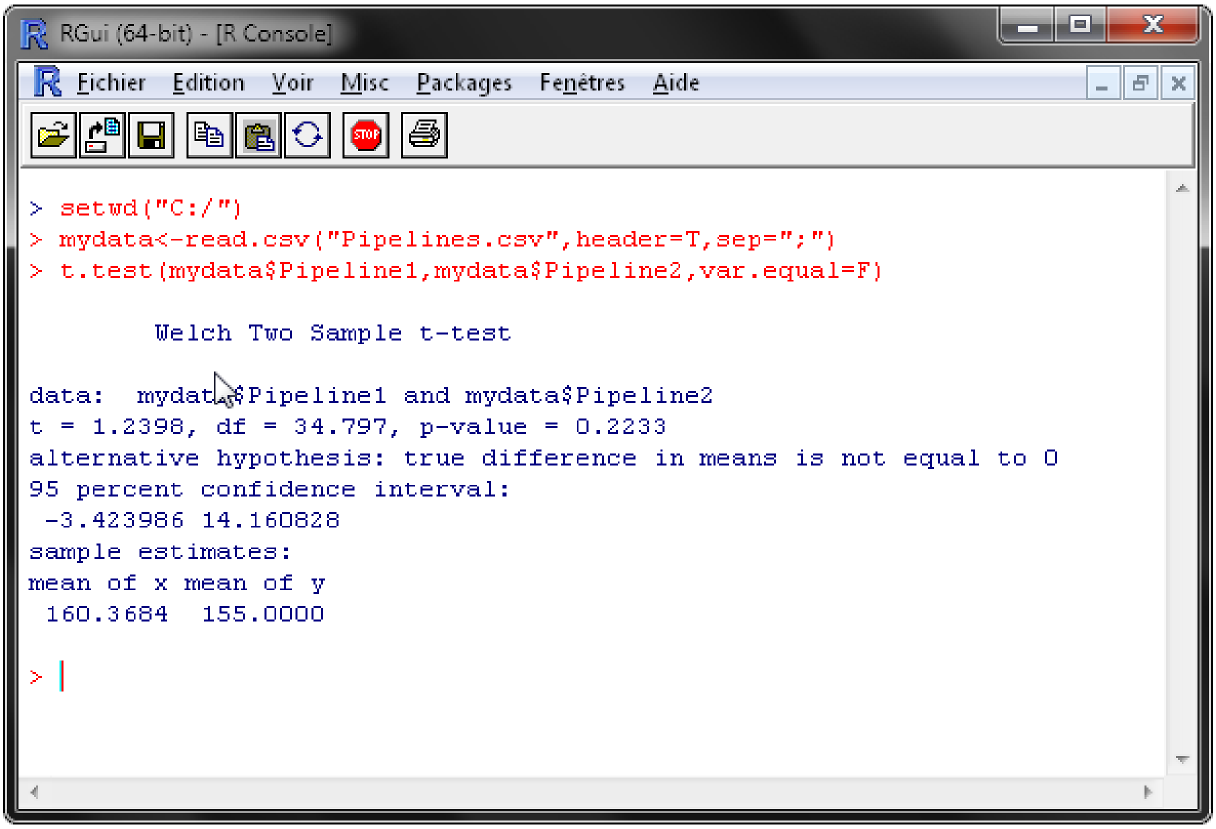
Nous retrouvons donc les bons résultats avec les mêmes conclusions.

Exemple 198.: Test t-Student hétéroscédastique bilatéral d'égalité de la moyenne (test de Welch)

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons donc la commande **t.test( )** :



Nous retrouvons donc à peu près les mêmes valeurs que Minitab et MS Excel. À la différence que Minitab donne des informations que R ne donne pas et réciproquement (alors qu'avec Microsoft Excel on pouvait bien évidemment avoir toutes les informations).

Nous retrouvons donc les bons résultats avec les mêmes conclusions.

Nous pouvons faire mieux avec le package **ggstatsplot** que nous recommandons fortement au lecteur de parcourir[23] sa documentation. Pour cela nous arrangeons un peu les données au préalable:

---

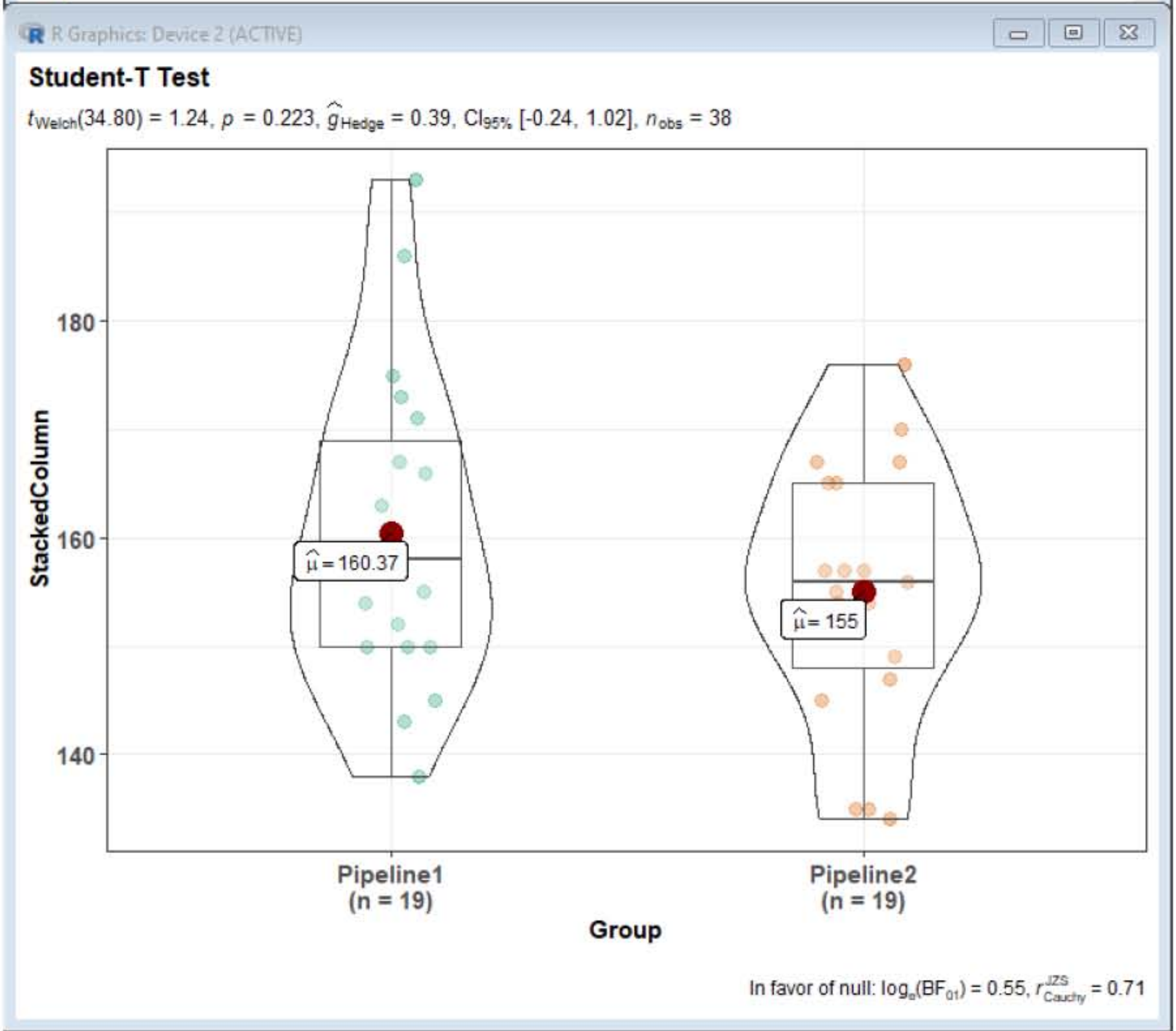
[23] https://github.com/IndrajeetPatil/ggstatsplot/blob/master/README.md

```
> mydata<-read.csv("c:/tmp/Pipelines.csv",header=T,sep=";")
> library("gtools")
> FirstSet<-data.frame(StackedColumn=mydata[,1],Group="Pipeline1")
> SecondSet<-data.frame(StackedColumn=mydata[,2],Group="Pipeline2")
> (mydata<-smartbind(FirstSet,SecondSet))
     StackedColumn     Group
1:1            163 Pipeline1
1:2            150 Pipeline1
1:3            171 Pipeline1
1:4            155 Pipeline1
1:5            186 Pipeline1
1:6            145 Pipeline1
1:7            154 Pipeline1
1:8            173 Pipeline1
1:9            152 Pipeline1
1:10           150 Pipeline1
1:11           143 Pipeline1
1:12           138 Pipeline1
1:13           166 Pipeline1
1:14           193 Pipeline1
1:15           158 Pipeline1
1:16           175 Pipeline1
1:17           167 Pipeline1
1:18           150 Pipeline1
1:19           158 Pipeline1
2:1            167 Pipeline2
2:2            157 Pipeline2
2:3            149 Pipeline2
2:4            145 Pipeline2
2:5            135 Pipeline2
2:6            157 Pipeline2
2:7            135 Pipeline2
2:8            167 Pipeline2
2:9            154 Pipeline2
2:10           165 Pipeline2
2:11           170 Pipeline2
2:12           165 Pipeline2
2:13           154 Pipeline2
2:14           176 Pipeline2
2:15           155 Pipeline2
2:16           157 Pipeline2
2:17           134 Pipeline2
2:18           156 Pipeline2
2:19           147 Pipeline2
>
```

Et ensuite allons-y pour le graphique:

R Console

```
> library("ggstatsplot")
> ggbetweenstats(
+   data = mydata,
+   x = Group,
+   y = StackedColumn,
+   title = "Student-T Test",
+   messages = FALSE
+ )
>
```



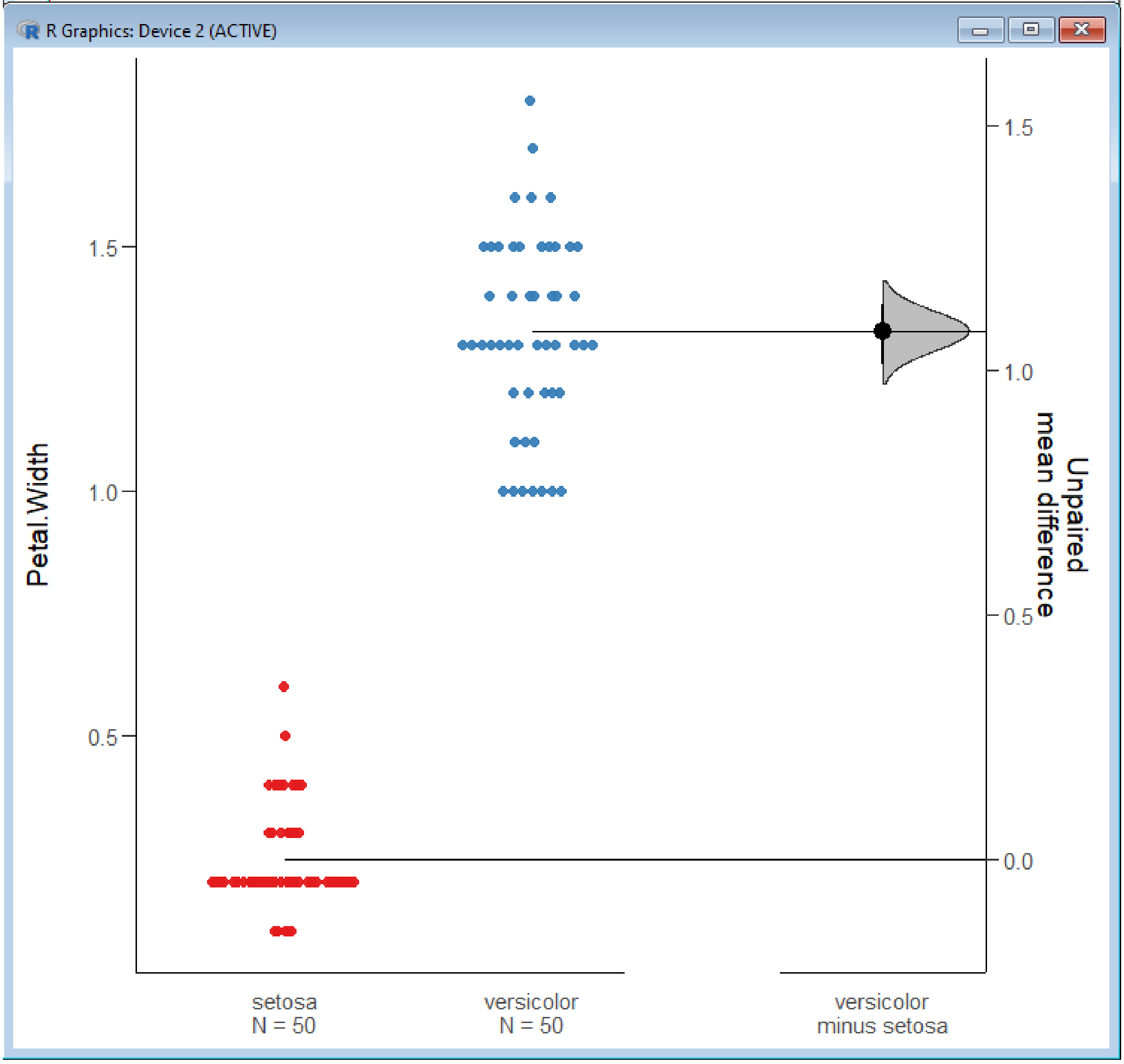
Exemple 199.: Graphique de Gardner-Altman pour test de Student des moyennes appariées ou non

R 3.4.1

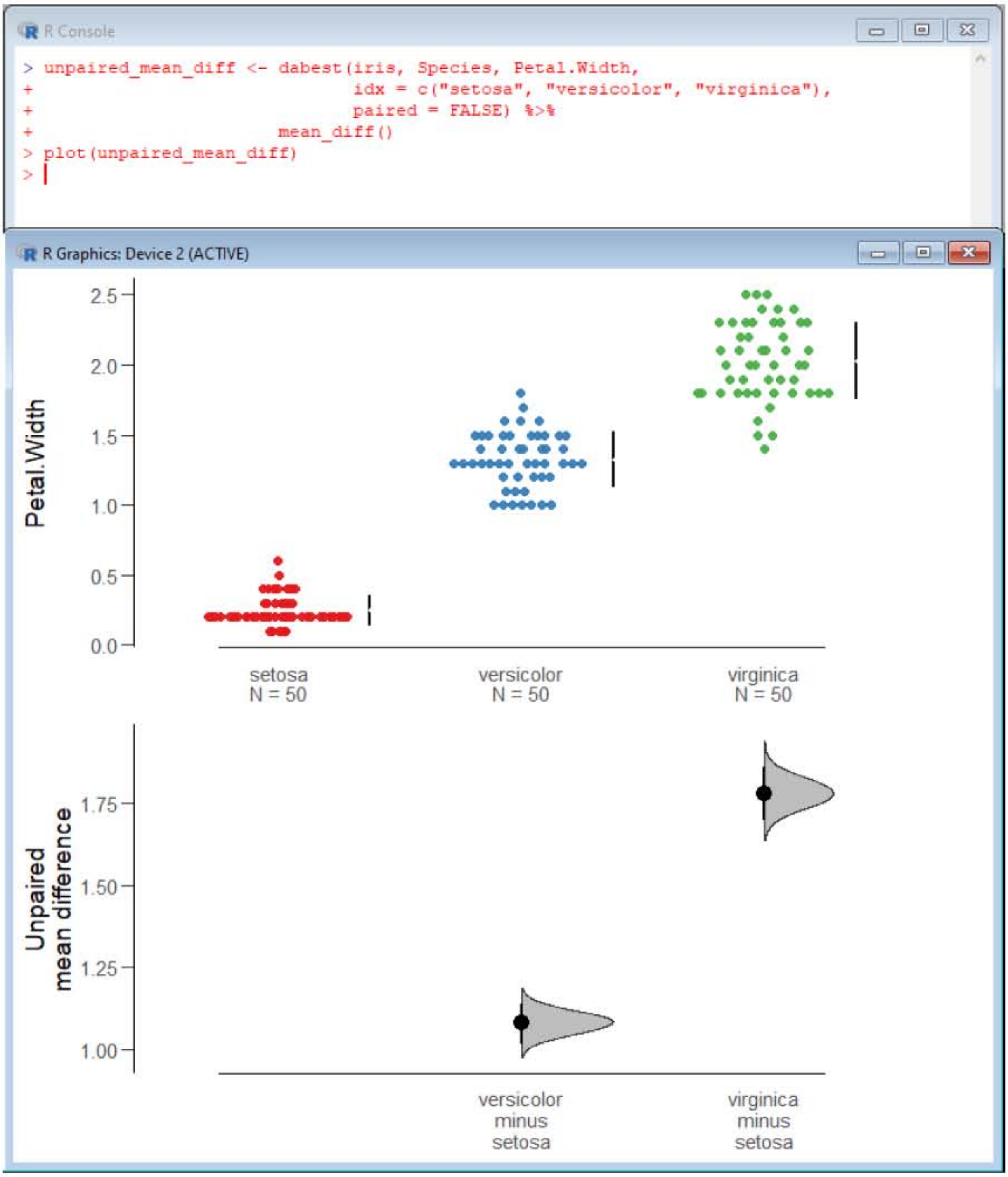
Pour faire ce graphique, nous allons utiliser le package **dabestr** et ses fonctions **dabest( )** et **plot( )**.

Voici donc deux graphiques différents (cas non appariés pour les deux) d'abord avec seulement deux groupes:

```
> library("dabestr")
>
> unpaired_mean_diff <- dabest(iris, Species, Petal.Width,
+                              idx = c("setosa", "versicolor"),
+                              paired = FALSE) %>%
+                   mean_diff()
> plot(unpaired_mean_diff)
>
```



et avec trois groupes:



Exemple 200.: Two one-sided test (TOST) de Student

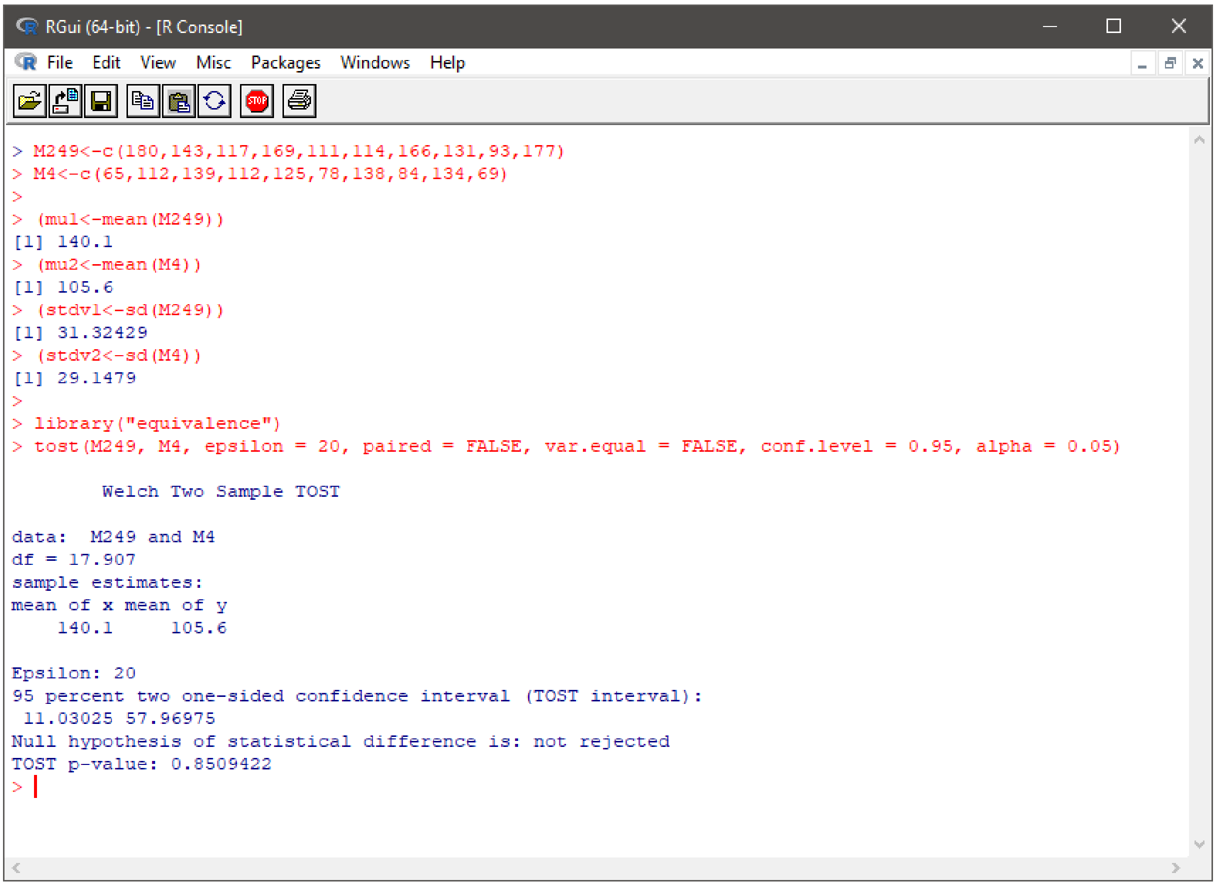
R 3.4.1

Nous avons étudié et démontré en détails dans le cours théorique le fameux test TOST de Student qui permet d'inverser la logique classique des tests NHST et donc au contraire de "prouver" l'hypothèse nulle au lieu de l'hypothèse alternative.

Nous avons fait un exemple numérique détaillé calculé à la main que nous allons reproduire donc ici avec R et voir si nous tombons sur les mêmes résultats et mêmes conclusions.

Nous allons comparer ici les packages **TOST** et **equivalence** de R pour le même cas et comparer les résultats mais aussi parce que les paramètres d'entrée doivent être saisis de manière très différente et les comparer n'est dénué d'intérêt.

Commençons par la fonction **tost( )** du package equivalence:

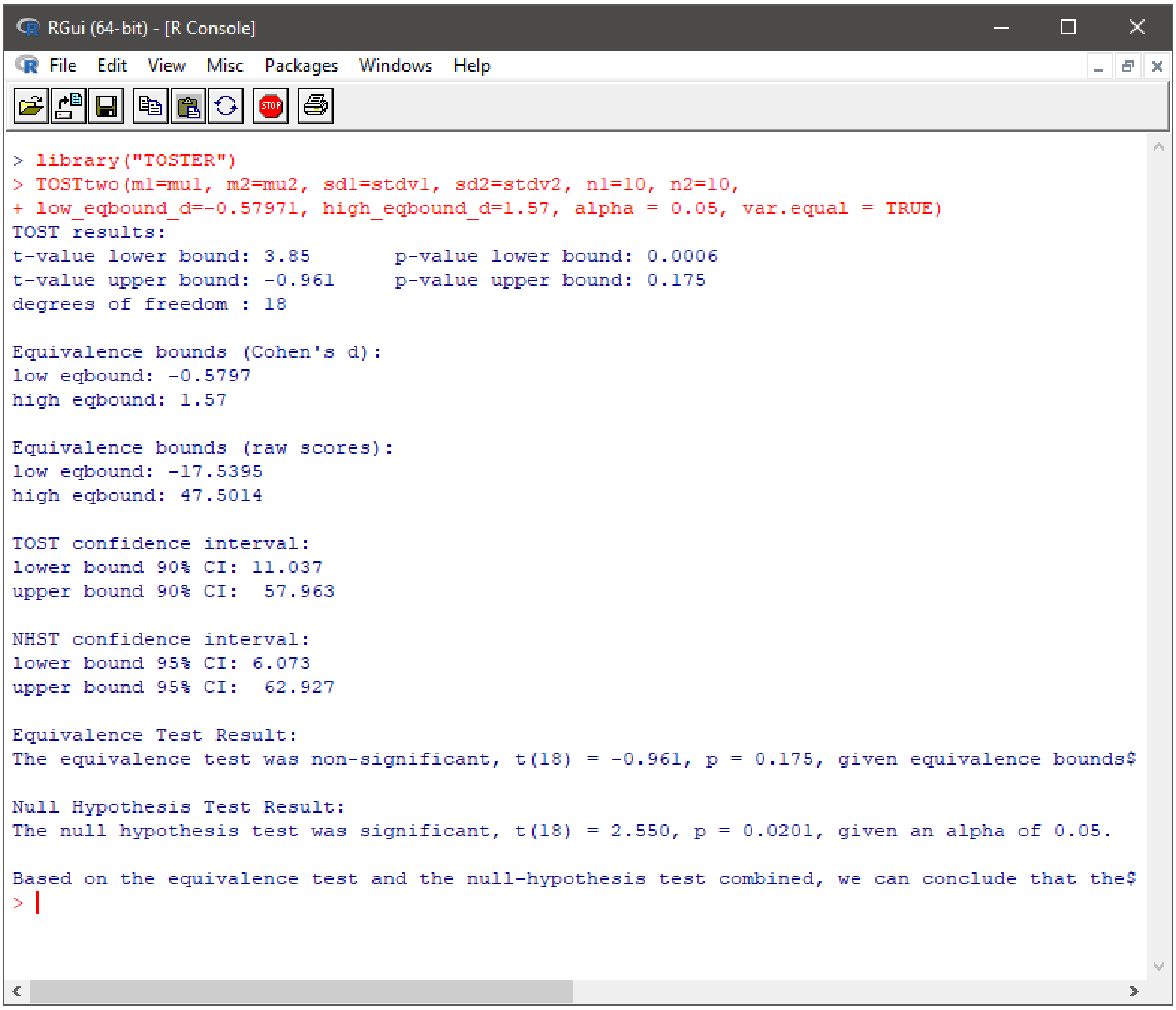


où le paramètre *epsilon* correspond bien évidemment à notre δ du cours théorique.

Même si l'intervalle de confiance et la *p*-value sont correcte (correspondent aux calculs à la main et aux valeurs renvoyées par Minitab) la question est pourquoi renvoyer qu'une seule des *p*-valeurs et un seule intervalle de confiance? Toutefois la conclusion correspond à celle du cours théorique: Nous échouons à rejeter l'hypothèse nulle que la différence absolue est supérieur à 20.

Bref… cette fonction de ce package est fiable au niveau des conclusions mais on est loin de détails que l'on obtient avec Minitab (à moins qu'il y ait une subtilité dans les paramètres d'entrée qu'un jour un lecteur nous indiquera????)!

Voyons si le package **TOSTER** et sa fonction **TOSTtwo( )** font mieux ? :



Commençons par parler des paramètres d'entrée *low_eqbound_d* et *high_eqbound_d* qui ne sont pas intuitifs dans le cas présent si on ne lit pas la documentation! Ces deux paramètres correspondent au nombre de fois qu'on doit retirer ou ajouter le *d* de Cohen:

$$d \text{ de Cohen} = \frac{|\mu_2 - \mu_1|}{\text{SE}}$$

à lui-même afin d'obtenir :

$$d \text{ de Cohen} + x(d \text{ de Cohen}) = \frac{|\mu_2 - \mu_1| \pm \delta}{\text{SE}}$$

Maintenant analysons les résultats! D'abord on peut remarquer que les t-values sont très similaires mais pas identiques à celles calculés à la main. Cela s'explique par le fait que le package utilise en réalité de la correction de Welch et fait le calcul non pas sur 18 degrés de liberté mais sur 17.907.

Les intervalles de confiance correspondent aux calculs à la main et nous pouvons constater que nous retrouvons un de ceux donnés par le package **equivalence**.

Maintenant parlons des phrases complètes non visibles ci-dessus dans la capture d'écran et qui sont:

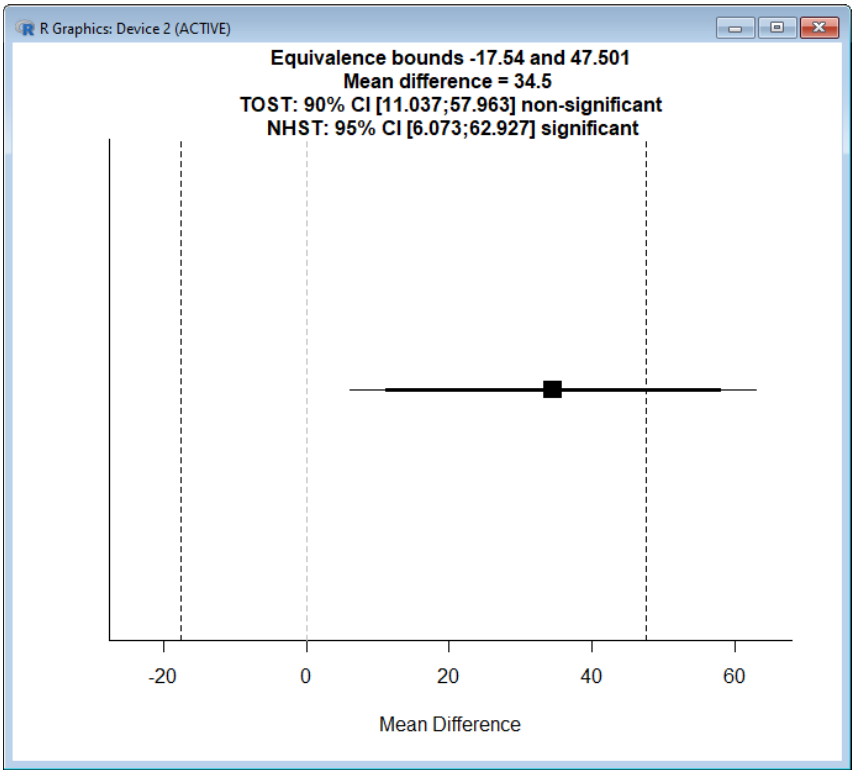
```
Equivalence Test Result:
The equivalence test was non-significant, t(18) = -0.961, p = 0.175,
given equivalence bounds of -17.540 and 47.501 (on a raw scale) and
an alpha of 0.05.

Null Hypothesis Test Result:
The null hypothesis test was significant, t(18) = 2.550, p = 0.0201,
given an alpha of 0.05.

Based on the equivalence test and the null-hypothesis test combined,
we can conclude that the observed effect is statistically different
from zero and statistically not equivalent to zero.
```

Bon les *t*-value et *p*-value obtenues ici ne correspondent ni à ceux obtenus dans le cours théorique, ni à celle renvoyée par le package **equivalence** ni par Minitab. Toutefois la conclusion reste la bonne et la même!

Et cette dernière fonction donne aussi le graphique suivant:



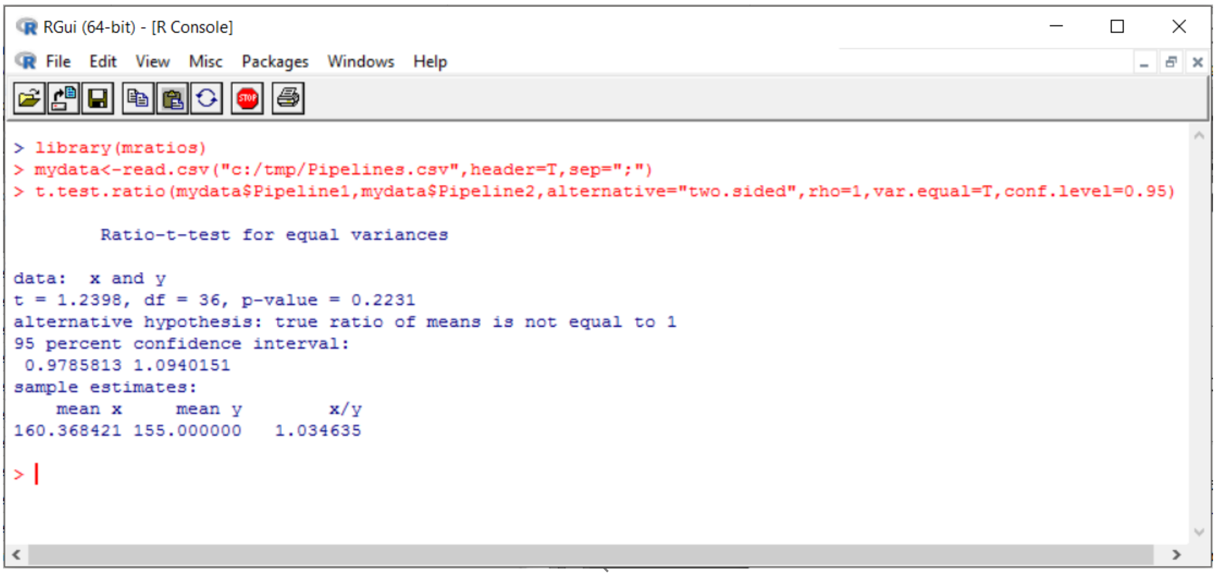
Notez que le package **TOSTER** a aussi une fonction pour faire un TOST des proportions nommée **TOSTtwo.prop( )** et aussi d'autres TOST de Student pour les données appariées!

Exemple 201.: Test du ratio des moyennes de Fieller

R 3.1.1

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours concernant le ratio de deux moyennes et son intervalle de confiance.

En utilisant les mêmes données que pour l'exemple précédent, nous obtenons (attention!!! le "rho" de cette fonction n'est pas la corrélation mais l'hypothèse nulle… hummm… hummm…):



```
> library(mratios)
> mydata<-read.csv("c:/tmp/Pipelines.csv",header=T,sep=";")
> t.test.ratio(mydata$Pipeline1,mydata$Pipeline2,alternative="two.sided",rho=1,var.equal=T,conf.level=0.95)

        Ratio-t-test for equal variances

data:  x and y
t = 1.2398, df = 36, p-value = 0.2231
alternative hypothesis: true ratio of means is not equal to 1
95 percent confidence interval:
 0.9785813 1.0940151
sample estimates:
    mean x     mean y        x/y
160.368421 155.000000   1.034635

>
```

Ce qui correspond à peu près aux calculs faits à la main où nous avions obtenu:

$$\theta_{1,2}=\frac{\dfrac{X}{Y}\pm\sqrt{\left(\dfrac{X}{Y}\right)^2 g+\dfrac{a}{b}g(1-g)}}{1-g}\cong\{0.987,1.084\}$$

Malheureusement nous sommes obligés de mettre des vecteurs dans la fonction de ce package ce qui limite son utilisation.

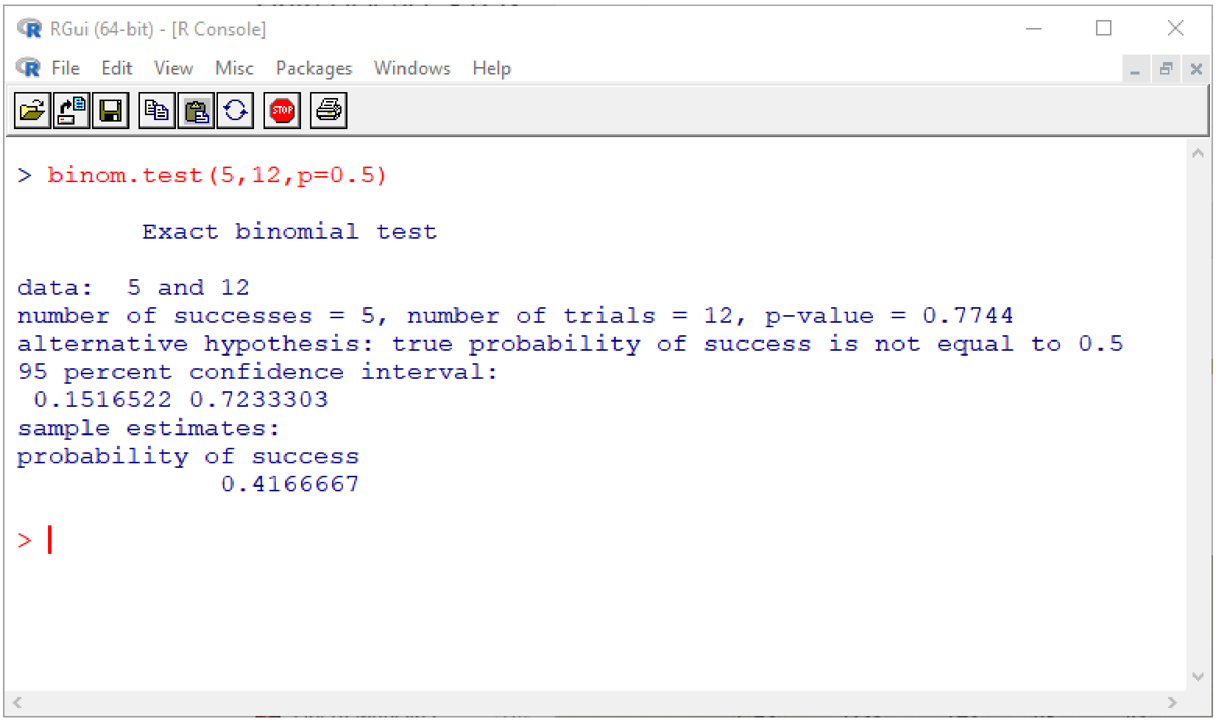
ToDo: Trouver une manière de contourner le faire de devoir mettre les vecteurs avec les valeurs sources.

Exemple 202.: Comparaison de proportions sur une même population (test binomial exact)

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons donc la commande **binom.test( )** :



Nous retrouvons donc exactement les mêmes données que celles calculées à la main et dans Microsoft Excel et Minitab (nous parlons alors parfois de test binomial exact de Clopper-Pearson).

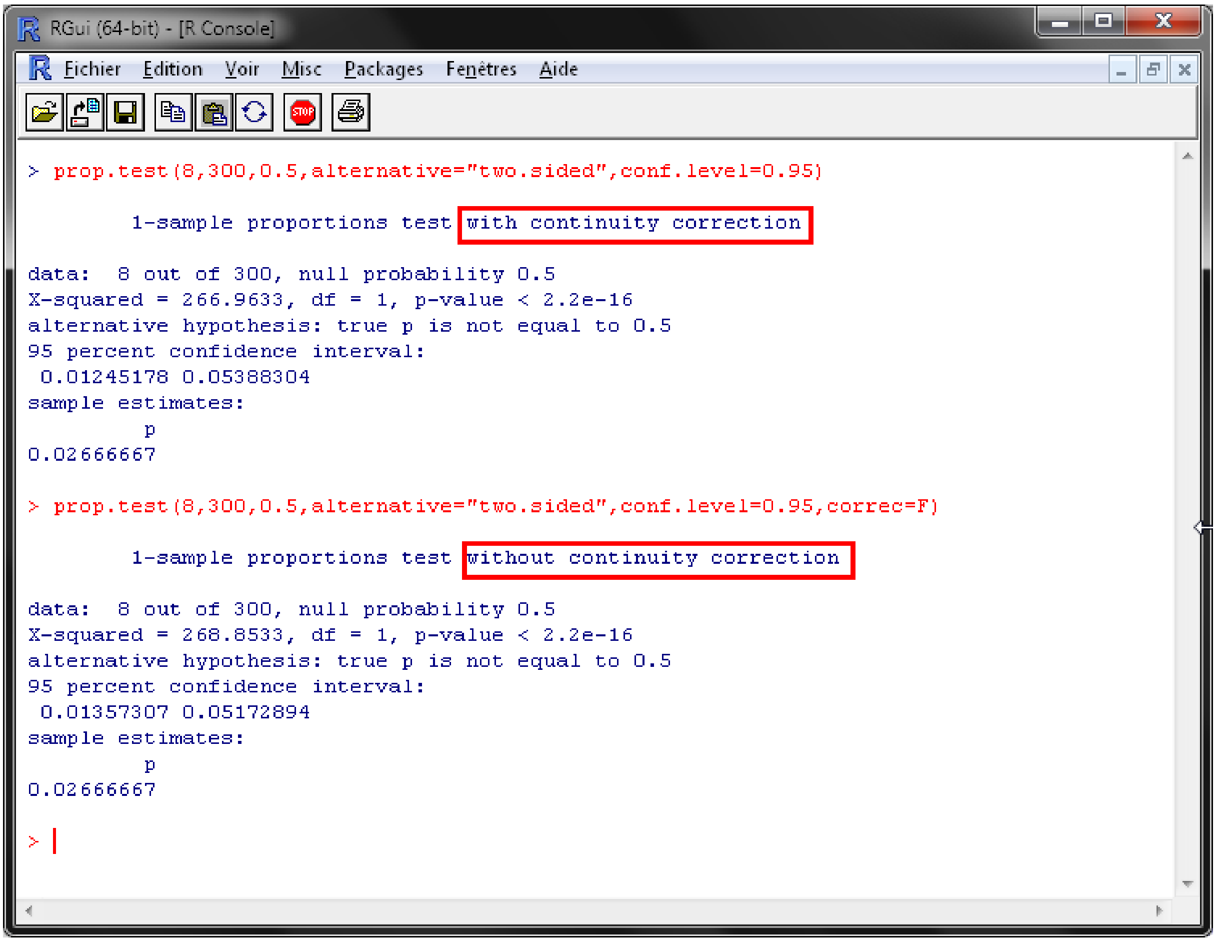
Nous retrouvons donc les bons résultats avec les mêmes conclusions.

Exemple 203.: Intervalle de confiance de la proportion

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons donc la commande **prop.test( )** avec ou sans correction de Yates:



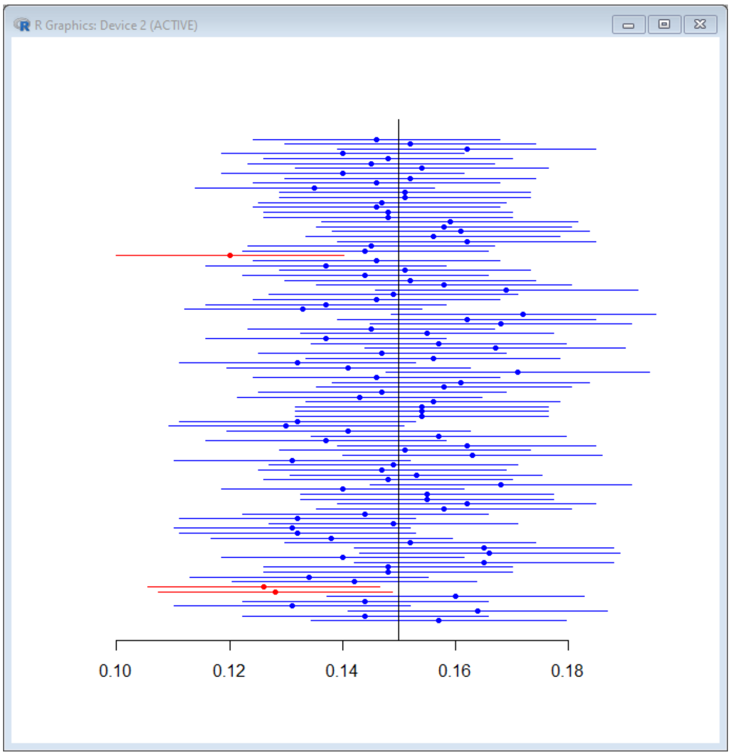
Dans les deux cas, nous voyons que nous ne retrouvons pas le même résultat que les calculs faits à la main et dans Microsoft Excel pour l'intervalle de confiance (mais pour le reste c'est OK...). Il y a une différence d'environ 10% en ce qui concerne les bornes c'est donc non négligeable. La raison de cette différence s'explique simplement par le fait que R utilise le test du chi-deux qui comme nous l'avons démontré dans le cours théorique est équivalent au carré du test Z.

Plus simplement: la p-valeur de prop.test de R est le score Z de Rao et l'intervalle de confiance et le CI de Wald.

Voyons aussi un cas ludique d'application copié du blog d'Arthur CHARPENTIER et qui est instructif aussi bien au niveau de l'apprentissage de R (manipulation de matrice et vecteurs et certains subtilités graphiques) que de la compréhension de statistiques:

```
> #exemple pris de: https://freakonometrics.hypotheses.org/18117
> ############################################
> #partie intervalle de confiance de la proprtion (binomiale)
> n <- 1000 #nombre d'événements (contrats d'assurance))
> xbar <- 150 #nombre d'événements d'intérêt (accidents déclarés)
> ns <- 100 #nombre de simulations
> #on produit un vecteur de 100*1000 éléments binaires de probabilité 150/1000
> v<-rbinom(n*ns,size=1,prob=xbar/n)
> #on les range en 1000 lignes dans une matrice
> #(l'idée étant que chaque colonne contienne une simulation)
> M<-matrix(v,nrow=n)
>
> #on crée une fonction pour calculer l'intervalle de confiance de le proportion à 95%
> fIC<-function(x) mean(x)+c(-1,1)*1.96*sqrt(mean(x)*(1-mean(x)))/sqrt(n)
> #on l'applique sur chacune des ns colonne de la matrice M
> #ce qui donnera donc 100 (ns) intervalles de confiance
> head(IC<-t(apply(M,2,fIC)),10)
          [,1]      [,2]
 [1,] 0.1344514 0.1795486
 [2,] 0.1222392 0.1657608
 [3,] 0.1410501 0.1869499
 [4,] 0.1100877 0.1519123
 [5,] 0.1222392 0.1657608
 [6,] 0.1372775 0.1827225
 [7,] 0.1072929 0.1487071
 [8,] 0.1054318 0.1465682
 [9,] 0.1203656 0.1636344
[10,] 0.1128862 0.1551138
> #on calcule aussi les 100 (ns) moyennes empiriques
> MN<-apply(M,2,mean)
> #on crée une matrice contenant de façon binaire les cas ou la moyenne attendu est en
> #dehors des intervalles simulés
> k<-(xbar/n<IC[,1])|(xbar/n>IC[,2])
> #on fait un plot pour mettre cela en évidence de manière sympa (de façon astucieuse)
> plot(MN,1:ns,xlim=range(IC),axes=FALSE,xlab="",ylab="",pch=19,cex=.7,col=c("blue","red")[1+k])
> axis(1)
> segments(IC[,1],1:ns,IC[,2],1:ns,col=c("blue","red")[1+k])
> abline(v=xbar/n)
>
```

Ce qui donne graphiquement:

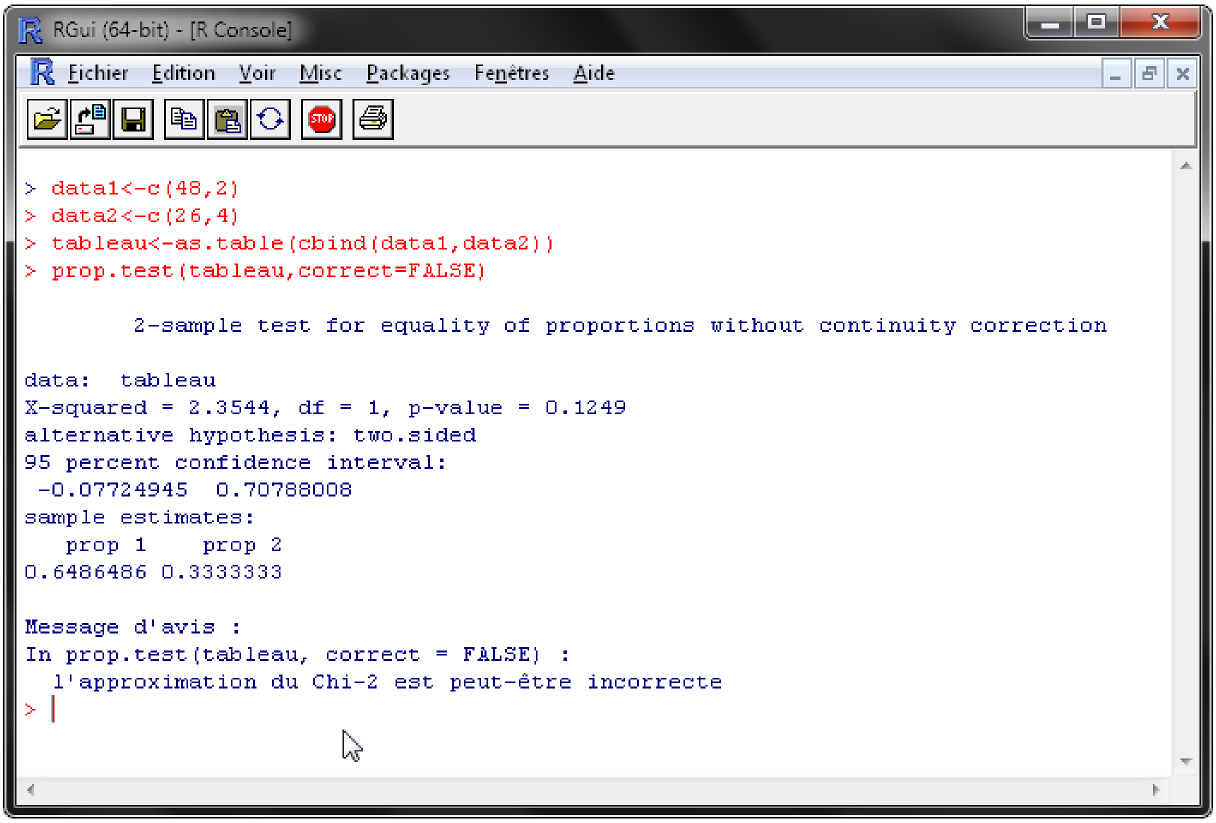


Exemple 204.: Comparaison de proportions sur 2 échantillons indépendants

R 3.0.2

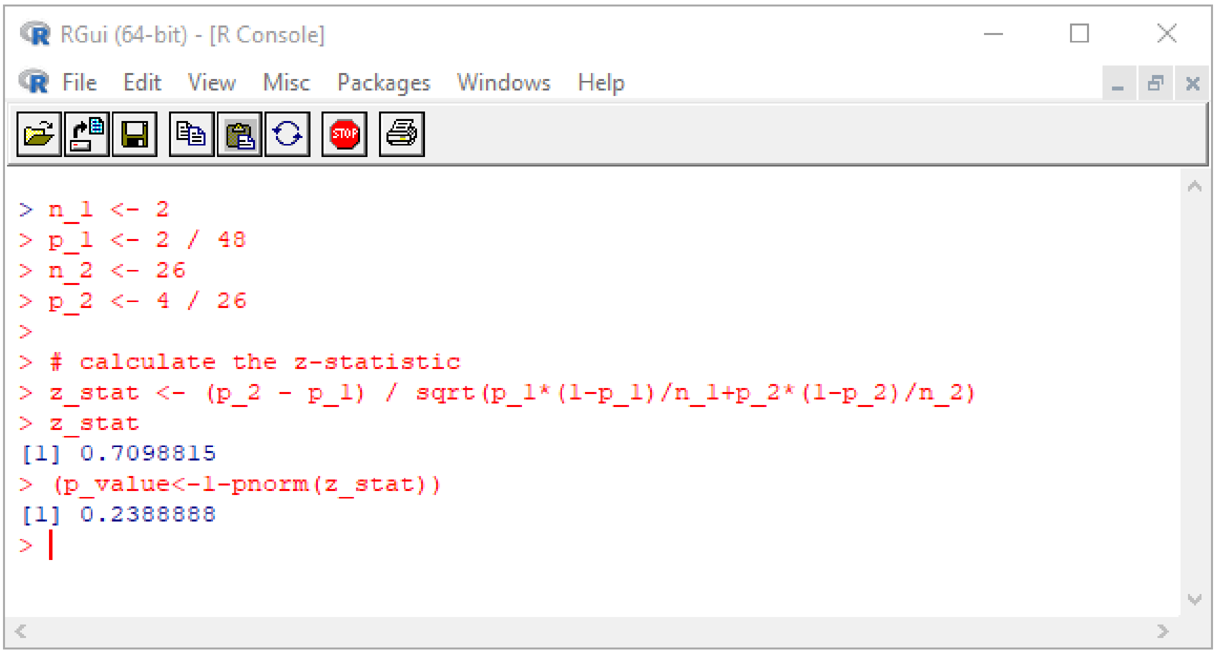
Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons toujours la commande **prop.test( )** sans correction de Yates:



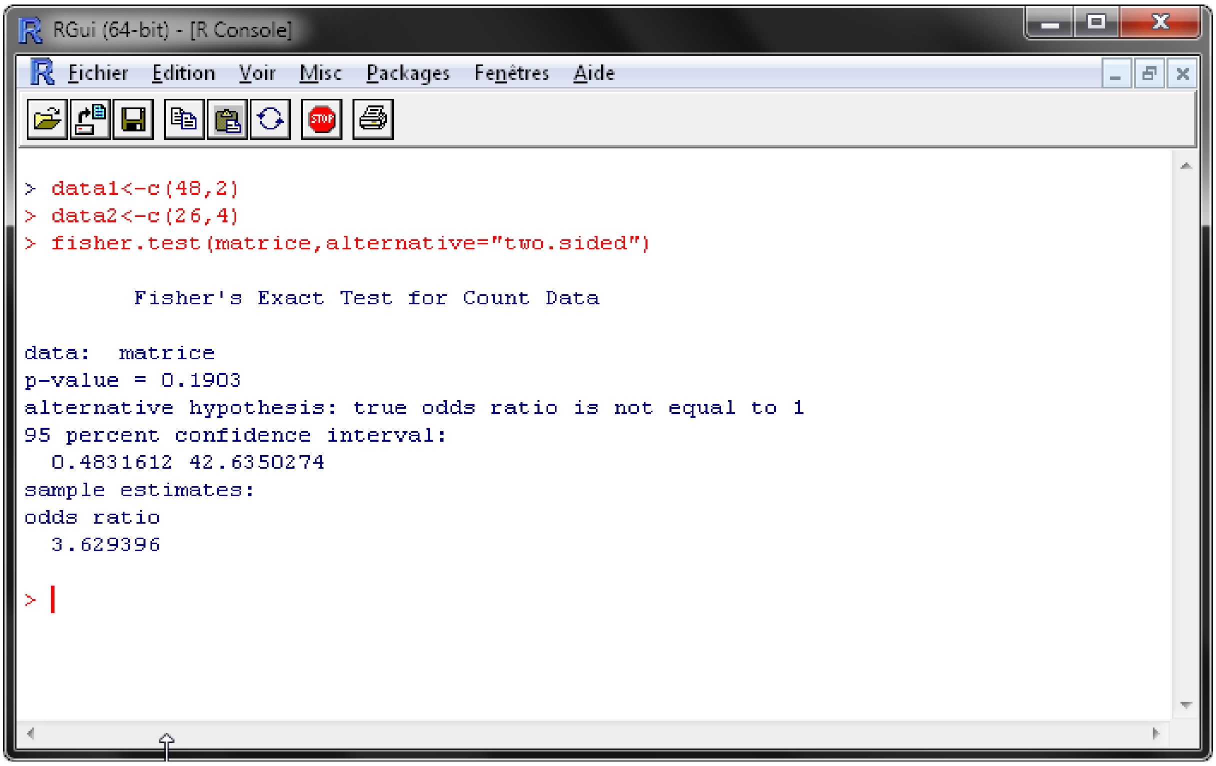
Donc nous ne retrouvons évidemment pas la même chose que dans Minitab et dans Microsoft Excel puisque nous y avions fait une approximation par la loi Normale alors que R fait une approximation par la loi de Khi-2 (bon la conclusion reste toutefois le même dans ce cas particulier). La raison de cette différence s'explique simplement par le fait que R utilise le test du chi-deux qui comme nous l'avons démontré dans le cours théorique est équivalent au carré du test Z.

Je n'ai pas trouvé de package faisant le test réel Z de deux proportions dans R, mais il peut facilement être écrit à la main faute de mieux:



Et on voit que la conclusion reste la même!

Si nous faisons comme Minitab immédiatement un test exact de Fisher (Minitab le fait automatiquement), nous avons avec R en utilisant la commande **fisher.test( )** :



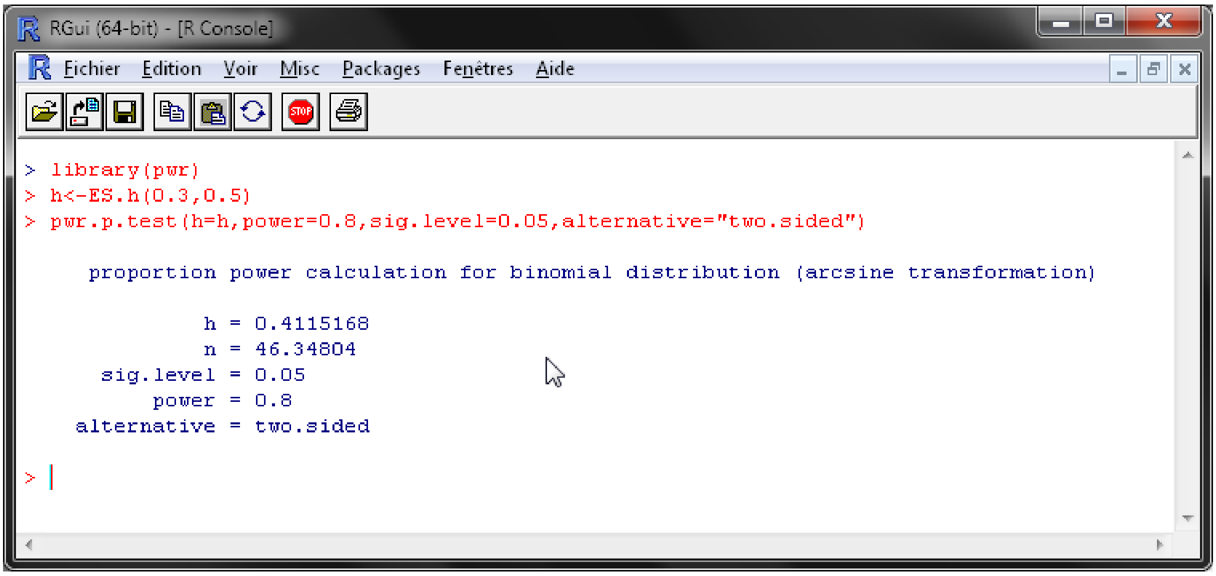
Le même résultat et la même conclusion qu'avec Minitab et MS Excel.

Exemple 205.: Taille d'échantillon (effectif) d'un test p (proportion) à 1 échantillon en bilatéral

R 3.0.2

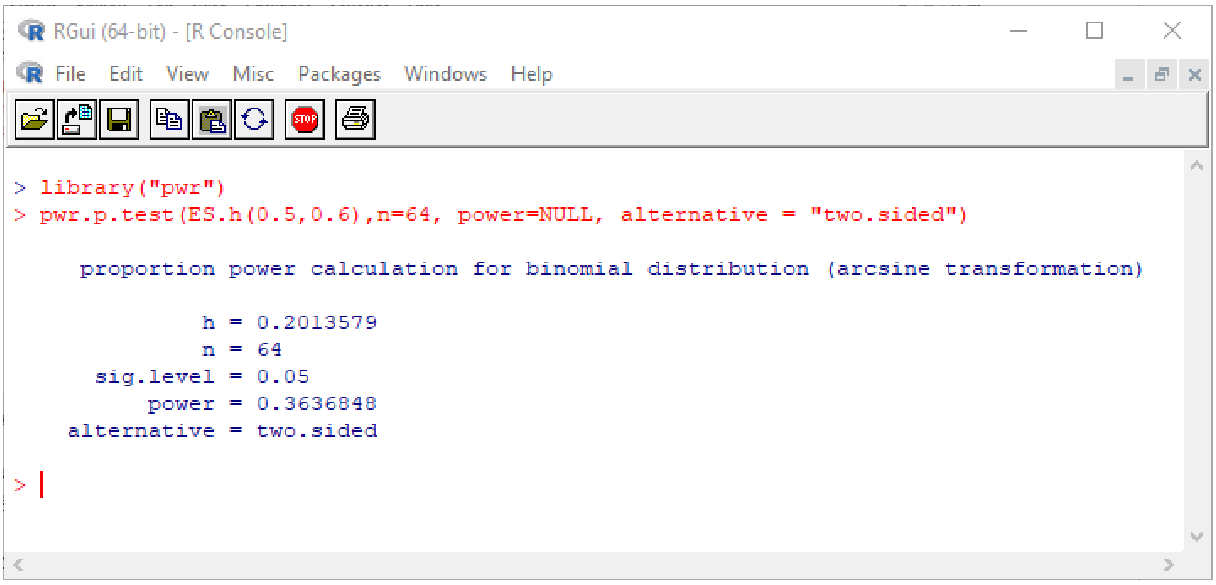
Calculer la taille nécessaire d'un échantillon a priori pour avoir une certaine puissance du test est aussi important dans la démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Pour cela, nous allons encore une fois utiliser la commande **pwr.p.test( )** du package **pwr**:



Nous retrouvons donc bien les mêmes valeurs que dans Minitab et toujours la même différence qu'avec ce que nous avons calculé à la main dans le cours théorique.

Sinon pour calculer la puissance, en se basant aussi sur le calcul du cours théorique, il semble que l'équivalent dans R soit:



au lieu des 0.3927 obtenus à la main (probablement… mais sans certitude (!)… à cause de l'approximation en arcsinus). Notez qu'on obtient exactement la même valeur numérique qu'avec Python.

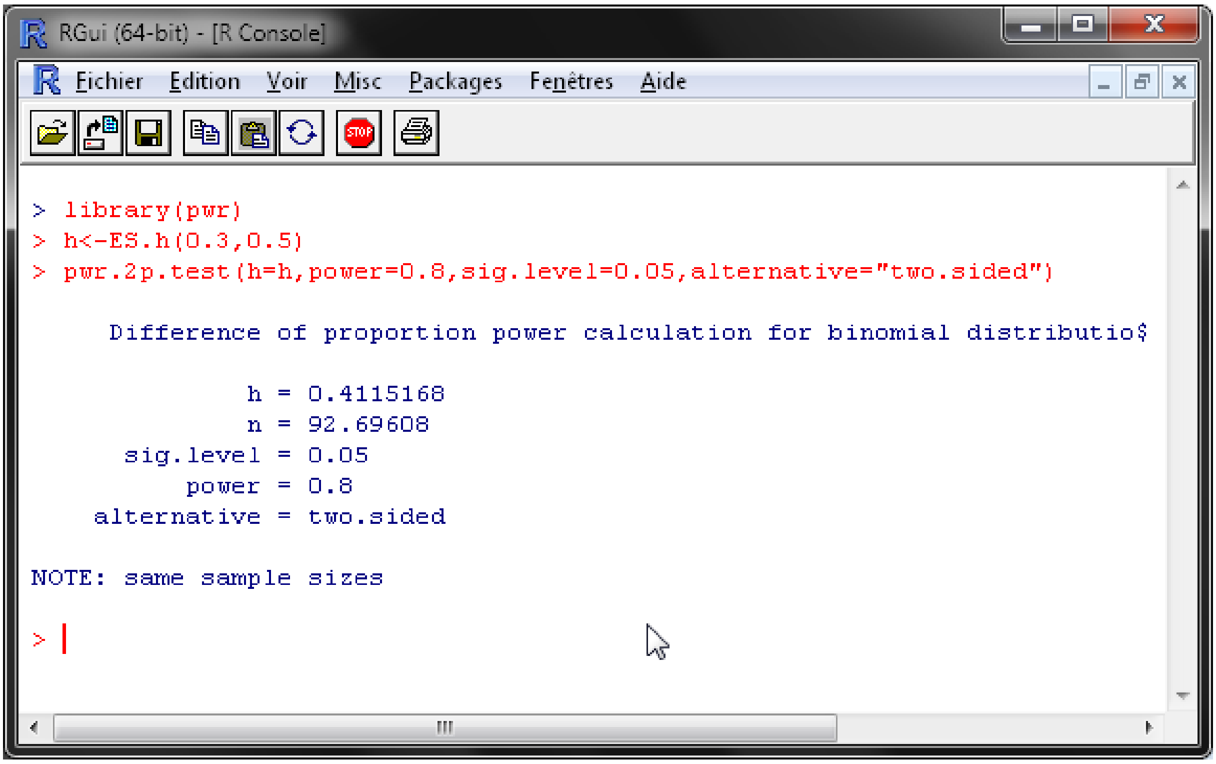
Exemple 206.: Taille d'échantillon (effectif) d'un test p (proportion) à 2 échantillons en bilatéral

R 3.0.2

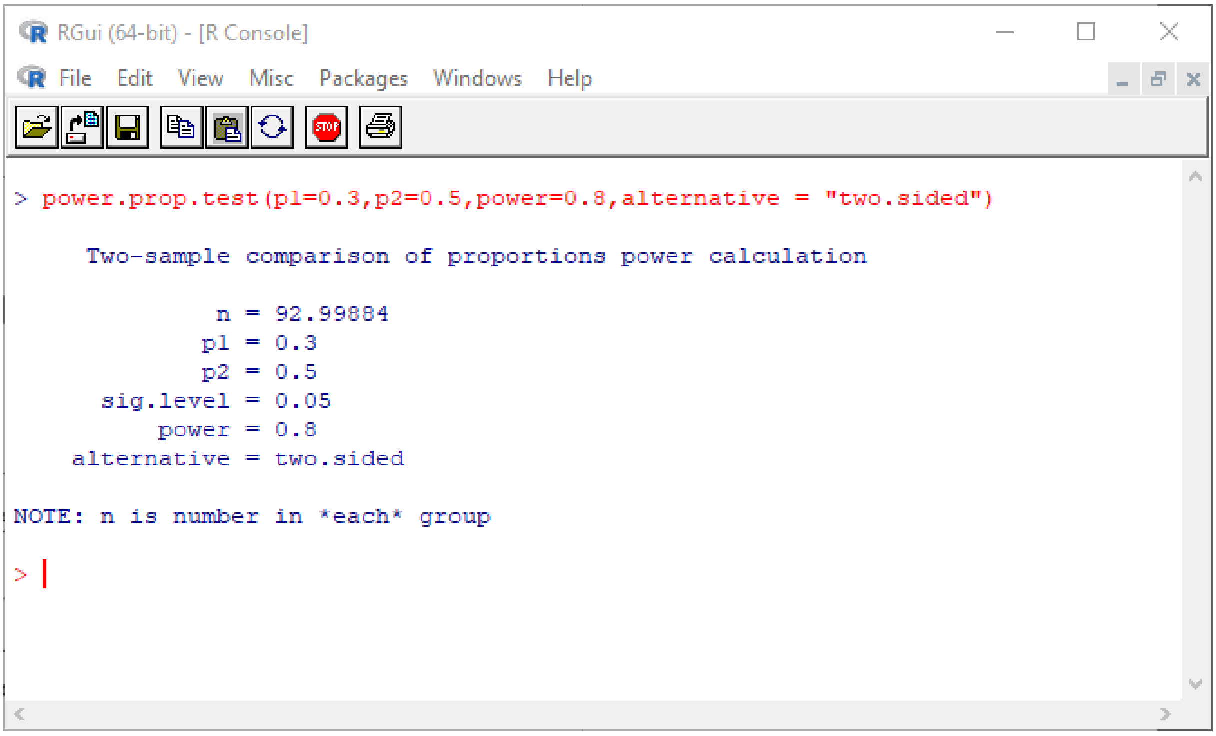
Calculer la taille nécessaire d'un échantillon a priori pour avoir une certaine puissance du test est aussi important dans le démarche scientifique. Nous allons ici vérifier que nous retrouvons ou non les mêmes résultats que ceux calculés à la main dans le cours théorique ou avec Minitab et MS Excel.

Imaginons que nous souhaiterions mettre en évidence une différence de 20% dans une étude (ou sondage) où nous nous attendons à avoir 50% de proportion expérimentale (donc une proportion de comparaison de 50-20=30%)

Pour cela, nous allons encore une fois utiliser la commande **pwr.2p.test( )** du package **pwr**:



On retrouve le même résultat avec la fonction native **power.prop.test( )**:

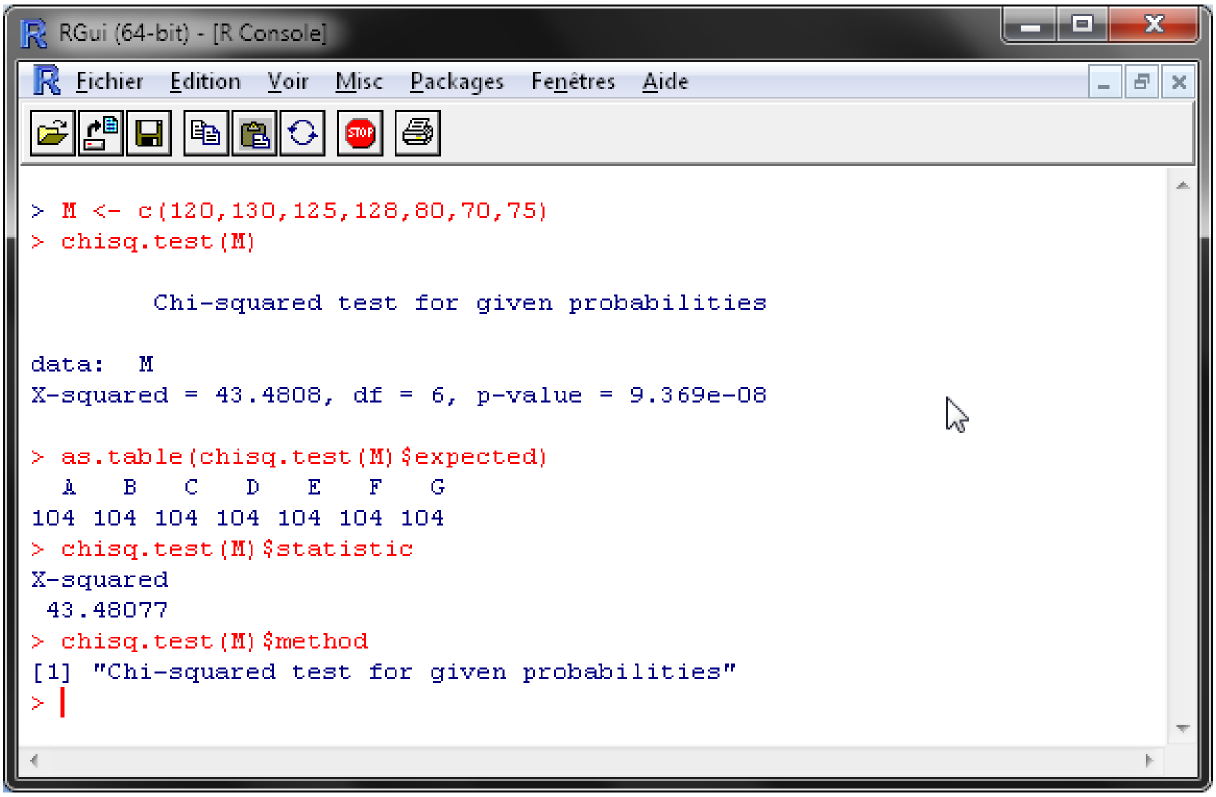


Nous retrouvons donc bien les mêmes valeurs que dans Minitab et toujours la même différence qu'avec ce que nous avons calculé à la main dans le cours théorique.

Exemple 207.: Test de *k*-proportions (test du khi-deux)

R 3.6.3

Rappelons que faire le test de (l'égalité) de *k*-proportions revient à faire un test (non-paramétrique) d'ajustement (indépendance) du khi-2 (voir page 1458):



Effectivement si on normalise le vecteur *M* ci-dessus en divisant chaque composante par le total des composantes, cela revient à comparer des proportions.

Dans la pratique on fait cependant l'inverse: pour satisfaire les conditions d'application du test, il faut multiplier le vecteur des proportions par 100!!!

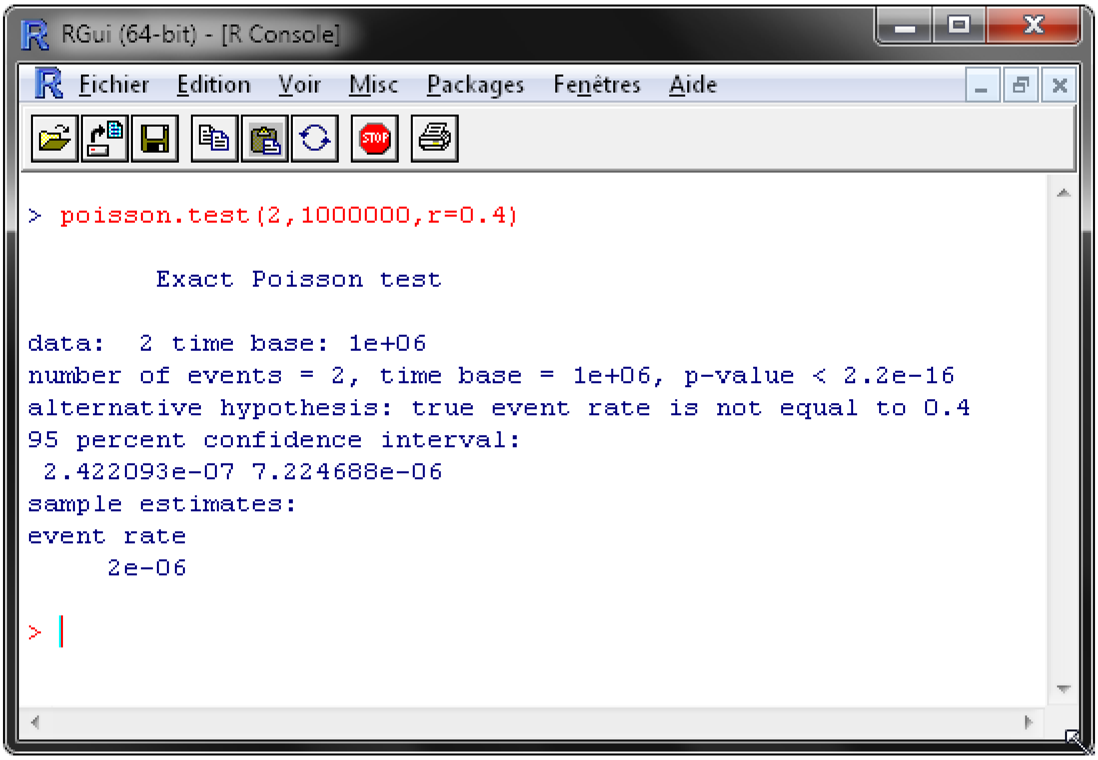
Exemple 208.: Test de Poisson à un échantillon unilatéral/bilatéral

R 3.0.2

Nous allons voir ici si nous retrouvons encore une fois les mêmes valeurs que celles calculées dans le cours théorique à la main ainsi que dans Minitab.

Nous partirons des mêmes exemples que dans le cours théorique.

Nous utilisons la commande native **poisson.test( )** et en bilatéral pour déterminer pour une compagnie d'aviation ayant eu 2 deux crashs en 1'000'000 de vols (événement très rare), quelle est l'intervalle de confiance en bilatéral à 95% sachant qu'au niveau mondial le nombre d'accidents par millions est de 0.4.

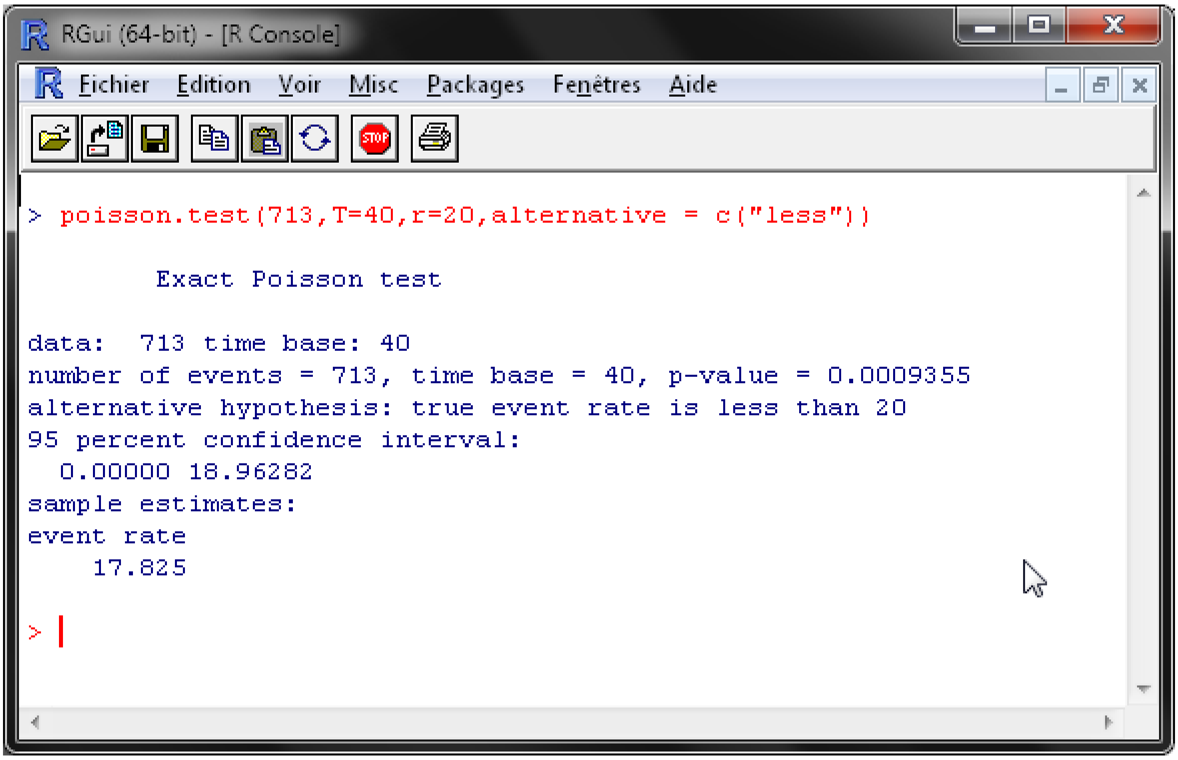


Donc là nous retrouvons les mêmes valeurs que dans le cours Minitab globalement, mais par contre pour la borne de gauche de l'intervalle, comme nous l'avons déjà mentionné dans le cours Minitab, le résultat ne correspond pas avec nos calculs faits à la main. Nous avions cependant montré dans le cours théorique quels étaient les calculs effectués en réalité dans l'algorithme (cependant sans en trouver la démonstration).

Maintenant procédons à l'exemple unilatéral fait aussi dans le cours Minitab avec les non-conformes.

Une société fabrique des télévisions en quantité constante et a mesuré le nombre d'appareils défectueux produits chaque trimestre pendant les dix dernières années (donc 4 fois 10 mesures = 40 trimestres). La direction décide que le nombre maximum acceptable d'unités défectueuses est de 20 par trimestre et souhaite déterminer si l'usine satisfait à ces exigences

(sous l'hypothèse que la distribution des défectueux suive une loi de Poisson) à un niveau de confiance de 5%.



Nous obtenons donc exactement le même résultat qu'avec Minitab!

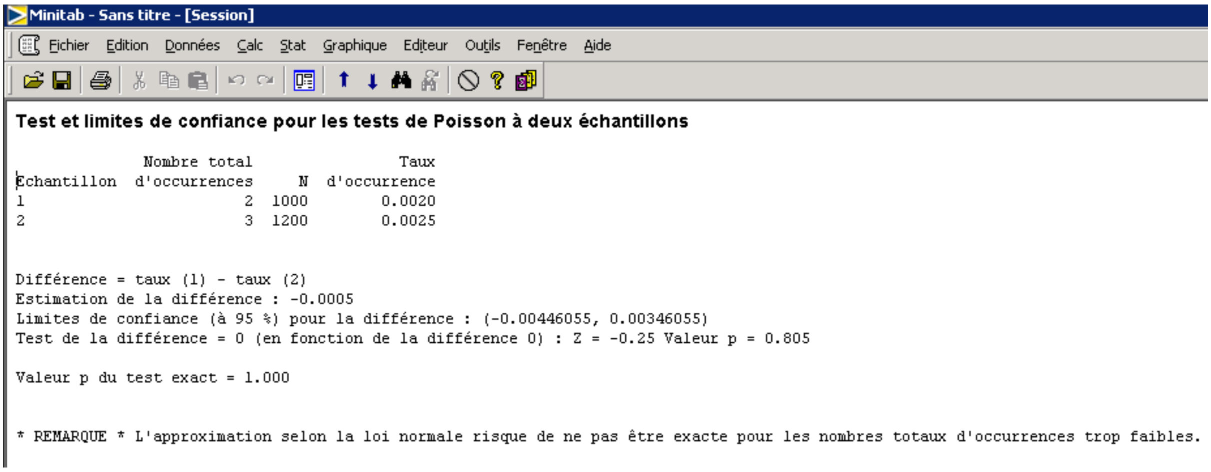
Exemple 209.: Test de Poisson de la différence de deux moyenne

R 3.2.0

Nous reproduisons comme à l'habitude l'exemple du cours théorique et fait avec Minitab et à la main:

Une compagnie d'aviation a eu 2 deux crashs en 1'000'000 de vols (événement très rare). Une autre compagnie a eu 3 crashs en 1'200'000 vols. Quel est l'intervalle de confiance en bilatéral à 95% en supposant que la différence devrait être nulle.

Minitab nous donnait pour info:



Test et limites de confiance pour les tests de Poisson à deux échantillons

```
                Nombre total              Taux
Echantillon  d'occurrences     N  d'occurrence
1                        2  1000        0.0020
2                        3  1200        0.0025

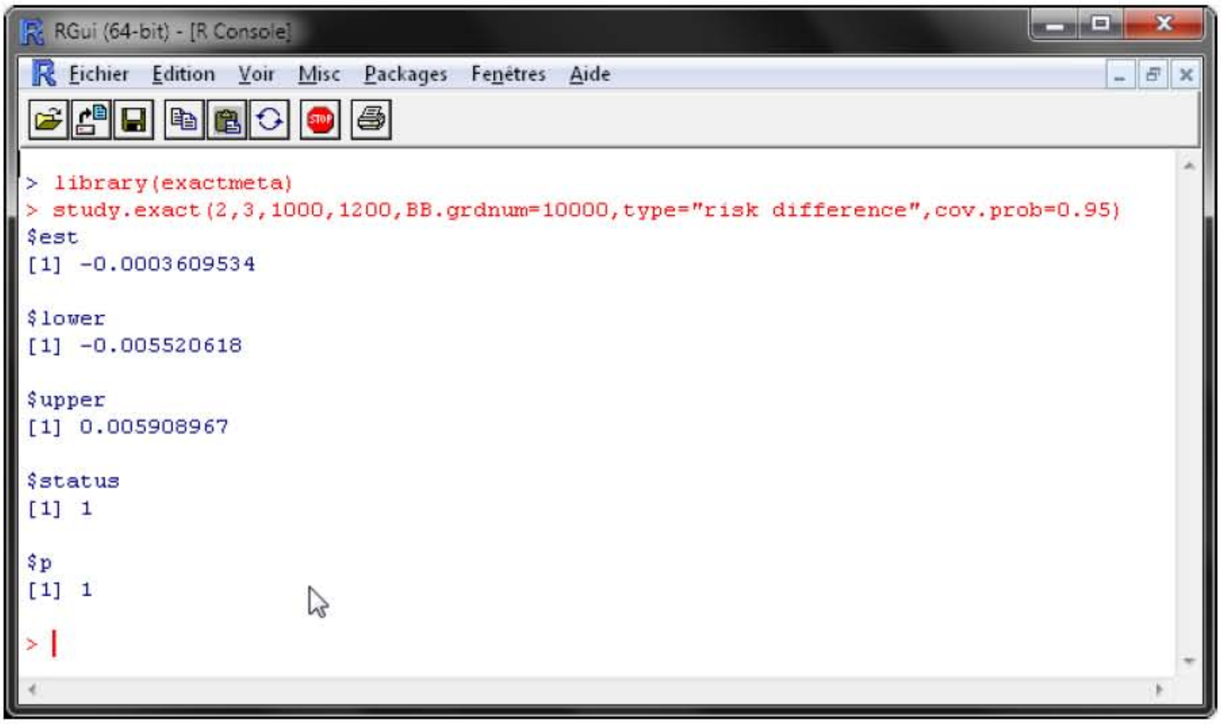

Différence = taux (1) - taux (2)
Estimation de la différence : -0.0005
Limites de confiance (à 95 %) pour la différence : (-0.00446055, 0.00346055)
Test de la différence = 0 (en fonction de la différence 0) : Z = -0.25 Valeur p = 0.805

Valeur p du test exact = 1.000


* REMARQUE * L'approximation selon la loi normale risque de ne pas être exacte pour les nombres totaux d'occurrences trop faibles.
```

R n'implémente pas cette méthode à ce jour avec les hypothèses sous-jacente du modèle que nous avons vu dans le cours théorique (hypothèse de continuité, stabilité de la loi de Poisson).

Par contre, la fonction **study.exact( )** du package **exactmeta** contient bien le test de Poisson de la différence des moyennes mais sans correction de continuité et comme nous pouvons le voir le résultat est significativement différent de la sortie de Minitab et des calculs effectués à la main même si la conclusion reste la même dans ce cas particulier (j'ai par ailleurs manifesté mon scepticisme des hypothèses utilisées pour construire ce test dans le cours théorique… donc comme quoi…).

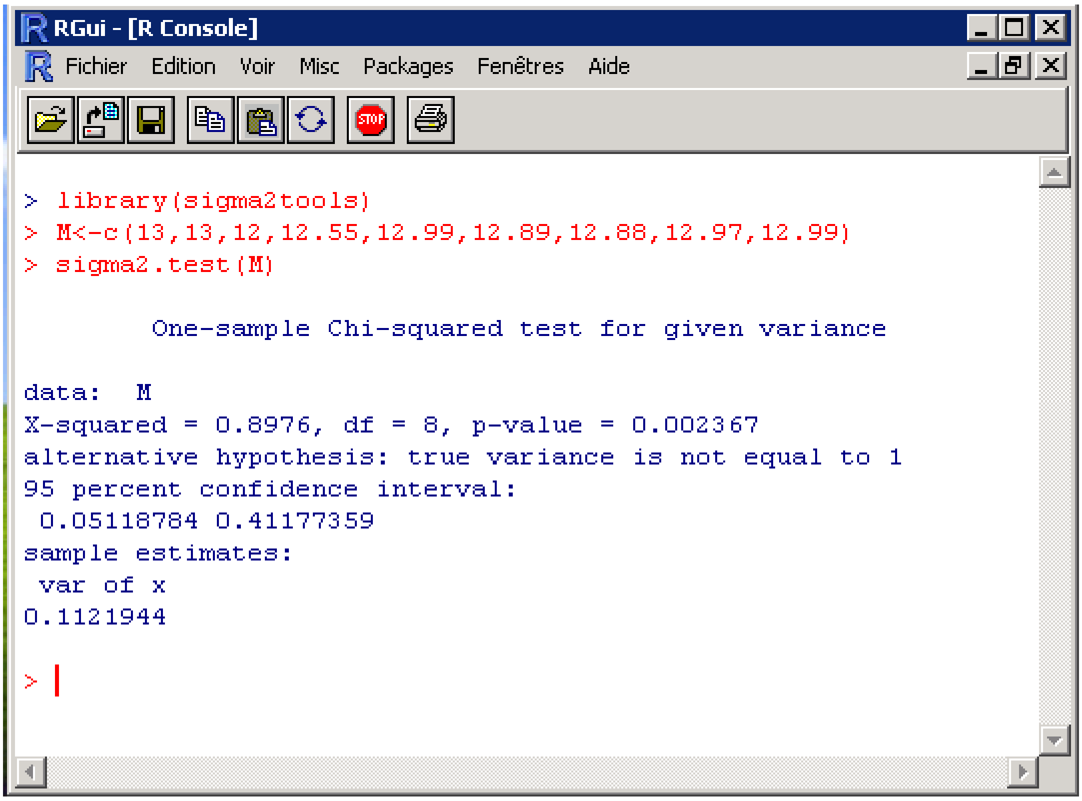

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(exactmeta)
> study.exact(2,3,1000,1200,BB.grdnum=10000,type="risk difference",cov.prob=0.95)
$est
[1] -0.0003609534
$lower
[1] -0.005520618
$upper
[1] 0.005908967
$status
[1] 1
$p
[1] 1
>

Exemple 210.: Intervalle de confiance de l'écart-type (test du khi-2 de la variance)

R 2.9.0

Nous voulons déterminer l'intervalle de confiance de cette variance (c'est comme-si nous cherchions à comparer avec un ratio où nous avons 1 au dénominateur) en utilisant la fonction **sigma2.test( )** du package **sigma2tools**.

Un échantillon de 9 vis a été tiré d'une ligne de production et la mesure de leur diamètre en [*mm*] a été reportée ci-dessous:



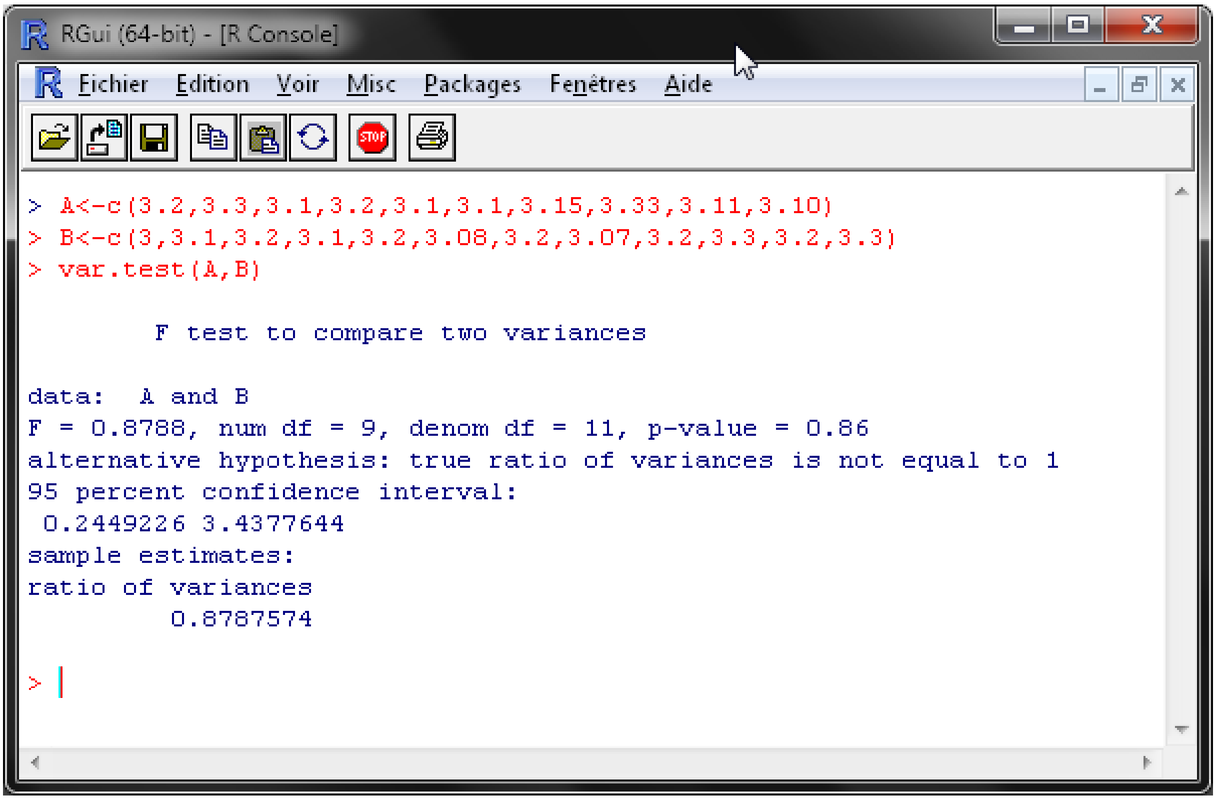
et tout est parfait! Nous retrouvons les mêmes résultats que dans le cours théorique et qu'avec Microsoft Excel et Minitab.

Exemple 211.: Test de Fisher d'égalité des variances

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Minitab et toujours avec les mêmes données.

Nous utilisons donc la commande **var.test( )** :



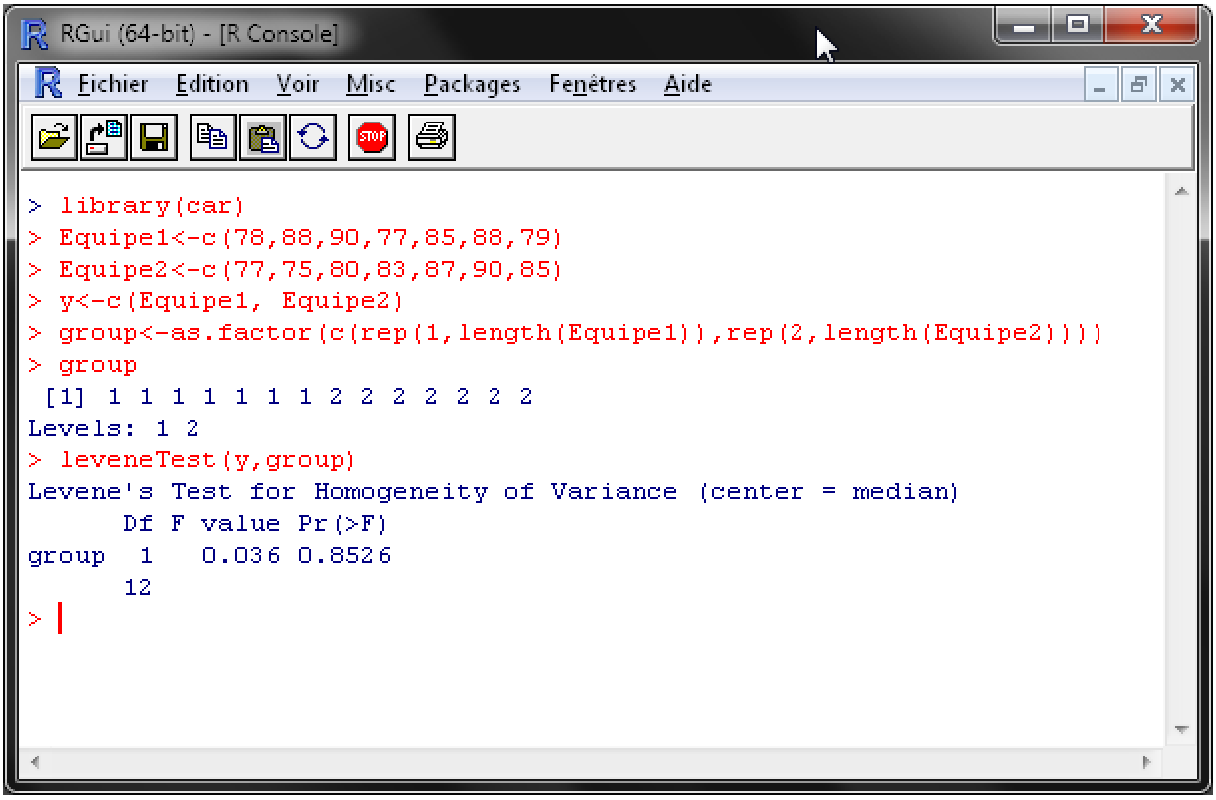
Nous retrouvons donc à peu près les mêmes valeurs que Minitab et MS Excel. À la différence que Minitab donne des informations que R ne donne pas et réciproquement (alors qu'avec Microsoft Excel on pouvait bien évidemment avoir toutes les informations).

Exemple 212.: Test de Levene et de Brown-Forsythe d'égalité de deux variances

R 3.0.2

Nous continuons donc avec le contrôle de conformité avec les démonstrations mathématiques faites en cours et les calculs faits à la main et dans Microsoft Excel(Minitab n'ayant pas le test de Levene mais de Brown-Forsythe même s'il indique le contraire) et toujours avec les mêmes données.

Nous utilisons donc la commande **leveneTest( )** du package **car** en préparant les données avec une structure peut triviale:



Nous retrouvons donc exactement les mêmes valeurs que celles calculées dans MS Excel. Par contre une valeur significativement différente de celle de Tanagra qui pour rappel était:

Levene's test 1

Parameters

Parameters	
Sort results	no

Results

Attribute_Y	Attribute_X	Description				Statistical test	
		Value	Examples	Average	Std-dev	Test	
Pièces	Equipe	Equipe 1	7	83,5714	5,4423	Levene's W	0,062976
		Equipe 2	7	82,4286	5,4116	df	2/18
		Equipe 3	7	83,8571	5,4292	p-value	0,939172
		All	21	83,2857	5,1879		

Computation time : 0 ms.
Created at 16/08/2013 16:34:39

Même si la conclusion reste cependant la même!

Voyons maintenant avec la variante robuste de Brown-Forsythe pour voir si là encore nous retrouvons les mêmes résultats (et bien évidemment avec les mêmes conclusions). Nous allons devoir utiliser la commande **levene.test( )** du package **lawstat**:

RGui (64-bit) - [R Console]

```
> library(lawstat)
> Equipe1<-c(78,88,90,77,85,88,79)
> Equipe2<-c(77,75,80,83,87,90,85)
> y<-c(Equipe1, Equipe2)
> group<-as.factor(c(rep(1,length(Equipe1)),rep(2,length(Equipe2))))
> group
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2
Levels: 1 2
> levene.test(y,group,location="median")

        modified robust Brown-Forsythe Levene-type test based on the absolute
        deviations from the median

data:  y
Test Statistic = 0.036, p-value = 0.8526

> 
```

Et nous retrouvons donc bien la même chose à la différence que R donne la statistique et la p-value avec plus de précision (comme à chaque fois quoi!). A comparer avec Tanagra aussi:

Brown - Forsythe's test 1

Parameters

Parameters	
Sort results	no

Results

Attribute_Y	Attribute_X	Description				Statistical test	
		Value	Examples	Average	Std-dev	Test	
Pièces	Equipe	Equipe 1	7	83,5714	5,4423	Brown & Forsythe's W	0,036036
		Equipe 2	7	82,4286	5,4116	df	1/12
		All	14	83,0000	5,2477	p-value	0,852614

Exemple 213.: Robustesse de tests statistiques

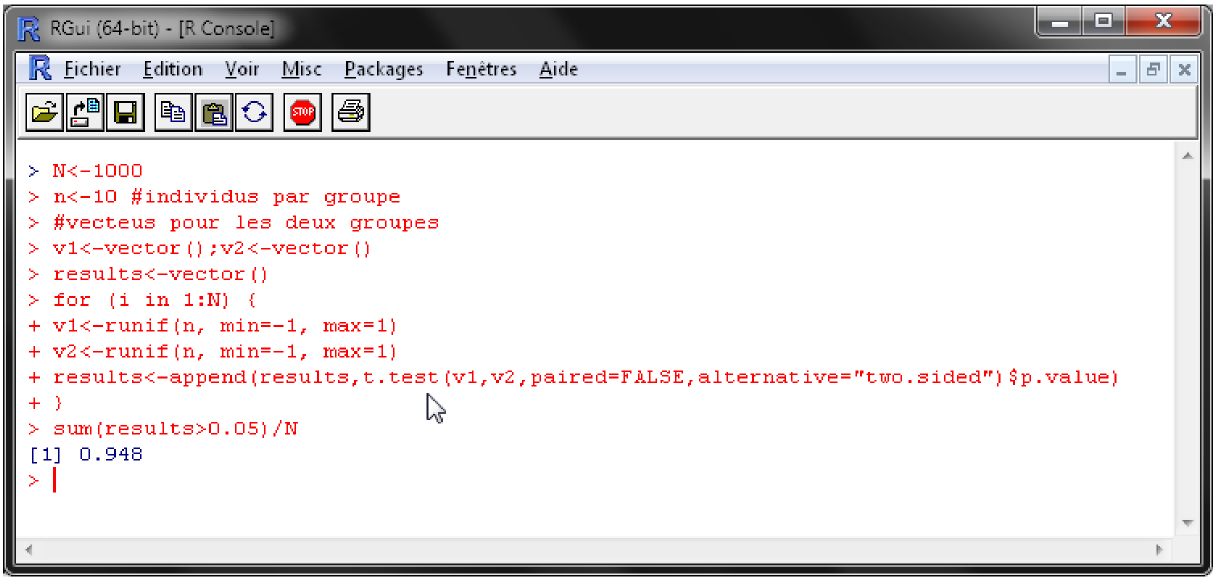
R 3.0.2

Nous allons voir ici comme il est facile de mettre en œuvre avec R ce que nous avons fait dans le cours Microsoft Excel et qui consistait à tester la robustesse de certains tests simples ou particulièrement complexes.

Nous n'allons pas ici refaire un exemple d'application pour chacun des tests (qu'ils soient paramétriques ou non paramétriques comme nous l'avions fait dans le cours MS Excel) car ce serait probablement plus ennuyeux qu'autre chose pour le lecteur.

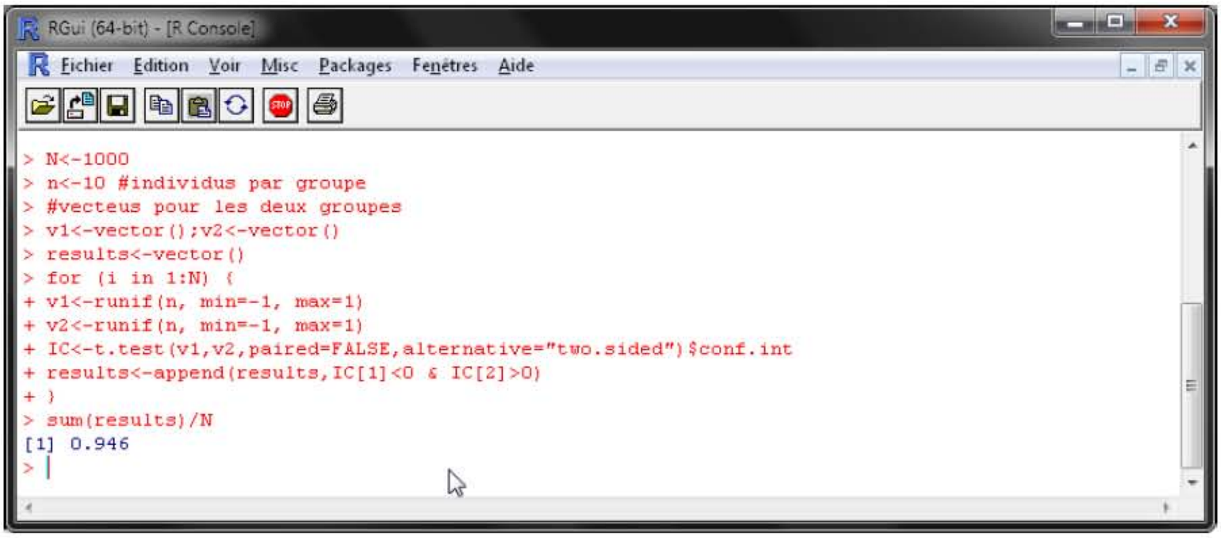
Afin de gagner du temps aussi de mon côté (car je déteste recréer la roue excepté si ce n'est pour faire mieux), j'ai reprise l'exemple mot pour mot, lettre par lettre du e-book de Vincent ZOONEKYND dont le but est de tester la robustesse du test de Student à la non-normalité des données (avec une distribution uniforme au lieu de Normale) et pour lequel il faudrait alors normalement utiliser un test non paramétrique de type Wilcoxon U.

Donc nous y allons pour faire 1'000 tests de Student pour échantillons indépendants en bilatéral et faire la moyenne des *p*-value qui sont au-delà de 5% (soit les vrais positifs dans le cas présent):



Donc nous avons 94.9% de vrais positifs. Donc le test est robuste pour une distribution uniforme qui est non pathologique relativement à la loi Normale. En tout cas cela donne l'approche de la méthode pour généraliser ensuite à tout autre test.

Nous pouvons faire de même avec l'intervalle de confiance (observez bien l'approche sémantique car c'est très pertinent!):


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> N<-1000
> n<-10 #individus par groupe
> #vecteus pour les deux groupes
> v1<-vector();v2<-vector()
> results<-vector()
> for (i in 1:N) {
+ v1<-runif(n, min=-1, max=1)
+ v2<-runif(n, min=-1, max=1)
+ IC<-t.test(v1,v2,paired=FALSE,alternative="two.sided")$conf.int
+ results<-append(results,IC[1]<0 & IC[2]>0)
+ }
> sum(results)/N
[1] 0.946
>

Exemple 214.: ANOVA (Analyses de la Variance)

Nous le savons, le nombre d'ANOVA possibles et imaginables existe en grand nombre. Nous allons donc ici uniquement nous concentrer sur celles étudiées et démontrées en détails dans le cours théorique.

Avant de commencer juste en petit rappel sur la notation dans R des modèles de régression s'avère après expérience indispensable:

Syntaxe(notation de Wilkinson)	Lecture
$Z \sim X + Y$	Le modèle est supposé du type $Z_i = X_i + Y_i$
$Z \sim X * Y$	Le modèle est supposé avec interactions du type $Z_i = X_i + Y_i + X_i Y_i$
$Z \sim X / Y$	Le modèle est supposé avec interactions imbriqué $Z_i = X_i + X_i Y_i$
$Z \sim X : Y$	Le modèle est seulement avec interactions $Z_i = X_i Y_i$
Z ~ factor(X)+Error(interaction(factor(X),factor(Y))	Le modèle est hiérarchique complet
Z ~ factor(X)+factor(Y)+Lettres	Modèle carré latin
	D'autres exemples suivront si besoin…

Tableau 2 Syntaxe pour les ANOVA

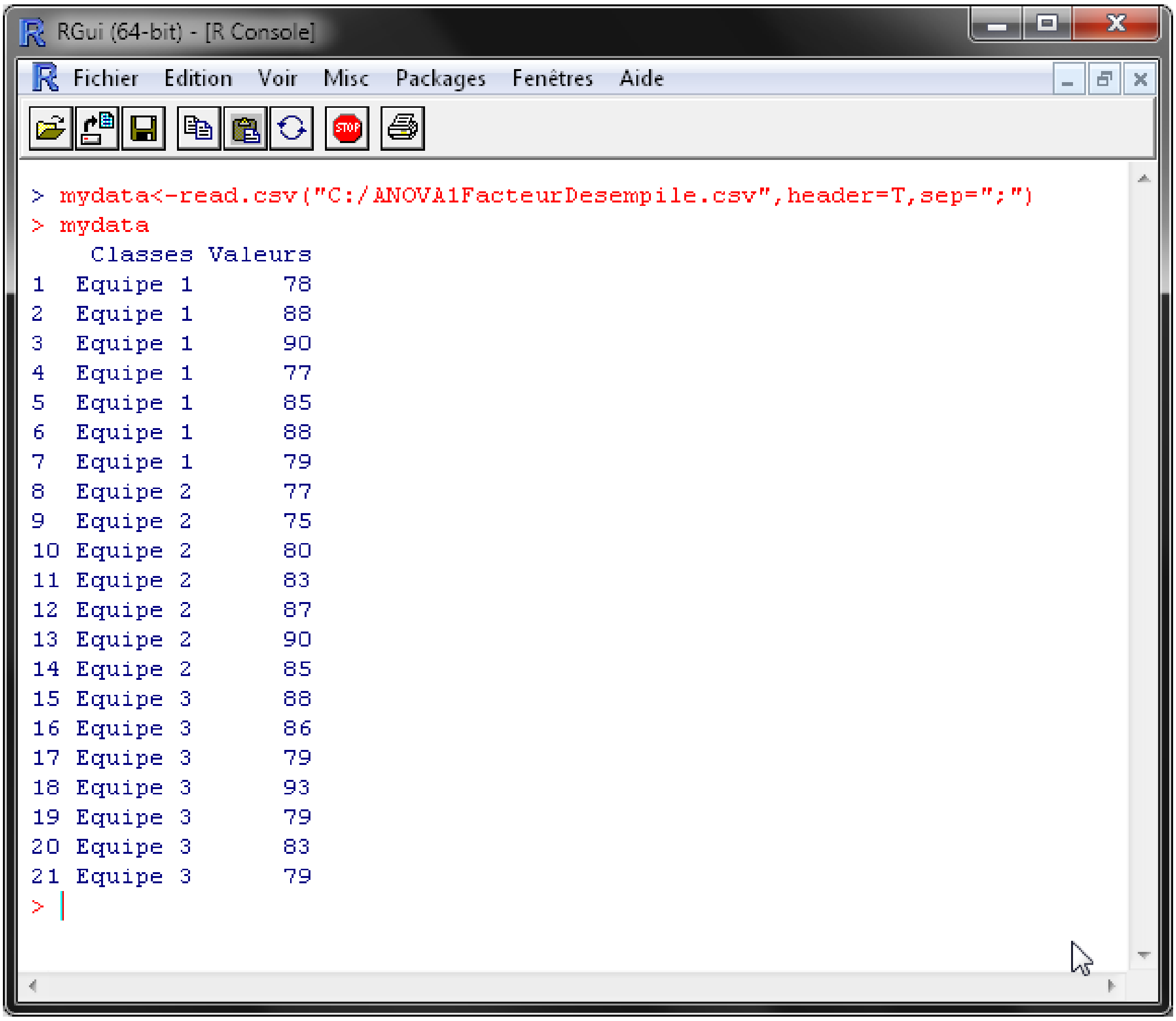
Attention! Z~X+Y+X*Y n'est pas égal à Z~Y+X+X*Y dans les cas non-balancés! (voir cours théorique pourquoi!).

Exemple 1.: ANOVA à un facteur fixe (ANOVA-1 canonique) empilé

R 3.0.2

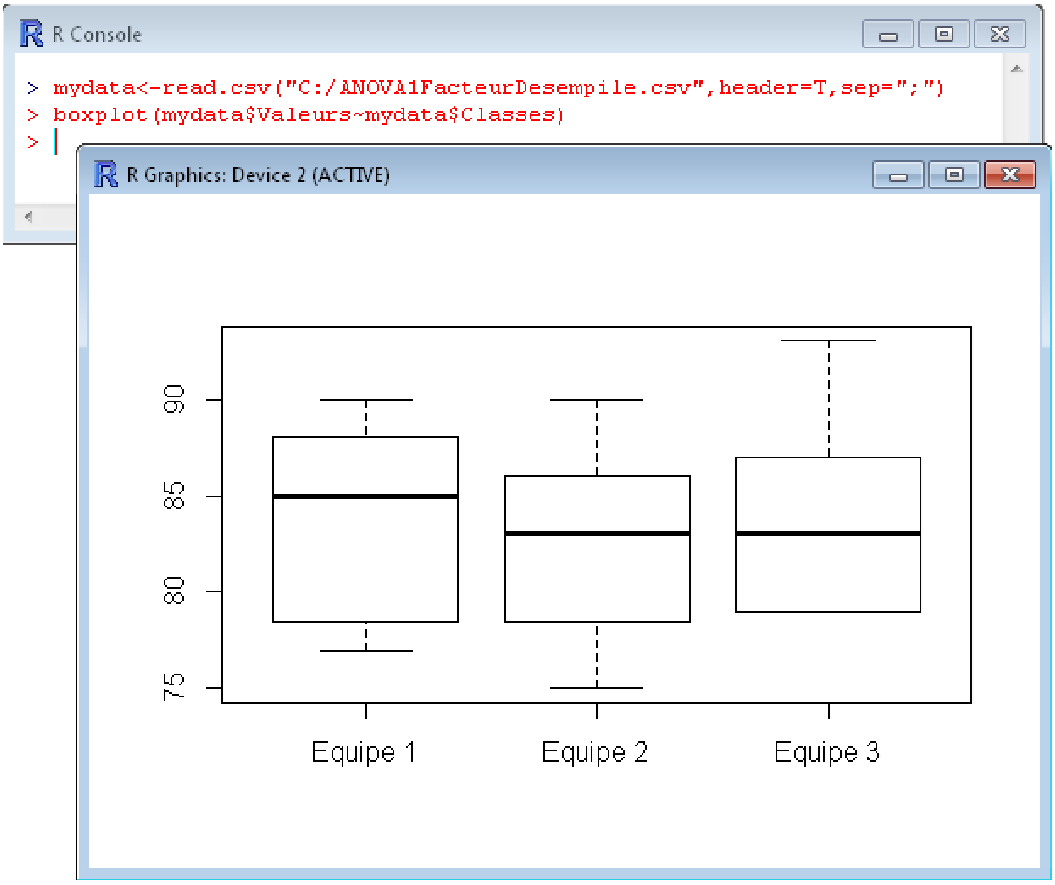
Imaginons une entreprise faisant les trois huit. Nous avons trois équipes qui travaillent sur une même machine. Nous souhaitons vérifier avec un seuil de confiance de 95% s'il y a une différence de productivité moyenne entre les trois équipes sur une semaine de travail.

Nous allons vérifier que nous retrouvons bien les valeurs calculées à la main dans le cours théorique ainsi que dans le cours Microsoft Excel et Minitab en utilisant les mêmes données:

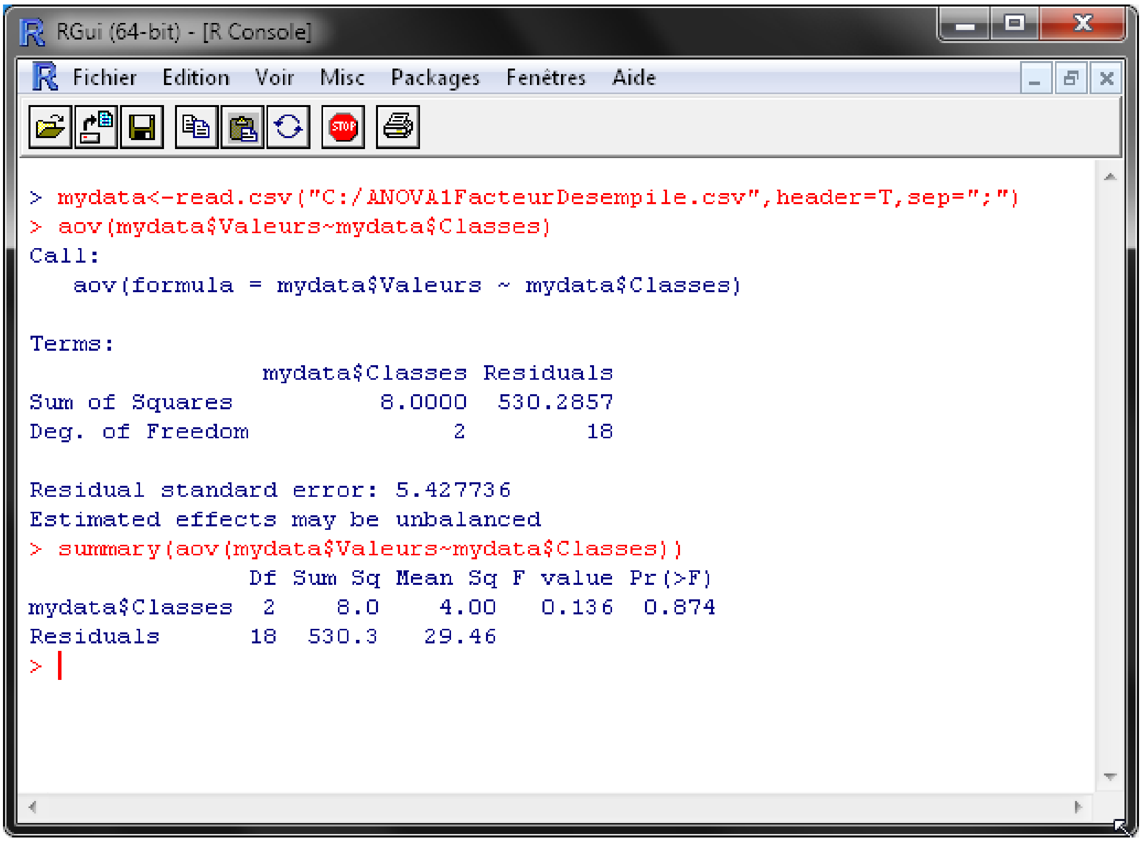



```
> mydata<-read.csv("C:/ANOVA1FacteurDesempile.csv",header=T,sep=";")
> mydata
    Classes Valeurs
1  Equipe 1      78
2  Equipe 1      88
3  Equipe 1      90
4  Equipe 1      77
5  Equipe 1      85
6  Equipe 1      88
7  Equipe 1      79
8  Equipe 2      77
9  Equipe 2      75
10 Equipe 2      80
11 Equipe 2      83
12 Equipe 2      87
13 Equipe 2      90
14 Equipe 2      85
15 Equipe 3      88
16 Equipe 3      86
17 Equipe 3      79
18 Equipe 3      93
19 Equipe 3      79
20 Equipe 3      83
21 Equipe 3      79
>
```

Déjà avant de sortir l'artillerie lourde une analyse qualitative peut parfois suffire:



Donc une analyse plus poussée se justifie bien! Nous utilisons alors la commande native **aov( )** de R:



Nous retrouvons donc bien les résultats de tout ce que nous avons calculé dans le cours théorique et Microsoft Excel et Minitab mais avec par contre moins d'informations.

La commande **plot.design( )** nous permet aussi d'afficher les effets moyens sous une autre forme que les box-plot:

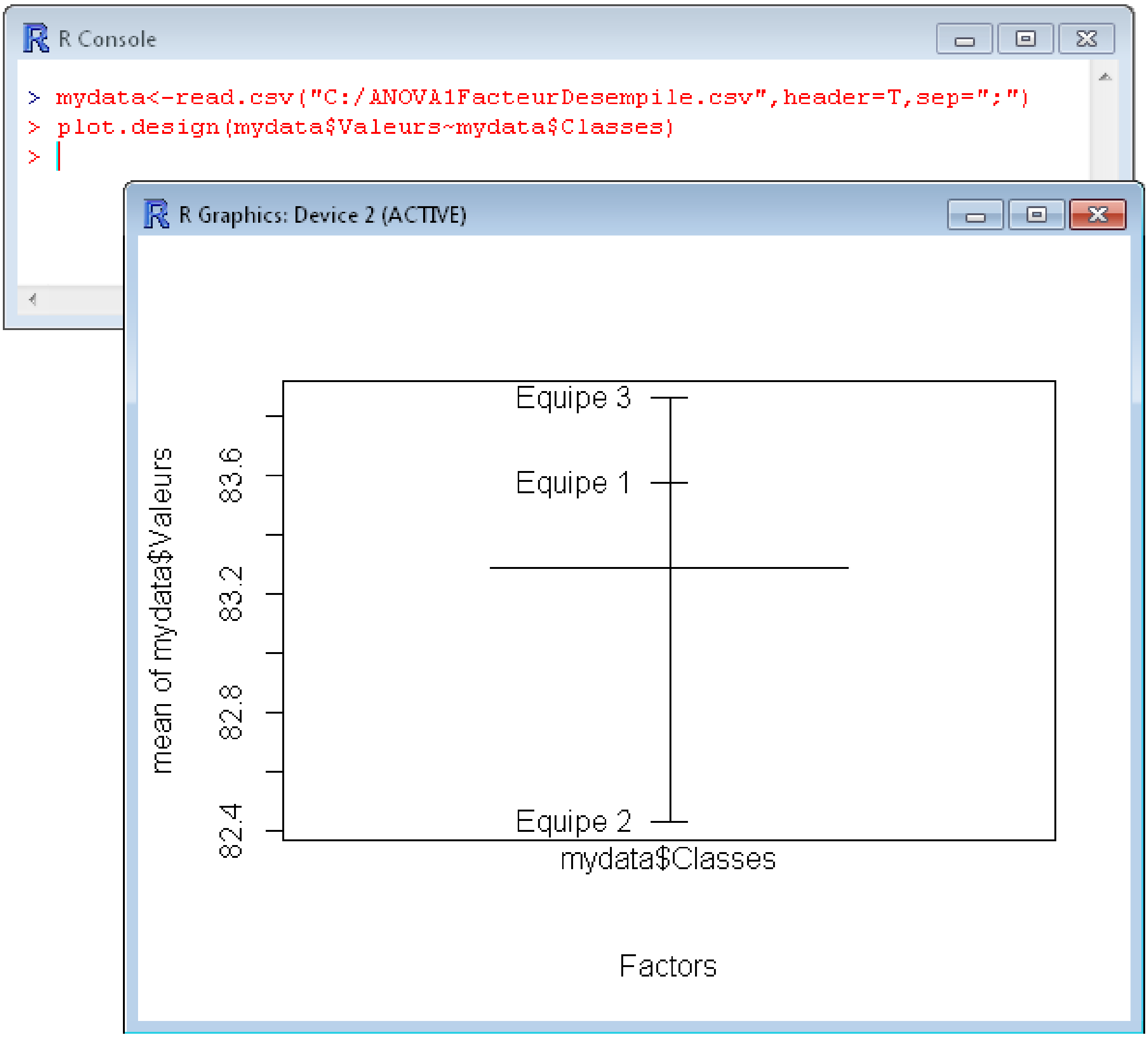
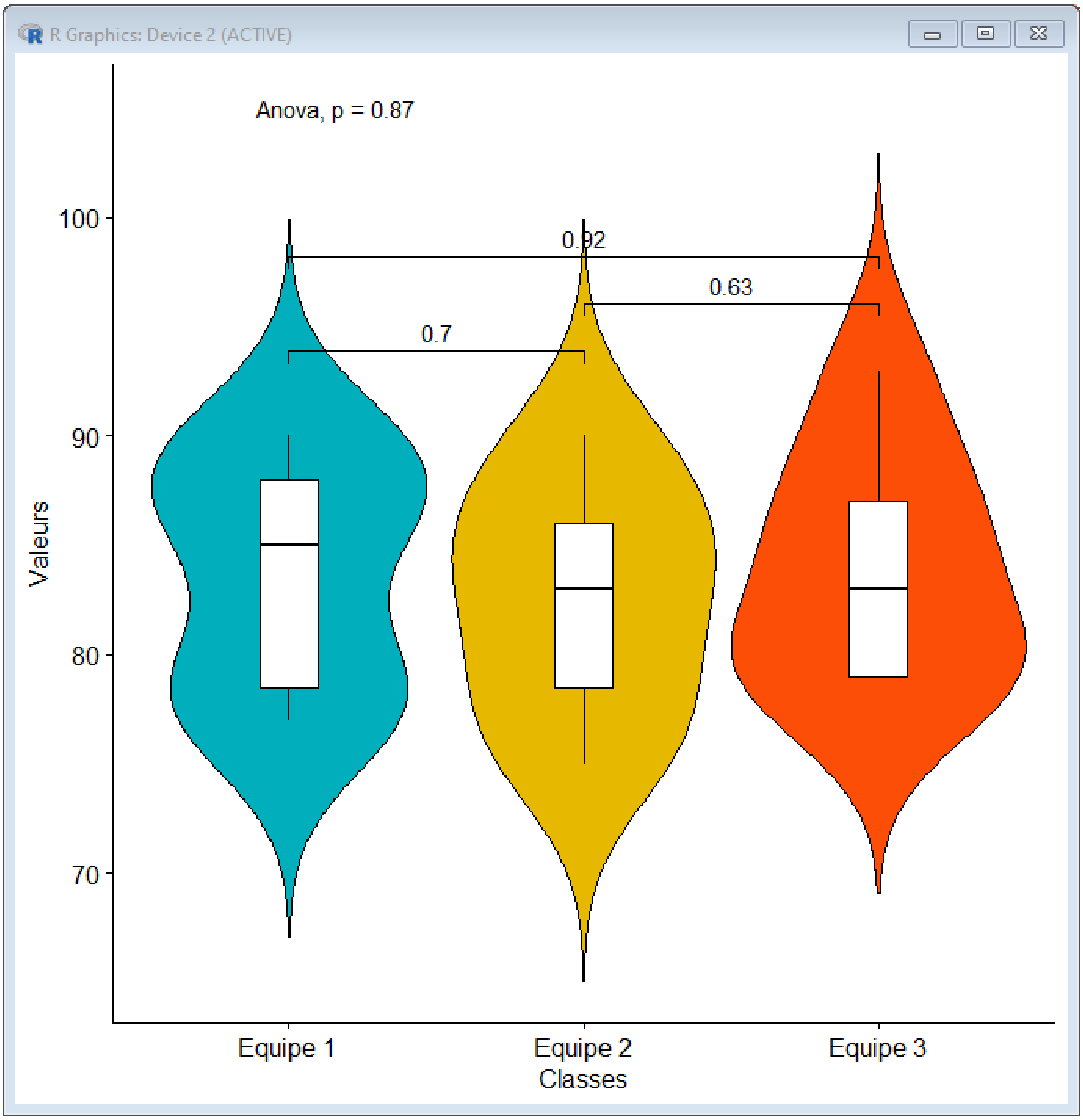


Figure 225 Graph ANOVA diagnostique des effets moyens

Sinon avec le package **ggpubr** et sa fonction **ggviolin( )** et sa couche **stat_compare_means( )** on peut arriver à un visuel très sympathique comme l'illustre l'exemple suivant:

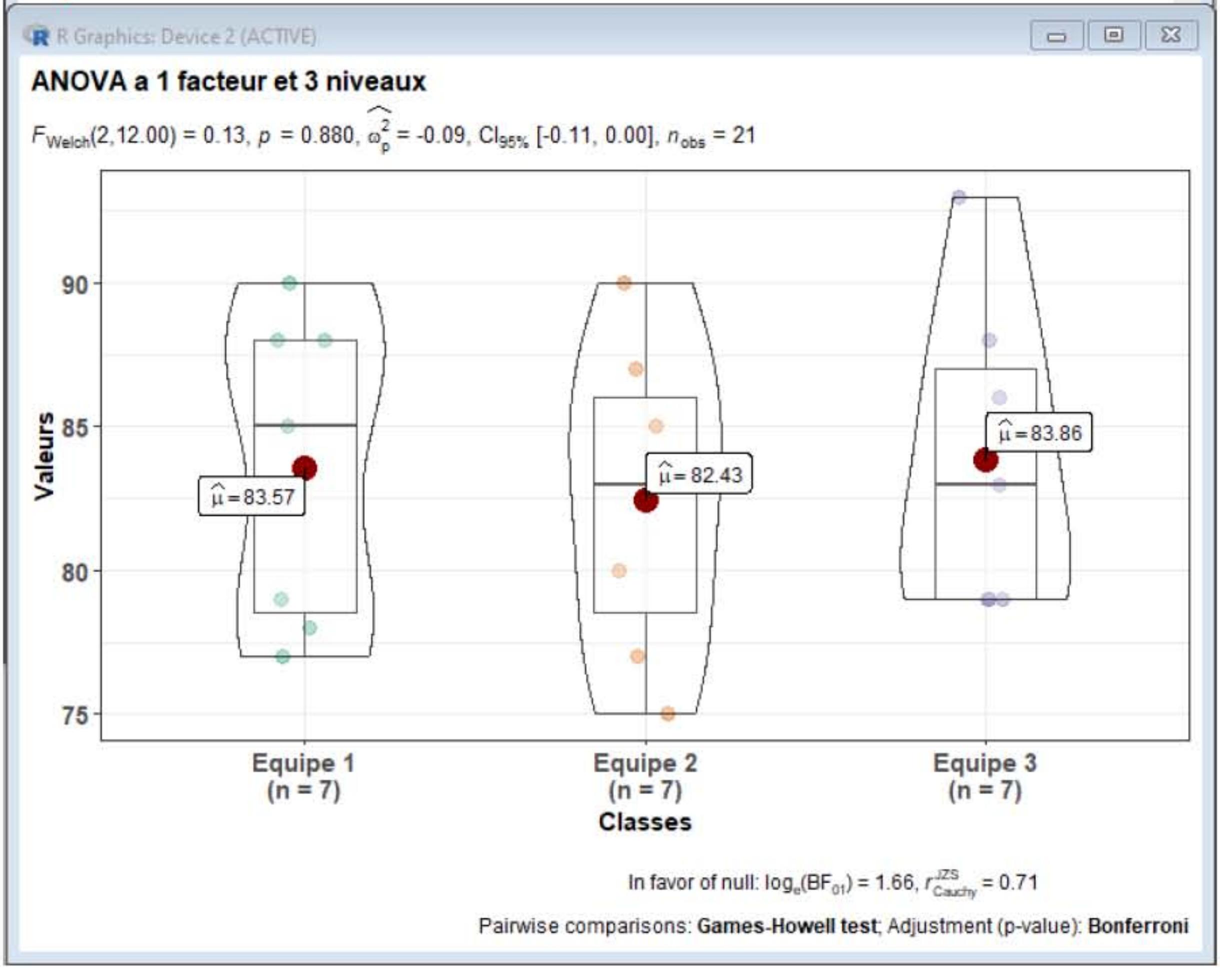
```
> library("ggpubr")
> my_comparisons<-list(c("Equipe 1","Equipe 2"),c("Equipe 2","Equipe 3"),c("Equipe 1","Equipe 3"))
> ggviolin(mydata,x="Classes",y="Valeurs",fill="Classes",palette=c("#00AFBB","#E7B800","#FC4E07"),
+ add="boxplot",add.params=list(fill="white"))+
+ #il y aussi comme method kruskall-wallis!
+ stat_compare_means(method = "anova", label.y=105)+
+ #ajout des p-values des comparaisons deux à deux
+ #l'ajustement ne fonctionne pas encore il bug pour l'instant (novembre 2018)!
+       stat_compare_means(comparisons = my_comparisons, method="t.test",p.adjust.method = "bonferroni"$
+ theme(legend.position="none")
> 
```

Ce qui donne:



Nous pouvons faire mieux avec le package **ggstatsplot** que nous recommandons fortement au lecteur de parcourir[24]:

```
> ggstatsplot::ggbetweenstats(
+   data = mydata,
+   x = Classes,
+   y = Valeurs,
+   pairwise.comparisons = TRUE, #affiche seulement si significatif
+   pairwise.annotation = "p.value",
+   p.adjust.method = "bonferroni",
+   title = "ANOVA a 1 facteur et 3 niveaux",
+   messages = FALSE
+ )
>
```

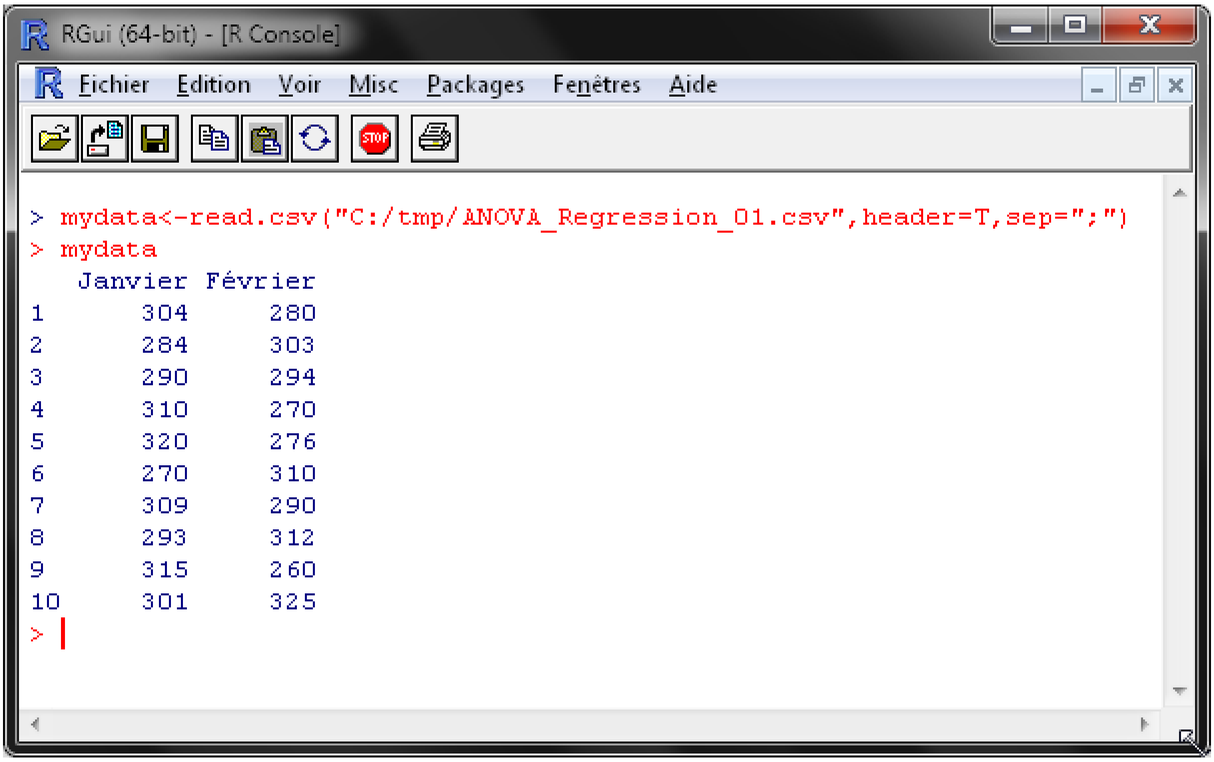


[24] https://github.com/IndrajeetPatil/ggstatsplot/blob/master/README.md

Exemple 2.: ANOVA à un facteur fixe (ANOVA-1 canonique) désempilé

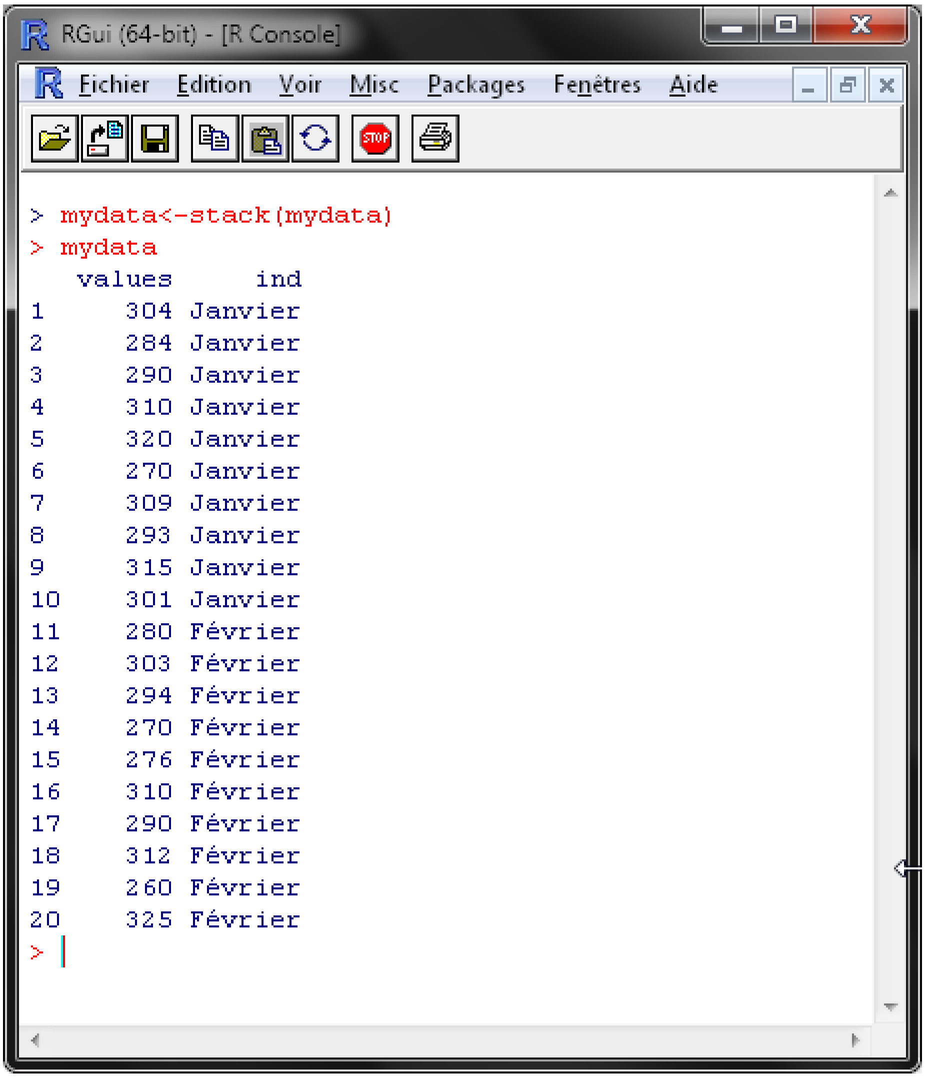
R 3.1.2

Pour le cas où les données sont désempilées, considérons l'exemple vu dans le cours théorique et que nous retrouverons lors de notre étude de la régression:

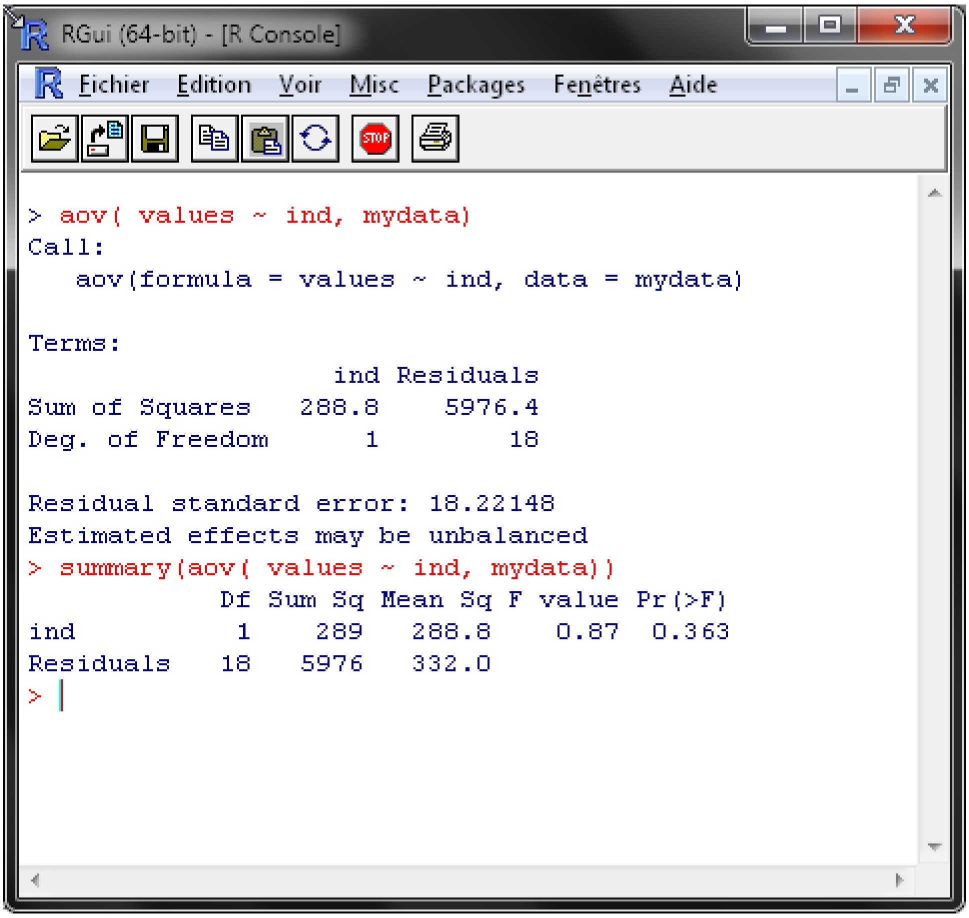



```
> mydata<-read.csv("C:/tmp/ANOVA_Regression_01.csv",header=T,sep=";")
> mydata
   Janvier Février
1      304     280
2      284     303
3      290     294
4      310     270
5      320     276
6      270     310
7      309     290
8      293     312
9      315     260
10     301     325
>
```

Nous devons donc restructurer les données pour R avec la fonction **stack( )** :



Ensuite, nous pouvons faire l'ANOVA à 1 facteur fixe:



Le résultat bien que peu explicite est cependant bien conforme aux calculs effectués dans le cours théorique.

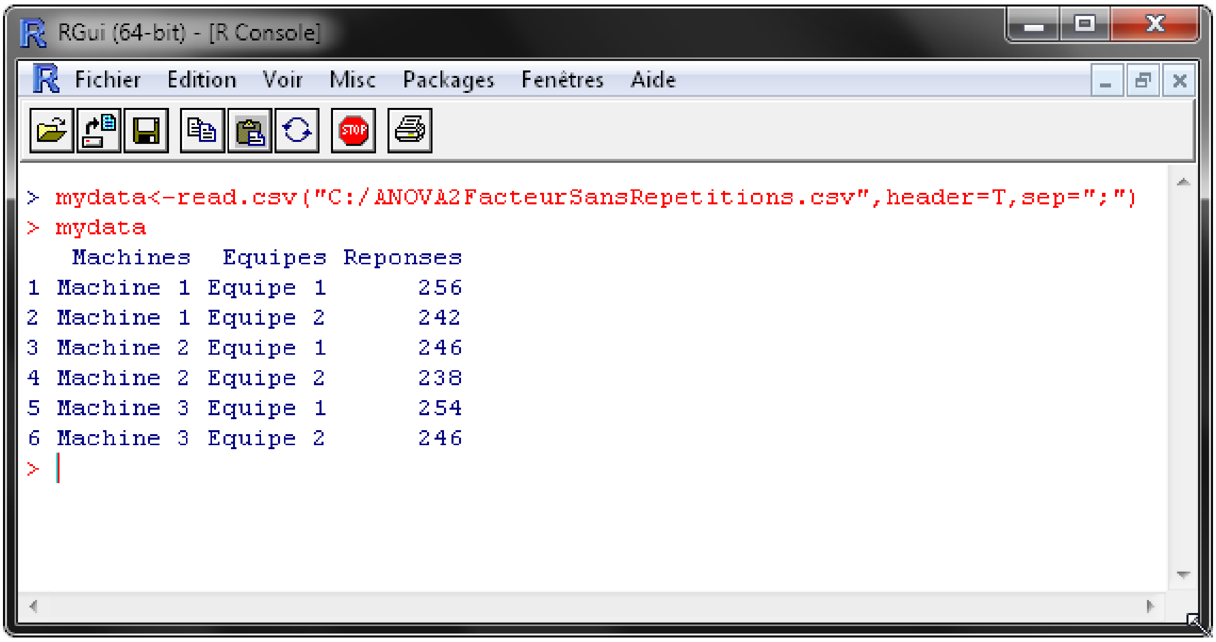
Exemple 3.: ANOVA à deux facteurs fixes (ANOVA-2 canonique) sans répétitions avec ou sans interactions

R 3.0.2

Imaginons une entreprise faisant les trois huit. Nous avons trois équipes qui travaillent sur une même machine. Nous souhaitons vérifier avec un seuil de confiance de 95% s'il y a une différence de productivité moyenne entre les trois équipes sur une semaine de travail (hypothèse que les moyennes sont égales).

Remarque: La variable est donc la Machine et sa modalité a trois niveaux!

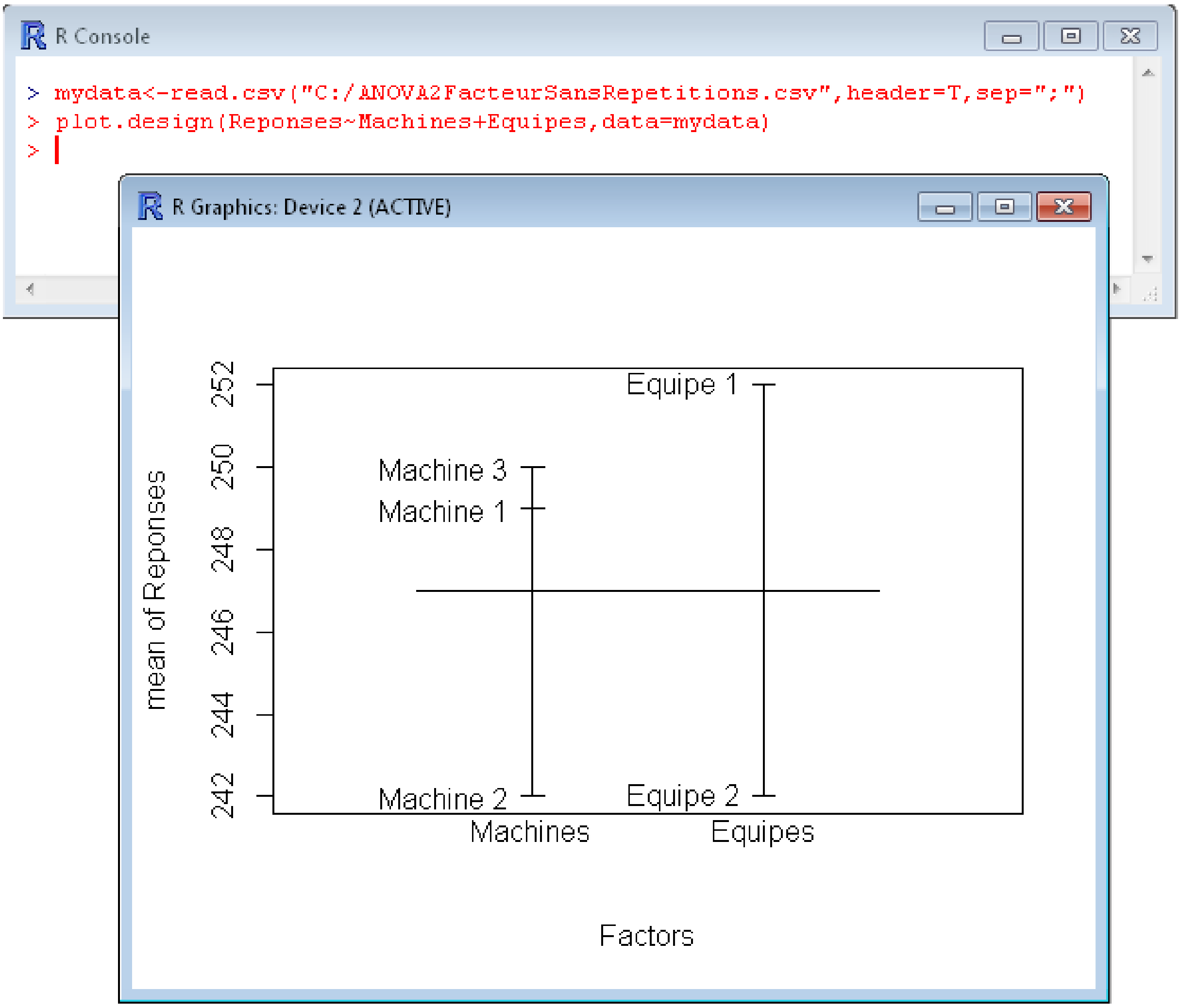
Nous allons donc ici appliquer la théorie vue en cours et vérifier si les résultats sont les mêmes que dans Minitab et MS Excel. Nous utiliserons évidemment les mêmes données:



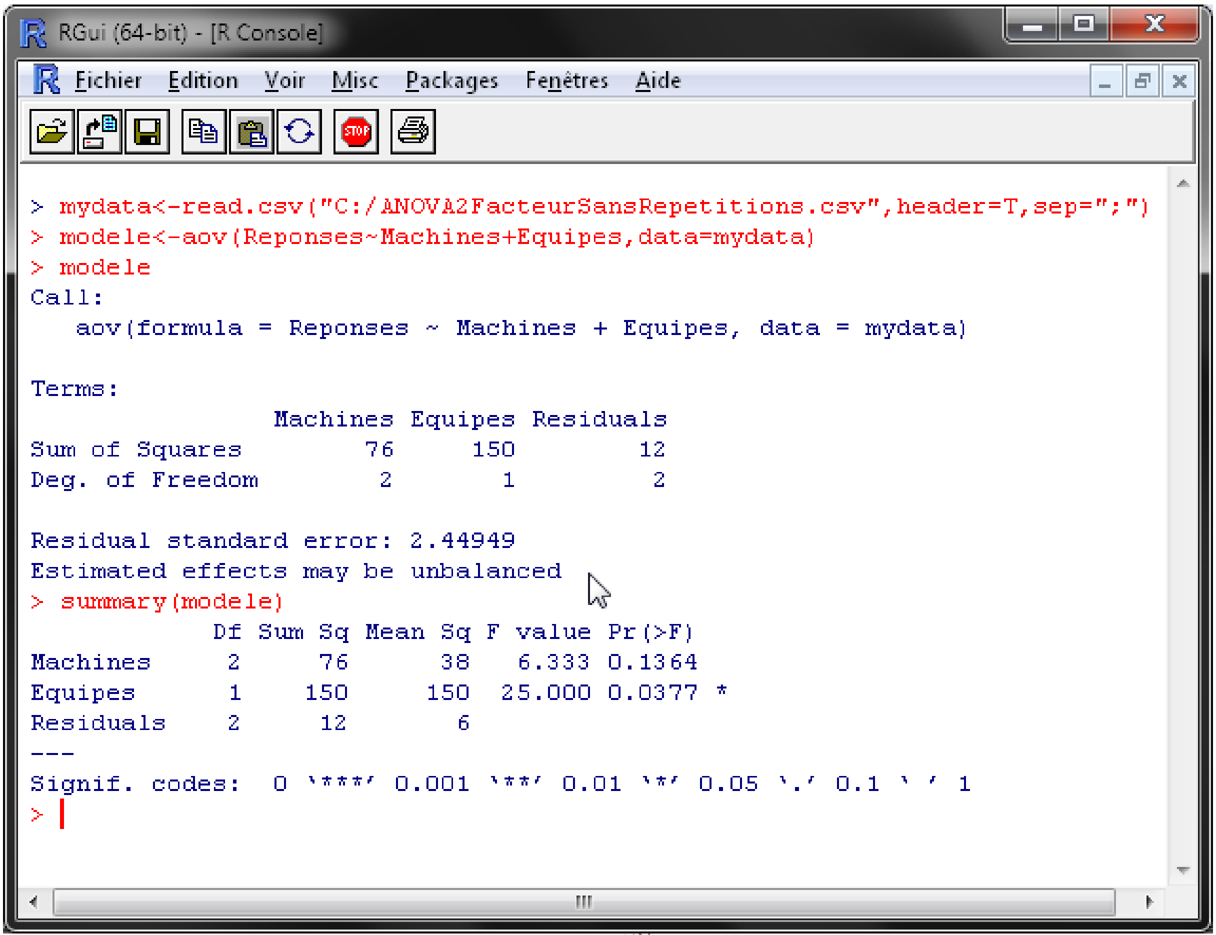
Nous pouvons d'abord faire une analyse qualitative avec le package **ggplot2**:



Ou encore une fois avec la commande native **plot.design( )** :

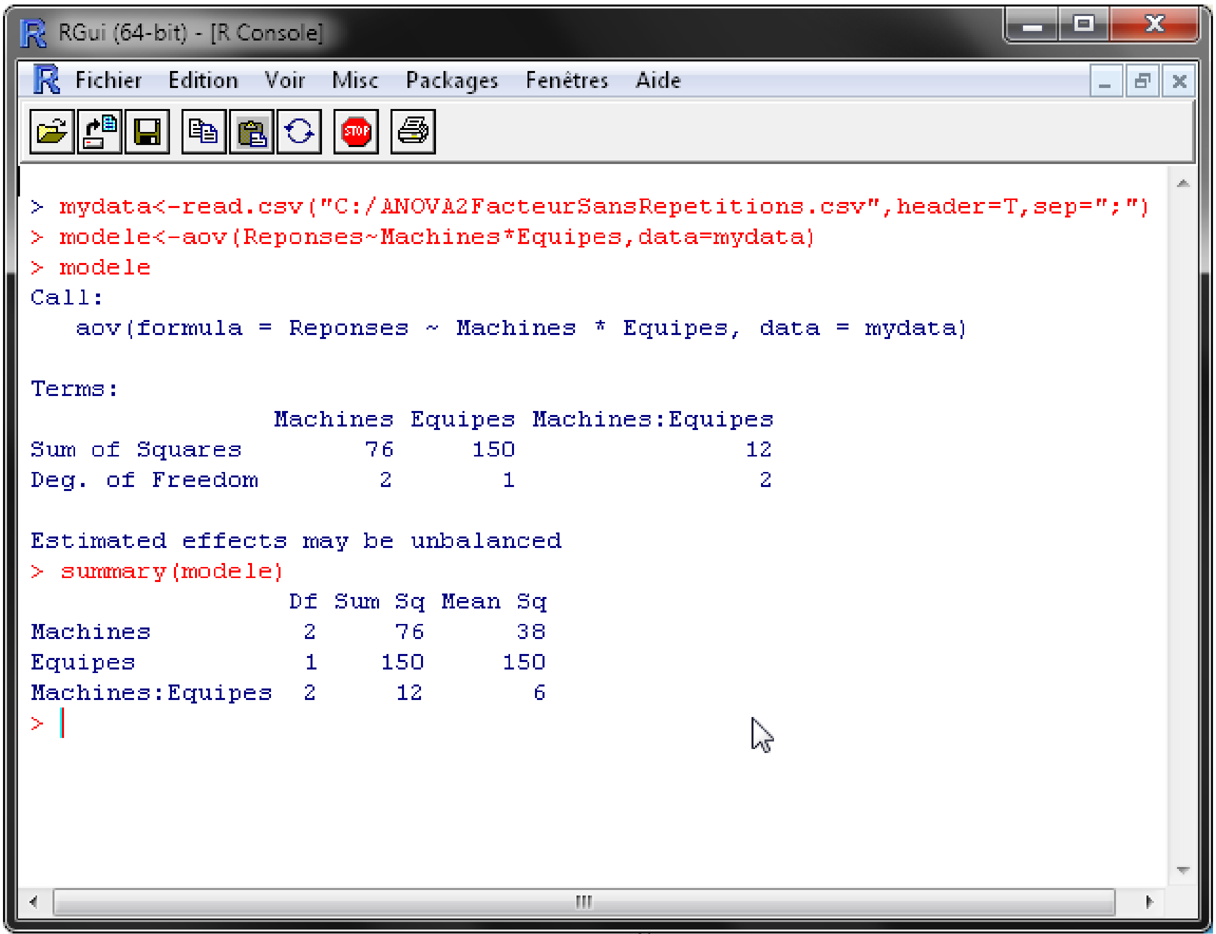


Nous avons alors d'abord en utilisant la commande native **aov( )** native à R pour avoir le modèle sans interactions:



Nous retrouvons donc les mêmes valeurs qu'avec Minitab et MS Excel.

Pour le modèle avec interactions (évidemment impossible dans le cas présent pour un logiciel de communiquer les statistiques puisqu'il n'y a pas de répétitions):



Avec la commande native **interaction.plot( )** nous pouvons avoir les diagrammes d'interactions:

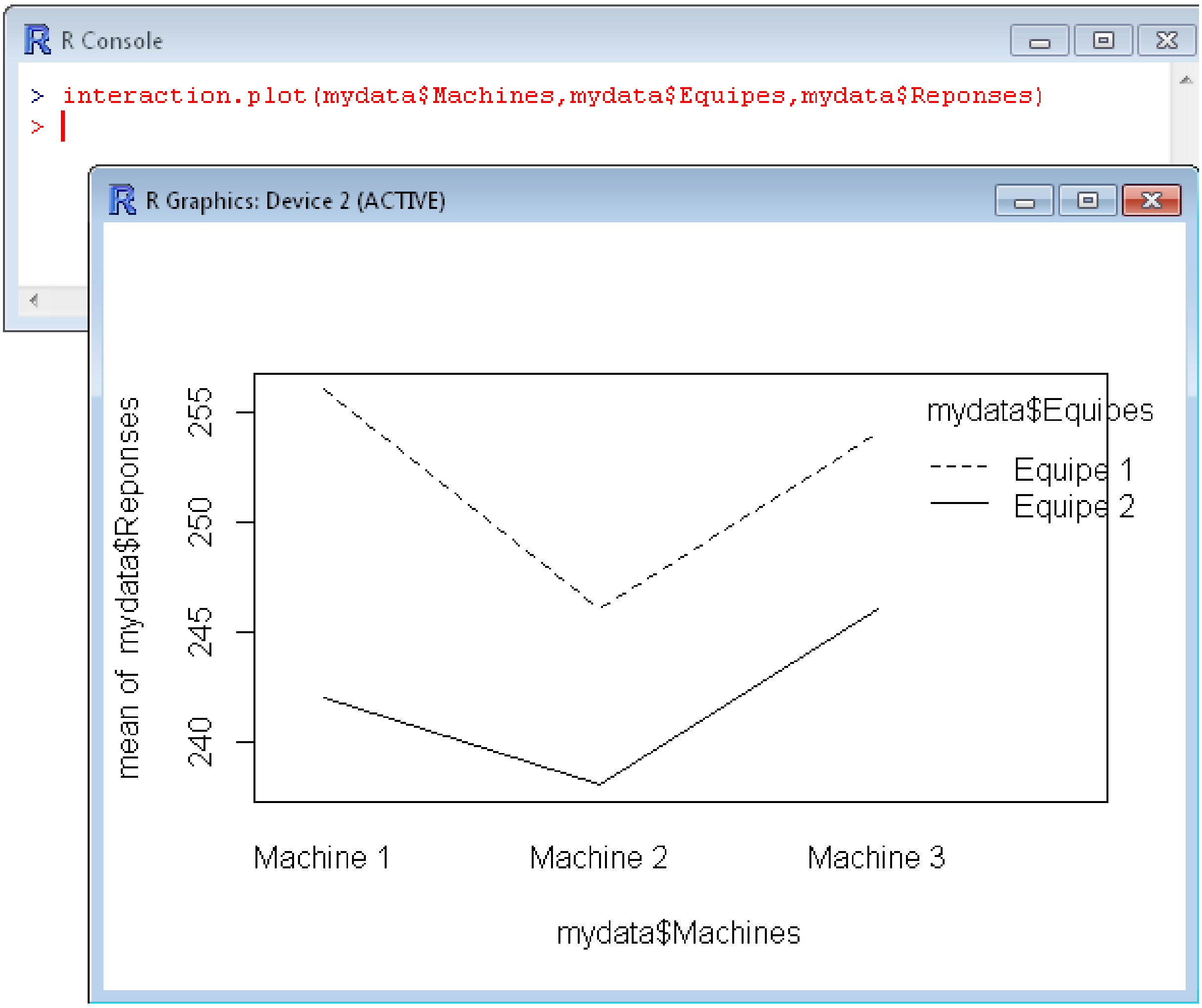
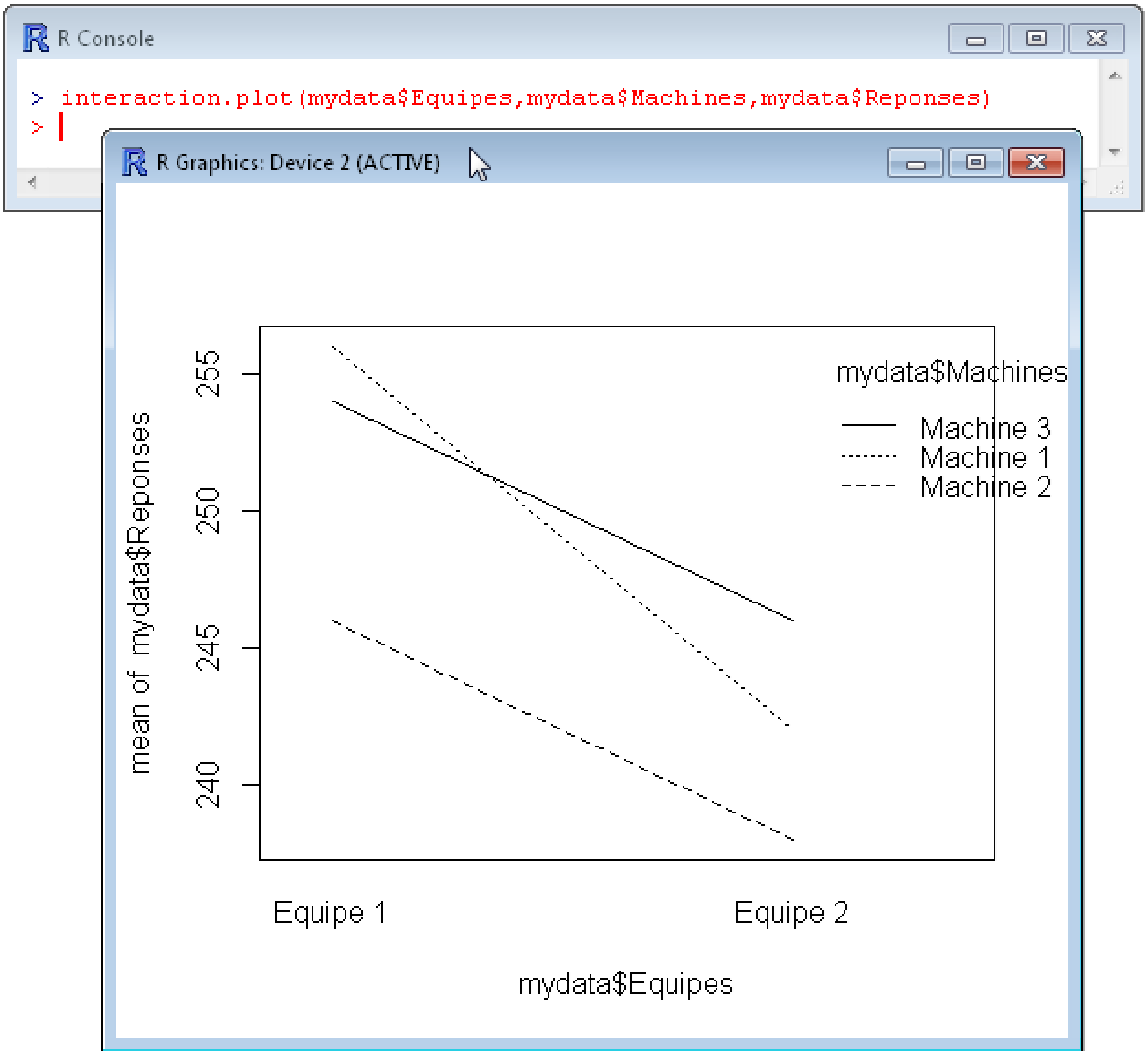
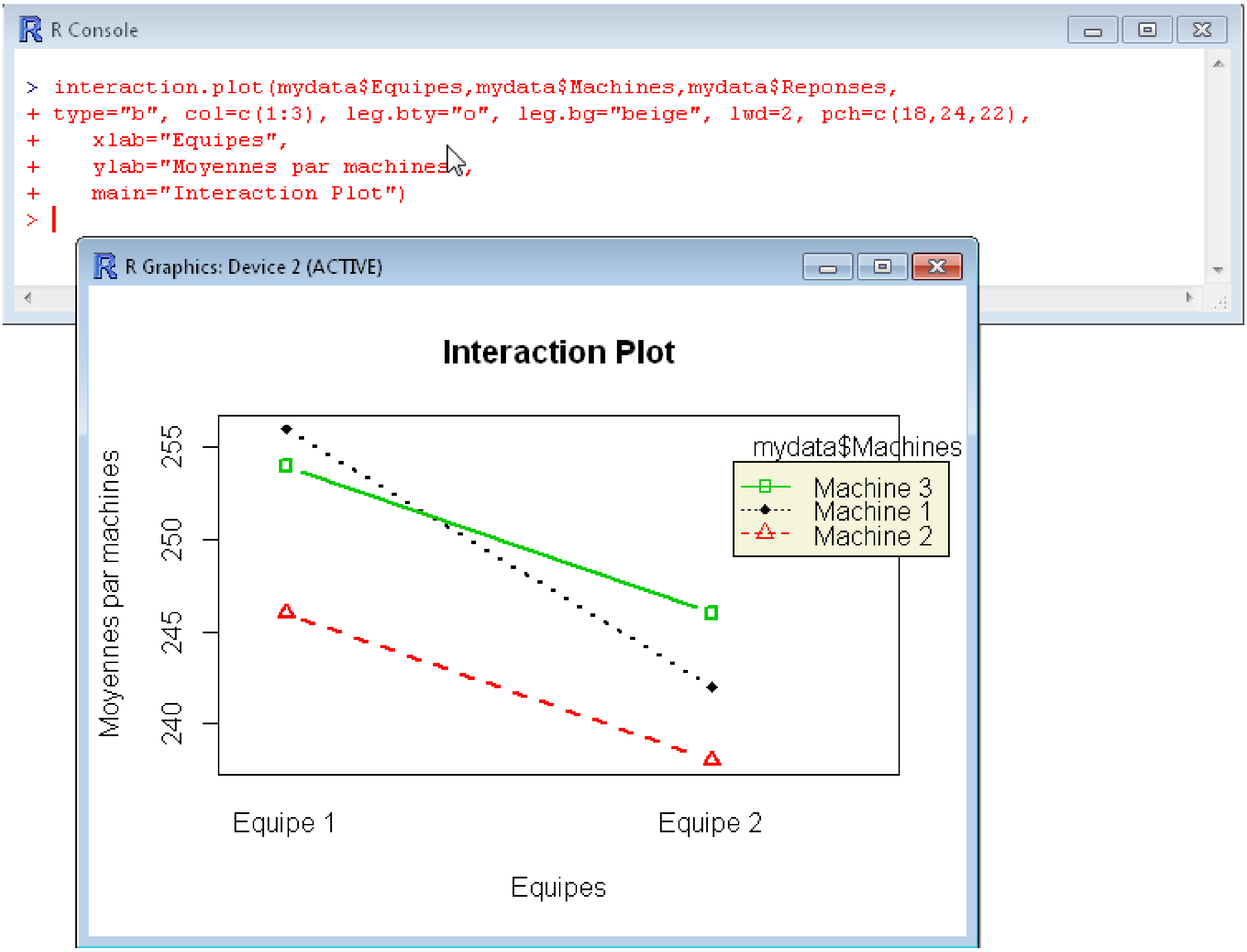


Figure 226 Graph ANOVA graphique des interactions

et respectivement:



On peut faire plus élégant:



ou pour revenir avec aux machines avec un visuel plus proche de celui de Minitab mais aussi meilleur que ce dernier:

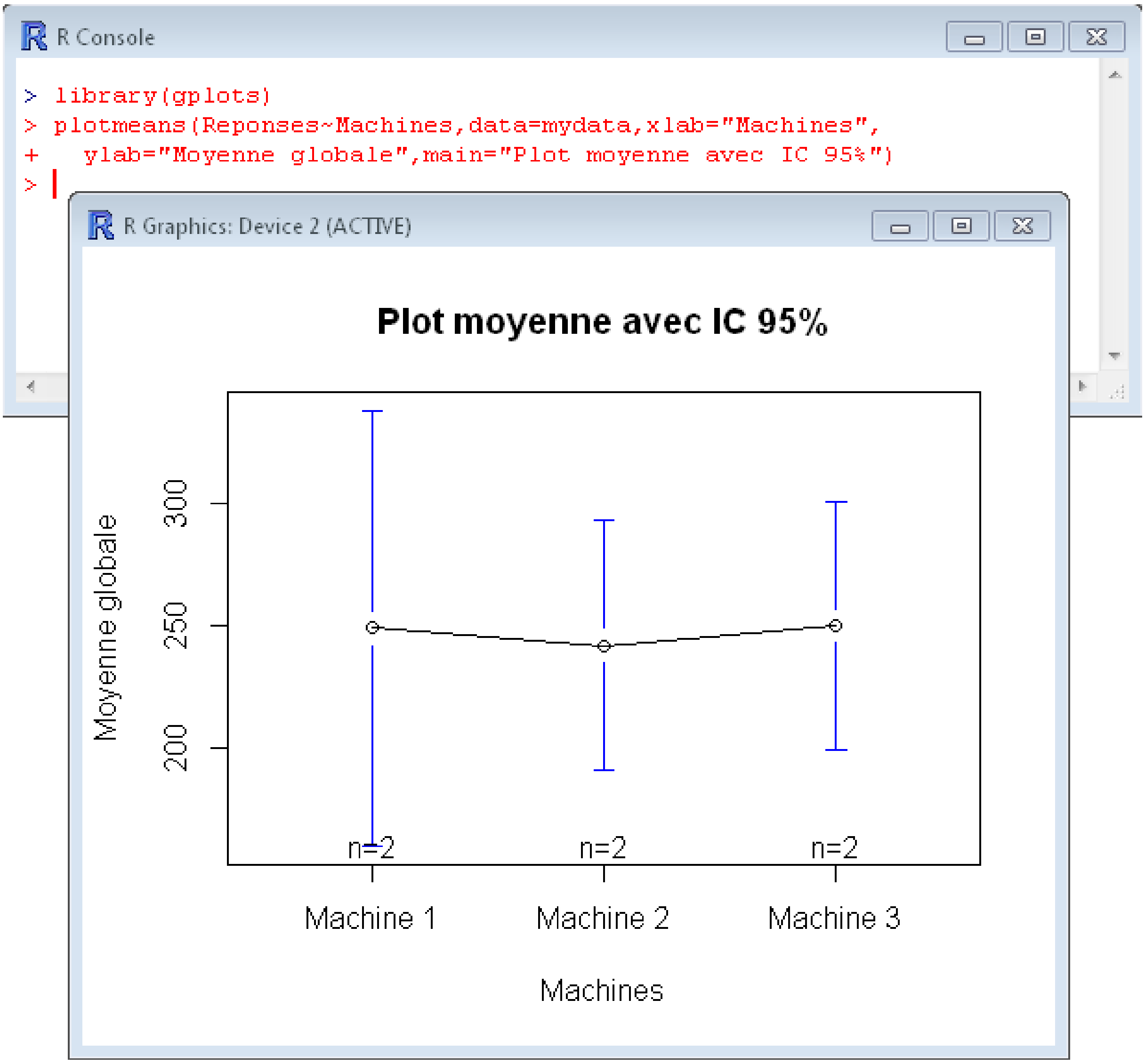


Figure 227 Graph ANOVA effets moyens avec intervalles de confiance

Exemple 4.: ANOVA à deux facteurs fixes (ANOVA-2 canonique) avec répétitions (réplications) avec ou sans interactions

R 3.0.2

Imaginons une entreprise faisant les trois huit. Nous avons trois équipes qui travaillent sur une même machine. Nous souhaitons vérifier avec un seuil de confiance de 95% s'il y a une différence de productivité moyenne entre les trois équipes sur une semaine de travail (hypothèse que les moyennes sont égales).

Remarque: La variable est donc la Machine et sa modalité a trois niveaux!

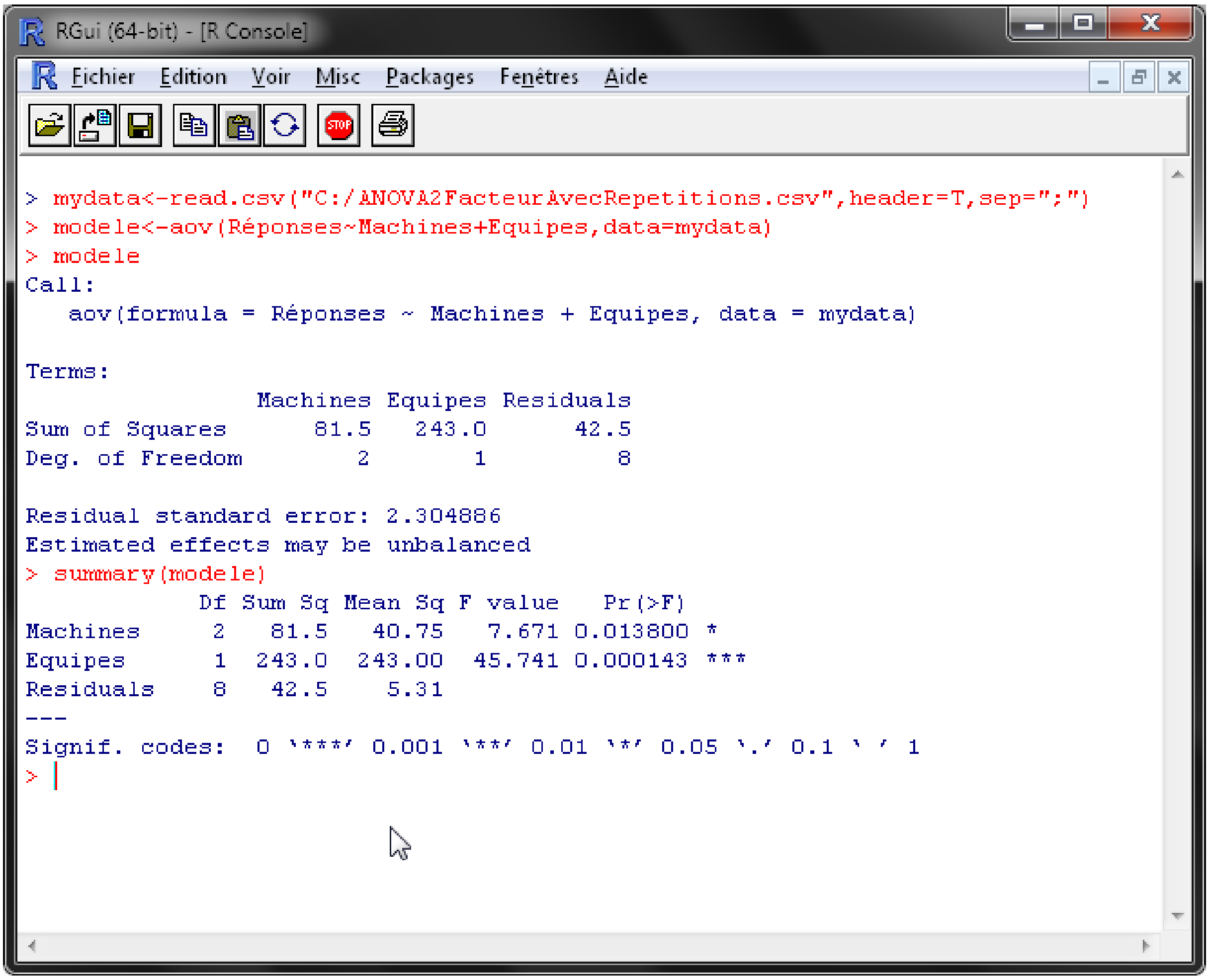
Nous allons donc ici appliquer la théorie vue en cours et vérifier si les résultats sont les mêmes que dans Minitab et MS Excel. Nous utiliserons évidemment les mêmes données et en même temps, utiliser la fonction **replications( )** pour avoir un petit résumé sur la manière dont l'expérience a été menée:

```
> mydata<-read.csv("C:/tmp/ANOVA2FacteursAvecRepetitions.csv",header=T,sep=";")
> mydata
    Equipes  Machines Réponses
1  Equipe 1 Machine 1      256
2  Equipe 1 Machine 1      251
3  Equipe 1 Machine 2      246
4  Equipe 1 Machine 2      248
5  Equipe 1 Machine 3      254
6  Equipe 1 Machine 3      251
7  Equipe 2 Machine 1      242
8  Equipe 2 Machine 1      240
9  Equipe 2 Machine 2      238
10 Equipe 2 Machine 2      240
11 Equipe 2 Machine 3      246
12 Equipe 2 Machine 3      246
> replications(Réponses~Machines+Equipes,data=mydata)
Machines  Equipes
       4        6
> replications(Réponses~Machines*Equipes,data=mydata)
        Machines          Equipes Machines:Equipes
               4                6                2
> |
```

Nous pouvons d'abord faire une analyse qualitative avec le package **ggplot2**:

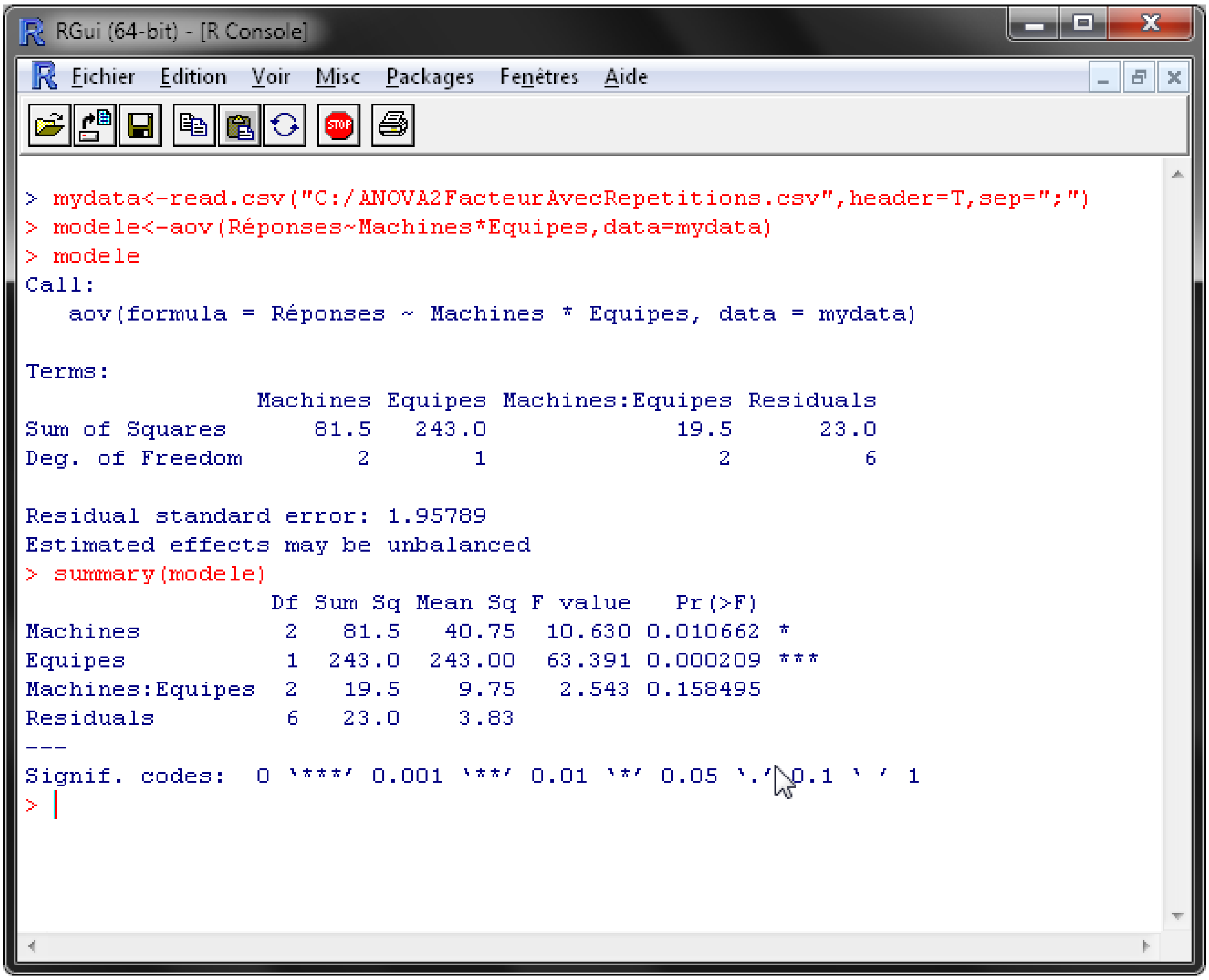


Nous avons alors d'abord en utilisant la commande native **aov( )** native à R pour avoir le modèle sans interactions:

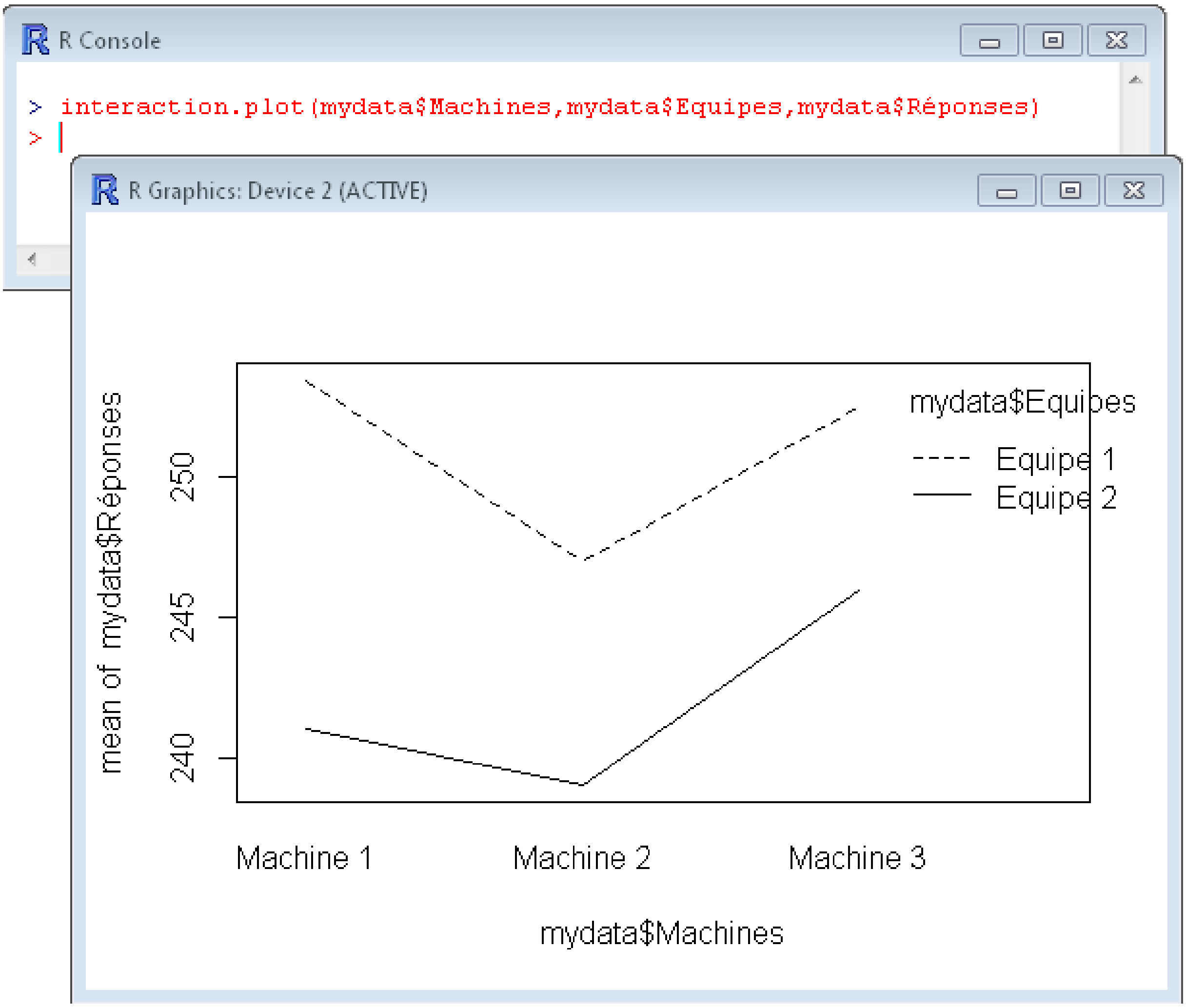


Nous retrouvons donc les mêmes valeurs qu'avec Minitab et MS Excel.

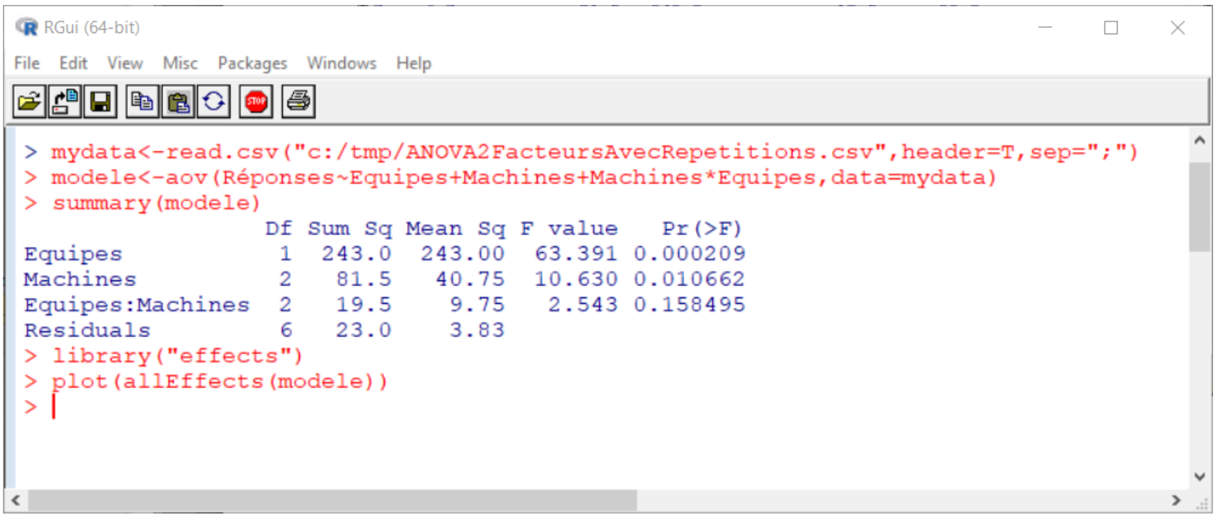
Pour le modèle avec interactions:



Avec la commande native **interaction.plot( )** nous pouvons avoir les diagrammes d'interactions:



Nous pouvons également combiner:



et afficher toutes les interactions de façon moderne avec le package **effects**:

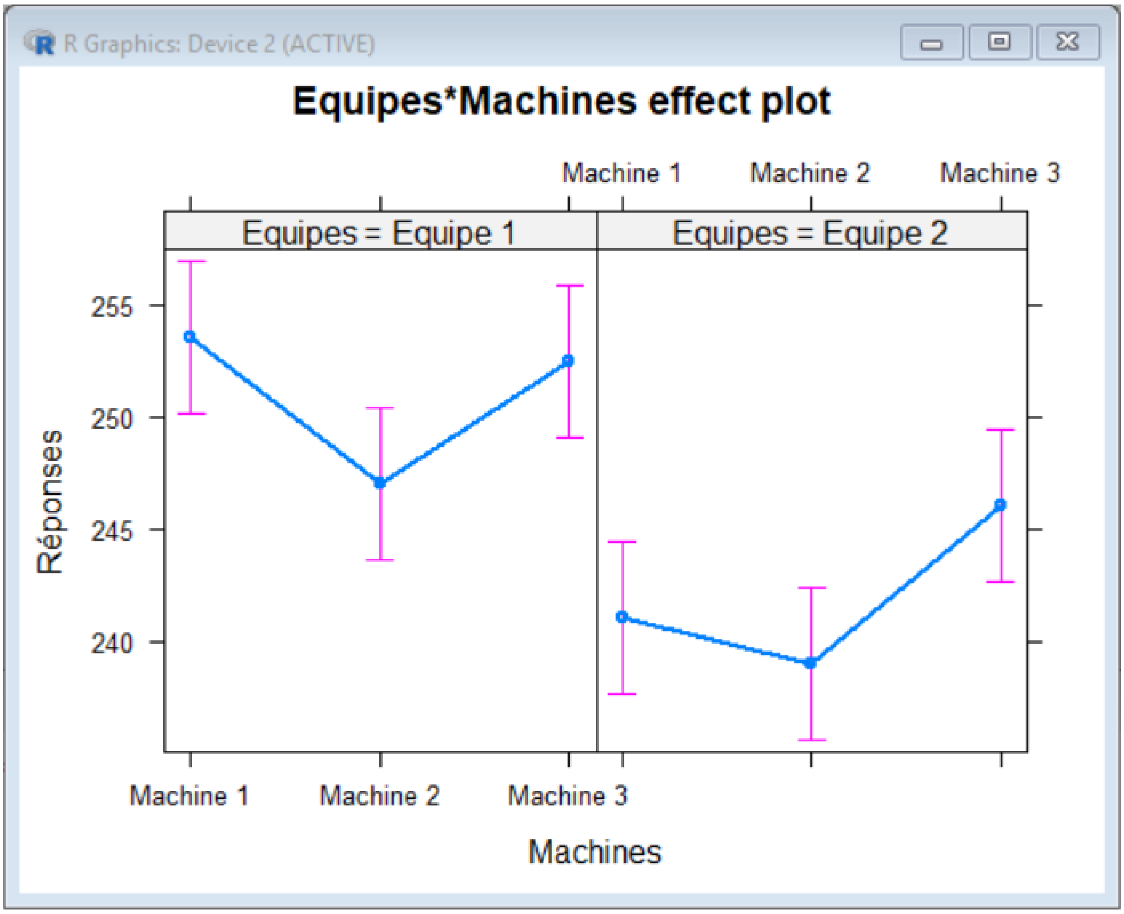


Figure 228 Graph ANOVA effets moyennes à lattices avec intervalles

Exemple 5.: ANOVA à deux facteurs fixes (ANOVA-2 canonique) avec répétitions (réplications) de type I, II et III

R 3.5.1

D'abord, vérifions bien que comme indiqué dans le cours théorique, si l'ANOVA est balancée, nous retrouvons bien les mêmes valeurs statistiques. Pour cela nous partons de:

```
> mydata<-read.csv("c:/tmp/ANOVA2FacteursAvecRepetitions.csv",header=T,sep=";")
> mydata
    Equipes  Machines Réponses
1  Equipe 1 Machine 1      256
2  Equipe 1 Machine 1      251
3  Equipe 1 Machine 2      246
4  Equipe 1 Machine 2      248
5  Equipe 1 Machine 3      254
6  Equipe 1 Machine 3      251
7  Equipe 2 Machine 1      242
8  Equipe 2 Machine 1      240
9  Equipe 2 Machine 2      238
10 Equipe 2 Machine 2      240
11 Equipe 2 Machine 3      246
12 Equipe 2 Machine 3      246
> 
```

et nous utilisons la commande **Anova( )** du package **car**:

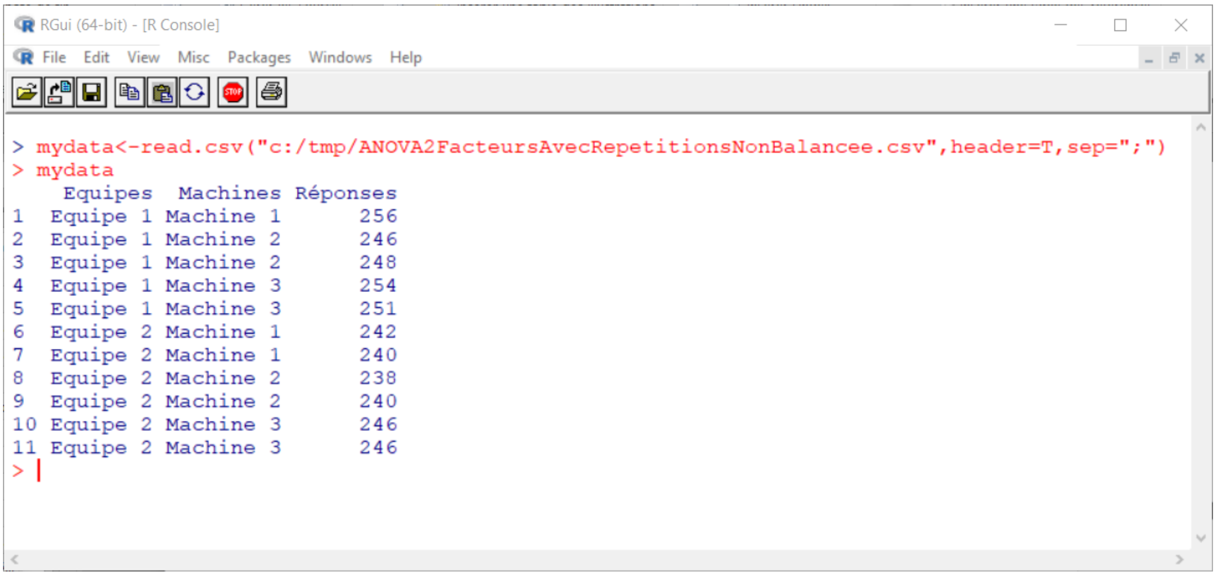
```
> #ANOVA balancée de type I
> anova(lm(Réponses~ Machines * Equipes, data=mydata))
Analysis of Variance Table

Response: Réponses
                 Df Sum Sq Mean Sq F value    Pr(>F)
Machines          2   81.5  40.750 10.6304 0.0106619
Equipes           1  243.0 243.000 63.3913 0.0002089
Machines:Equipes  2   19.5   9.750  2.5435 0.1584954
Residuals         6   23.0   3.833
> #ANOVA balancée de type II
> Anova(lm(Réponses~ Machines * Equipes, data=mydata),Type=2)
Anova Table (Type II tests)

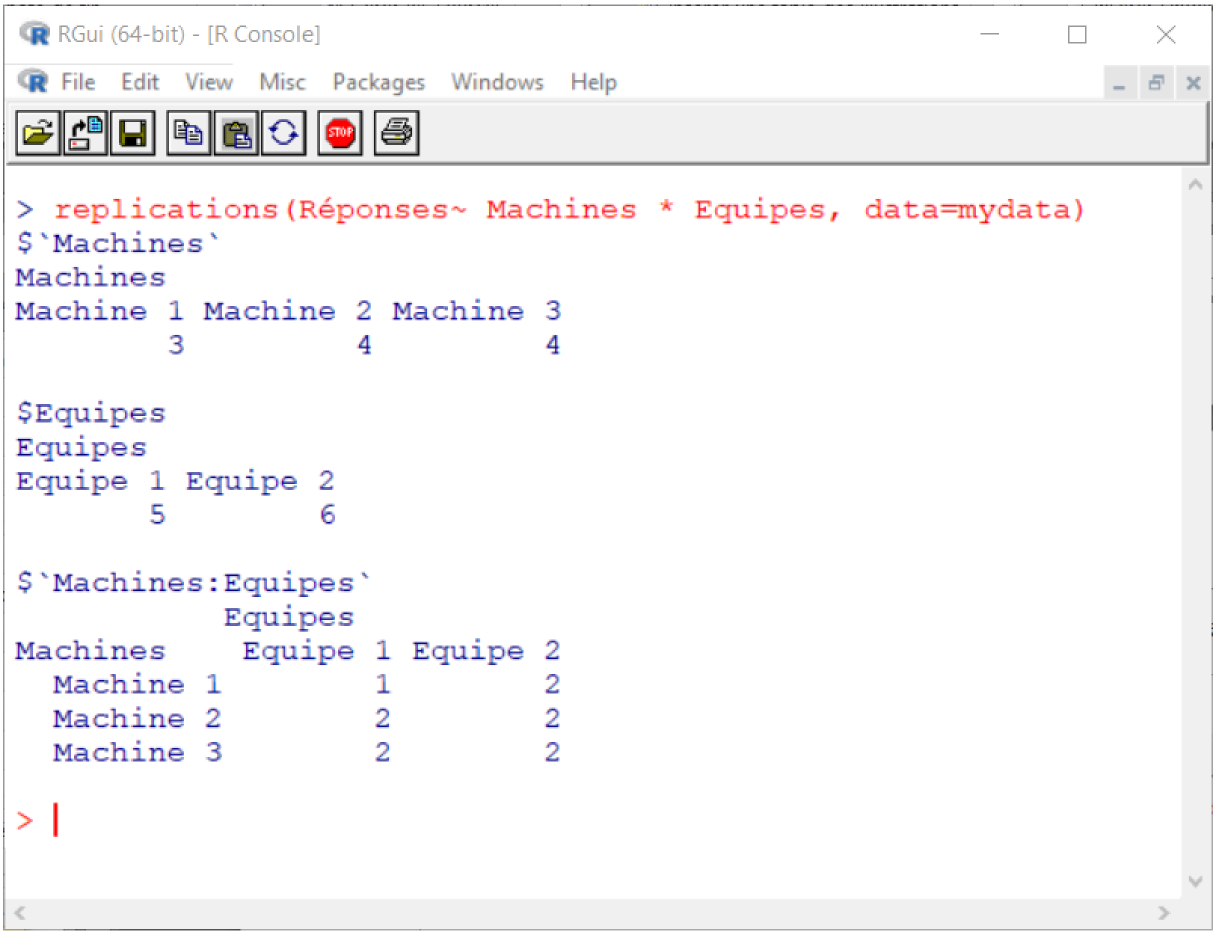
Response: Réponses
                 Sum Sq Df F value    Pr(>F)
Machines           81.5  2 10.6304 0.0106619
Equipes           243.0  1 63.3913 0.0002089
Machines:Equipes   19.5  2  2.5435 0.1584954
Residuals          23.0  6
> #ANOVA balancée de type III
> Anova(lm(Réponses~ Machines * Equipes, data=mydata),Type=3)
Anova Table (Type II tests)

Response: Réponses
                 Sum Sq Df F value    Pr(>F)
Machines           81.5  2 10.6304 0.0106619
Equipes           243.0  1 63.3913 0.0002089
Machines:Equipes   19.5  2  2.5435 0.1584954
Residuals          23.0  6
> 
```

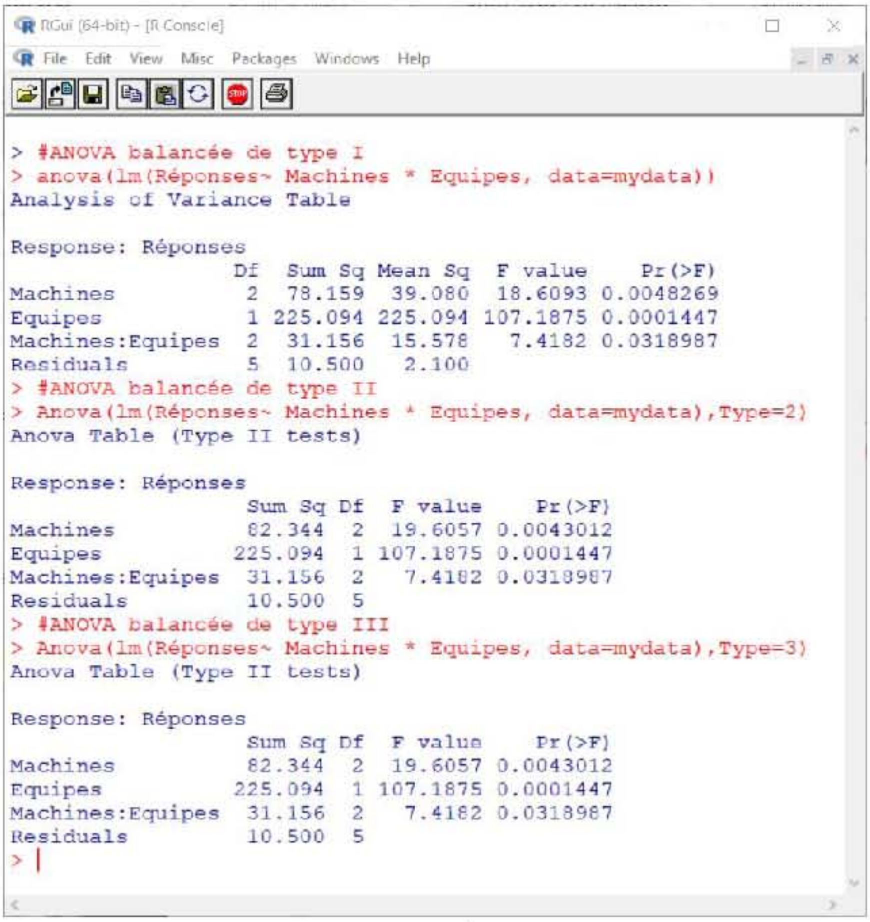
Maintenant, déséquilibrons le plan en supprimant la troisième ligne. Il reste alors:



Nous pouvons vérifier le déséquilibre toujours avec la fonction **replications( )** :



Et ensuite allons-y pour les trois ANOVA:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #ANOVA balancée de type I
> anova(lm(Réponses~ Machines * Equipes, data=mydata))
Analysis of Variance Table

Response: Réponses
                 Df  Sum Sq Mean Sq  F value    Pr(>F)
Machines          2  78.159  39.080  18.6093 0.0048269
Equipes           1 225.094 225.094 107.1875 0.0001447
Machines:Equipes  2  31.156  15.578   7.4182 0.0318987
Residuals         5  10.500   2.100
> #ANOVA balancée de type II
> Anova(lm(Réponses~ Machines * Equipes, data=mydata),Type=2)
Anova Table (Type II tests)

Response: Réponses
                  Sum Sq Df  F value    Pr(>F)
Machines          82.344  2  19.6057 0.0043012
Equipes          225.094  1 107.1875 0.0001447
Machines:Equipes  31.156  2   7.4182 0.0318987
Residuals         10.500  5
> #ANOVA balancée de type III
> Anova(lm(Réponses~ Machines * Equipes, data=mydata),Type=3)
Anova Table (Type II tests)

Response: Réponses
                  Sum Sq Df  F value    Pr(>F)
Machines          82.344  2  19.6057 0.0043012
Equipes          225.094  1 107.1875 0.0001447
Machines:Equipes  31.156  2   7.4182 0.0318987
Residuals         10.500  5
> 
```

Exemple 6.: ANOVA à deux facteurs fixes (ANOVA-2 canonique) avec répétitions (réplications) de type I non-balancée avec moyenne par moindres carrées

R 3.6.0

D'abord la partie classique préparatoire:

```
> mydata<-read.csv("c:/tmp/ANOVA2FacteursAvecRepetitionsNonBalancee.csv",header=T,sep=";")
> mydata
    Equipes  Machines Réponses
1  Equipe 1 Machine 1      256
2  Equipe 1 Machine 2      246
3  Equipe 1 Machine 2      248
4  Equipe 1 Machine 3      254
5  Equipe 1 Machine 3      251
6  Equipe 2 Machine 1      242
7  Equipe 2 Machine 1      240
8  Equipe 2 Machine 2      238
9  Equipe 2 Machine 2      240
10 Equipe 2 Machine 3      246
11 Equipe 2 Machine 3      246
>  (model<-lm(Réponses~Machines*Equipes,data=mydata))

Call:
lm(formula = Réponses ~ Machines * Equipes, data = mydata)

Coefficients:
                      (Intercept)                  MachinesMachine 2
                            256.0                               -9.0
                MachinesMachine 3                     EquipesEquipe 2
                             -3.5                              -15.0
MachinesMachine 2:EquipesEquipe 2  MachinesMachine 3:EquipesEquipe 2
                              7.0                                8.5

> anova(model)
Analysis of Variance Table

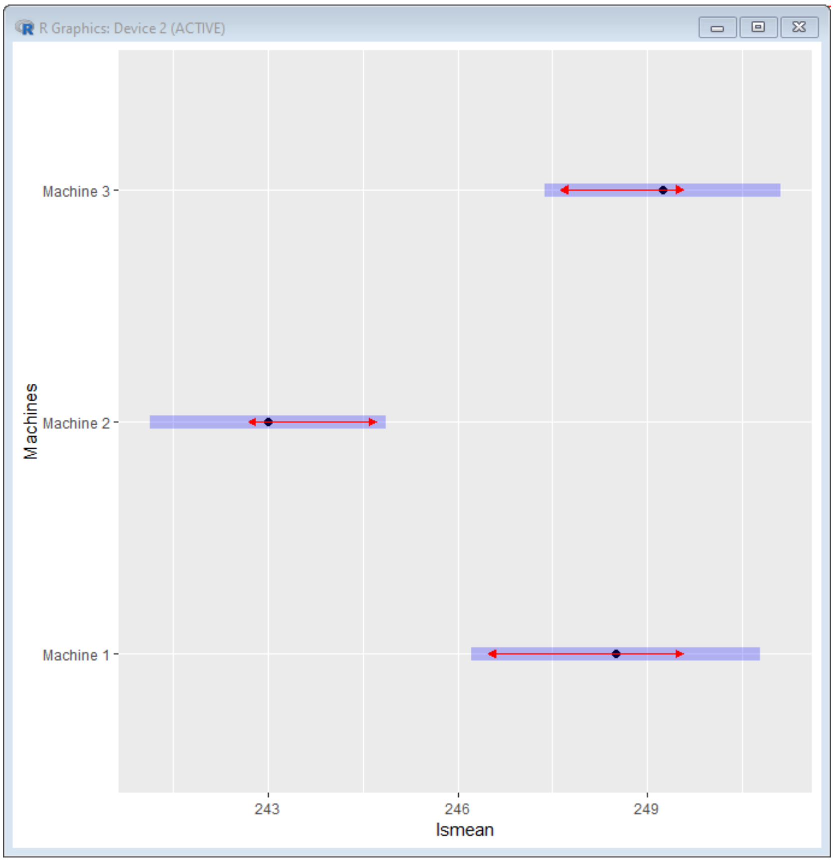
Response: Réponses
                 Df  Sum Sq Mean Sq  F value    Pr(>F)
Machines          2  78.159  39.080  18.6093 0.0048269
Equipes           1 225.094 225.094 107.1875 0.0001447
Machines:Equipes  2  31.156  15.578   7.4182 0.0318987
Residuals         5  10.500   2.100
>
```

Voyons maintenant basiquement ce que permettent de faire la fonction **LSmeans( )** du package **doBy** et en comparaison, la fonction **lsmeans( )** du package **lsmeans**:

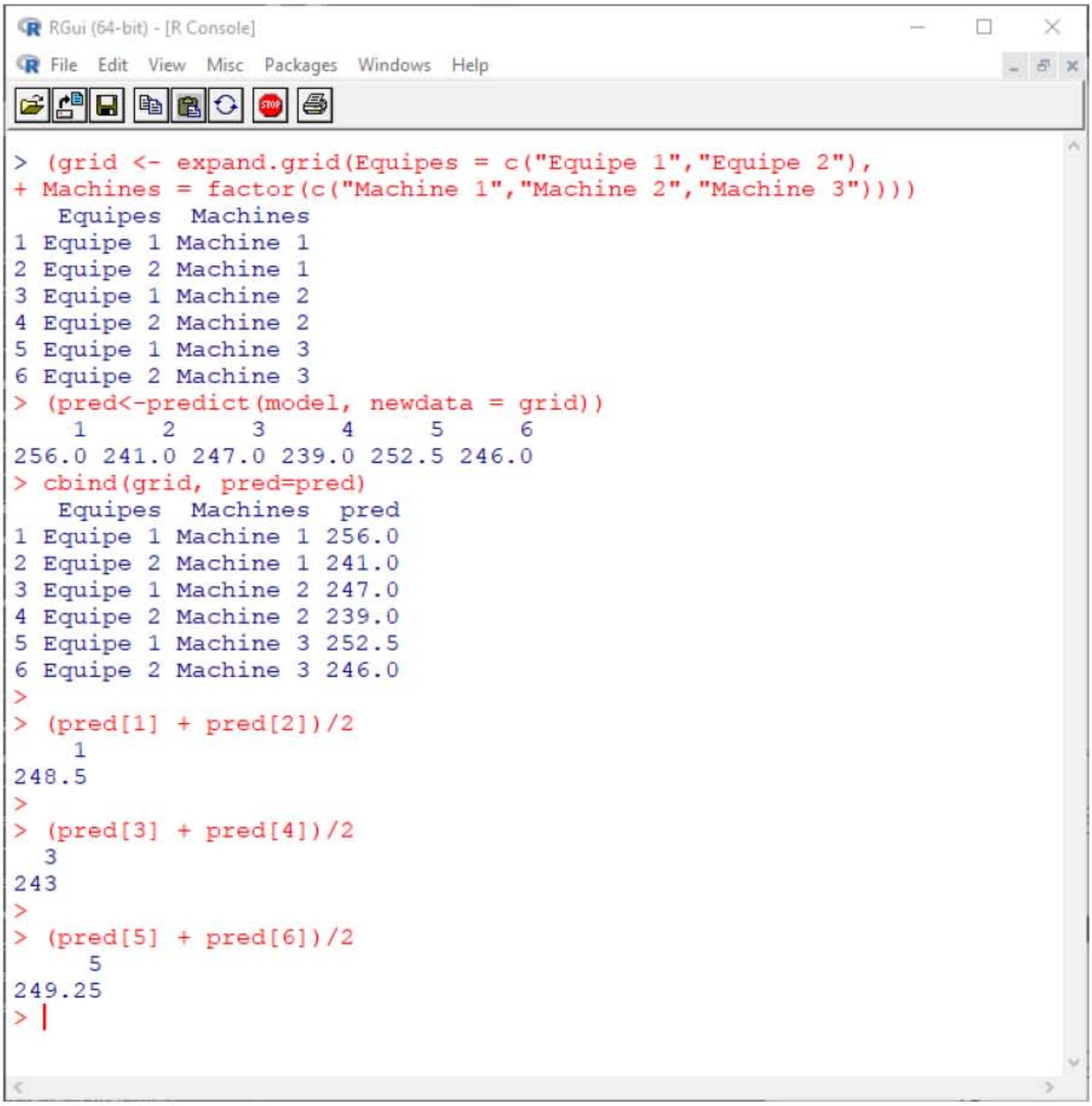
```
> #d'abord avec le package doBy
> library("doBy")
> LSmeans(model, effect="Machines")
Coefficients:
      estimate        se        df    t.stat p.value
[1,] 248.50000   0.88741   5.00000 280.02778       0
[2,] 243.00000   0.72457   5.00000 335.37186       0
[3,] 249.25000   0.72457   5.00000 343.99768       0
>
> #en pour vérification avec le package lsmeans
> library("lsmeans")
> (lsmc<-lsmeans(model, ~Machines))
NOTE: Results may be misleading due to involvement in interactions
 Machines  lsmean    SE df lower.CL upper.CL
 Machine 1    248 0.887  5      246      251
 Machine 2    243 0.725  5      241      245
 Machine 3    249 0.725  5      247      251

Results are averaged over the levels of: Equipes
Confidence level used: 0.95
> plot(lsmc,comparison=TRUE,alpha=0.05)
>
```

Ce qui donne pour le graphique:



Vérifions que nous retrouvons bien les mêmes LS-Means en faisant les calculs à la mains (des valeurs ponctuelles seules!) en utilisant les fonctions **expand.grid( )** et **predict( )**:



```
> (grid <- expand.grid(Equipes = c("Equipe 1","Equipe 2"),
+ Machines = factor(c("Machine 1","Machine 2","Machine 3"))))
   Equipes  Machines
1 Equipe 1 Machine 1
2 Equipe 2 Machine 1
3 Equipe 1 Machine 2
4 Equipe 2 Machine 2
5 Equipe 1 Machine 3
6 Equipe 2 Machine 3
> (pred<-predict(model, newdata = grid))
    1     2     3     4     5     6 
256.0 241.0 247.0 239.0 252.5 246.0 
> cbind(grid, pred=pred)
   Equipes  Machines  pred
1 Equipe 1 Machine 1 256.0
2 Equipe 2 Machine 1 241.0
3 Equipe 1 Machine 2 247.0
4 Equipe 2 Machine 2 239.0
5 Equipe 1 Machine 3 252.5
6 Equipe 2 Machine 3 246.0
>
> (pred[1] + pred[2])/2
    1 
248.5 
>
> (pred[3] + pred[4])/2
  3 
243 
>
> (pred[5] + pred[6])/2
     5 
249.25 
> 
```

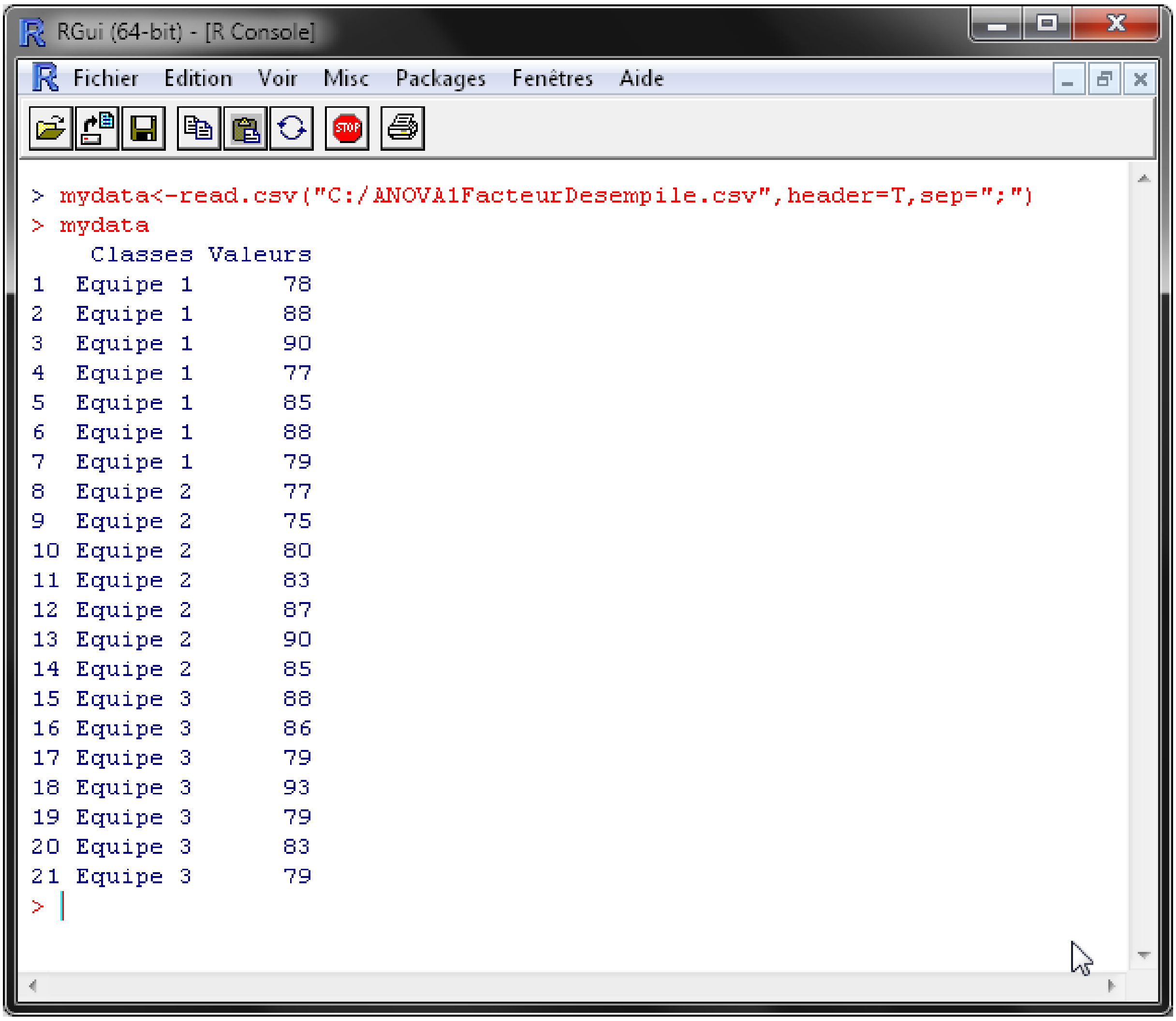
Exemple 7.: Comparaisons multiples du test de Student avec correction de Bonferroni

R 3.0.2

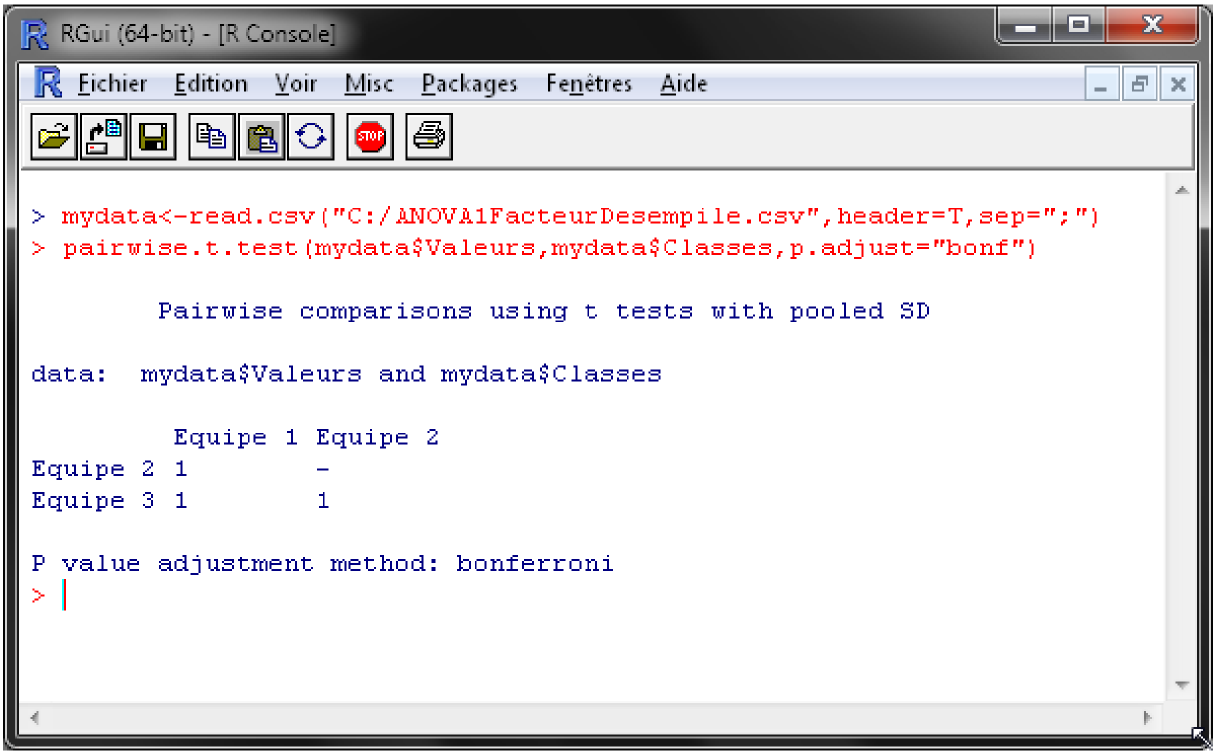
Dans le cours théorique nous avons mentionné la possibilité d'utiliser la technique de Bonferroni avec le test de Student. Par ailleurs, nous avons même démontré mathématiquement l'origine de la correction du niveau de seuil tel que proposé par Bonferroni. Nous n'avons pas pu mettre en œuvre cette technique dans Minitab. Voyons comment alors gérer cela avec R.

Remarque: Nous verrons juste après un test équivalent considéré comme plus robuste qu'est le test de (l'étendue) Tukey HSD.

Nous partons des mêmes données que dans le cours théorique et que l'ANOVA à 1 facteur fixe faite plus haut:



et nous utilisons la commande native **pairwise.t.test( )** de R:



Le résultat peut surprendre dans le cas présent mais c'est simplement que la différence n'est jamais significative et donc que les *p*-value sont in extenso très proches de l'unité.

Une autre approche consiste à utiliser le package **emmeans** (expected marginal means) et sa fonction du même nom:

```
> (anova<-aov(mydata$Valeurs~mydata$Classes))
Call:
   aov(formula = mydata$Valeurs ~ mydata$Classes)

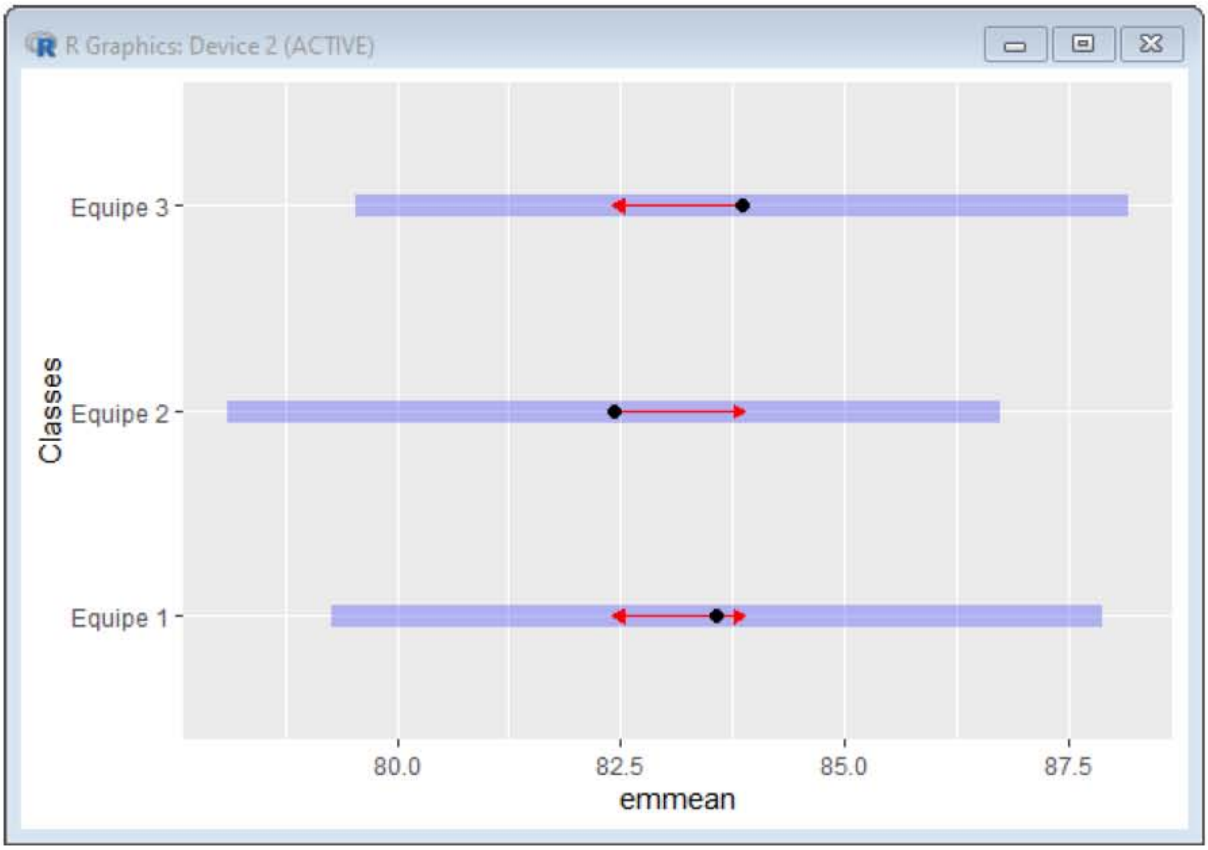
Terms:
                mydata$Classes Residuals
Sum of Squares          8.0000  530.2857
Deg. of Freedom              2        18

Residual standard error: 5.427736
Estimated effects may be unbalanced
> library("emmeans")
> (emm<-emmeans(anova,~Classes))
 Classes  emmean   SE df lower.CL upper.CL
 Equipe 1   83.6 2.05 18     79.3     87.9
 Equipe 2   82.4 2.05 18     78.1     86.7
 Equipe 3   83.9 2.05 18     79.5     88.2

Confidence level used: 0.95
> pairs(emm)
 contrast            estimate  SE df t.ratio p.value
 Equipe 1 - Equipe 2    1.143 2.9 18   0.394  0.9184
 Equipe 1 - Equipe 3   -0.286 2.9 18  -0.098  0.9947
 Equipe 2 - Equipe 3   -1.429 2.9 18  -0.492  0.8758

P value adjustment: tukey method for comparing a family of 3 estimates
> plot(emm, comparisons = TRUE)
> 
```

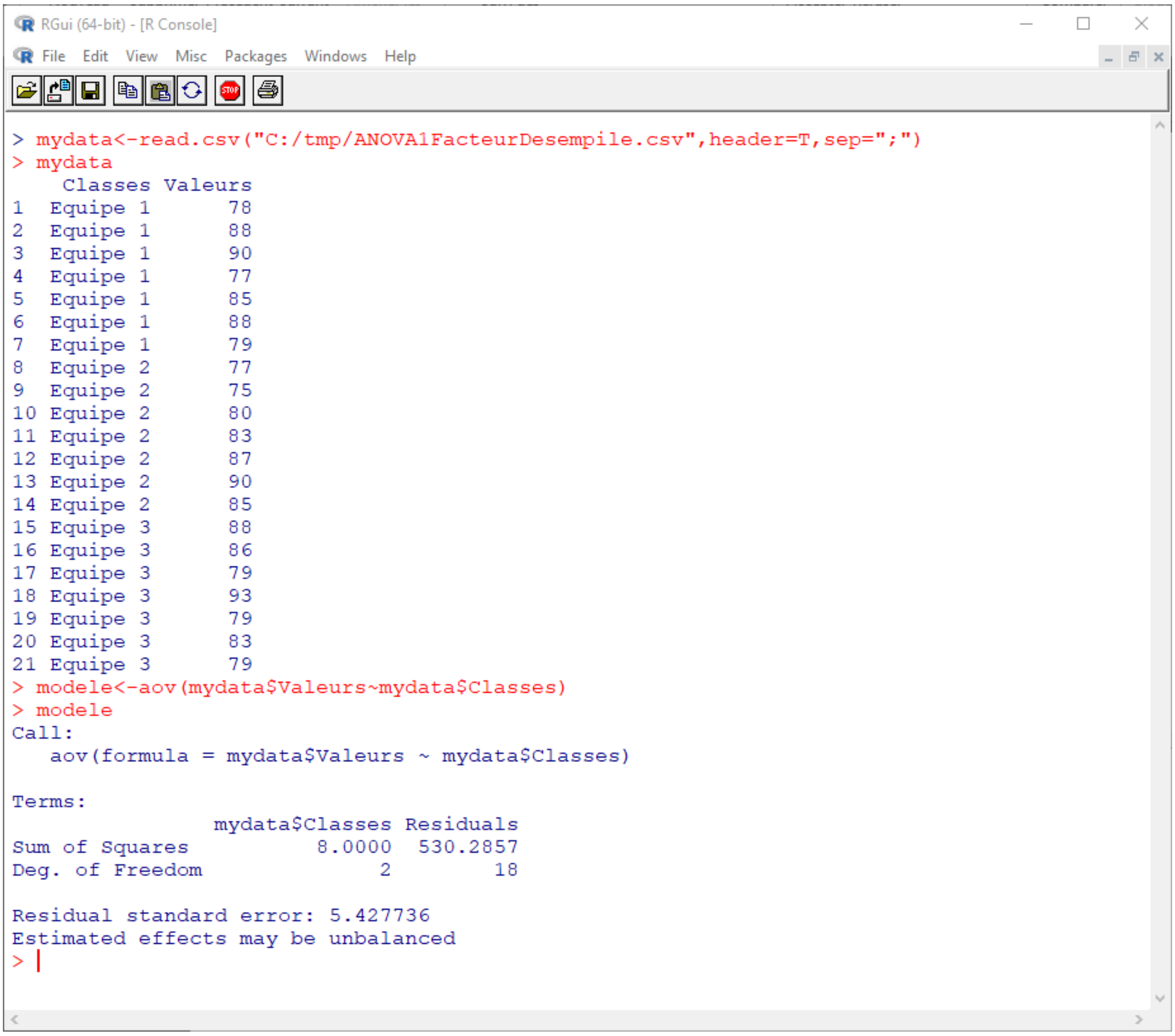
et le graphique correspondant:


R Graphics: Device 2 (ACTIVE)
Equipe 3
Equipe 2
Equipe 1
Classes
80.0
82.5
85.0
87.5
emmean

Exemple 8.: Test post hoc de Fisher (Least Significant Difference)

R 3.6.0

Nous avons vu dans le cours théorique que le test LSD de Fisher est le plus simple des tests posthoc qu'on peut imaginer pour l'ANOVA. Voyons donc déjà un premier exemple avec notre jeu de données habituel et avec un ANOVA classique (nous verrons des cas d'applications plus complexes plus tard!):



```
> mydata<-read.csv("C:/tmp/ANOVA1FacteurDesempile.csv",header=T,sep=";")
> mydata
    Classes Valeurs
1  Equipe 1      78
2  Equipe 1      88
3  Equipe 1      90
4  Equipe 1      77
5  Equipe 1      85
6  Equipe 1      88
7  Equipe 1      79
8  Equipe 2      77
9  Equipe 2      75
10 Equipe 2      80
11 Equipe 2      83
12 Equipe 2      87
13 Equipe 2      90
14 Equipe 2      85
15 Equipe 3      88
16 Equipe 3      86
17 Equipe 3      79
18 Equipe 3      93
19 Equipe 3      79
20 Equipe 3      83
21 Equipe 3      79
> modele<-aov(mydata$Valeurs~mydata$Classes)
> modele
Call:
   aov(formula = mydata$Valeurs ~ mydata$Classes)

Terms:
                mydata$Classes Residuals
Sum of Squares          8.0000  530.2857
Deg. of Freedom              2        18

Residual standard error: 5.427736
Estimated effects may be unbalanced
> 
```

Ensuite nous faisons un test LSD à l'aide du package **agricolae** (**qui contient un grand nombre de différents tests de type LSD!!!**) et sa fonction **LSD.test( )**:

```
> out<-LSD.test(modele,trt="mydata$Classes",p.adj="bonferroni")
> out
$statistics
   MSerror Df     Mean       CV  t.value      MSD
  29.46032 18 83.28571 6.517007 2.639145 7.656811

$parameters
        test  p.ajusted         name.t ntr alpha
  Fisher-LSD bonferroni mydata$Classes   3  0.05

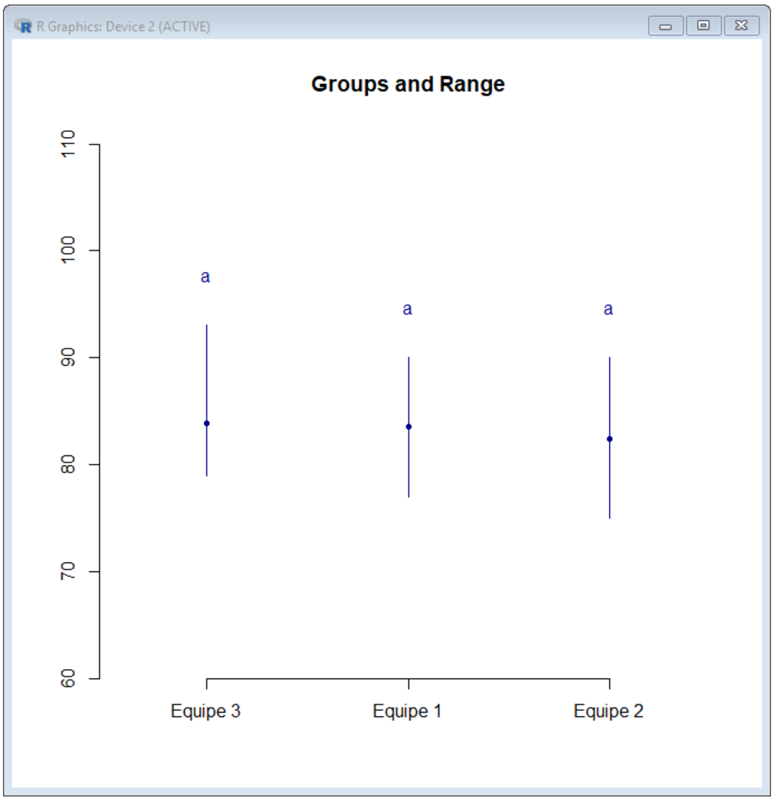
$means
         mydata$Valeurs      std r      LCL      UCL Min Max  Q25 Q50 Q75
Equipe 1       83.57143 5.442338 7 79.26141 87.88145  77  90 78.5  85  88
Equipe 2       82.42857 5.411628 7 78.11855 86.73859  75  90 78.5  83  86
Equipe 3       83.85714 5.429198 7 79.54712 88.16717  79  93 79.0  83  87

$comparison
NULL

$groups
         mydata$Valeurs groups
Equipe 3       83.85714      a
Equipe 1       83.57143      a
Equipe 2       82.42857      a

attr(,"class")
[1] "group"
> plot(out)
>
```

Il n'y a donc aucune différence significative entre les différentes équipes prises par paires! Et pour ceux qui préfèrent les graphiques, voici le plot correspondant:



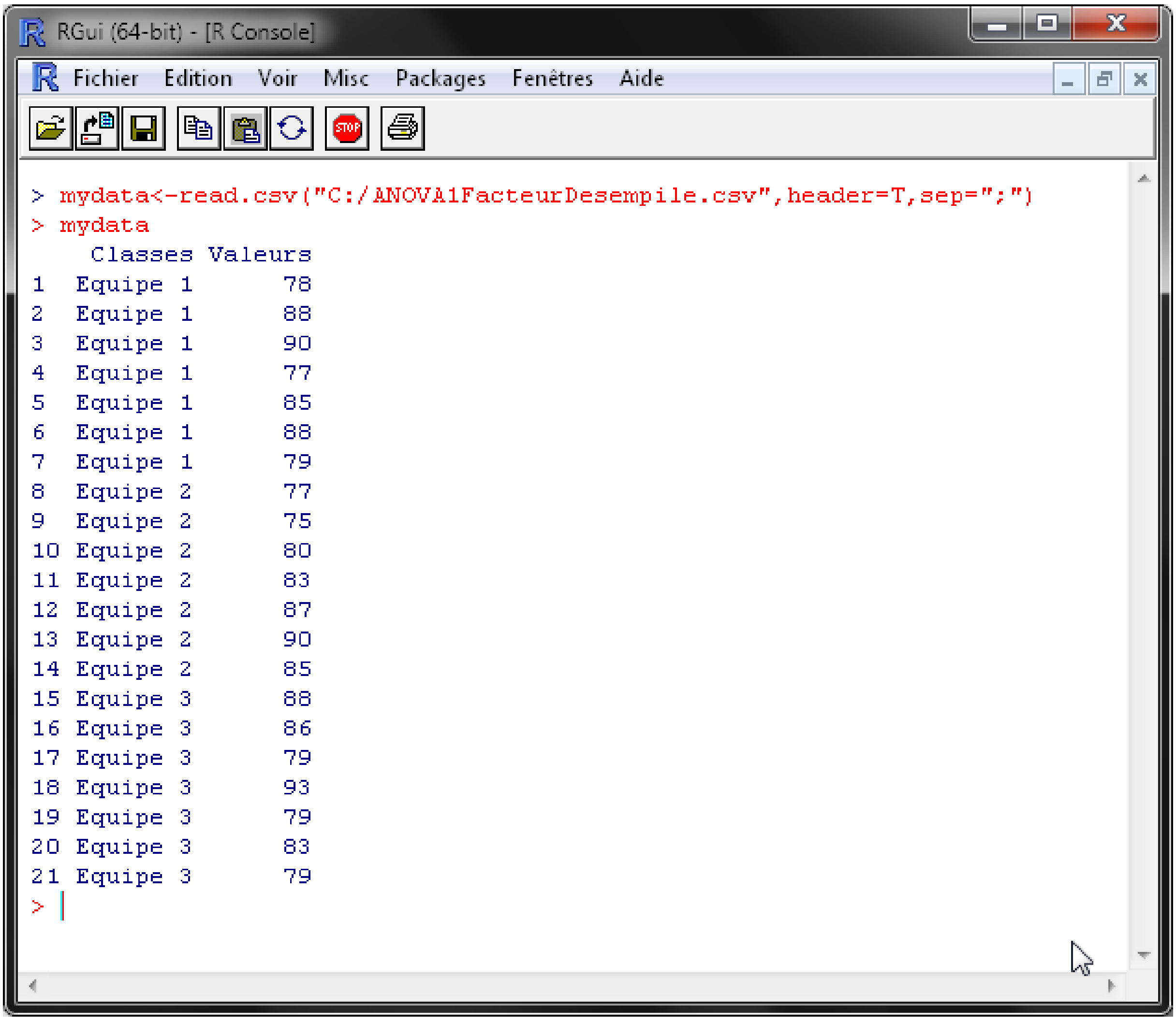
Exemple 9.: Test post hoc de (l'étendue) de Tukey HSD

R 3.0.2

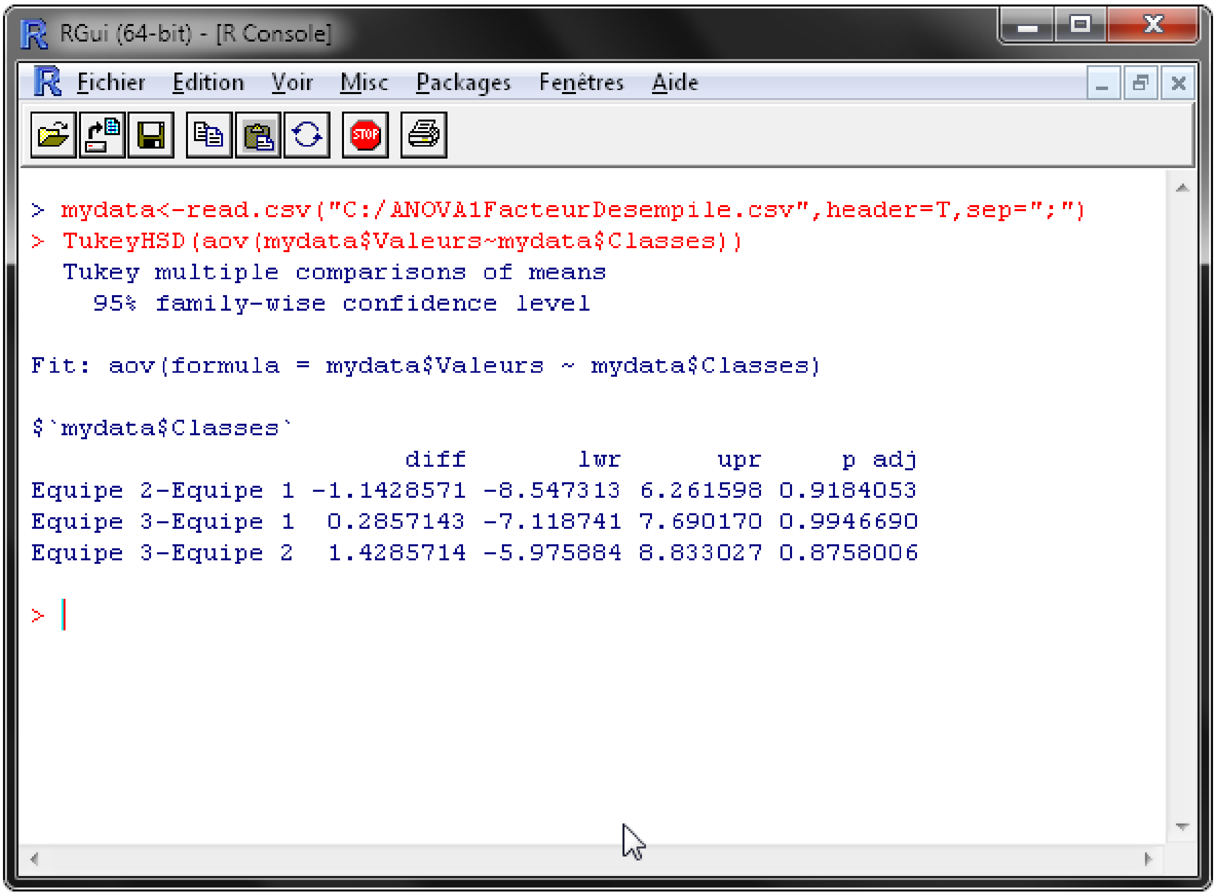
Ici, nous allons vérifier si les concepts démontrés dans le cours de statistique théorique concernant le test de Tukey sont appliqués de façon identique dans R par rapport à Minitab.

Nous n'avons pas en classe dans le cours théorique simulé les distributions de l'étendue studentisée plus par flemme que par manque de temps (les simulations de Monte-Carlo se basant toujours sur le même principe...), donc il n'y aura pas de comparaison par rapport avec Microsoft Excel. Nous ferons ici juste une vérification des résultats renvoyés par Minitab en utilisant les tables des étendues studentisées.

Nous partons des mêmes données que dans le cours Minitab et que l'ANOVA à 1 facteur fixe faite plus haut:



Et ensuite nous utilisons la commande native **TukeyHSD( )** de R:



Nous retrouvons bien les mêmes valeurs que dans le cours Minitab avec un visuel plus sobre et donc moins confus et donc aussi avec les mêmes conclusions.

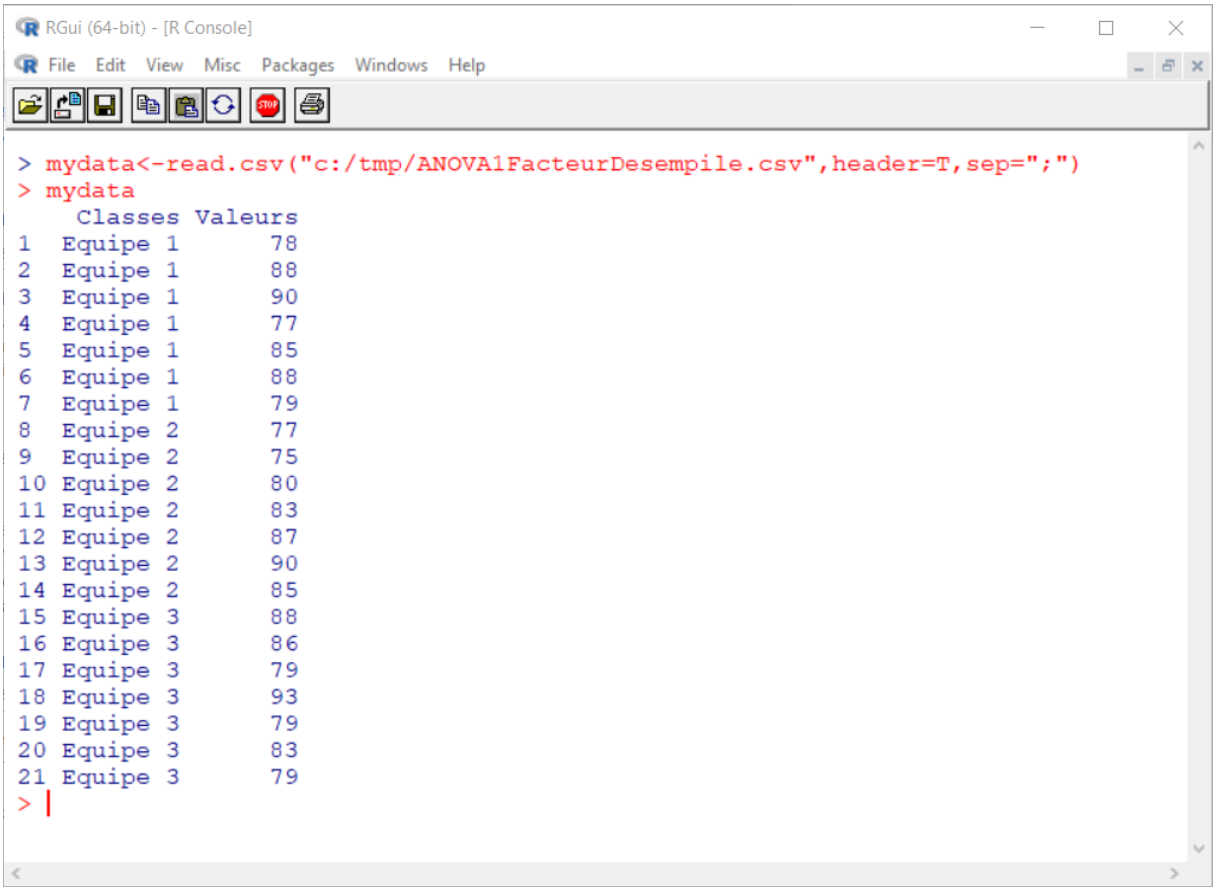
Qu'il n'y a priori pas d'option (contrairement à Minitab) pour faire un test par paire individuelles avec le taux d'erreur individuel de Fisher.

Exemple 10.: Test post hoc de Dunnett

R 3.4.3

Ici, nous allons vérifier si les concepts démontrés dans le cours de statistique théorique concernant le test de de Dunnett sont appliqués de façon identique dans R par rapport à Minitab.

Nous partons des mêmes données que dans le cours Minitab et que l'ANOVA à 1 facteur fixe faite plus haut:



Nous créons un modèle ANOVA avec la fonction **aov( )** et ensuite nous le passons à la fonction **glht( )** du package **multicomp**:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> aov
Call:
   aov(formula = Valeurs ~ Classes, data = mydata)

Terms:
                 Classes Residuals
Sum of Squares    8.0000  530.2857
Deg. of Freedom        2        18

Residual standard error: 5.427736
Estimated effects may be unbalanced
> library("multcomp")
> summary(glht(aov,lincfct=mcp(Group="Dunnett")))

         Simultaneous Tests for General Linear Hypotheses

Fit: aov(formula = Valeurs ~ Classes, data = mydata)

Linear Hypotheses:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept) == 0      83.5714     2.0515  40.737   <1e-04
ClassesEquipe 2 == 0  -1.1429     2.9012  -0.394    0.947
ClassesEquipe 3 == 0   0.2857     2.9012   0.098    0.999
(Adjusted p values reported -- single-step method)

> |
```


Nous pouvons déjà constater que les résultats sont conformes à Minitab même si format de sortie est bien moche… (nous remarquons au passage que cette fonction R prend le premier groupe comme étant automatiquement le groupe de contrôle!) :

Tests de simultanéité de Dunnett pour la moyenne du niveau – moyenne du contrôle

Différence des niveaux	Différence des moyennes	Erreur type de la différence	IC à 95 %	Valeur de T	Valeur de p ajustée
Equipe 2 - Equipe 1	-1.14	2.90	(-8.10; 5.82)	-0.39	0.894
Equipe 3 - Equipe 1	0.29	2.90	(-6.67; 7.24)	0.10	0.993

Niveau de confiance individuel = 97.25 %

Attention! Le package **multicomp** utilise les simulations de Monte Carlo pour calculer les p-values donc à chaque fois que le texte est exécuté, si un seed n'est pas fixé, les résultats seront sensiblement différents à chaque fois.

Nous pouvons aussi vérifier concernant les intervalles de confiance:

```
> confint(glht(aov,lincfct=mcp(Group="Dunnett")))

         Simultaneous Confidence Intervals

Fit: aov(formula = Valeurs ~ Classes, data = mydata)

Quantile = 2.5091
95% family-wise confidence level


Linear Hypotheses:
                     Estimate lwr      upr
(Intercept) == 0      83.5714  78.4240 88.7189
ClassesEquipe 2 == 0 -1.1429  -8.4224  6.1367
ClassesEquipe 3 == 0  0.2857  -6.9939  7.5653

> plot(glht(aov,lincfct=mcp(Group="Dunnett")))
```


Nous voyons que les IC diffèrent un peu…

Et finalement, le graphique de comparaison par rapport au groupe de contrôle:

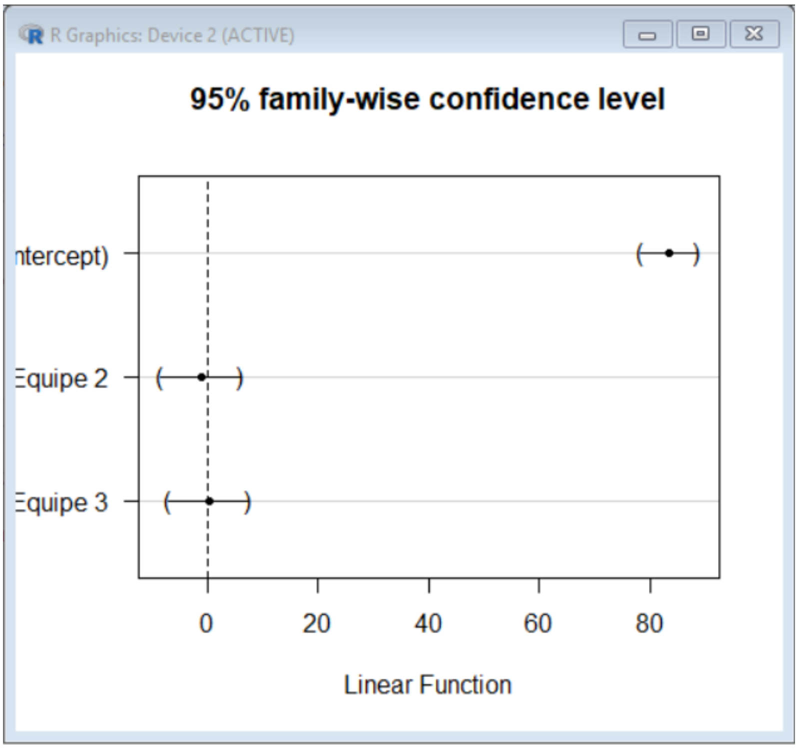


Figure 229 Graph de diagnostique du test de Dunett

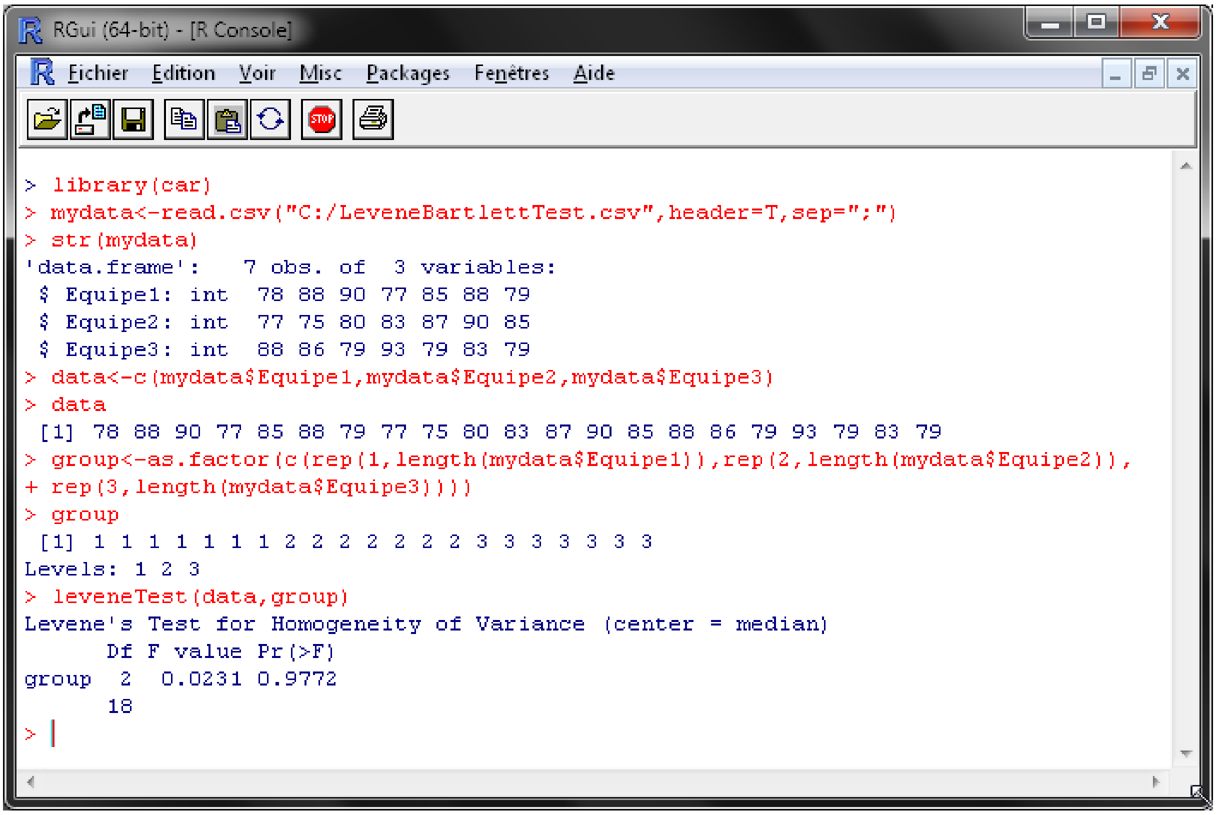
Exemple 11.: Test de Levene et Bartlett d'égalité des variances d'une ANOVA canonique

R 3.0.2

L'objectif ici va être le cas plus intéressant d'application du test de Levene aux données d'une ANOVA empilée et de vérifier encore une fois si les résultats sont conformes aux simulations de Monte-Carlo effectuées dans le cours théoriques et à l'application numérique effectuée dans Microsoft Excel et Minitab.

Nous allons donc ici réutiliser le test de Levene **testLevene( )** du package **car** vu plus haut mais donc aussi le test Bartlett qui sera nouveau dans le cas présent.

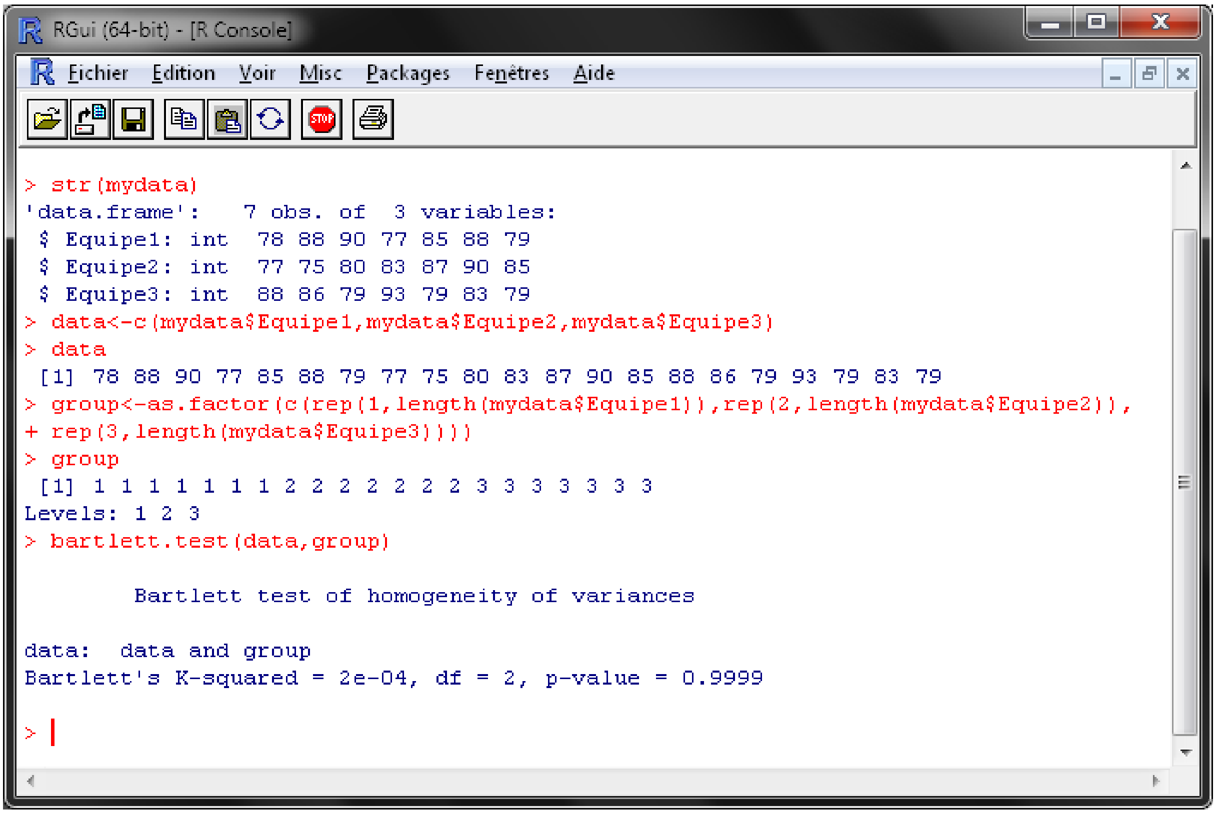
Commençons par le cas du test de Levene:



```
> library(car)
> mydata<-read.csv("C:/LeveneBartlettTest.csv",header=T,sep=";")
> str(mydata)
'data.frame':   7 obs. of  3 variables:
 $ Equipe1: int  78 88 90 77 85 88 79
 $ Equipe2: int  77 75 80 83 87 90 85
 $ Equipe3: int  88 86 79 93 79 83 79
> data<-c(mydata$Equipe1,mydata$Equipe2,mydata$Equipe3)
> data
 [1] 78 88 90 77 85 88 79 77 75 80 83 87 90 85 88 86 79 93 79 83 79
> group<-as.factor(c(rep(1,length(mydata$Equipe1)),rep(2,length(mydata$Equipe2)),
+ rep(3,length(mydata$Equipe3))))
> group
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
Levels: 1 2 3
> leveneTest(data,group)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  0.0231 0.9772
      18
> 
```

Nous retrouvons donc les mêmes valeurs qu'avec Minitab (et donc avec les mêmes conclusions) ou que dans le cours théorique.

Même si nous n'avons pas fait la démonstration du test de Bartlett en cours puisque les détails de celle-ci est introuvable dans les livres et même dans l'article d'origine de Bartlett lui-même (donc si quelqu'un à la démonstration détaillée qu'il n'hésite pas à se manifester!) regardons quand même si nous retrouvons la même valeur qu'avec Minitab en utilisant la fonction **bartlett.test( )** :




```
> str(mydata)
'data.frame':   7 obs. of  3 variables:
 $ Equipe1: int  78 88 90 77 85 88 79
 $ Equipe2: int  77 75 80 83 87 90 85
 $ Equipe3: int  88 86 79 93 79 83 79
> data<-c(mydata$Equipe1,mydata$Equipe2,mydata$Equipe3)
> data
 [1] 78 88 90 77 85 88 79 77 75 80 83 87 90 85 88 86 79 93 79 83 79
> group<-as.factor(c(rep(1,length(mydata$Equipe1)),rep(2,length(mydata$Equipe2)),
+ rep(3,length(mydata$Equipe3))))
> group
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
Levels: 1 2 3
> bartlett.test(data,group)

        Bartlett test of homogeneity of variances

data:  data and group
Bartlett's K-squared = 2e-04, df = 2, p-value = 0.9999

>
```

Donc bien que ce soit beaucoup plus précis c'est le même résultat que celui donné par Minitab.

Exemple 12.: ANOVA Imbriquée (emboîtée)/Hiérarchique complète

R 3.1.1

Non sans mal nous avons démontré dans le cours théorique le concept et les développements relatifs à l'ANOVA hiérarchique complète (full hierarchical ANOVA). Nous allons vérifier ici que nous obtenons les mêmes résultats que Minitab et surtout du NIST lui-même.

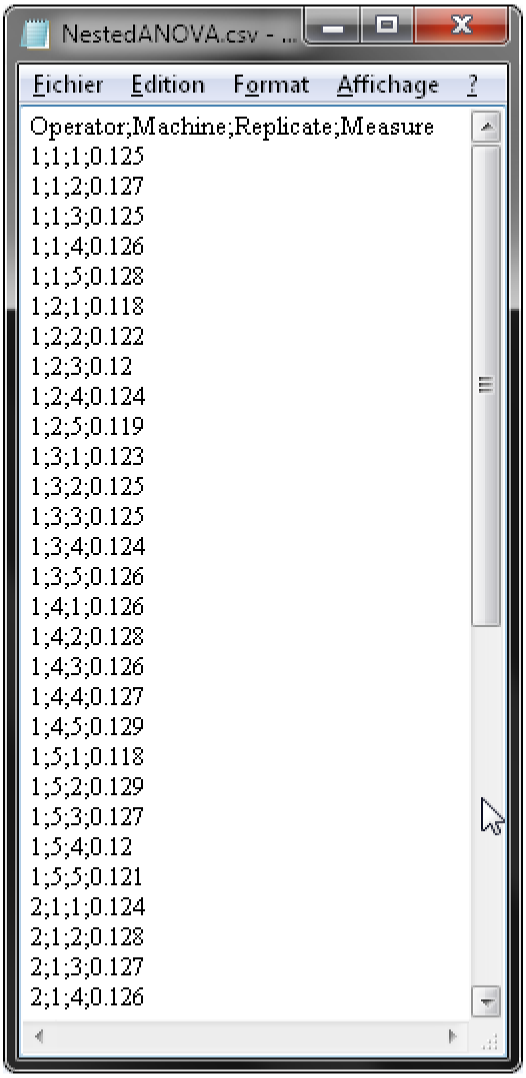
Donc en partant de l'exemple du lien suivant:

http://www.itl.nist.gov/div898/handbook/ppc/section2/ppc233.htm

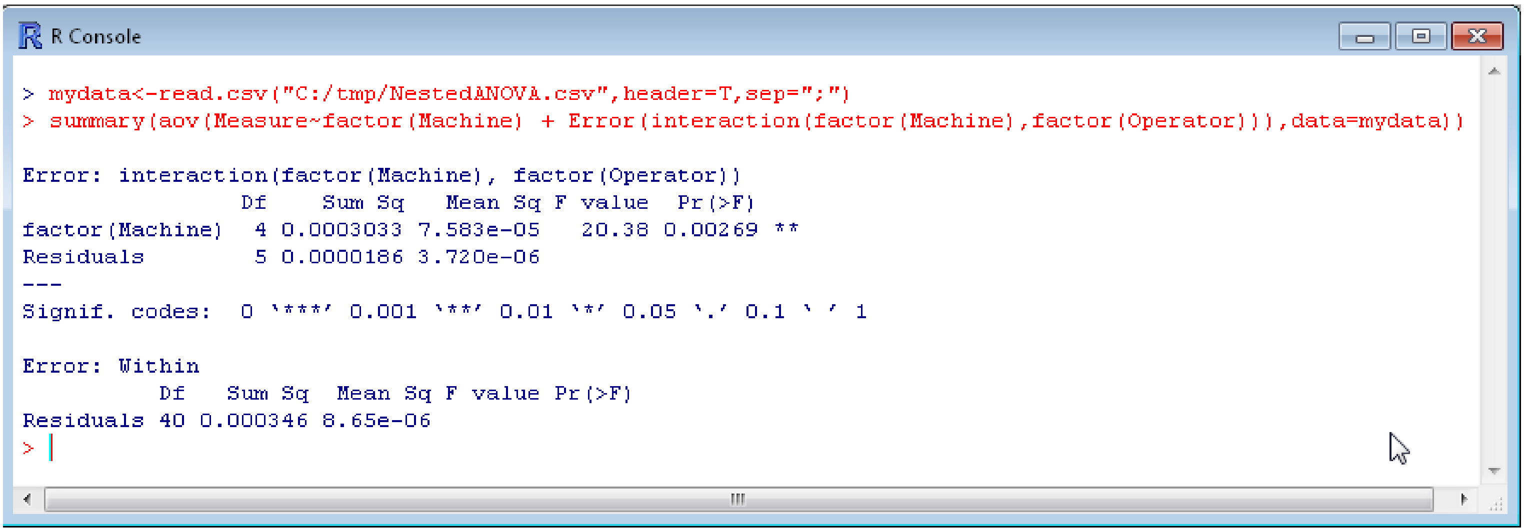
qui contient cette ANOVA hiérarchisée:

	Machines				
	1	**2**	**3**	**4**	**5**
Operator Day	0.125	0.118	0.123	0.126	0.118
	0.127	0.122	0.125	0.128	0.129
	0.125	0.120	0.125	0.126	0.127
	0.126	0.124	0.124	0.127	0.120
	0.128	0.119	0.126	0.129	0.121
Operator Night	0.124	0.116	0.122	0.126	0.125
	0.128	0.125	0.121	0.129	0.123
	0.127	0.119	0.124	0.125	0.114
	0.126	0.125	0.126	0.130	0.124
	0.129	0.120	0.125	0.124	0.117

Donc avec le fichier suivant:



Nous avons alors très subtilement:



Nous pouvons voir que R ne renvoie pas la ligne des totaux ni le deuxième test de Fisher mais sinon le reste de sortie correspond bien à la documentation du NIST:

Source	Sum of Squares	Deg. of Freedom	Mean Square	F_0
Machine	3.03e-4	4	7.58e-5	20.38
Operator(Machine)	1.86e-5	5	3.72e-6	0.428
Residuals	3.46e-4	40	8.70e-6	
Corrected Total	6.68e-4	49		

Pour combler cette lacune nous allons alors procéder en deux étapes. La première ci-dessus a nous a permis d'obtenir le premier test de Fisher, celle ci-dessous nous permet d'obtenir la deuxième:

```
R Console
> summary(aov(Measure~factor(Machine)/factor(Operator),data=mydata))
                                 Df    Sum Sq   Mean Sq F value   Pr(>F)    
factor(Machine)                   4 0.0003033 7.583e-05   8.766 3.52e-05 ***
factor(Machine):factor(Operator)  5 0.0000186 3.720e-06   0.430    0.825    
Residuals                        40 0.0003460 8.650e-06                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> 
```

Sinon la conclusion est la même qu'avec Minitab car ce dernier nous affichait les p-value pour les deux tests de Fisher: Les Machines sont le seul facteur significatif donc les seules améliorations doivent être procédées sur l'outil et non sur l'humain.

Exemple 13.: ANOVA Carré Latin

R 3.1.1

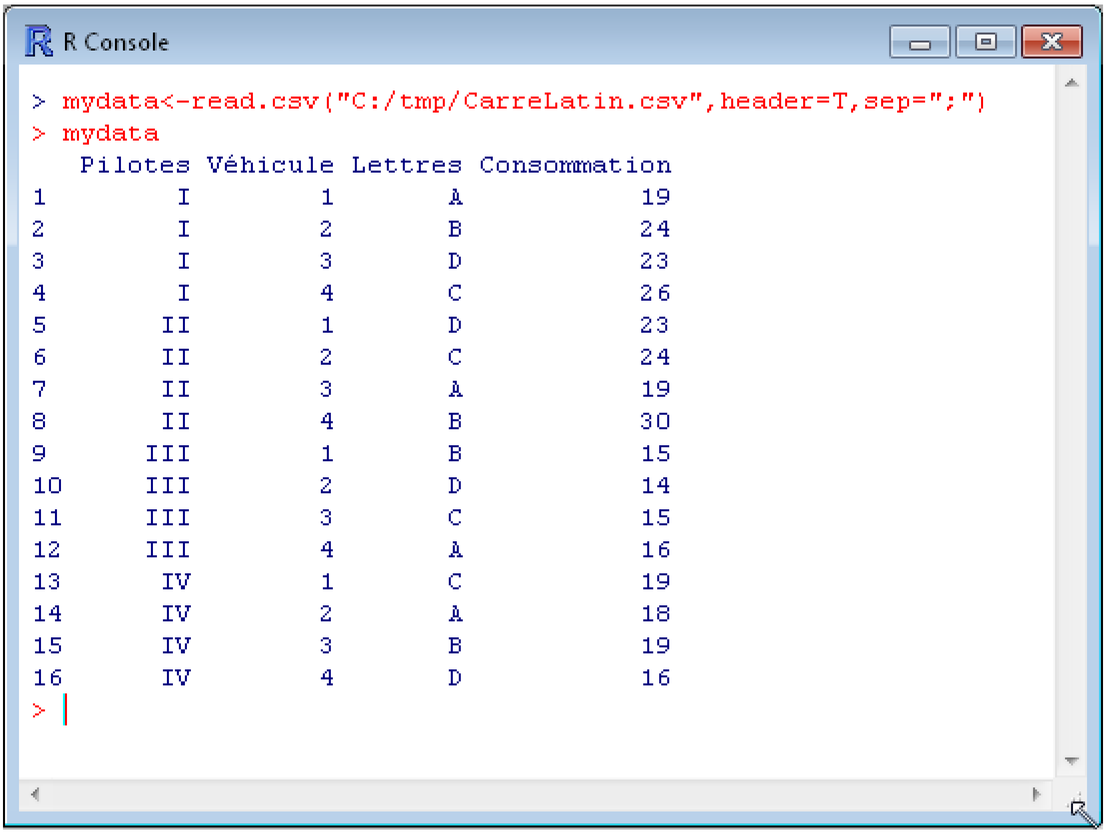
R ne propose pas explicitement au même titre que Minitab de type d'ANOVA qui dans le cadre des plans d'expériences pourrait être utilisé pour chercher la combinaison optimale… donc il va être difficile de l'optimiser… La seule chose que nous pouvons faire a priori c'est une analyse de la variance d'où la présence de ce sujet dans le chapitre des ANOVA.

Nous allons comme à l'habitude reproduire l'exemple du cours théorique et vérifier que nous obtenons bien les mêmes résultats que ceux calculés à la main et obtenus avec Minitab. Pour cela, nous partons des données suivantes reproduisant le tableau carré latin du cours théorique:

Considérons par exemple le cas d'une analyse de rapidité de 4 véhicules différents {1,2,3,4} par 4 pilotes différents {I,II,III,IV} sur 4 routes différentes {A, B, D, C} et les valeurs mesurées sont les consommations de carburant pour parcourir une distance fixe:

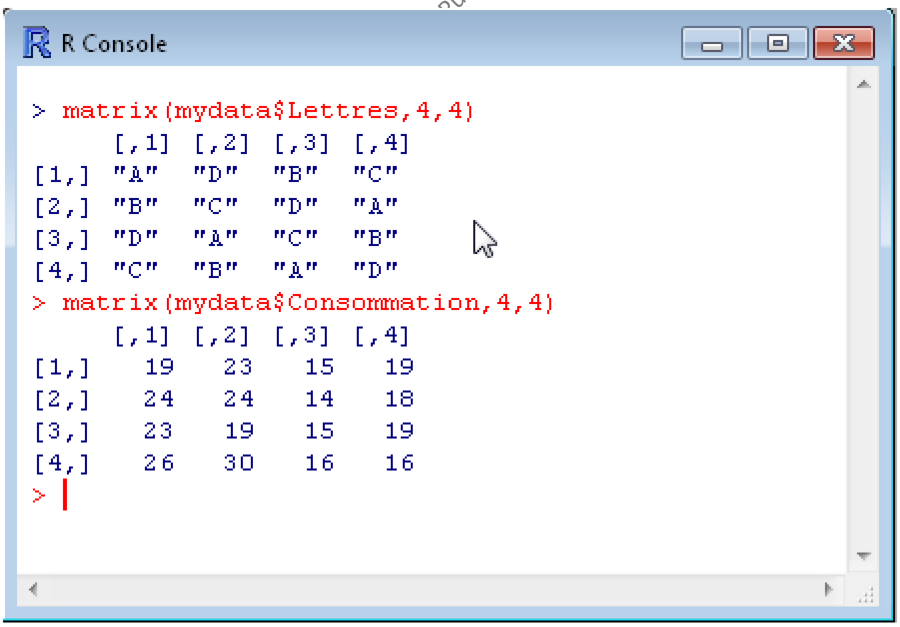
		Véhicules			
		1	2	3	4
Pilotes	I	A 19	B 24	D 23	C 26
	II	D 23	C 24	A 19	B 30
	III	B 15	D 14	C 15	A 16
	IV	C 19	A 18	B 19	D 16

Dans R nous importons un fichier CSV structuré comme le montre la capture d'écran ci-dessous:



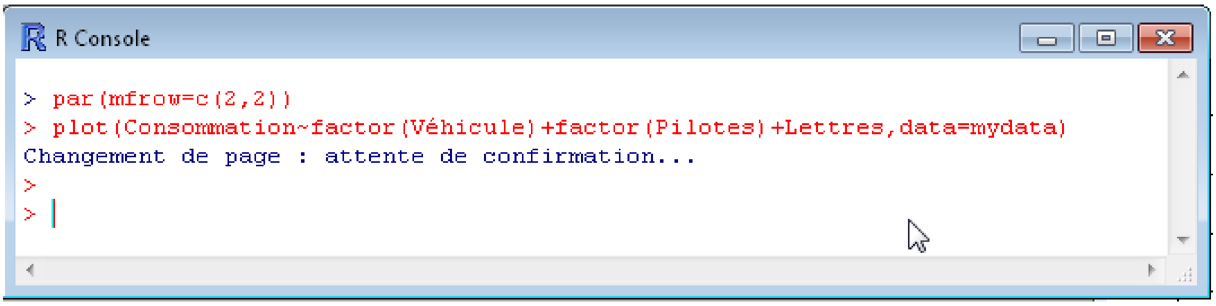
```
> mydata<-read.csv("C:/tmp/CarreLatin.csv",header=T,sep=";")
> mydata
   Pilotes Véhicule Lettres Consommation
1        I        1       A           19
2        I        2       B           24
3        I        3       D           23
4        I        4       C           26
5       II        1       D           23
6       II        2       C           24
7       II        3       A           19
8       II        4       B           30
9      III        1       B           15
10     III        2       D           14
11     III        3       C           15
12     III        4       A           16
13      IV        1       C           19
14      IV        2       A           18
15      IV        3       B           19
16      IV        4       D           16
>
```

Remarquons que nous pouvons reconstruire le carré latin (bon ici il est transposé mais cela ne change rien sur le fond):



```
> matrix(mydata$Lettres,4,4)
     [,1] [,2] [,3] [,4]
[1,] "A"  "D"  "B"  "C"
[2,] "B"  "C"  "D"  "A"
[3,] "D"  "A"  "C"  "B"
[4,] "C"  "B"  "A"  "D"
> matrix(mydata$Consommation,4,4)
     [,1] [,2] [,3] [,4]
[1,]   19   23   15   19
[2,]   24   24   14   18
[3,]   23   19   15   19
[4,]   26   30   16   16
>
```

Nous pouvons faire une analyse qualitative avec les boîtes à moustache:



Nous pouvons déjà constater ci-dessous que les pilotes on une influence certaine alors que pour les autres rien n'est moins sûr ce que nous vérifierons avec les tests de Fisher:

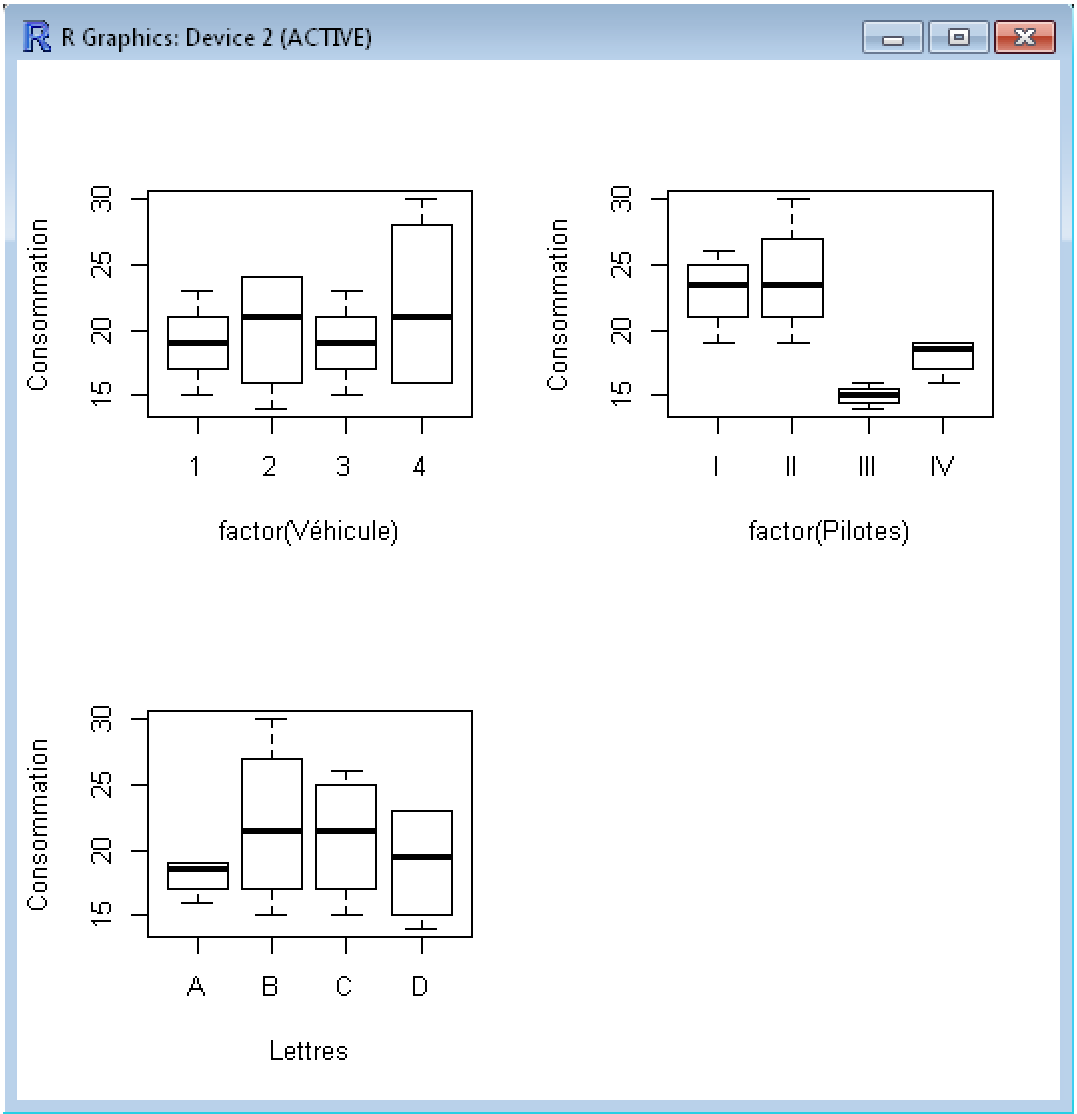
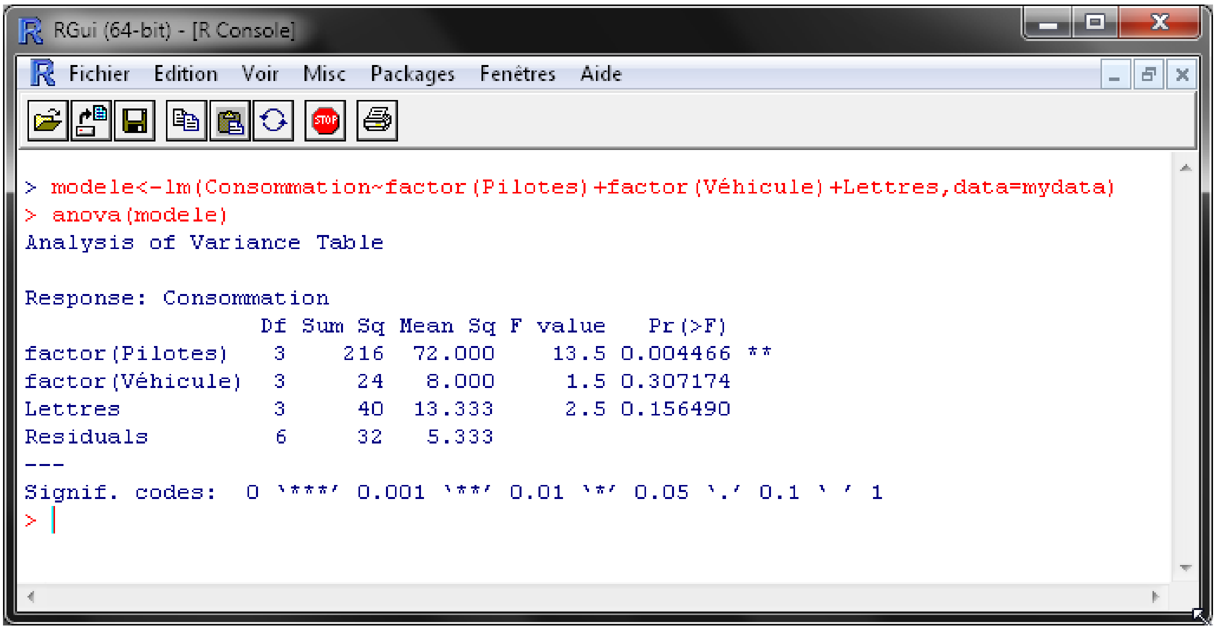


Figure 230 Graph d'effets moyens Box-plot d'un ANOVA carré latin

Maintenant nous faisons l'ANOVA du carré Latin:



Nous retrouvons bien les valeurs calculées à la main! Avec les mêmes conclusions: Seuls les pilotes influencent significativement le temps écoulé!

Exemple 14.: ANOVA Gréco-Latin

R 3.6.0

Nous allons considérer l'expérience comprenant:

- Quatre ouvriers assembleurs
- Quatre méthodes d'assemblages
- Quatre ordres d'assemblages
- Quatre lieux d'assemblages

Nous considérons donc le carré gréco-latin suivant (temps d'assemblage en heures):

		Opérateur			
		1	**2**	**3**	**4**
Ordre Assemblage	**1**	$C\beta = 10$	$B\gamma = 10$	$D\delta = 12$	$A\alpha = 7$
	2	$B\alpha = 8$	$C\delta = 15$	$A\gamma = 7$	$D\beta = 14$
	3	$A\delta = 6$	$D\alpha = 14$	$B\beta = 11$	$C\gamma = 13$
	4	$D\gamma = 11$	$A\beta = 8$	$C\alpha = 10$	$B\delta = 8$

Où:

- Les méthodes d'assemblages sont indexées par les lettres latines
- Les lieux d'assemblages par les lettres grecques

Voyons d'abord comment construire des plans gréco-latins de manière randomisée étant donné que ce n'est pas un exercice simple pour l'humain moyen en utilisant le package **agricolae** et sa fonction **design.graeco( )**:

```
> library("agricolae")
> latines<-c("A","B","C","D")
> grecques<- c(1,2,3,4)
>
> #premier tableau gréco-latin
> table<-design.graeco(trt1=latines,trt2=grecques,seed=1)
> table$sketch
     [,1]  [,2]  [,3]  [,4] 
[1,] "A 3" "B 4" "D 1" "C 2"
[2,] "B 1" "A 2" "C 3" "D 4"
[3,] "D 2" "C 1" "A 4" "B 3"
[4,] "C 4" "D 3" "B 2" "A 1"
>
> #une deuxième tableau gréco-latin du même type
> table<- design.graeco(trt1=latines,trt2=grecques,seed=10)
> table$sketch
     [,1]  [,2]  [,3]  [,4] 
[1,] "A 4" "B 2" "D 1" "C 3"
[2,] "B 1" "A 3" "C 4" "D 2"
[3,] "D 3" "C 1" "A 2" "B 4"
[4,] "C 2" "D 4" "B 3" "A 1"
> 
```

Mais pour reproduire le tableau donné plus haut, nous écrirons:

```
> #Construction du tableau
> Ordre_Assemblage<-c(1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)
> Operateur<-c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4)
> Methode_Assemblage<-c("C","B","A","D","B","C","D","A","D","A","B","C","A","D","C","B")
> Lieu_Travail<-c(2,1,4,3,3,4,1,2,4,3,2,1,1,2,3,4)
> Temps_Assemblage<-c(10,8,6,11,10,15,14,8,12,7,11,10,7,14,13,8)
>
> #Conversions
> Ordre_Assemblage<-factor(Ordre_Assemblage)
> Operateur<-factor(Operateur)
> Methode_Assemblage<-factor(Methode_Assemblage)
> Lieu_Travail<-factor(Lieu_Travail)
> 
```

Maintenant nous pouvons lancer l'analyse en utilisant simplement les fonctions natives **lm( )** et **aov( )**:

```
> modele<-lm(Temps_Assemblage~Ordre_Assemblage+Operateur+Methode_Assemblage+Lieu_Travail)
> modele

Call:
lm(formula = Temps_Assemblage ~ Ordre_Assemblage + Operateur +
    Methode_Assemblage + Lieu_Travail)

Coefficients:
       (Intercept)   Ordre_Assemblage2   Ordre_Assemblage3   Ordre_Assemblage4
              4.50                1.25                1.25               -0.50
        Operateur2          Operateur3          Operateur4  Methode_AssemblageB
              3.00                1.25                1.75                2.25
Methode_AssemblageC Methode_AssemblageD       Lieu_Travail2       Lieu_Travail3
              5.00                5.75                1.00                0.50
      Lieu_Travail4
              0.50

> anova<-aov(modele)
> summary(anova)
                   Df Sum Sq Mean Sq F value Pr(>F)
Ordre_Assemblage    3    9.5   3.167   2.714 0.2170
Operateur           3   18.5   6.167   5.286 0.1024
Methode_Assemblage  3   83.5  27.833  23.857 0.0135
Lieu_Travail        3    2.0   0.667   0.571 0.6714
Residuals           3    3.5   1.167
> 
```

Nous retrouvons donc bien les valeurs attendues!

On peut donc voir que seulement la méthode d'assemblage est significative!

Faisons maintenant un test LSD de Fisher à l'aide de la fonction **LSD.test( )** pour identifier quels sont les groupes significativement différents:

```
> out<-LSD.test(y=anova,trt="Methode_Assemblage",group=T,console=T)

Study: anova ~ "Methode_Assemblage"

LSD t Test for Temps_Assemblage

Mean Square Error:  1.166667

Methode_Assemblage,  means and individual ( 95 %) CI

  Temps_Assemblage       std r        LCL        UCL Min Max
A             7.00 0.8164966 4  5.281283  8.718717   6   8
B             9.25 1.5000000 4  7.531283 10.968717   8  11
C            12.00 2.4494897 4 10.281283 13.718717  10  15
D            12.75 1.5000000 4 11.031283 14.468717  11  14

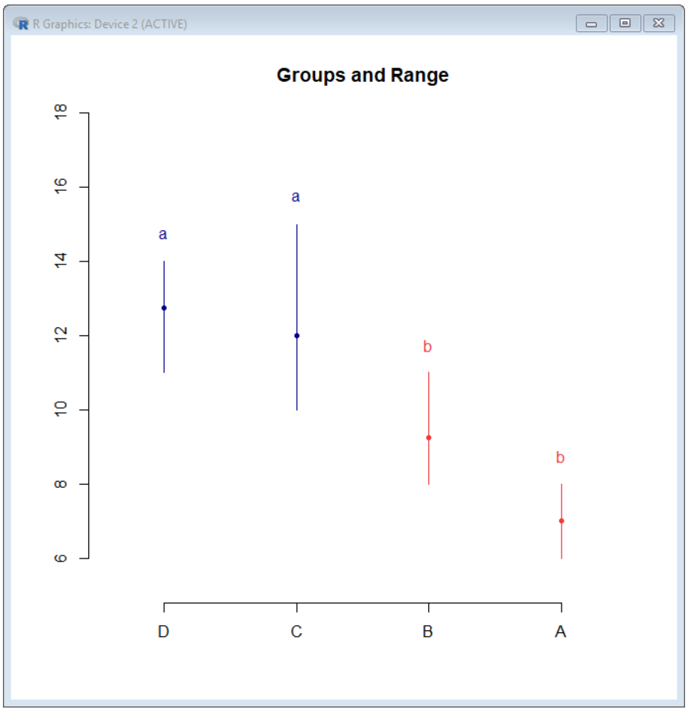
Alpha: 0.05 ; DF Error: 3
Critical Value of t: 3.182446

least Significant Difference: 2.430634

Treatments with the same letter are not significantly different.

  Temps_Assemblage groups
D            12.75      a
C            12.00      a
B             9.25      b
A             7.00      b
> plot(out)
>
```

Avec le graphique correspondant:



Donc les méthodes d'assemblage qui nécessitent le moins de temps d'assemblage moyen sont A et B. Celles nécessitant le plus grand temps d'assemblage sont D et C.

Exemple 15.: ANCOVA (Analyse de la Covariance) dixit régression hiérarchique modérée

R 3.1.1

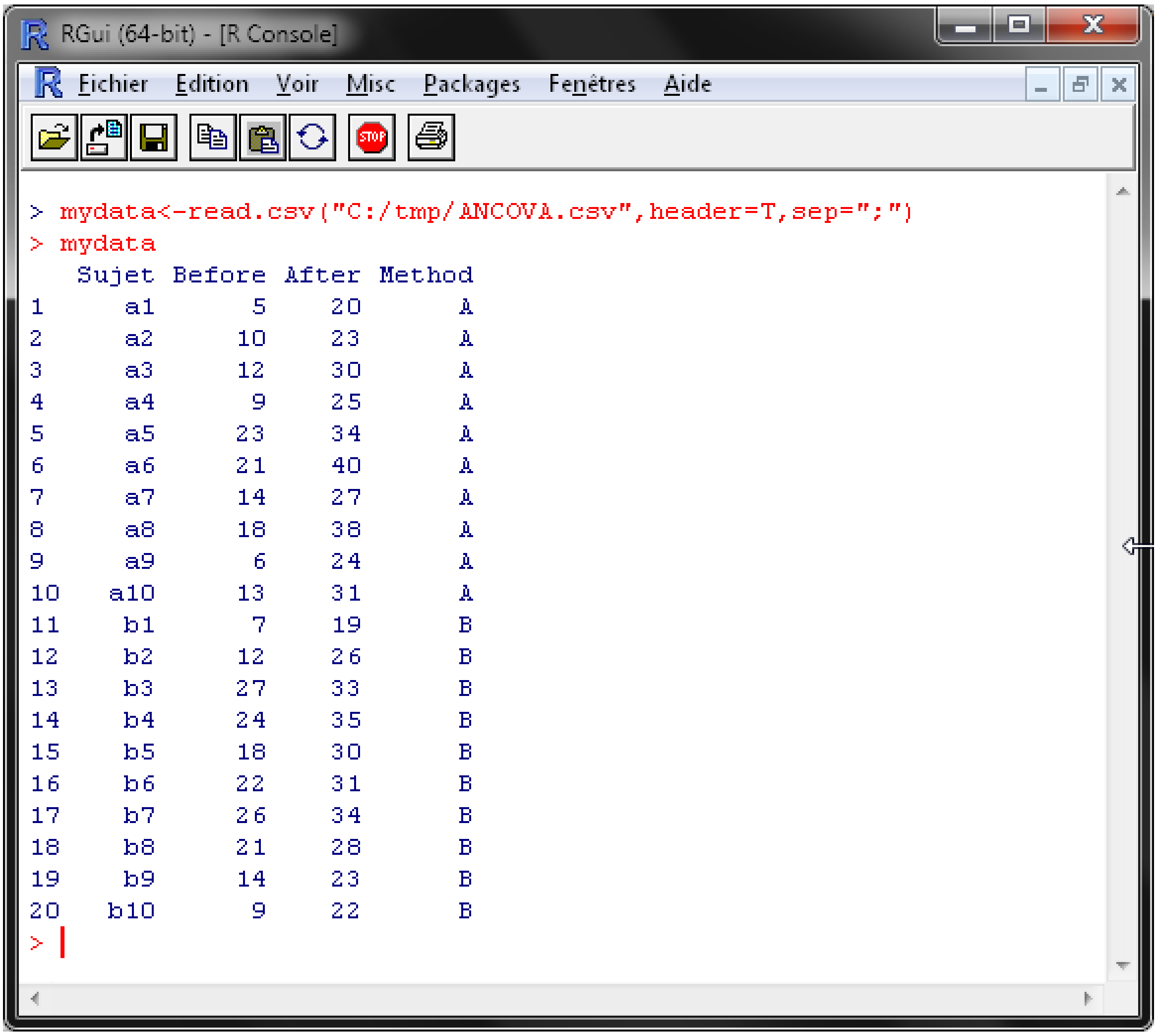
Notre but ici est de vérifier si nous réobtenons bien les résultats des calculs effectués à la main dans le cours théorique et ce avec quel niveau de détails.

D'abord, nous reprenons les données du cours théorique qui sous forme esthétique est donnée pour rappel par:

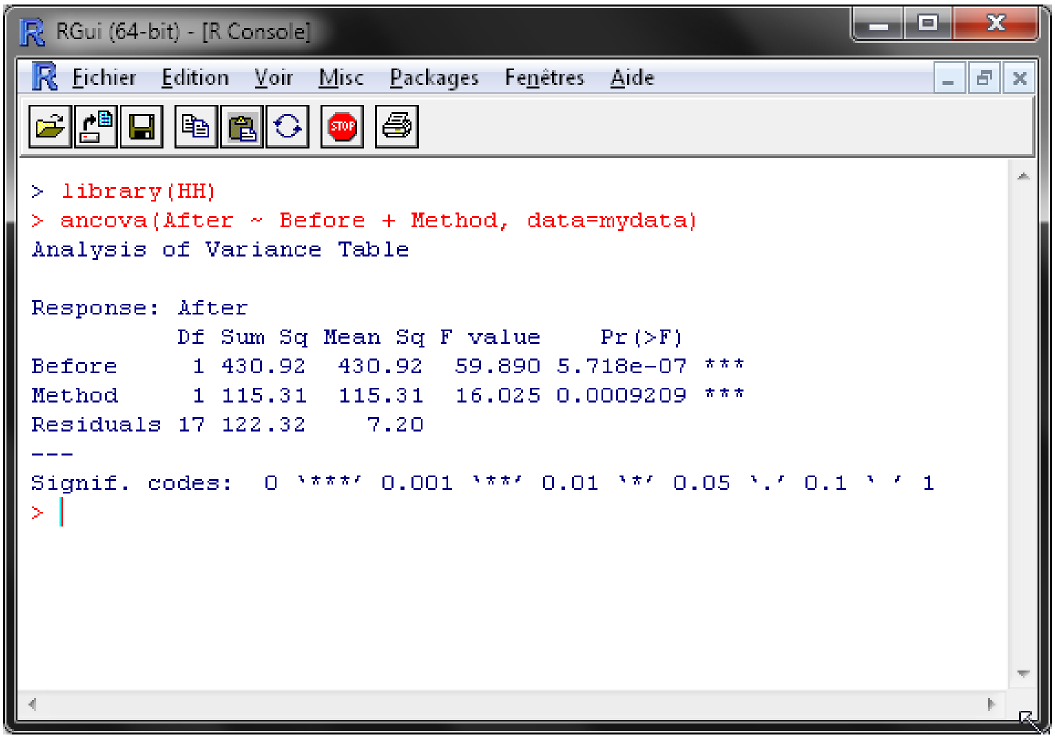
	Méthode A				Méthode B	
Sujet	X_a	Y_a		Sujets	X_b	Y_b
a1	5	20		b1	7	19
a2	10	23		b2	12	26
a3	12	30		b3	27	33
a4	9	25		b4	24	35
a5	23	34		b5	18	30
a6	21	40		b6	22	31
a7	14	27		b7	26	34
a8	18	38		b8	21	28
a9	6	24		b9	14	23
a10	13	31		b10	9	22
Moyenne	13.1	29.2			18.0	28.1

Donc la *Méthode* est selon le vocabulaire du domaine de la régression hiérarchique, la variable de "modération".

Ce qui nous donne dans R:

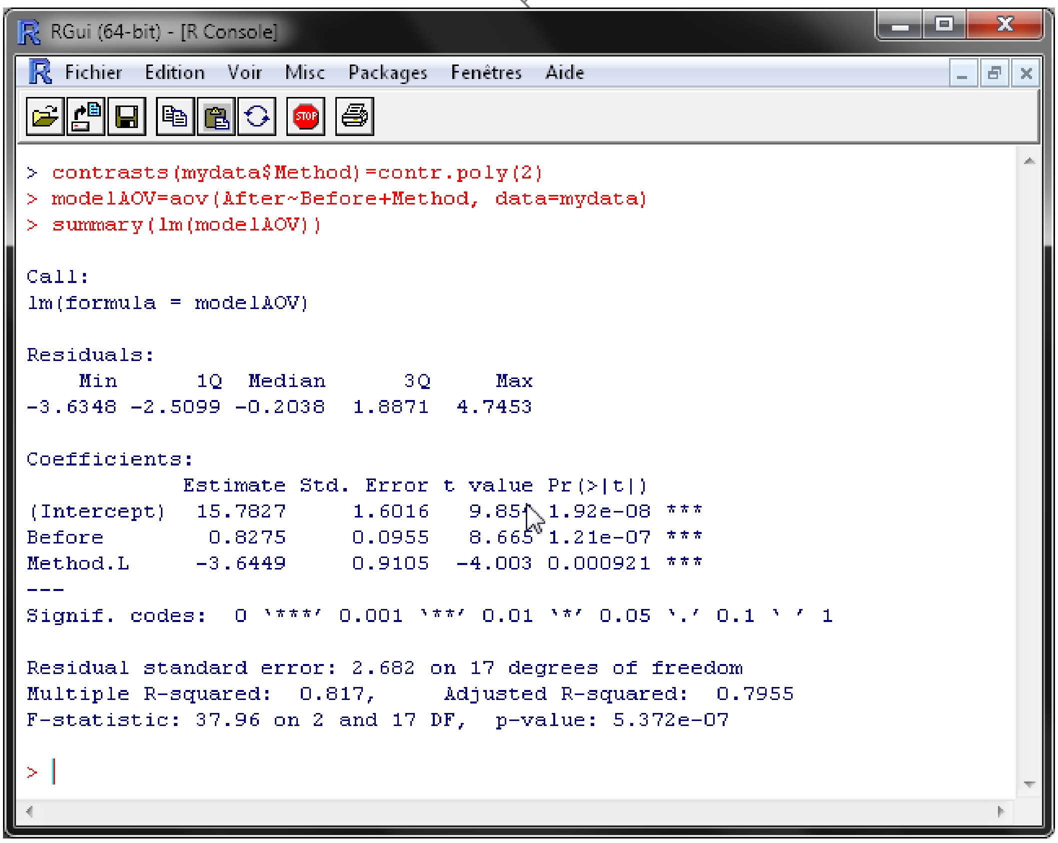


Ensuite nous utilisons pour simplifier la procédure la commande **ancova( )** du package **HH** pour obtenir:



C'est un peu basique comme résumé puisqu'il n'y a aucun test post-hoc mais bon sinon nous retrouvons bien les résultats calculés dans le cours théorique à main avec la même conclusion!

Pour avoir la partie de la régression de Minitab nous utilisons (très…) subtilement la commande **contrasts( )** conjointement avec **aov( )** et **lm( )** :



Rappelons que l'ANCOVA suppose implicitement aucune interaction. Donc voyons comment grâce à la syntaxe précédente, nous pouvons inclure une interaction de la variable de modération:

```
> modelAOV<-aov(After~Before+Method+Before*Method,data=mydata)
> lm(modelAOV)

Call:
lm(formula = modelAOV)

Coefficients:
    (Intercept)            Before          Method.L  Before:Method.L
        15.7657            0.8486           -0.9291          -0.1793

> modelAOV
Call:
   aov(formula = After ~ Before + Method + Before * Method, data = mydata)

Terms:
                  Before   Method Before:Method Residuals
Sum of Squares  430.9238 115.3060       12.3371  109.9831
Deg. of Freedom        1        1             1        16

Residual standard error: 2.621821
Estimated effects may be unbalanced
> summary(lm(modelAOV))

Call:
lm(formula = modelAOV)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8562 -1.7500  0.0696  1.8982  4.0207

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     15.76566    1.56549  10.071 2.49e-08
Before           0.84856    0.09466   8.964 1.23e-07
Method.L        -0.92909    2.21394  -0.420    0.680
Before:Method.L -0.17935    0.13387  -1.340    0.199

Residual standard error: 2.622 on 16 degrees of freedom
Multiple R-squared:  0.8355,    Adjusted R-squared:  0.8046
F-statistic: 27.09 on 3 and 16 DF,  p-value: 1.655e-06

>
```

Exemple 16.: MANOVA

R 3.1.1

Nous allons à nouveau contrôler que nous obtenons bien certains éléments calculés à la main dans le cours théorique.

D'abord nous préparons le jeu de données en conséquence:

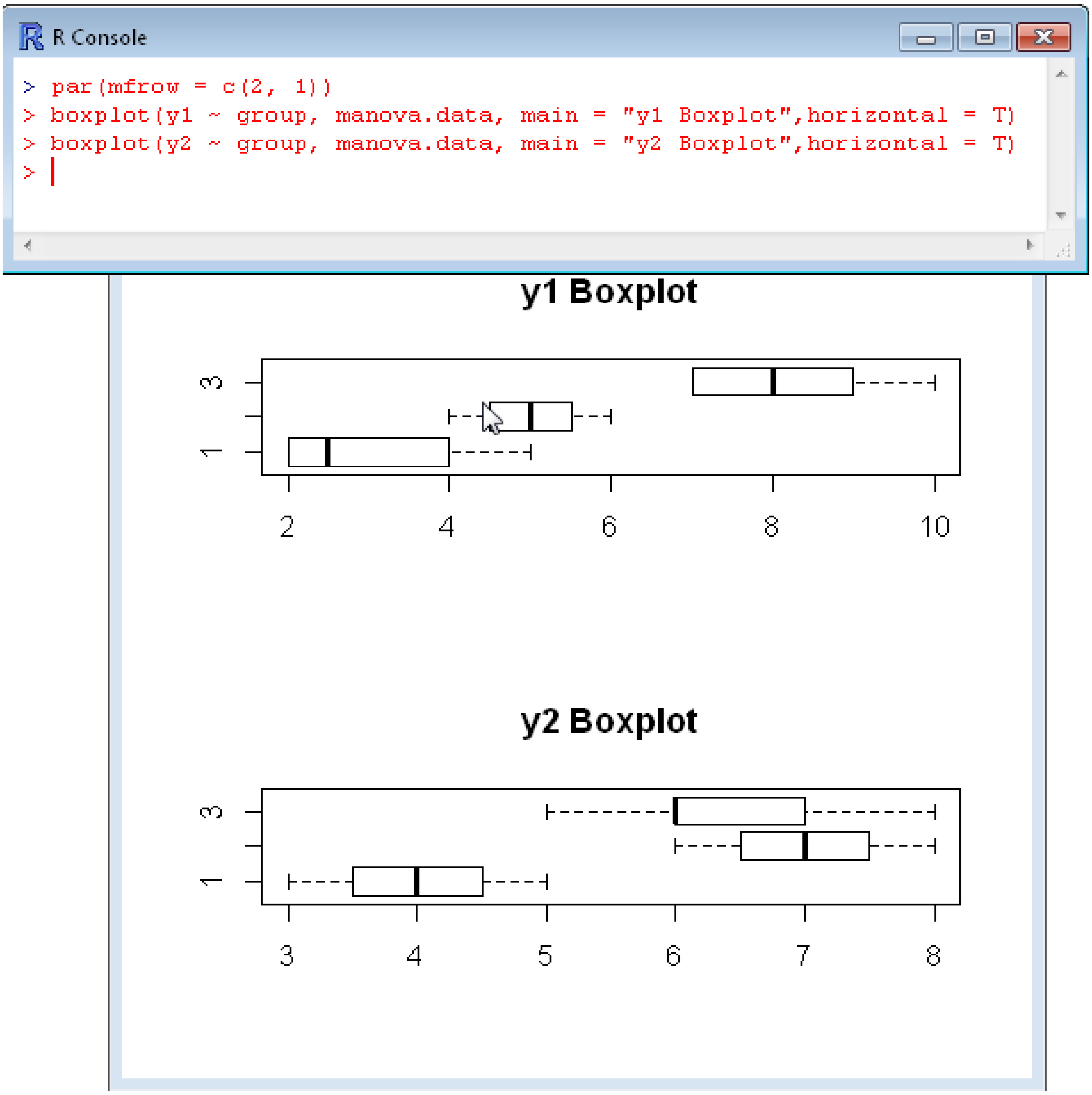
```
> manova.data <- data.frame(group = as.factor(rep(1:3,
+  c(4, 3, 5))), y1 = c(2, 3, 5, 2, 4, 5, 6,
+  7, 8, 10, 9, 7), y2 = c(3, 4, 4, 5, 8, 6,
+  7, 6, 7, 8, 5, 6))
> manova.data
   group y1 y2
1      1  2  3
2      1  3  4
3      1  5  4
4      1  2  5
5      2  4  8
6      2  5  6
7      2  6  7
8      3  7  6
9      3  8  7
10     3 10  8
11     3  9  5
12     3  7  6
> 
```

Ensuite, nous calculons les moyennes de chacun de variables dépendantes pour chacun des groupes:

```
> with(manova.data, tapply(y1, group, mean))
  1   2   3 
3.0 5.0 8.2 
> with(manova.data, tapply(y2, group, mean))
  1   2   3 
4.0 7.0 6.4 
> 
```

Nous obtenons bien les calculs effectués à la main.

Ensuite, rien ne nous empêche de faire une analyse graphique visuelle:



Nous pouvons déjà constater qu'entre les variables dépendantes ET entre les variables indépendantes il y dans les deux cas des variations qui semblent être très fortes.

Pour la suite, nous utilisons la fonction native **manova( )** de R:

```
> m1 <- manova(cbind(y1, y2) ~ group, manova.data)
> m1
Call:
   manova(cbind(y1, y2) ~ group, manova.data)

Terms:
                   group Residuals
resp 1          61.86667  14.80000
resp 2             19.05      9.20
Deg. of Freedom        2         9

Residual standard errors: 1.282359 1.01105
Estimated effects may be unbalanced
> summary(m1, test = "Wilks")
          Df    Wilks approx F num Df den Df    Pr(>F)
group      2 0.089674   9.3575      4     16 0.0004271 ***
Residuals  9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> 
```

Donc conformément à ce que nous pouvions nous attendre, nous rejetons l'hypothèse nulle.

Exemple 17.: ANOVA par permutation

R 3.1.1

Comme nous l'avons mentionné dans le cours théorique, selon différentes recherches, il semblerait que les ANOVA par permutation soient plus robustes que les ANOVA non-paramétriques de Friedman ou de Kruskal-Wallis. Voyons donc comment exécuter un de des dernières avec deux packages différentes: le package **permuco** et le package **lmperm**!

Voyons donc **permuco** et sa fonction **aovperm( )** et **lmPerm** avec sa fonction **aovp( )**:

```
> mydata<-read.csv("c:/tmp/ANOVA1FacteurDesempile.csv",header=T,sep=";")
> mydata
    Classes Valeurs
1  Equipe 1      78
2  Equipe 1      88
3  Equipe 1      90
4  Equipe 1      77
5  Equipe 1      85
6  Equipe 1      88
7  Equipe 1      79
8  Equipe 2      77
9  Equipe 2      75
10 Equipe 2      80
11 Equipe 2      83
12 Equipe 2      87
13 Equipe 2      90
14 Equipe 2      85
15 Equipe 3      88
16 Equipe 3      86
17 Equipe 3      79
18 Equipe 3      93
19 Equipe 3      79
20 Equipe 3      83
21 Equipe 3      79
> #d'abord la méthode classique
> summary(aov(mydata$Valeurs~mydata$Classes))
               Df Sum Sq Mean Sq F value Pr(>F)
mydata$Classes  2    8.0    4.00   0.136  0.874
Residuals      18  530.3   29.46
>
> #enfin la méthode par permutation qui devrai gérer
> #mieux les outliers et d'autres conditions non satisfaites
> #évidemment on peut faire des ANOVA plus compliquées!
> library("permuco")
> numb_permu=5000
> aovperm(mydata$Valeurs~mydata$Classes,np=numb_permu)
Anova Table
Permutation test using freedman_lane to handle nuisance variables and 5000 permutations.
                  SS df      F parametric P(>F) permutation P(>F)
mydata$Classes   8.0  2 0.1358           0.8739            0.8764
Residuals      530.3 18
>
> library("lmPerm")
> summary(aovp(mydata$Valeurs~mydata$Classes))
[1] "Settings:  unique SS "
Component 1 :
               Df R Sum Sq R Mean Sq Iter Pr(Prob)
mydata$Classes  2     8.00      4.00   92   0.8478
Residuals      18   530.29     29.46

> |
```

Exemple 215.: Test de Hosmer-Lemeshow (HL)

R 3.4.1

L'exemple et la vérification qui suivent sont grandement inspirés du très bon article de Benjami Smith:

https://bensstats.wordpress.com/2021/04/21/robservations-9-the-hosmer-lemeshow-test-follows-a-chi-square-distribution-water-is-wet/

D'abord, comme nous l'avons mentionné dans le cours théorique, il semblerait qu'il n'y ait pas de preuve formelle que:

$$G_{HL}=\sum_{j=1}^{g}\frac{(o_j-e_j)^2}{e_j}$$

Suit une loi du khi-deux du type:

$$G_{HL}\sim\chi^2_{g-2}$$

D'après la lecture de l'article des auteurs de ce test, il semblerait qu'ils soient arrivés à cette conclusions par des simulations numériques. Voyons donc si nous pouvons rejeter cette conclusion ou pas?

D'abord nous faisons des simulations d'une dizaine de milliers de régressions logistiques et nous utilisons la fonction **hoslem.test( )** du package **ResourceSelection** pour stocker les statistiques avec la valeur empirique recommandée de 10 groupes.:

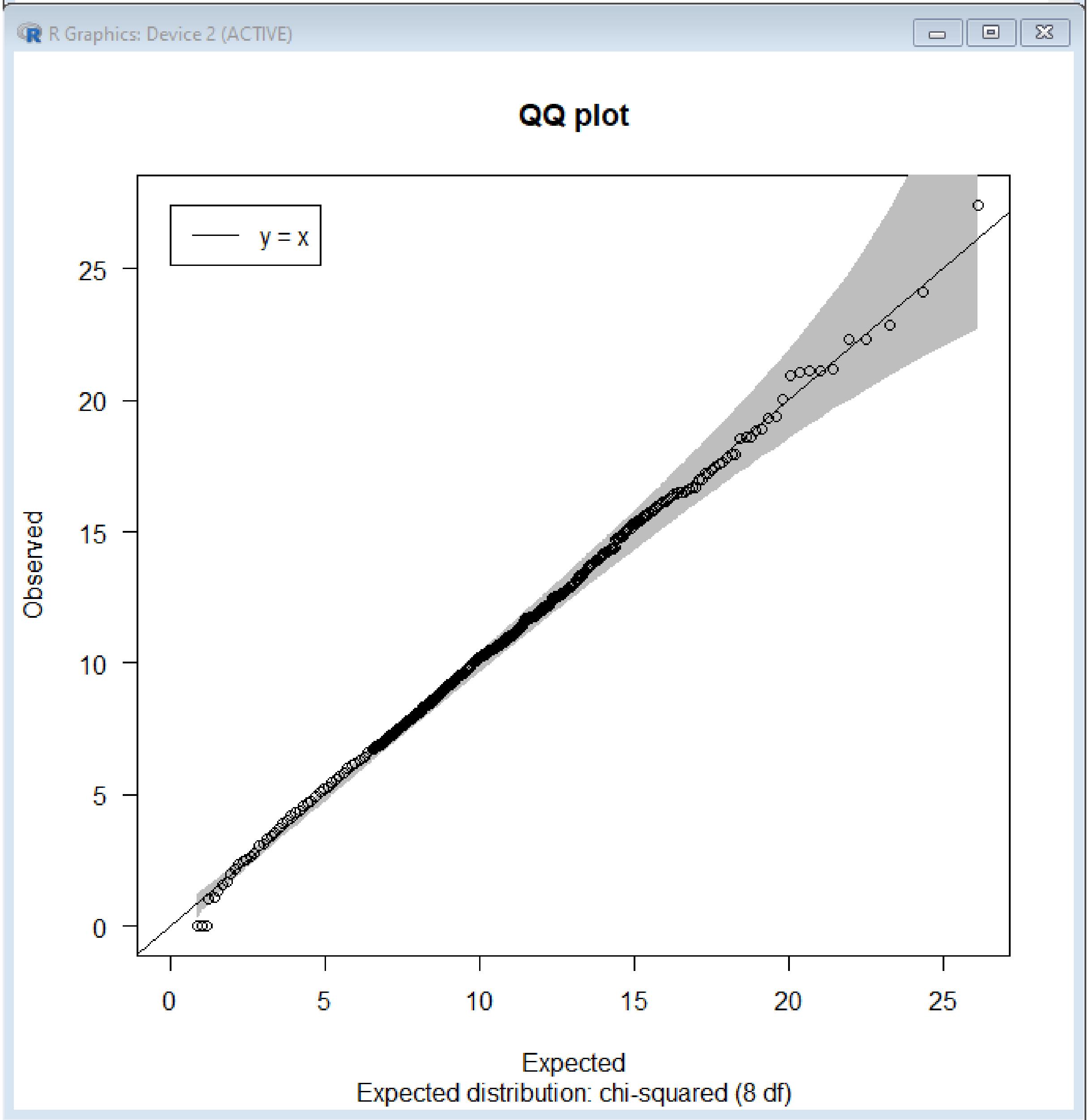
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #pour la reproductibilité
> set.seed(1234)
>
> #on charge un libraire contentant le test de HL
> library("ResourceSelection")
>
>
> #On définit où on stocke les valeurs du test de HL
> hlvals<-c()
>
> # On lance la simulation
> for (i in 10:1000){
+
+   #Taille de l'échantillon
+   n<-i
+   #Rendre les données dépendantes les unes des autres
+   x1<-rnorm(n)
+   x2<-0.5*x1+rnorm(n)
+   xb<-x1-x2
+
+   #On crée la fonction de lien logistique
+   pr<- exp(xb)/(1+exp(xb))
+
+   #On crée une variable de réponse au hasard
+   y <- 1*(runif(n)<pr)
+
+   #On crée un data frame pour le passer à la régression logistique
+   dt<- data.frame(y,x1,x2)
+
+   #La régression logistique habituelle
+   model<-glm(y~x1+x2, data=dt, family=binomial(link="logit"))
+
+   #On exécute le test HL
+   hl<-ResourceSelection::hoslem.test(model$y,fitted(model),g=10)
+
+   #On sauvegarde le résultat du test de HL avec l'indexation adaptée
+   hlvals[i-9]<-hl$statistic
+ }
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
3: glm.fit: algorithm did not converge
4: glm.fit: fitted probabilities numerically 0 or 1 occurred
5: glm.fit: algorithm did not converge
6: glm.fit: fitted probabilities numerically 0 or 1 occurred
> |
```

Pour créer un graphique Q-Q pour la distribution du khi-deux, nous utilisons le package **snpStats**[25] et sa fonction **qq.chisq( )**:

[25] Parce que ce package n'est pas disponible dans les versions actuelles de R si nécessaire, nous installons le package directement depuis Github avec la commande devtools::install_github("NikNakk/snpStats")

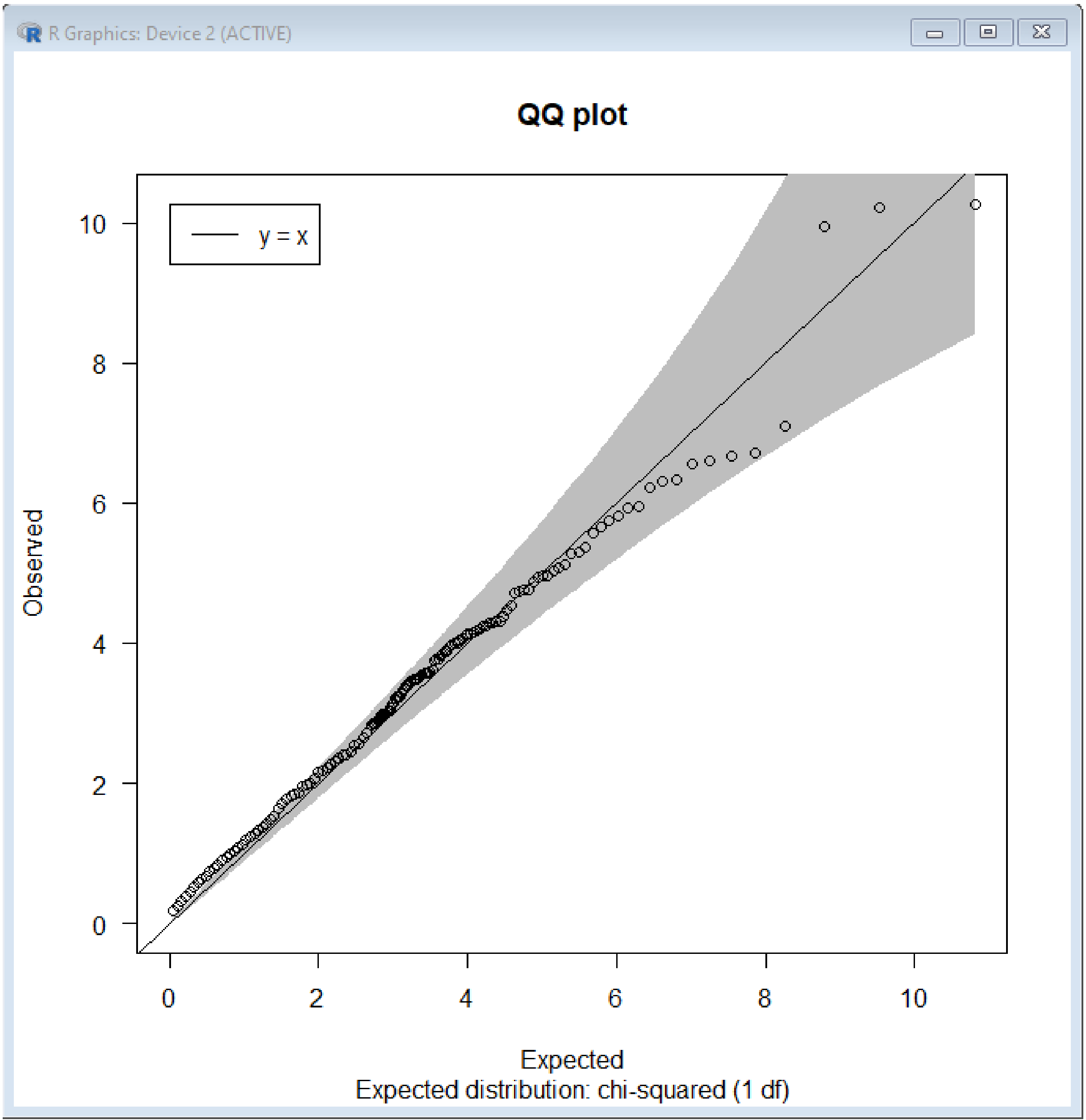
```
> library("snpStats")
> # g= 10 donc les ddl sont de 8
> qq.chisq(hlvals,df=8,slope.one = TRUE)
         N    omitted     lambda
991.000000   0.000000   1.035931
>
```



Il semblerait donc que nous n'allons pas rejeter l'hypothèse nulle comme quoi:

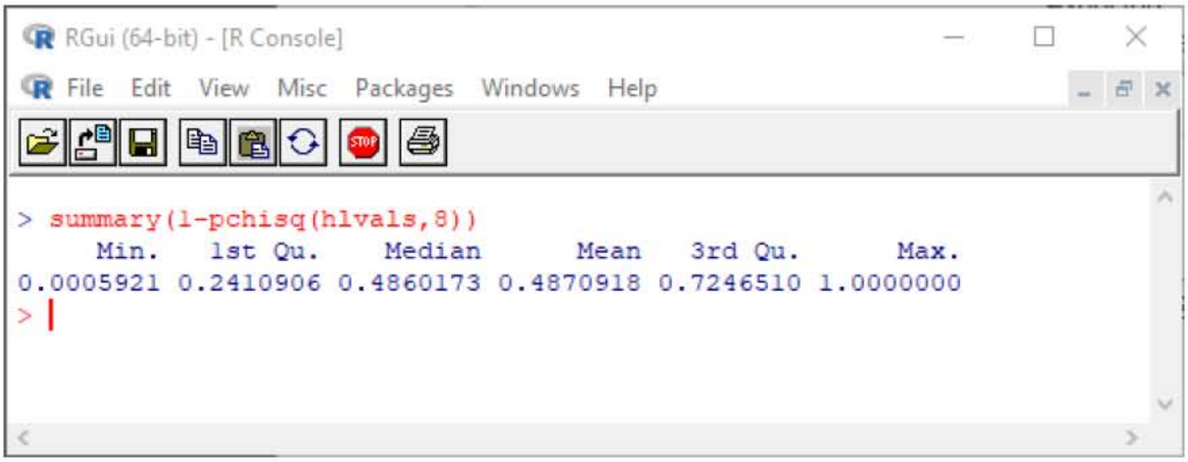
$$G_{HL} \sim \chi^2_{g-2}$$

Le lecteur peut s'amuser avec le cas limite $g > (p+1)$ c'est-à-dire $g > 2$ puis ensuite en adaptant df=1 dans le code précédent pour voir que nous obtenons:



Ce qui reste pas mauvais…

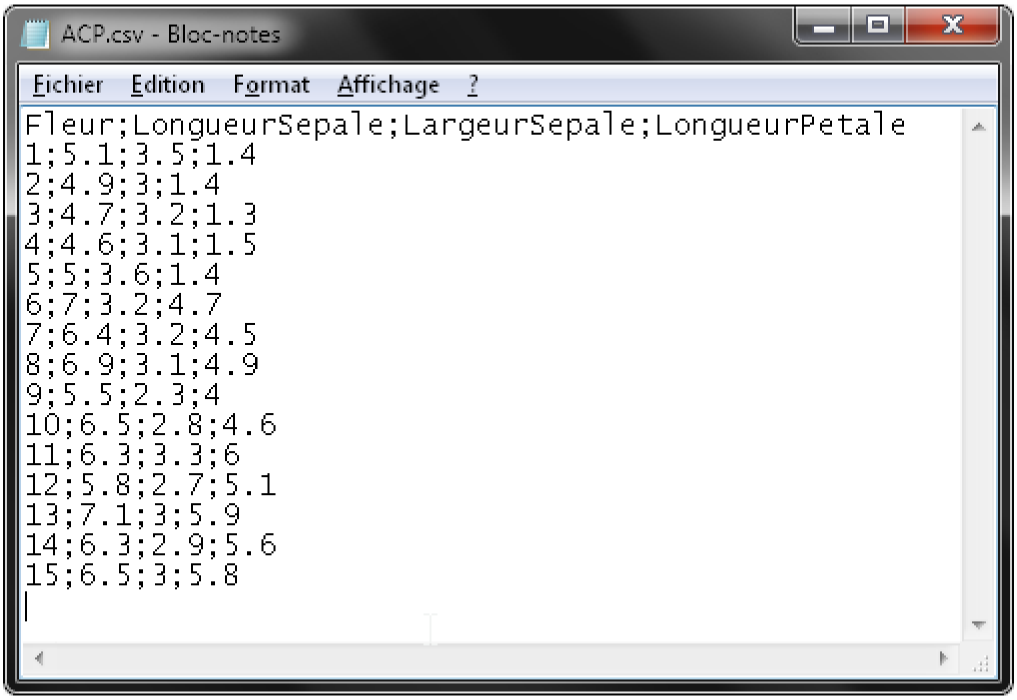
La statistique du test HL ne prend pas en compte le surapprentissage et a tendance à avoir une faible puissance. Cela est évident si nous regardons le résumé des *p*-valeurs des statistiques HL car la moyenne et la médiane sont très faibles:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> summary(1-pchisq(hlvals,8))
Min. 1st Qu. Median Mean 3rd Qu. Max.
0.0005921 0.2410906 0.4860173 0.4870918 0.7246510 1.0000000
>

Exemple 216.: ACP (Analyses en Composantes Principales) paramétrique

R 3.0.2

Nous allons ici vérifier que nous retrouvons les résultats de Minitab et obtenus avec Microsoft Excel avec les relations obtenues lors des démonstrations mathématiques de la méthode dans le cours théorique. Nous reprenons donc les données partielles des Iris de Fisher (histoire de faire original...):



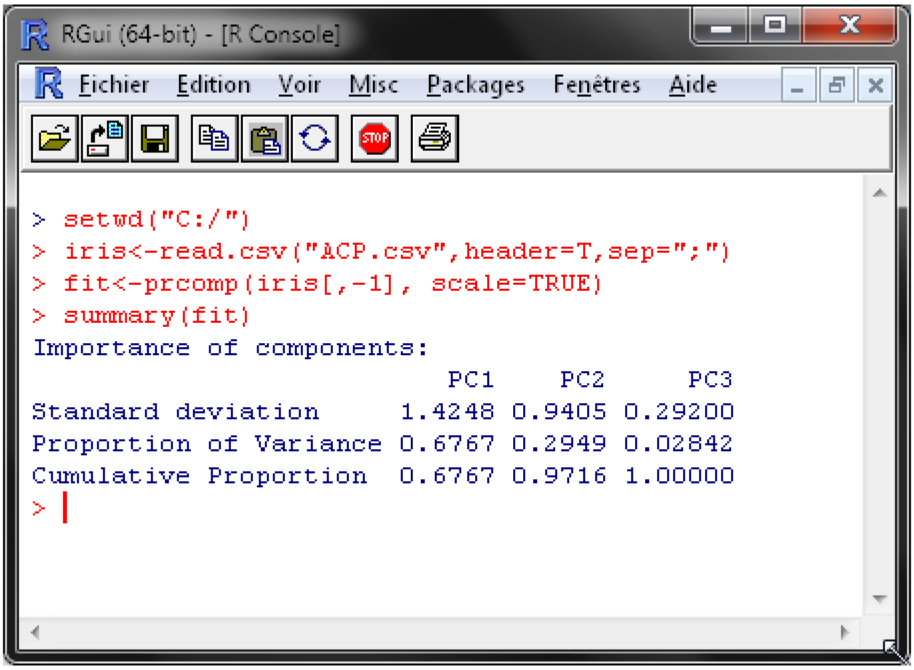
ACP.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
Fleur;LongueurSepale;LargeurSepale;LongueurPetale
1;5.1;3.5;1.4
2;4.9;3;1.4
3;4.7;3.2;1.3
4;4.6;3.1;1.5
5;5;3.6;1.4
6;7;3.2;4.7
7;6.4;3.2;4.5
8;6.9;3.1;4.9
9;5.5;2.3;4
10;6.5;2.8;4.6
11;6.3;3.3;6
12;5.8;2.7;5.1
13;7.1;3;5.9
14;6.3;2.9;5.6
15;6.5;3;5.8
```

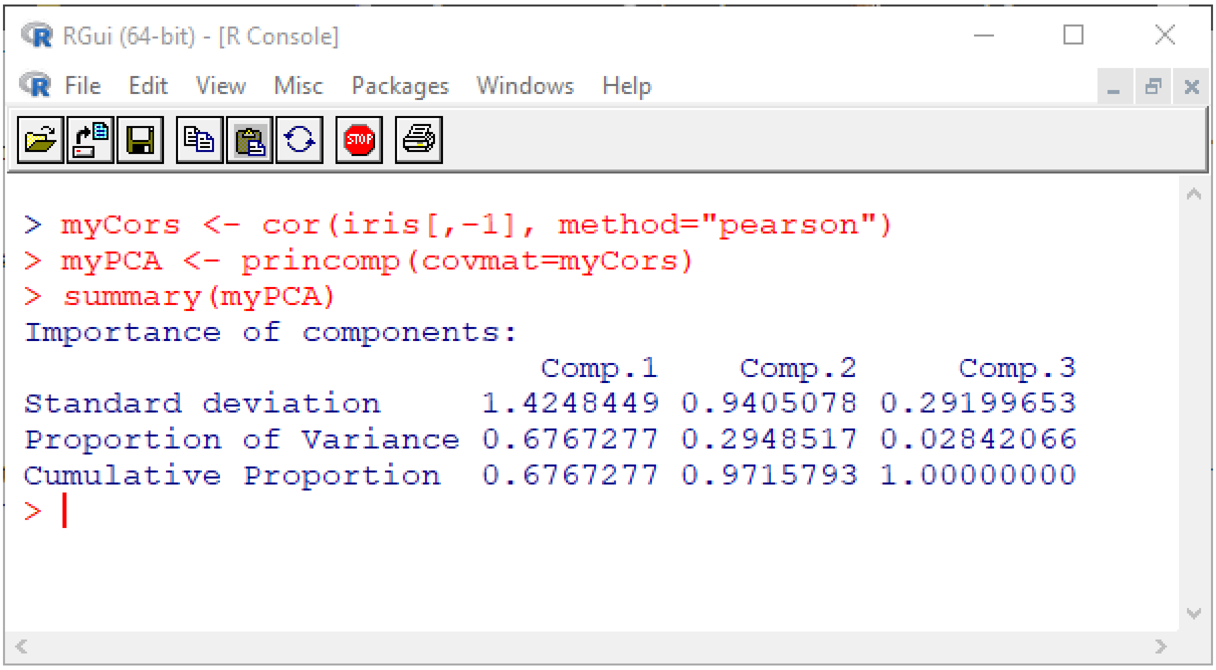
Remarque: Rappelons que l'ACP est très utilisée aussi en finance!

Nous avons donc en utilisant la commande **prcomp( )** qui exécute une ACP basée sur la décomposition en valeurs singulières SVD (d'après la documentation):



Donc à part la première ligne que nous n'avons pas calculée dans le cours théorique (et que Minitab ne donne pas), le reste est parfaitement conforme aux calculs faits à la main et dans Microsoft Excel ainsi que dans Minitab.

Nous pouvons obtenir le même résultat avec la fonction **princomp( )** qui exécute une ACP basée sur le théorème spectral (moins stable numériquement que la SVD):



Ensuite, nous pouvons plotter les valeurs propres ("scree plot[26]"):

[26] Pour rappel il s'agit d'un "scree plot" (en anglais "scree" se sont les éboulis qui descendent des montagnes).

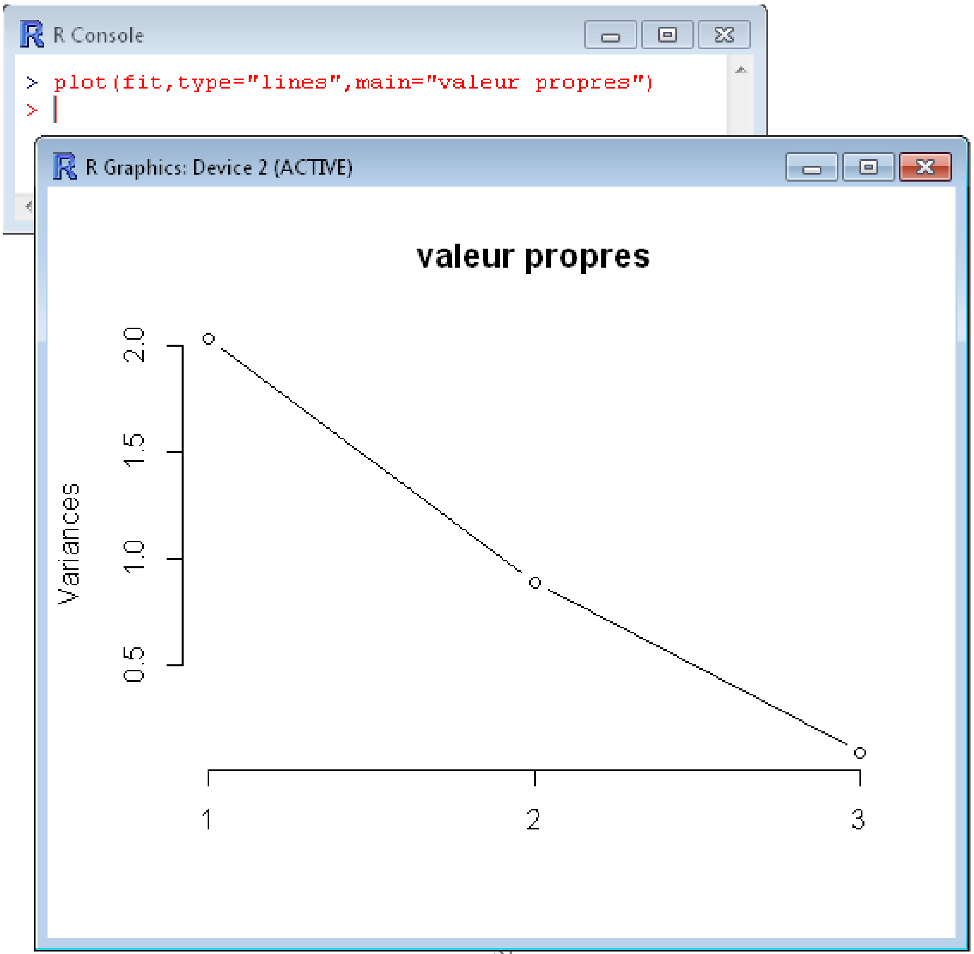
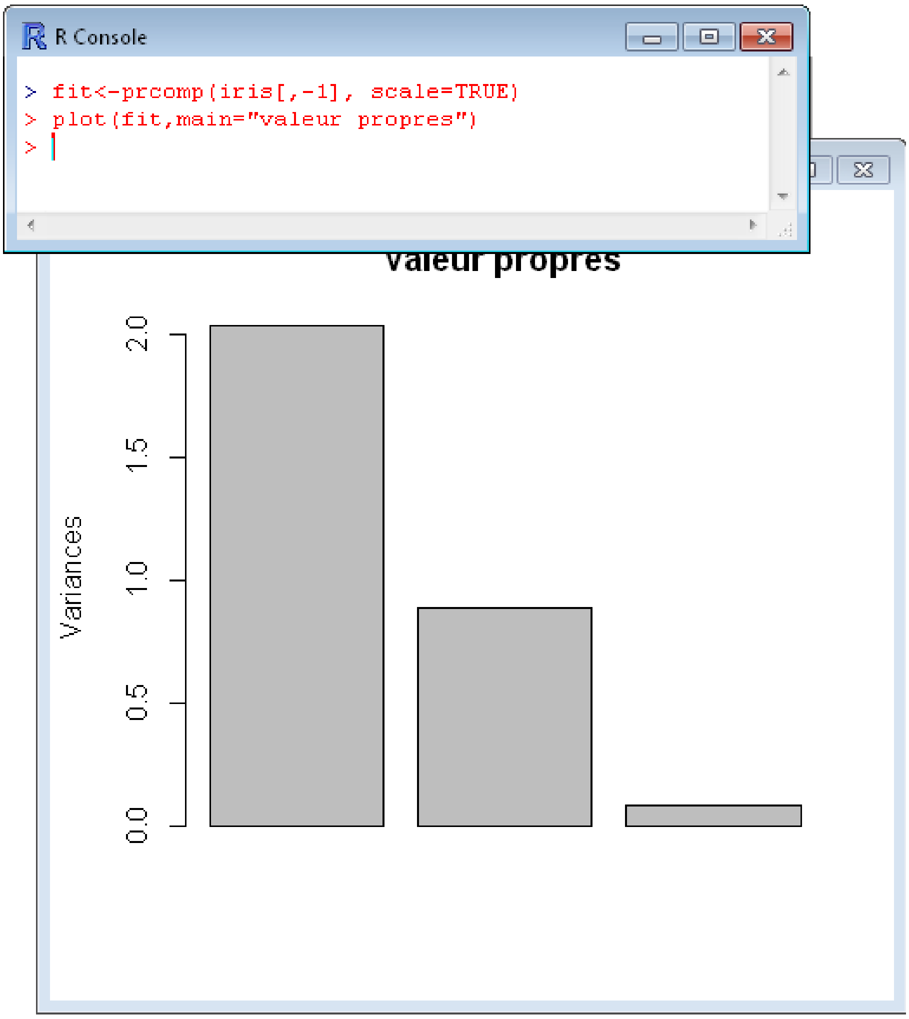


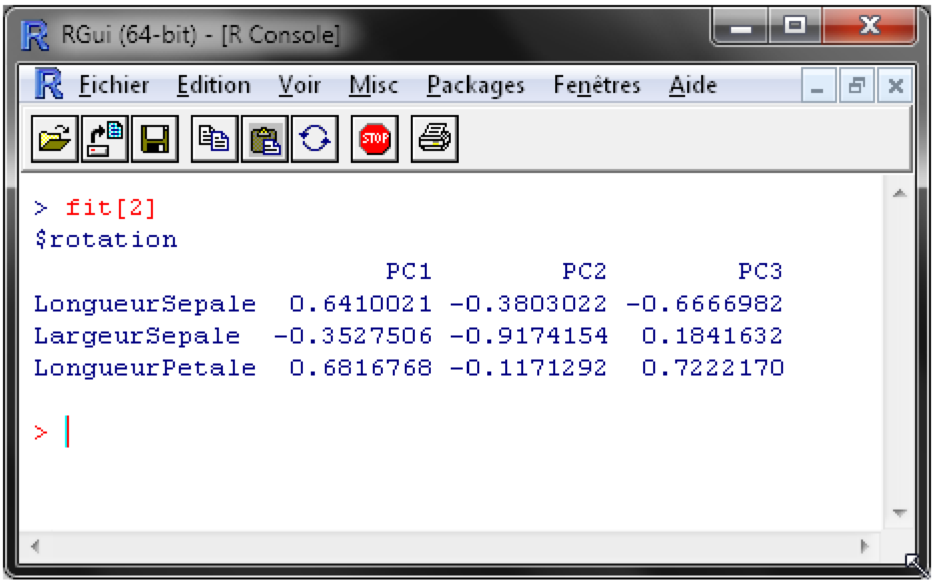
Figure 231 Graph "scree plot" des valeurs propres de l'ACP

ou plus classiquement:



Ce qui est parfaitement conforme à ce que nous avons obtenu dans les autres cours.

Nous pouvons aussi obtenir la matrice des vecteurs propres (aussi parfaitement conforme):



Nous pouvons également ploter le diagramme de double projection avec la commande **biplot( )** :

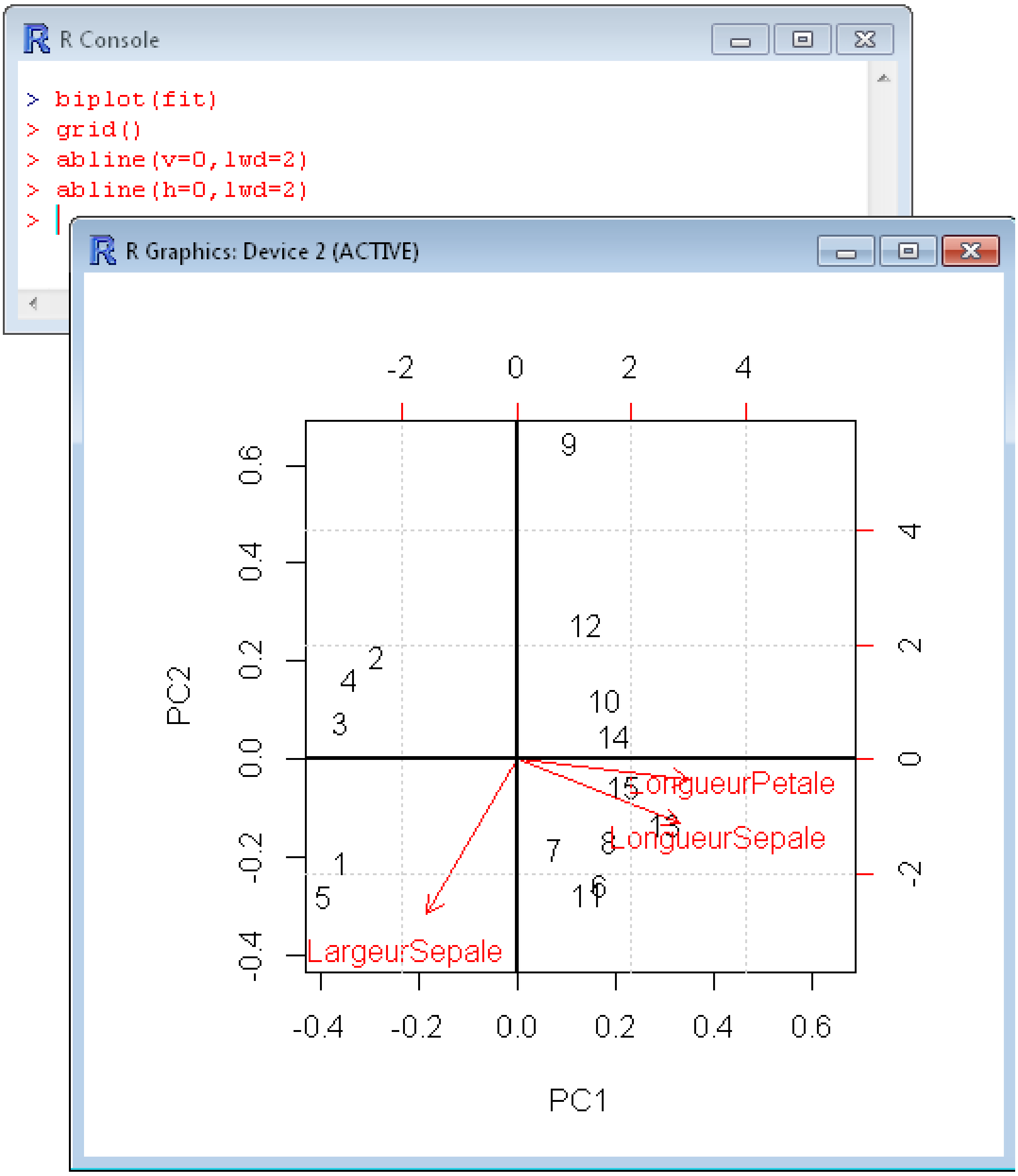
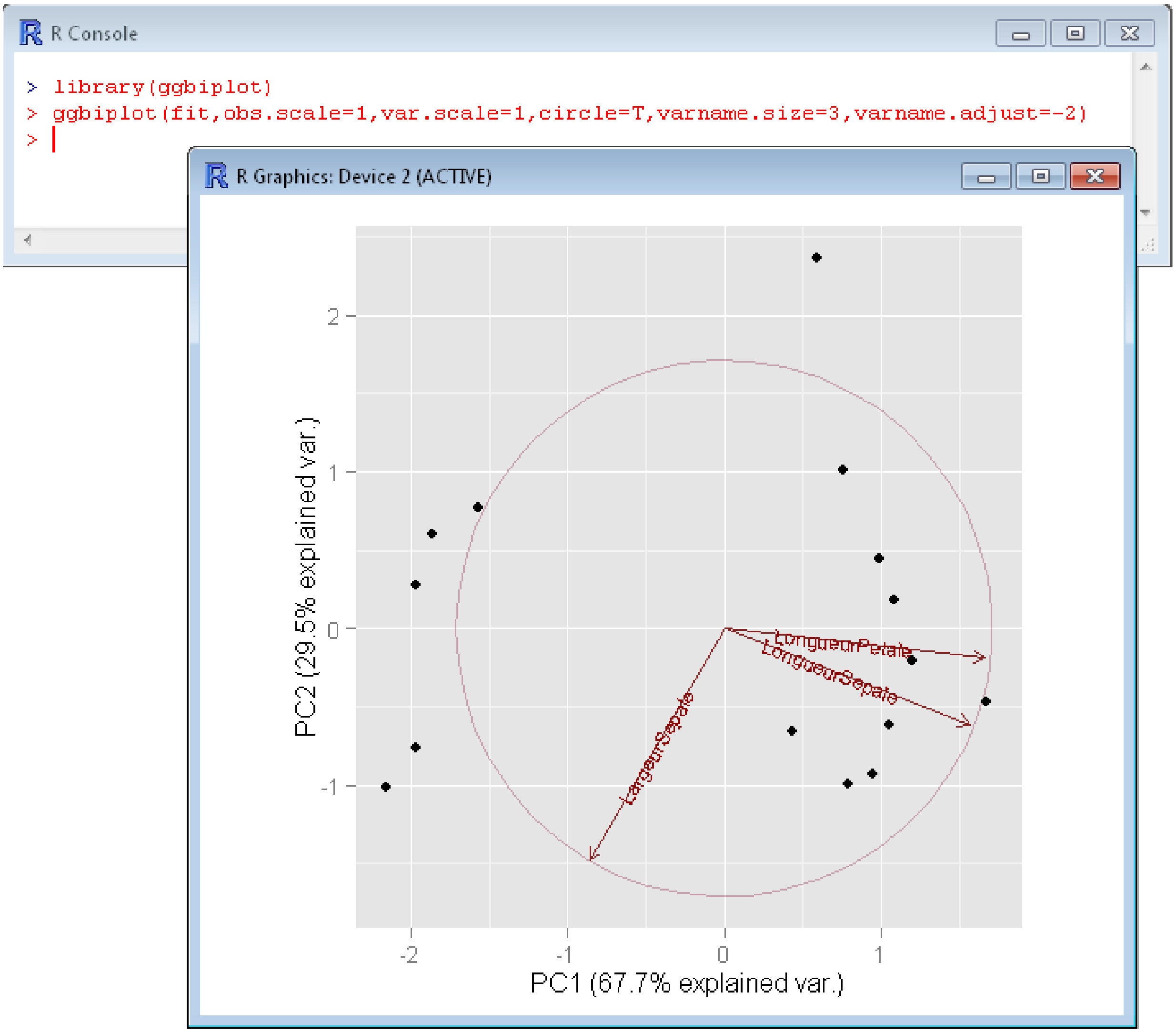


Figure 232 Graph bi-plot de l'ACP

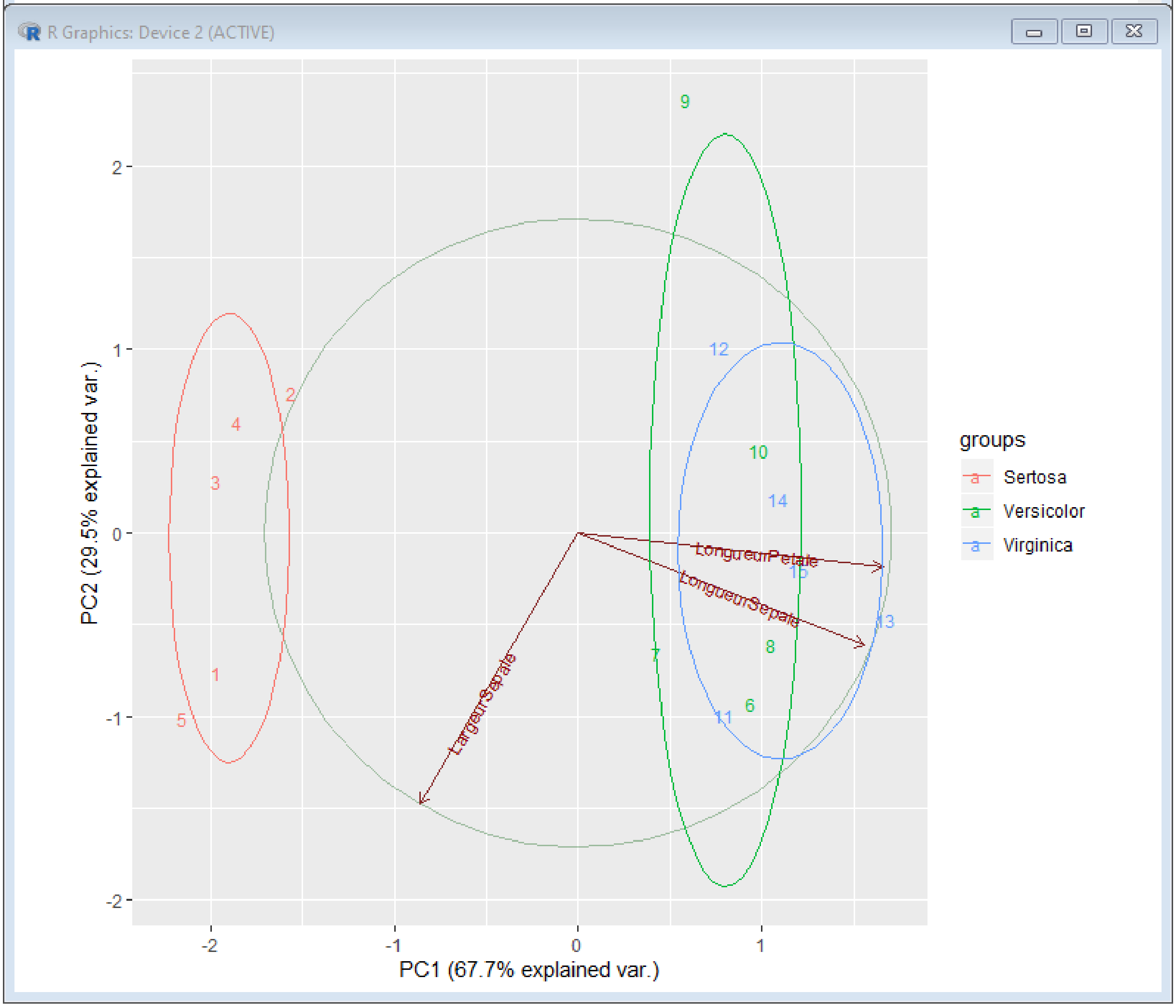
Donc ici, contrairement à Minitab, nous retrouvons les valeurs telles que calculées dans le cours théorique à la main pour les points par contre pour les vecteurs propres cela est différent de ce que nous avons calculé dans le cours théorique ET différent de ce que Minitab ou MATLAB retournent...

Ou dans un autre style avec la commande **ggbiplot( )** package **ggbiplot**:



On peut faire beaucoup plus avec la fonction ggbiplot( )

```
> iris.type<-c(rep("Sertosa",5),rep("Versicolor",5),rep("Virginica",5))
> ggbiplot(fit,obs.scale=1,var.scale=1,ellipse=T,circle=T,varname.size=3,
+ varname.adjust=-2,labels=rownames(iris),groups=iris.type)
>
```

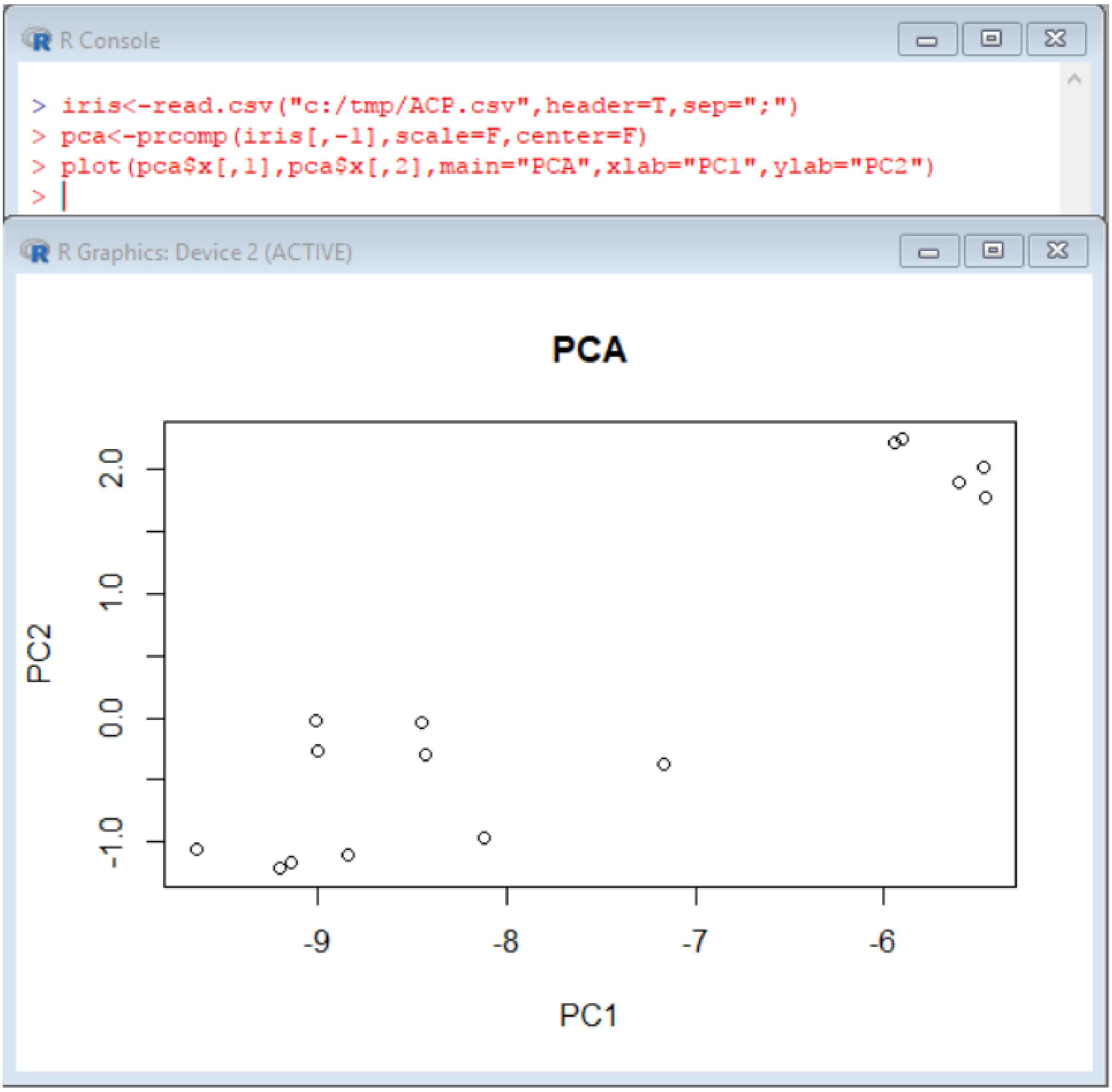


Et encore cela n'est qu'un aperçu!!!

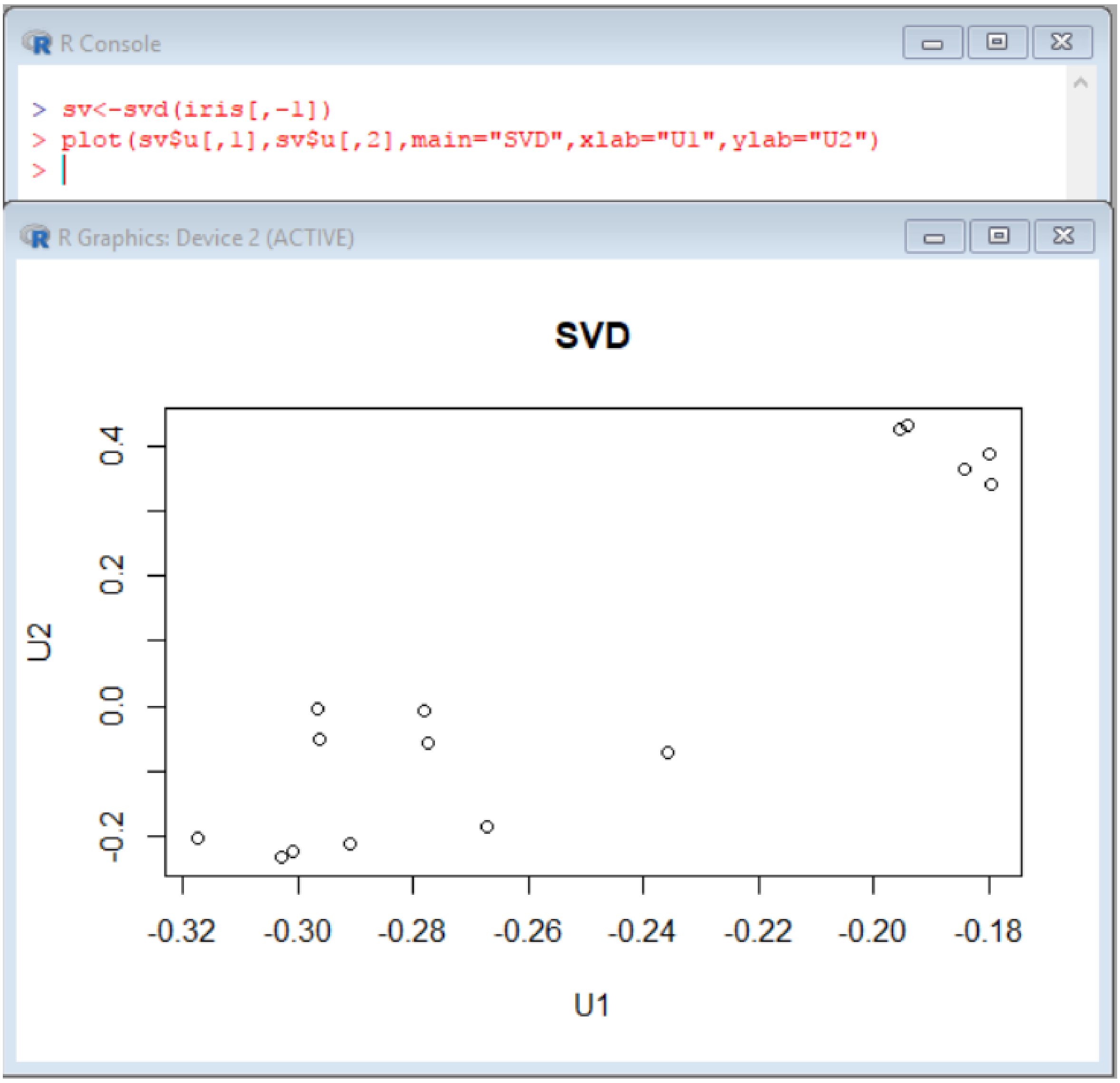
ACP (Analyses en Composantes Principales) paramétrique vs SVD (Décomposition en Valeurs Singulières)

Nous avons vu dans le cours théorique que l'ACP est intimement liée à la SVD et au théorème spectral. Au fait l'ACP étant une SVD sur les données centrées. Voyons cela explicitement avec R!

Donc d'abord revoyons l'ACP identique à celle d'avant à la différence que nous ne centrons (*center*=F) ni réduisons (*scale*=F) les données:



Et maintenant effectuons la décomposition en valeurs singulières:



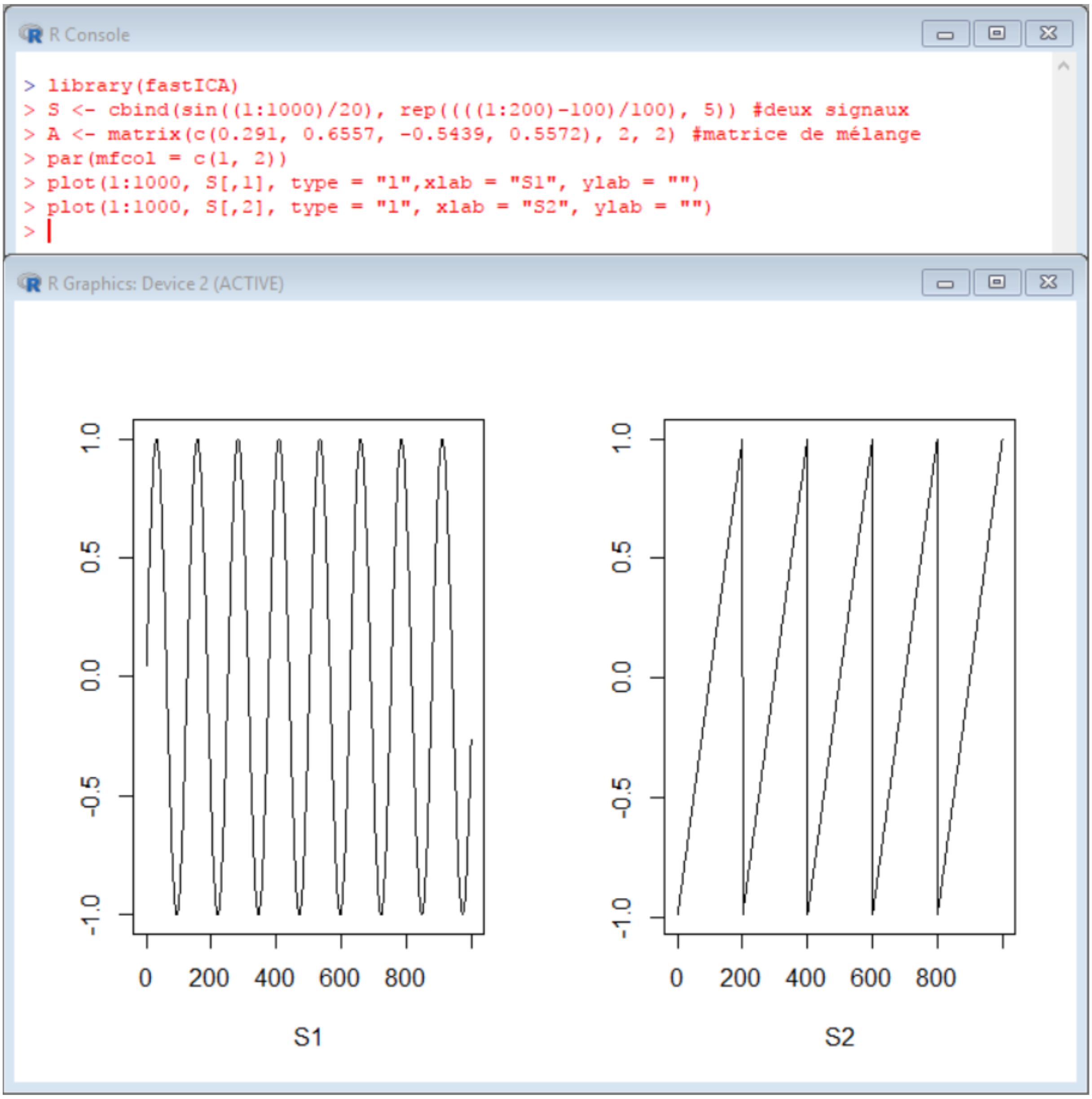
C.Q.F.D. Nous obtenons bien la même chose!

Exemple 217.: Analyse en Composantes Indépendantes (ICA) WP

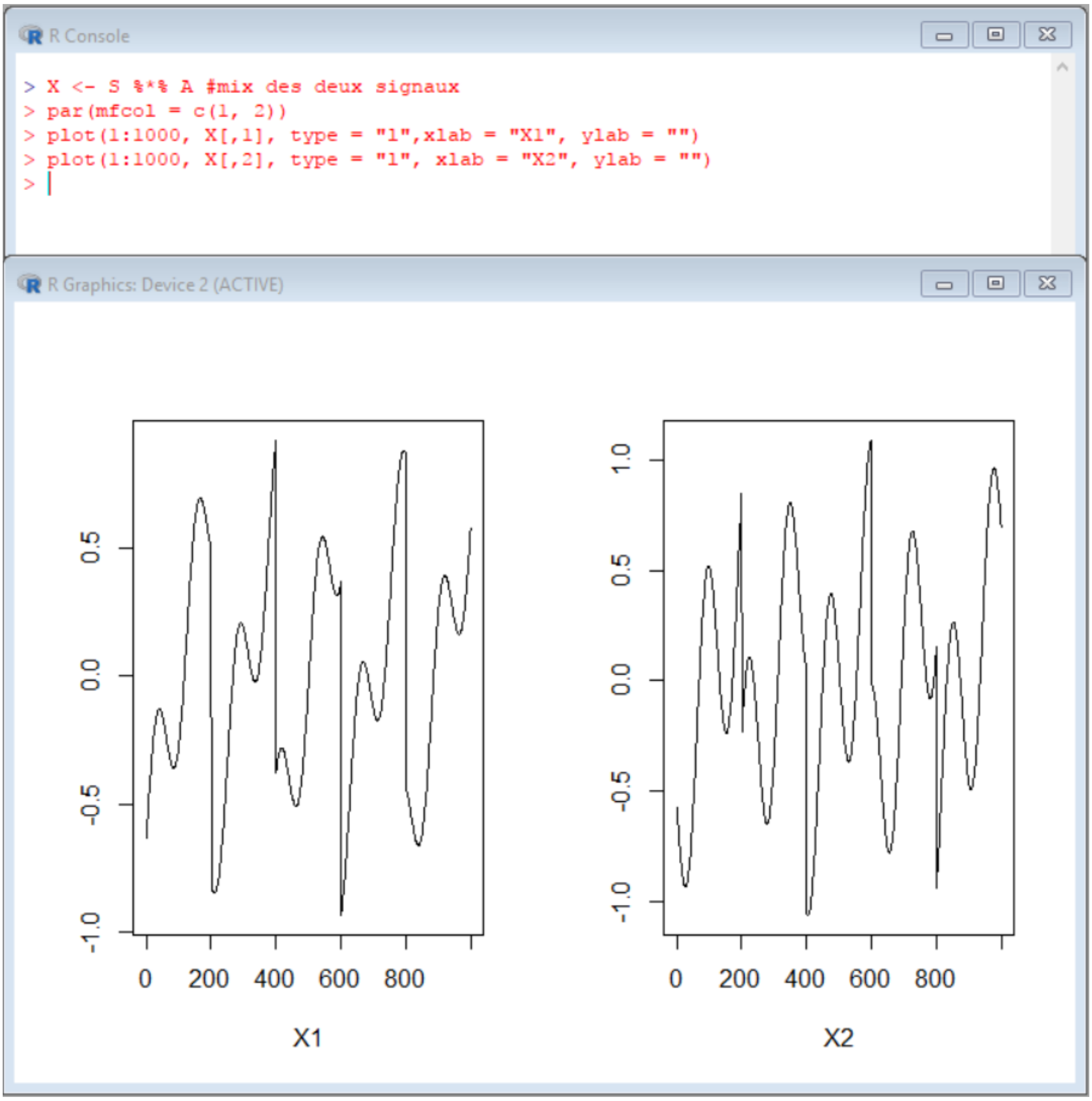
R 3.4.1

Donc voici un algorithme dont nous n'avons pas pu voir la démonstration mathématique dans le cours théorique et qui permet contrairement à l'ACP qui cherche à maximiser la variance, lui cherche à minimiser l'information mutuelle (cf. Théorie de l'information).

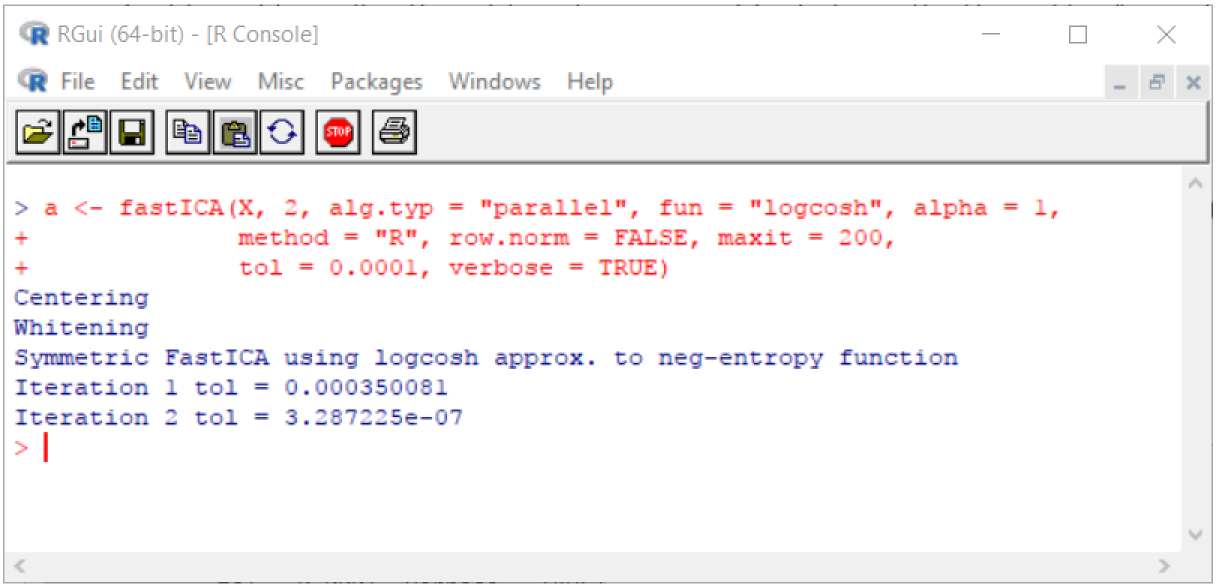
D'abord nous chargeons le package **fastICA** et créons deux signaux ainsi que la matrice de mélange:



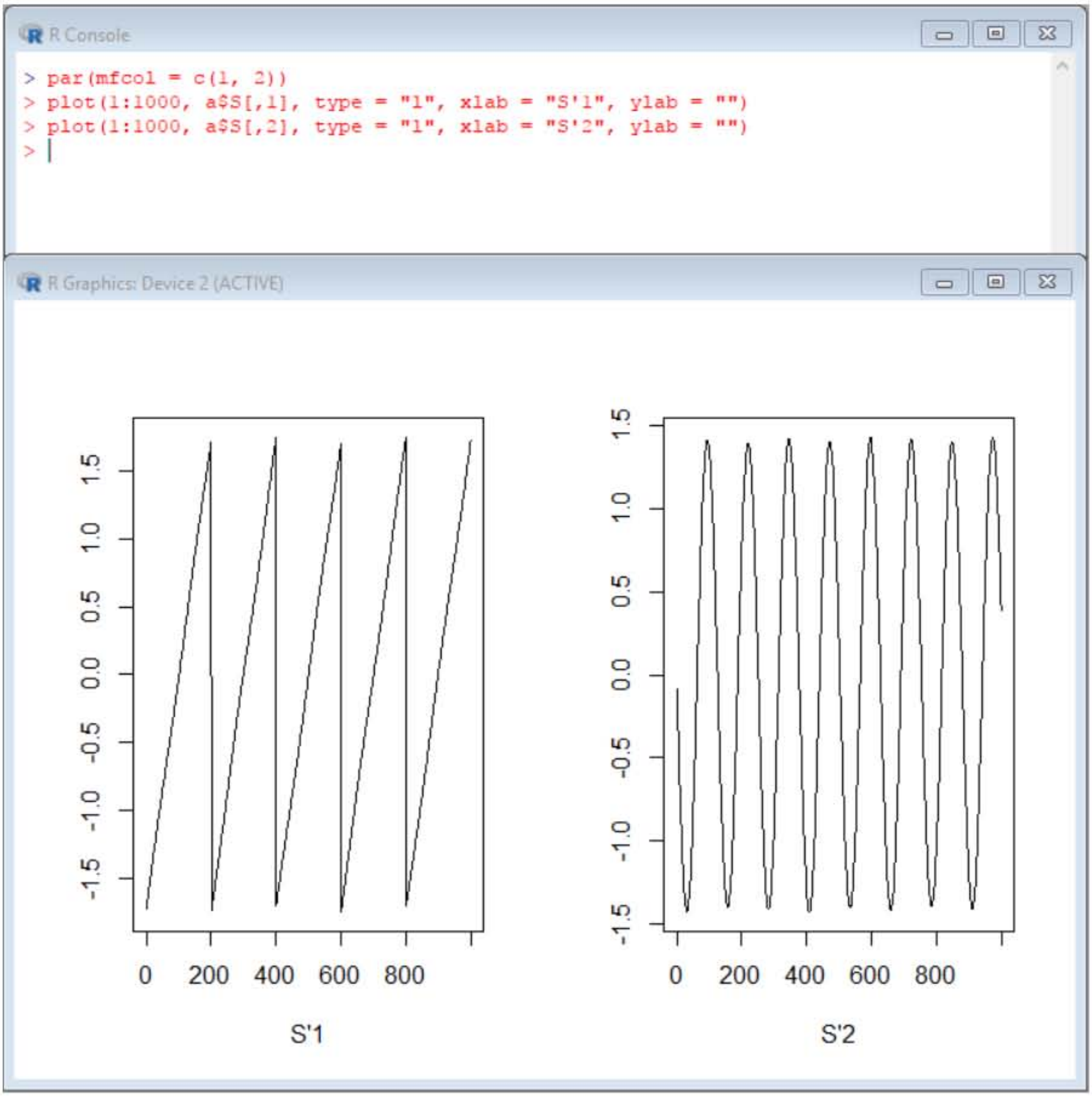
Ensuite nous mixons les deux signaux:



Ensuite on utilise l'ICA pour retrouver au mieux les signaux d'origine avec la fonction **fastICA( )** :



et on affiche le résultat:


R Console
> par(mfcol = c(1, 2))
> plot(1:1000, a$S[,1], type = "l", xlab = "S'1", ylab = "")
> plot(1:1000, a$S[,2], type = "l", xlab = "S'2", ylab = "")
>
R Graphics: Device 2 (ACTIVE)
-1.5 -1.0 -0.5 0.0 0.5 1.0 1.5
0 200 400 600 800
S'1
-1.5 -1.0 -0.5 0.0 0.5 1.0 1.5
0 200 400 600 800
S'2

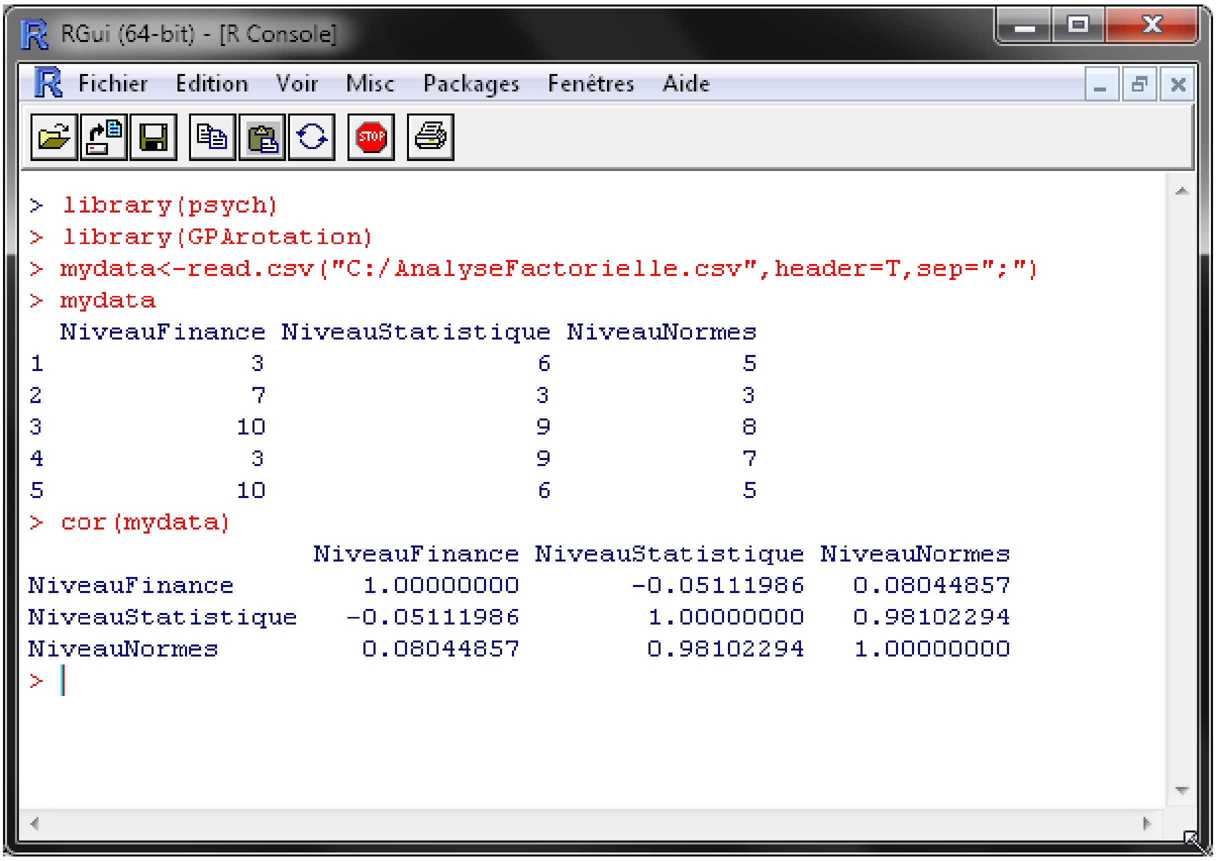
Exemple 218.: Analyse factorielle exploratoire (AFE)

R 3.0.2

Le but va être ici de vérifier comme à l'habitude que les résultats calculés à la main (et donc les conclusions y relatives) sont conformes aux démonstrations mathématiques faites dans le cours théorique avec l'exemple particulier que nous avions choisi ainsi que de vérifier s'ils sont conformes à ce que nous avions dans Microsoft Excel et Minitab.

Nous utiliserons la commande **fa( )** du package **psych** qui nécessite lui-même le chargement du package **GPArotation** pour contrôler le type de rotation.

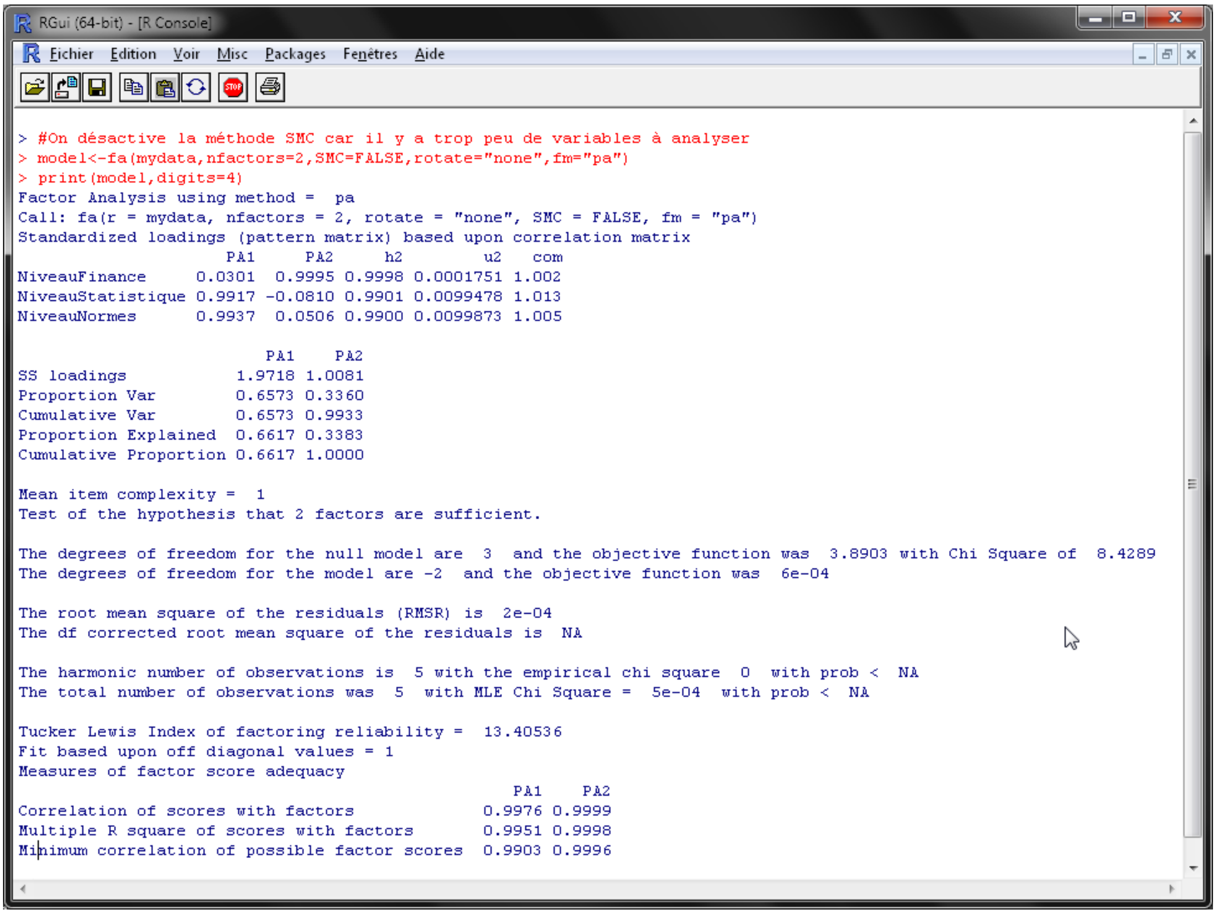
D'abord nous montrons les données que nous utilisons et la matrice de covariance y relative:



```
> library(psych)
> library(GPArotation)
> mydata<-read.csv("C:/AnalyseFactorielle.csv",header=T,sep=";")
> mydata
  NiveauFinance NiveauStatistique NiveauNormes
1             3                 6            5
2             7                 3            3
3            10                 9            8
4             3                 9            7
5            10                 6            5
> cor(mydata)
                  NiveauFinance NiveauStatistique NiveauNormes
NiveauFinance        1.00000000       -0.05111986   0.08044857
NiveauStatistique   -0.05111986        1.00000000   0.98102294
NiveauNormes         0.08044857        0.98102294   1.00000000
>
```

AFE avec méthode ACP sans rotation!

Ensuite, nous lançons l'AFE (ou plus simplement dit "l'analyse factorielle") sans rotation (*rotate="none"*) avec la méthode factorielle ACP (*fm="pa"*):



```
> #On désactive la méthode SMC car il y a trop peu de variables à analyser
> model<-fa(mydata,nfactors=2,SMC=FALSE,rotate="none",fm="pa")
> print(model,digits=4)
Factor Analysis using method =  pa
Call: fa(r = mydata, nfactors = 2, rotate = "none", SMC = FALSE, fm = "pa")
Standardized loadings (pattern matrix) based upon correlation matrix
                     PA1     PA2     h2        u2   com
NiveauFinance     0.0301  0.9995 0.9998 0.0001751 1.002
NiveauStatistique 0.9917 -0.0810 0.9901 0.0099478 1.013
NiveauNormes      0.9937  0.0506 0.9900 0.0099873 1.005

                         PA1    PA2
SS loadings           1.9718 1.0081
Proportion Var        0.6573 0.3360
Cumulative Var        0.6573 0.9933
Proportion Explained  0.6617 0.3383
Cumulative Proportion 0.6617 1.0000

Mean item complexity =  1
Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the null model are  3  and the objective function was  3.8903 with Chi Square of  8.4289
The degrees of freedom for the model are -2  and the objective function was  6e-04

The root mean square of the residuals (RMSR) is  2e-04
The df corrected root mean square of the residuals is  NA

The harmonic number of observations is  5 with the empirical chi square  0  with prob <  NA
The total number of observations was  5  with MLE Chi Square =  5e-04  with prob <  NA

Tucker Lewis Index of factoring reliability =  13.40536
Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                  PA1    PA2
Correlation of scores with factors             0.9976 0.9999
Multiple R square of scores with factors       0.9951 0.9998
Minimum correlation of possible factor scores  0.9903 0.9996
```

Nous retrouvons donc presque les mêmes résultats que dans le cours théorique et Minitab/MS Excel. C'est principalement la communalité qui diffère et scores des facteurs à quelques pourcents.

Par contre nous n'avons pas démontré ni parlé dans le cours théorique de toutes les informations données en-dessous des deux tableaux ni de ce paramètre *u*2 que renvoie R et qui semble (mais à vérifier...) être simplement 1-*h*2. De plus les noms choisis par R pour les colonnes ne sont pas des plus subtils... Il en sera de même pour les deux autres méthodes que nous allons voir ci-dessous et qui ont toutes deux décortiquées dans le cours théorique.

AFE avec méthode ACP et rotation VariMax!

Maintenant lançons l'AFE (ou plus simplement dit "l'analyse factorielle") avec rotation varimax (*rotate="varimax"*) toujours avec la méthode factorielle ACP (*fm="pa"*):

```
> model<-fa(mydata,nfactors=2,SMC=FALSE,rotate="varimax",fm="pa")
> print(model,digits=4)
Factor Analysis using method =  pa
Call: fa(r = mydata, nfactors = 2, rotate = "varimax", SMC = FALSE,
    fm = "pa")
Standardized loadings (pattern matrix) based upon correlation matrix
                     PA1     PA2     h2        u2   com
NiveauFinance     0.0165  0.9998 0.9998 0.0001751 1.001
NiveauStatistique 0.9927 -0.0675 0.9901 0.0099478 1.009
NiveauNormes      0.9929  0.0642 0.9900 0.0099873 1.008


                         PA1    PA2
SS loadings           1.9717 1.0082
Proportion Var        0.6572 0.3361
Cumulative Var        0.6572 0.9933
Proportion Explained  0.6617 0.3383
Cumulative Proportion 0.6617 1.0000

Mean item complexity =  1
Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the null model are  3  and the objective function was  3.8903 with Chi Square of  8.4289
The degrees of freedom for the model are -2  and the objective function was  6e-04

The root mean square of the residuals (RMSR) is  2e-04
The df corrected root mean square of the residuals is  NA

The harmonic number of observations is  5 with the empirical chi square  0  with prob <  NA
The total number of observations was  5  with MLE Chi Square =  5e-04  with prob <  NA

Tucker Lewis Index of factoring reliability =  13.40536
Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                 PA1    PA2
Correlation of scores with factors            0.9976 0.9999
Multiple R square of scores with factors      0.9951 0.9998
Minimum correlation of possible factor scores 0.9903 0.9996
```

Donc encore une fois à quelques % près non significatif, nous pouvons considérer les résultats identiques à ceux de Minitab et Microsoft Excel ainsi qu'à ceux obtenus dans le cours théorique.

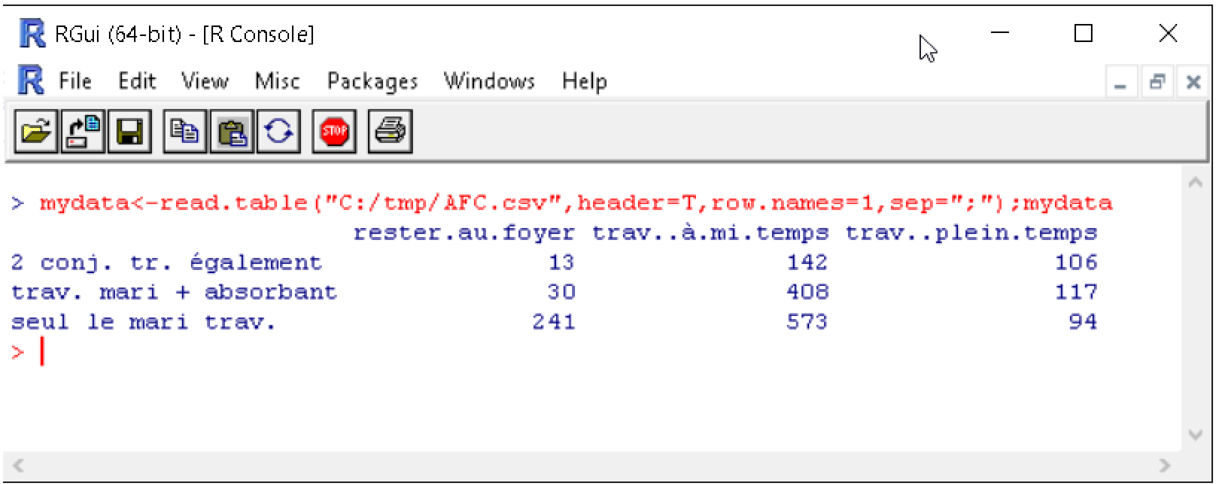
AFC Analyse Factorielle des Correspondances

Nous allons ici reprendre les valeurs numériques du cours théoriques dont pour rappel la démonstration mathématique et l'origine proviennent entièrement de l'ouvrage *Analyse de données avec R* de F. Husson, S. Lê et J. Pagès (ISBN: 978-2-7535-0938-2).

Nous partons du fichier suivant:



Nous chargeons les données en tant que table et nous retrouvons bien la table du cours théorique:



Le test du chi-2 est donné par:

```
> res.test.chi2<-chisq.test(mydata[,1:3])
> res.test.chi2

        Pearson's Chi-squared test

data:  mydata[, 1:3]
X-squared = 233.43, df = 4, p-value < 2.2e-16

> round(res.test.chi2$expected,1)
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également             43.0            170.0              48.0
trav. mari + absorbant            91.4            361.5             102.1
seul le mari trav.               149.6            591.5             167.0
> round(res.test.chi2$residuals^2,2)
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également            20.93             4.62             70.12
trav. mari + absorbant           41.27             5.98              2.19
seul le mari trav.               55.88             0.58             31.88
> round(100*res.test.chi2$residuals^2/res.test.chi2$stat,2)
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également             8.96             1.98             30.04
trav. mari + absorbant           17.68             2.56              0.94
seul le mari trav.               23.94             0.25             13.66
> 
```

Ensuite pour les profils lignes nous avons:

```
> dd<-rbind(mydata,apply(mydata[,1:3],2,sum))
> dd
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également               13              142               106
trav. mari + absorbant              30              408               117
seul le mari trav.                 241              573                94
4                                  284             1123               317
> rownames(dd)[4]<-"Profil moyen"
> dd
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également               13              142               106
trav. mari + absorbant              30              408               117
seul le mari trav.                 241              573                94
Profil moyen                       284             1123               317
> round(prop.table(as.matrix(dd),margin=1),3)
                       rester.au.foyer trav..à.mi.temps trav..plein.temps
2 conj. tr. également            0.050            0.544             0.406
trav. mari + absorbant           0.054            0.735             0.211
seul le mari trav.               0.265            0.631             0.104
Profil moyen                     0.165            0.651             0.184
>
```

et les profils colonnes:

```
> dd<-cbind(mydata,apply(mydata[,1:3],1,sum))
> dd
                       rester.au.foyer trav..à.mi.temps trav..plein.temps apply(mydata[, 1:3], 1, sum)
2 conj. tr. également               13              142               106                          261
trav. mari + absorbant              30              408               117                          555
seul le mari trav.                 241              573                94                          908
> colnames(dd)[4]<-"Profil moyen"
> dd
                       rester.au.foyer trav..à.mi.temps trav..plein.temps Profil moyen
2 conj. tr. également               13              142               106          261
trav. mari + absorbant              30              408               117          555
seul le mari trav.                 241              573                94          908
> round(prop.table(as.matrix(dd),margin=2),3)
                       rester.au.foyer trav..à.mi.temps trav..plein.temps Profil moyen
2 conj. tr. également            0.046            0.126             0.334        0.151
trav. mari + absorbant           0.106            0.363             0.369        0.322
seul le mari trav.               0.849            0.510             0.297        0.527
>
```

Nous faisons maintenant le plot de l'AFC avec la fonction **ca( )** du package **FactoMineR**:

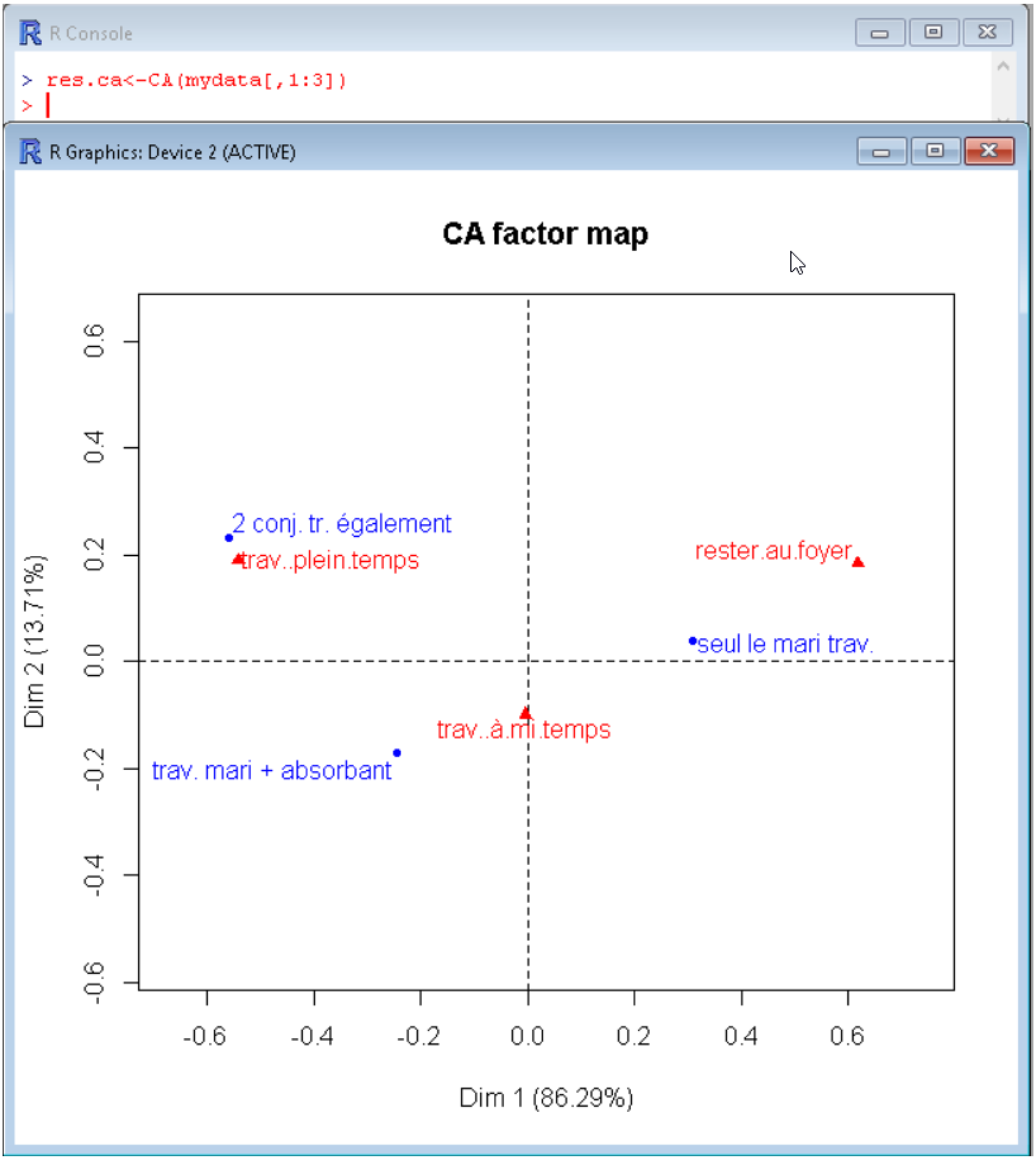
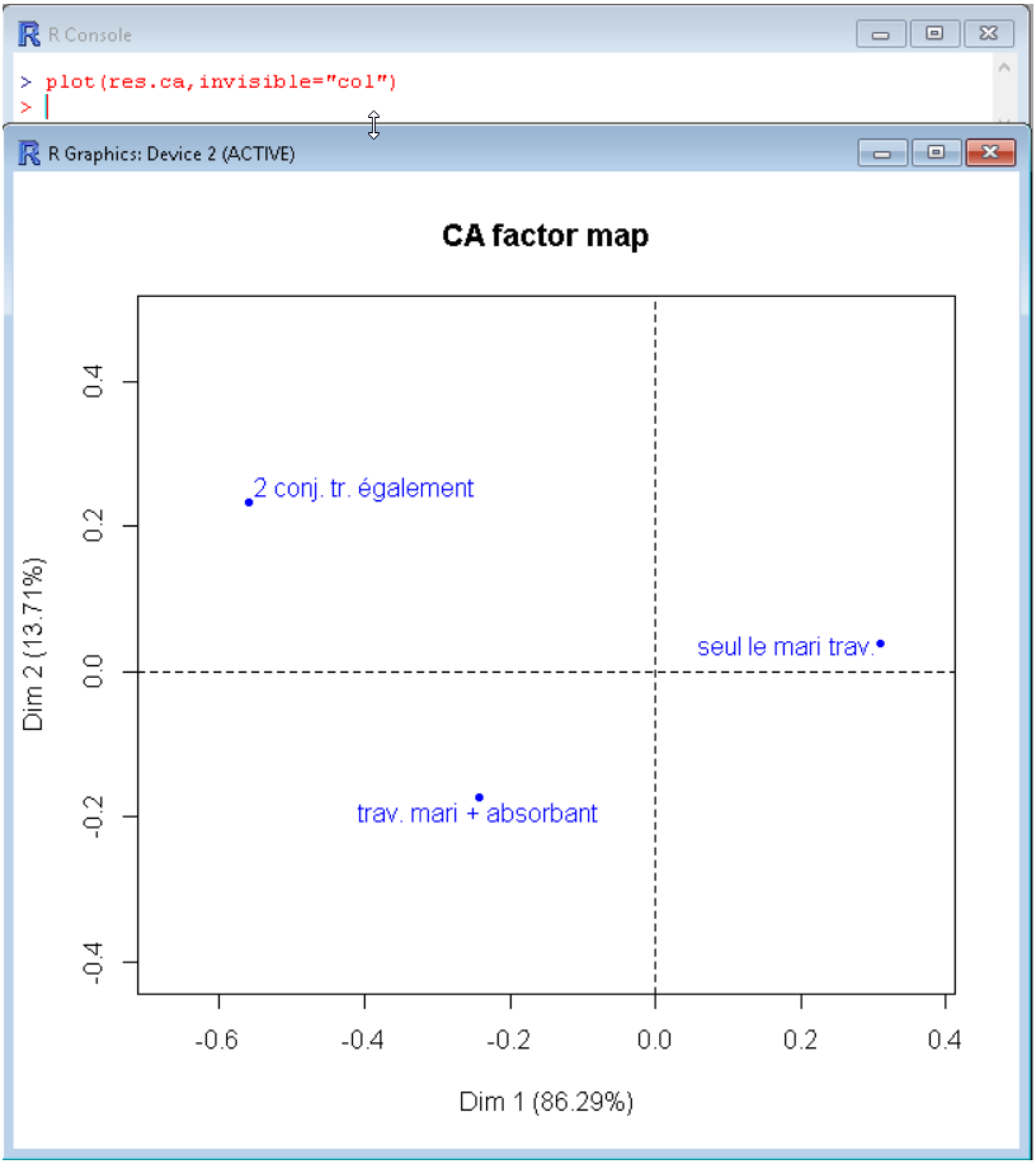
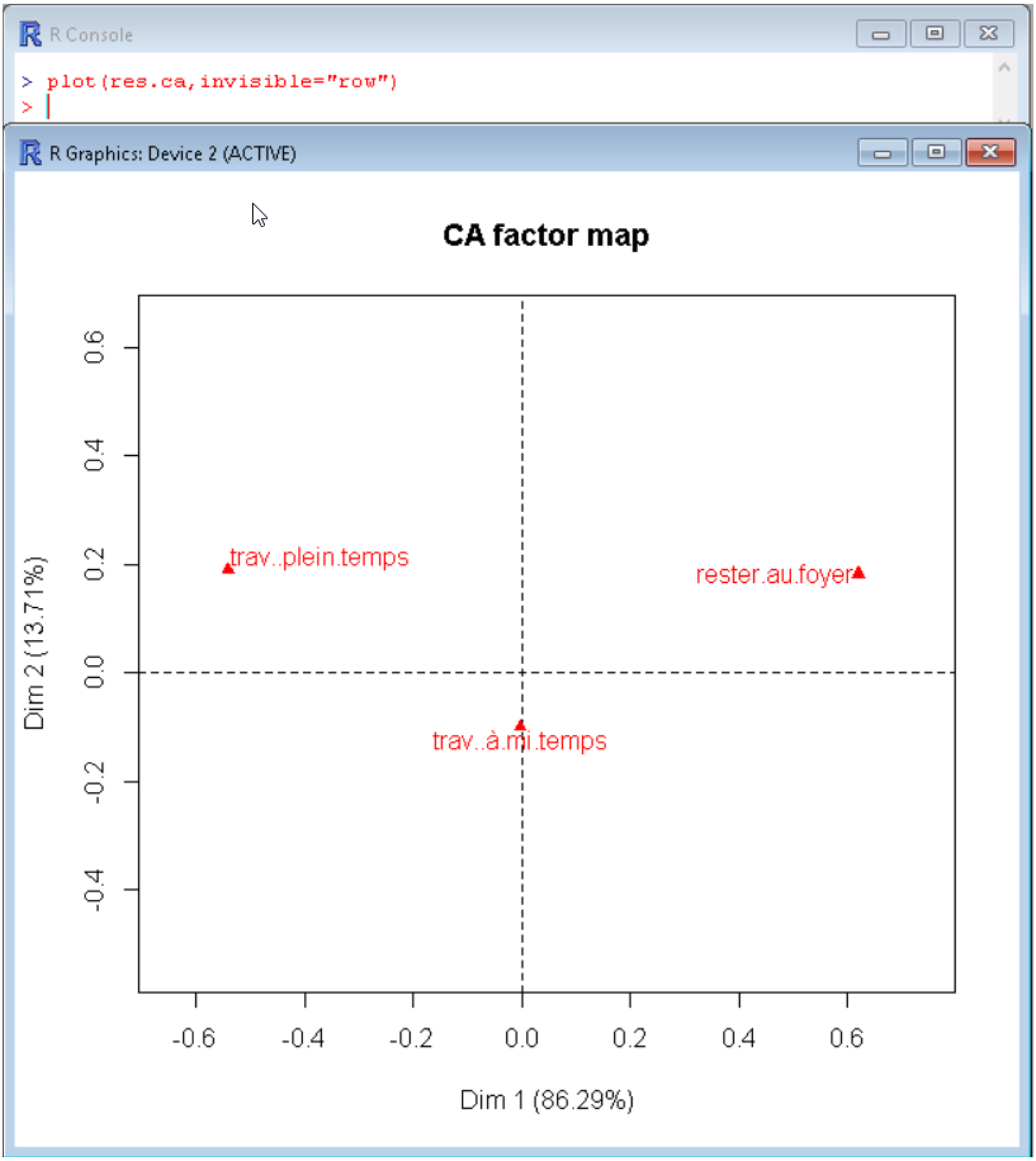


Figure 233 Graph de carte de l'analyse factorielle des correspondances

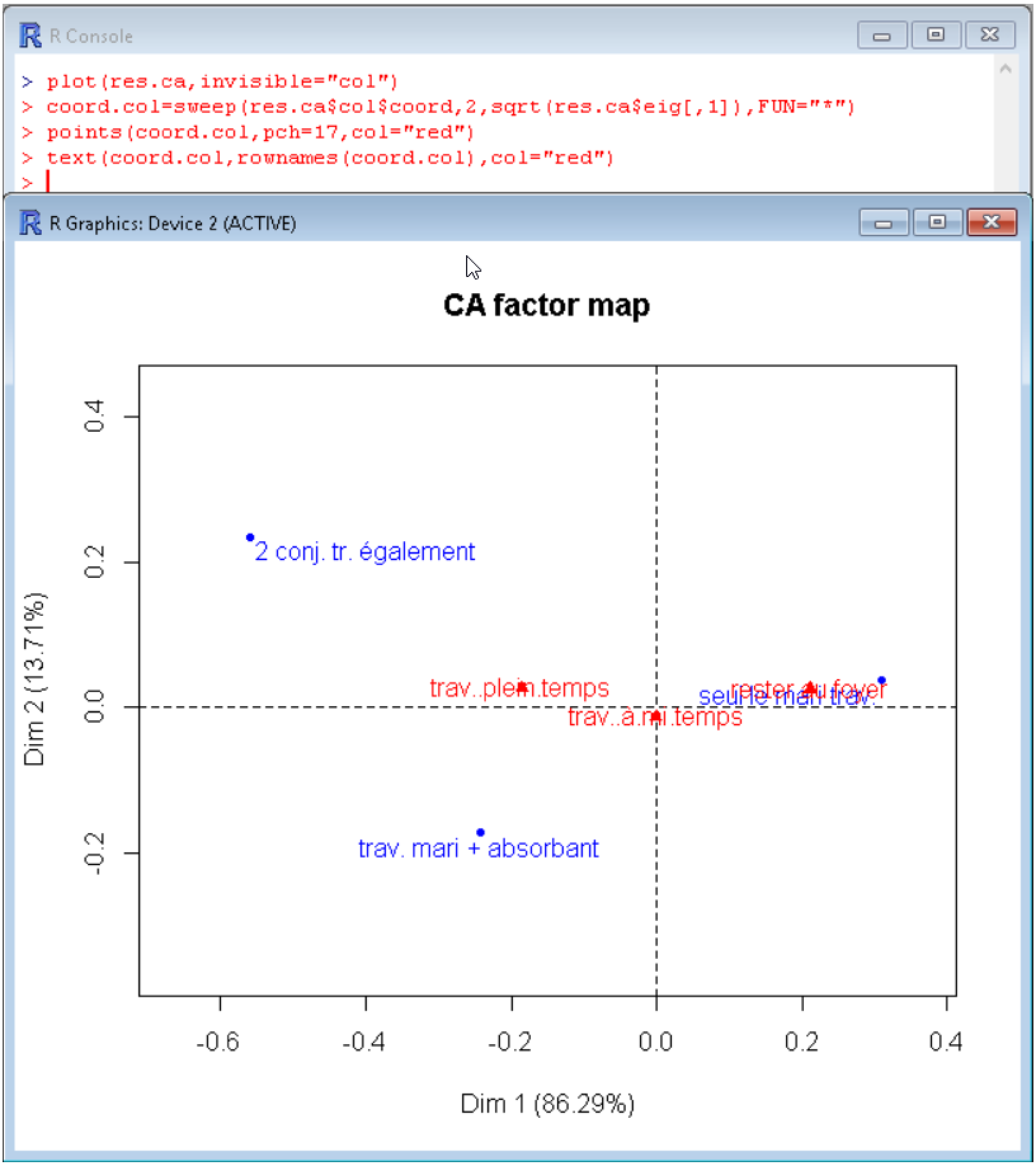
On peut masquer les données de colonnes:



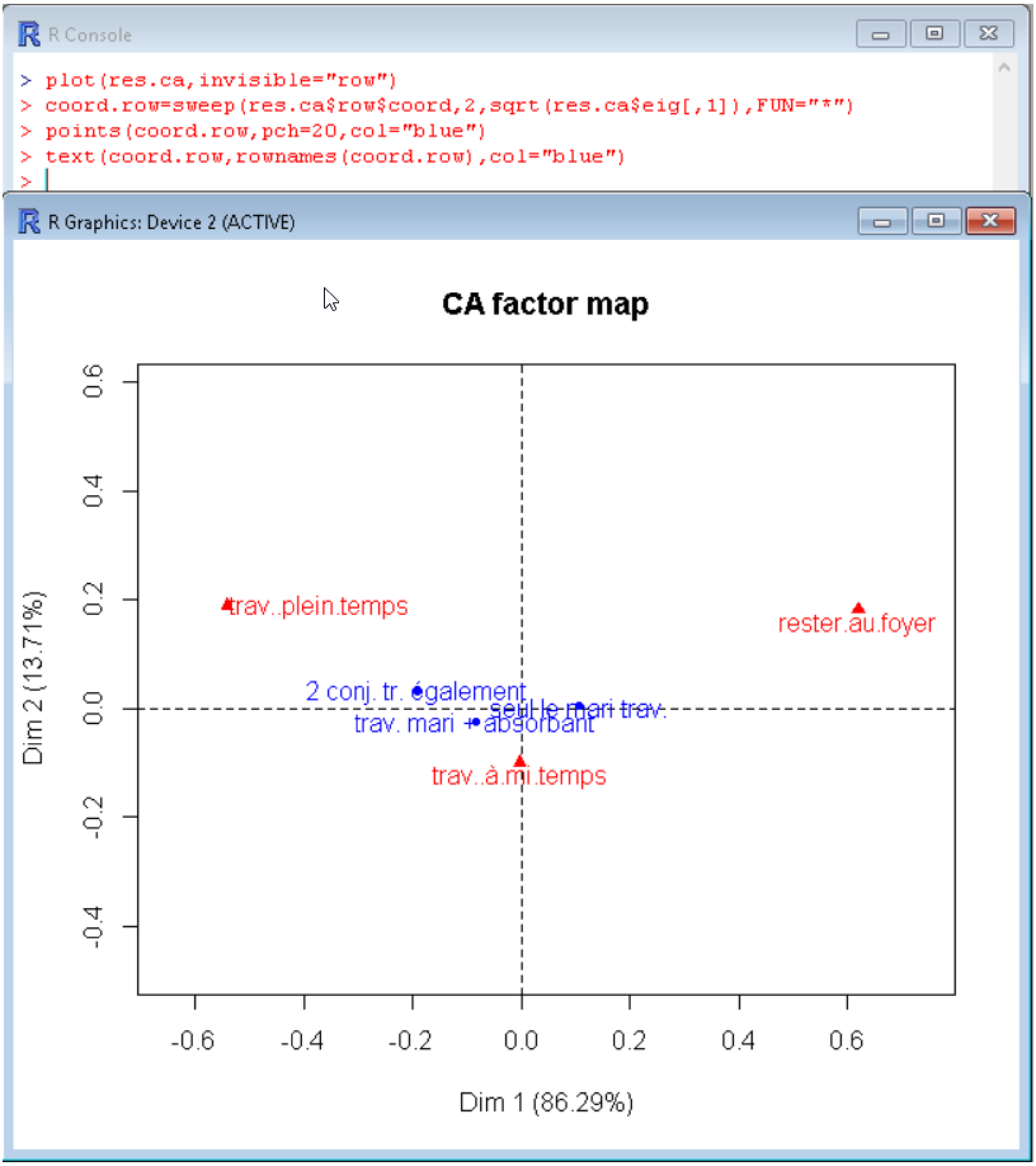
ou masquer les données de lignes:



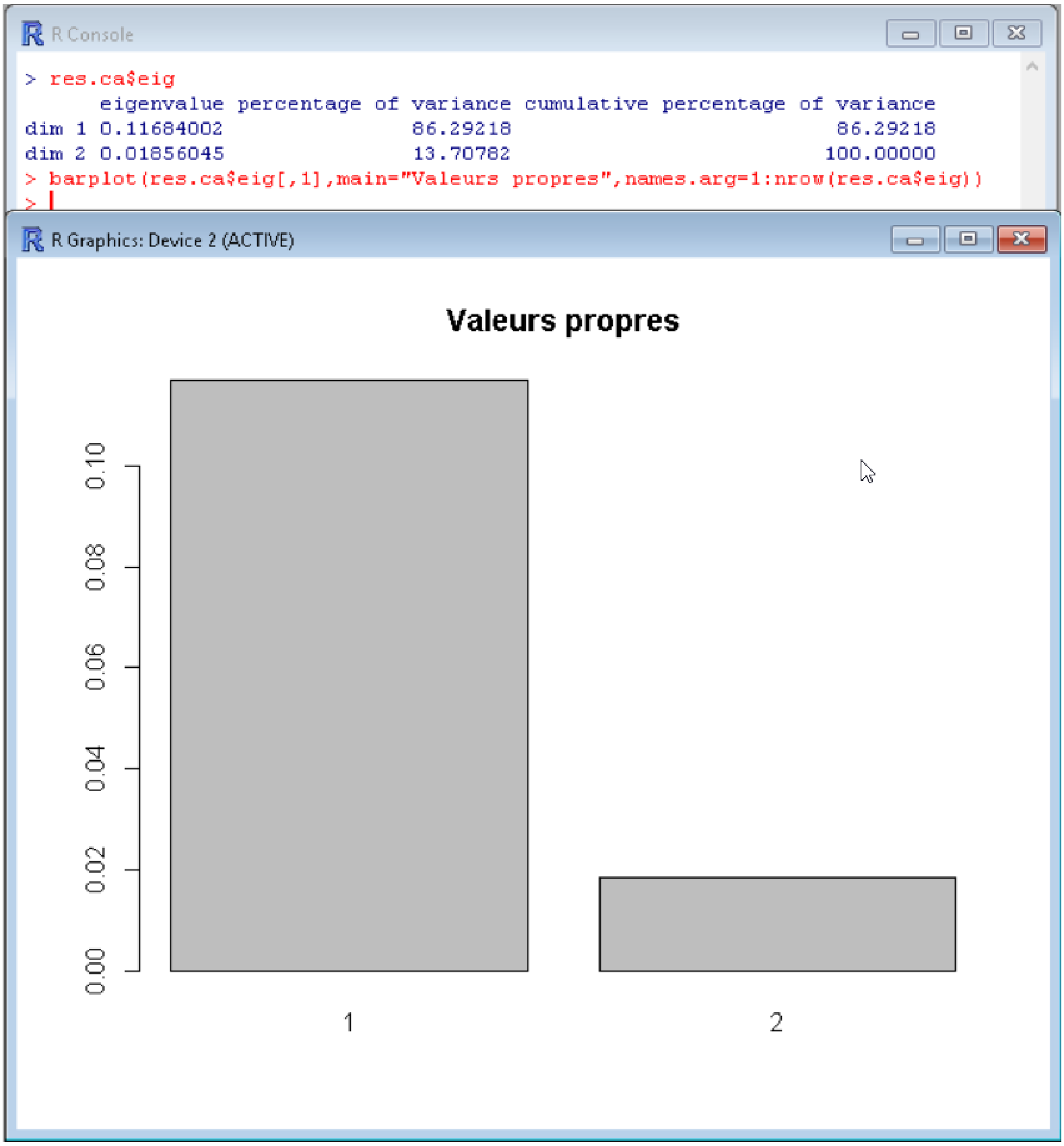
Nous pouvons normaliser les données en s'occupant d'abord de l'analyse éliminant les colonnes:



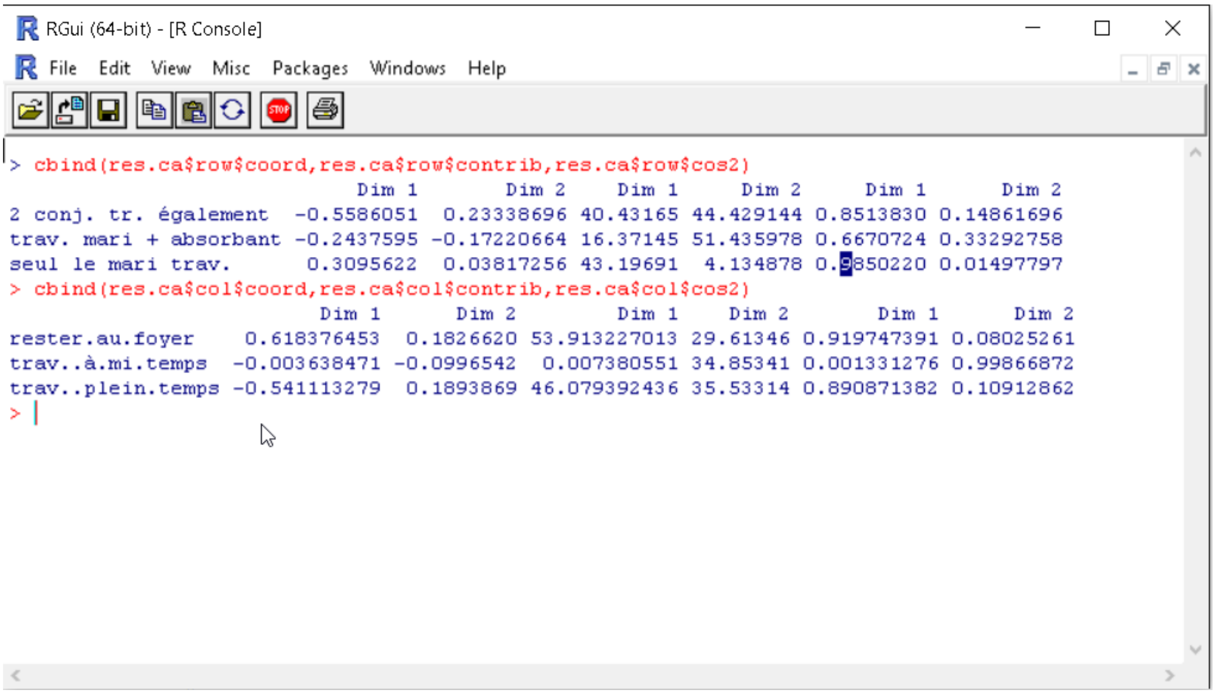
Idem pour les lignes:



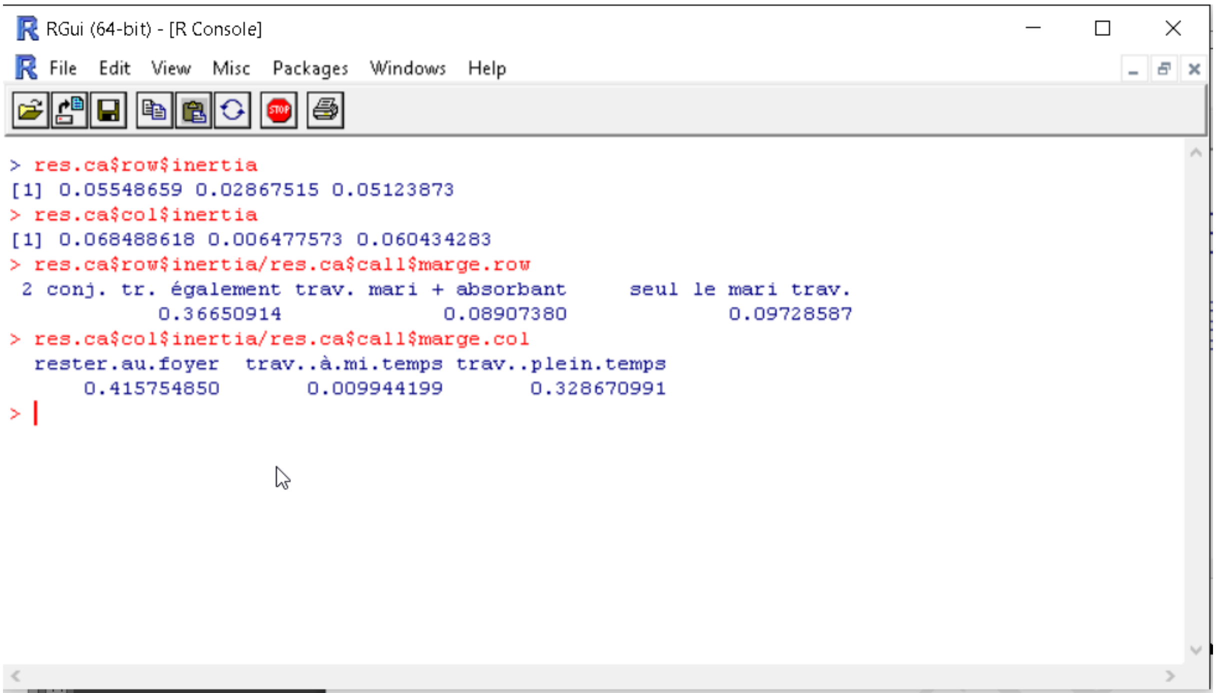
Le tableur des valeurs propres et le graph correspondant sont donnés par:



Le tableau des contributions et celui des qualités de représentations des lignes et des colonnes sont obtenus par:



Les inerties des lignes et des colonnes sont obtenus directement tandis que les distances au carré doivent être recalculées en utilisant la marge ligne et la marge colonne:



Exemple 219.: Analyse factorielle des correspondances

R 4.0.1

Ce qui suit est un copie traduite de l'excellent article disponible ici (de Tim Bock):

https://www.displayr.com/math-correspondence-analysis/

On commence par construire la table avec les fréquences observées:

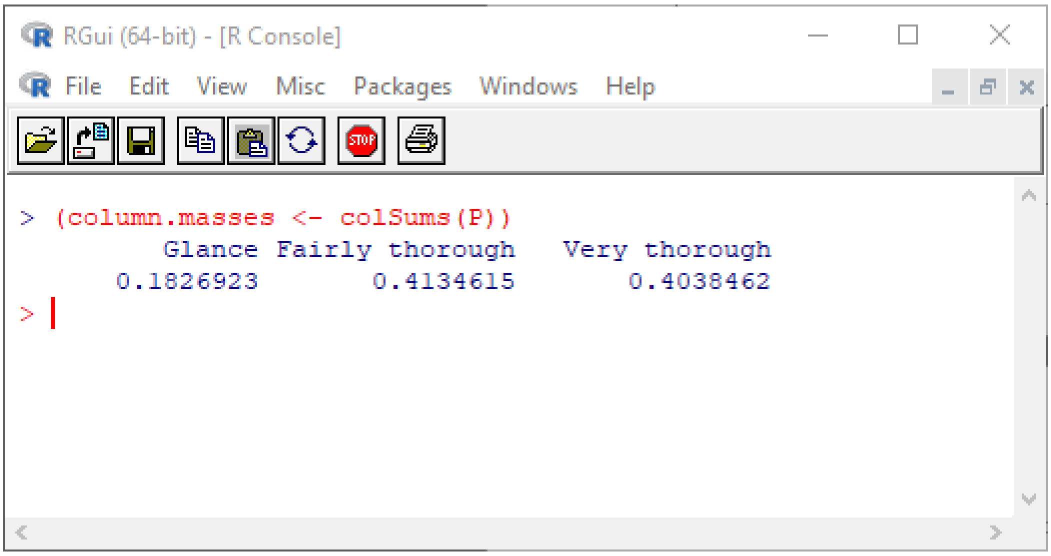
```
> N<-matrix(c(5, 18, 19, 12, 3, 7, 46, 29, 40, 7, 2, 20, 39, 49, 16),
+  nrow = 5, dimnames=list("Level of education" = c("Some primary", "Primary completed",
+  "Some secondary", "Secondary completed", "Some tertiary"),
+  "Category of readership" = c("Glance", "Fairly thorough", "Very thorough")))
> N
                     Category of readership
Level of education     Glance Fairly thorough Very thorough
  Some primary              5               7             2
  Primary completed        18              46            20
  Some secondary           19              29            39
  Secondary completed      12              40            49
  Some tertiary             3               7            16
>
```

La première étape de l'analyse des correspondances consiste à additionner toutes les valeurs du tableau. Notons ce total *n*.

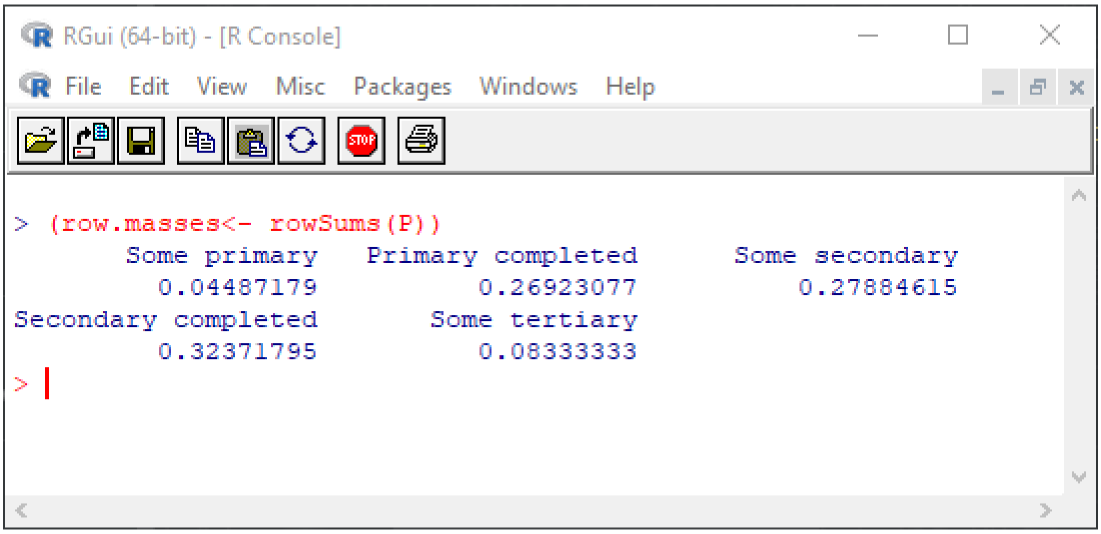
Ensuite, nous calculons le tableau des proportions, *P*. Ce qui donnera:

```
> (n<-sum(N))
[1] 312
> (P<-N/n)
                     Category of readership
Level of education         Glance Fairly thorough Very thorough
  Some primary        0.016025641      0.02243590   0.006410256
  Primary completed   0.057692308      0.14743590   0.064102564
  Some secondary      0.060897436      0.09294872   0.125000000
  Secondary completed 0.038461538      0.12820513   0.157051282
  Some tertiary       0.009615385      0.02243590   0.051282051
>
```

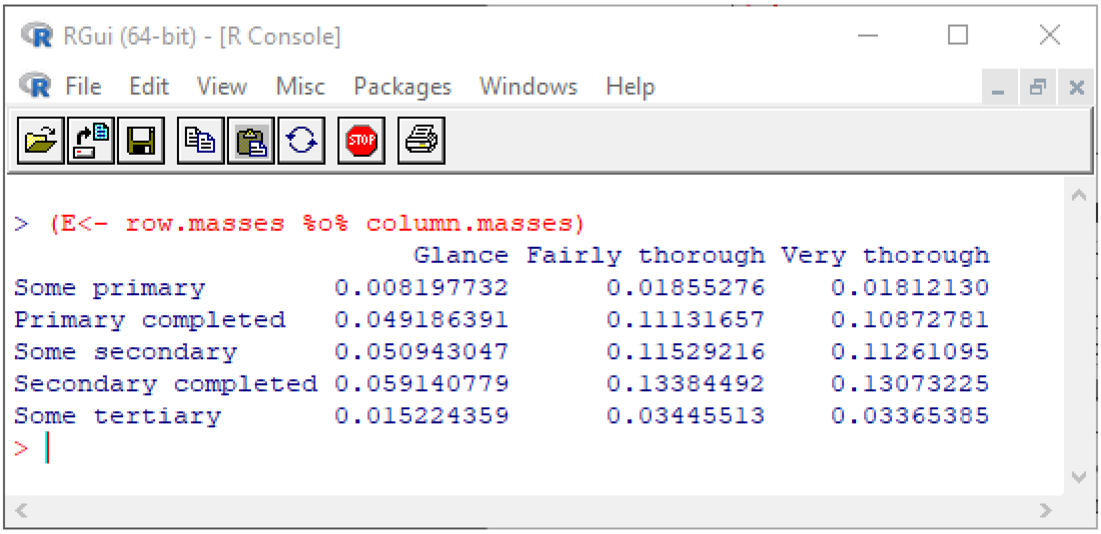
Dans le langage de l'analyse des correspondances, les sommes des lignes et des colonnes du tableau des proportions sont appelées "masses". Ce sont les entrées de nombreux calculs différents. Les masses de colonnes dans cet exemple montrent que *Glance*, *Fairly thorough* et *Very thorough* décrivent les habitudes de lecture de 18.3 %, 41.3 % et 40.4 % de l'échantillon respectivement. Nous pouvons calculer les masses des colonnes en utilisant le code R suivant :



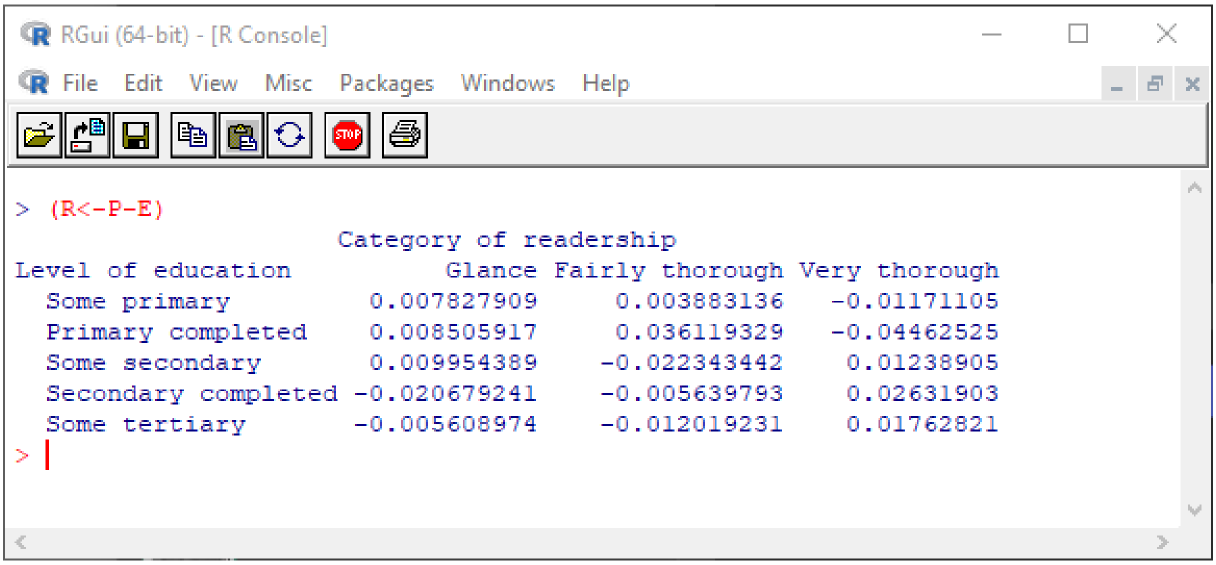
et respectivement pour les lignes:



En se référant au tableau des proportions d'origine, 1.6 % des personnes ont jeté un coup d'oeil (*glanced*) et ont fait des études primaires (*some primary*). Ce nombre est-il grand ou petit ? Nous pouvons calculer la valeur que nous nous attendrions à voir en supposant qu'il n'y a pas de relation entre l'éducation (*education*) et le lectorat (*readership*). La proportion qui *glanced* à un journal est de 18.2 % et 4.5 % n'ont qu'un peu d'enseignement primaire. Ainsi, s'il n'y a pas de relation entre l'éducation et le lectorat, nous nous attendrions à ce que 4.5% des 18.2% des personnes (c'est-à-dire 0.008 = 0.8%) aient à la fois jeté un coup d'œil et aient fait des études primaires. Nous pouvons calculer les proportions attendues de toutes les cellules du tableau de la même manière:



Nous calculons les résidus en soustrayant les proportions attendues des proportions observées. Les résidus dans l'analyse des correspondances ont un rôle différent de celui qui est typique dans les statistiques. En général, en statistique, les résidus quantifient l'étendue de l'erreur dans un modèle. Dans l'analyse des correspondances, en revanche, tout l'accent est mis sur l'examen des résidus.



Le plus grand résidu est de -0.045 pour *Primary completed* et *Very thoroug*. C'est-à-dire que la proportion observée de personnes qui n'ont terminé que l'école primaire et qui sont ont fait un lecture très approfondie est de 6.4 %, ce qui est inférieur de 4.5 % à la proportion attendue de 10.9 %, qui est calculée en supposant qu'il n'y a aucune relation entre le lectorat des journaux et l'éducation. Ainsi, la conclusion provisoire que nous pouvons en tirer est qu'il existe une association négative entre le fait d'avoir terminé l'enseignement primaire et la lecture très approfondie. C'est-à-dire que les personnes n'ayant fait qu'une éducation primaire sont moins susceptibles de lire très attentivement que la personne moyenne.

Calculons maintenant l'index:

```
> (I<-R/E)
                          Category of readership
Level of education          Glance Fairly thorough Very thorough
  Some primary           0.9548872       0.20930233    -0.6462585
  Primary completed      0.1729323       0.32447398    -0.4104308
  Some secondary         0.1954023      -0.19379845     0.1100164
  Secondary completed -0.3496613      -0.04213677     0.2013201
  Some tertiary         -0.3684211      -0.34883721     0.5238095
>
```

Les résidus indexés ont une interprétation simple. Plus la valeur du tableau est éloignée, plus la proportion observée est grande par rapport à la proportion attendue.

Nous pouvons maintenant voir une tendance claire. La plus grande valeur sur la table est le 0.9548882 pour *Some primary* et *Glance*. Cela nous indique que les personnes ayant une éducation primaire sont presque deux fois plus susceptibles de jeter un œil à un journal que ce à quoi nous nous attendrions s'il n'y avait pas de relation entre l'éducation et la lecture. En d'autres termes, la valeur observée est supérieure de 95 % à la valeur attendue.

En lisant cette première ligne, nous voyons qu'il y a un résidu indexé plus faible, mais positif, de 0.21 pour *Fairly thorough* et *Some primary*. Cela nous indique que les personnes ayant fait des études primaires étaient 21% plus susceptibles d'être des lecteurs assez approfondis que nous attendrions. Et, un score de -0.65 pour *Veru thorough*, nous indique que les personnes ayant un niveau d'instruction élémentaire étaient 65% moins susceptibles d'être des lecteurs très approfondis que prévu. En lisant tous les chiffres sur la table, la tendance générale est que des niveaux d'éducation plus élevés équivalent à un lectorat plus approfondi.

Comme nous le verrons plus loin, l'analyse des correspondances est une technique conçue pour visualiser ces valeurs indexées.

Pour faire un graphique de ce tableau des Index, nous faisons une décomposition en valeur singulières (SVD)

Exemple 220.: Copulas Gaussiens et Student bivariés

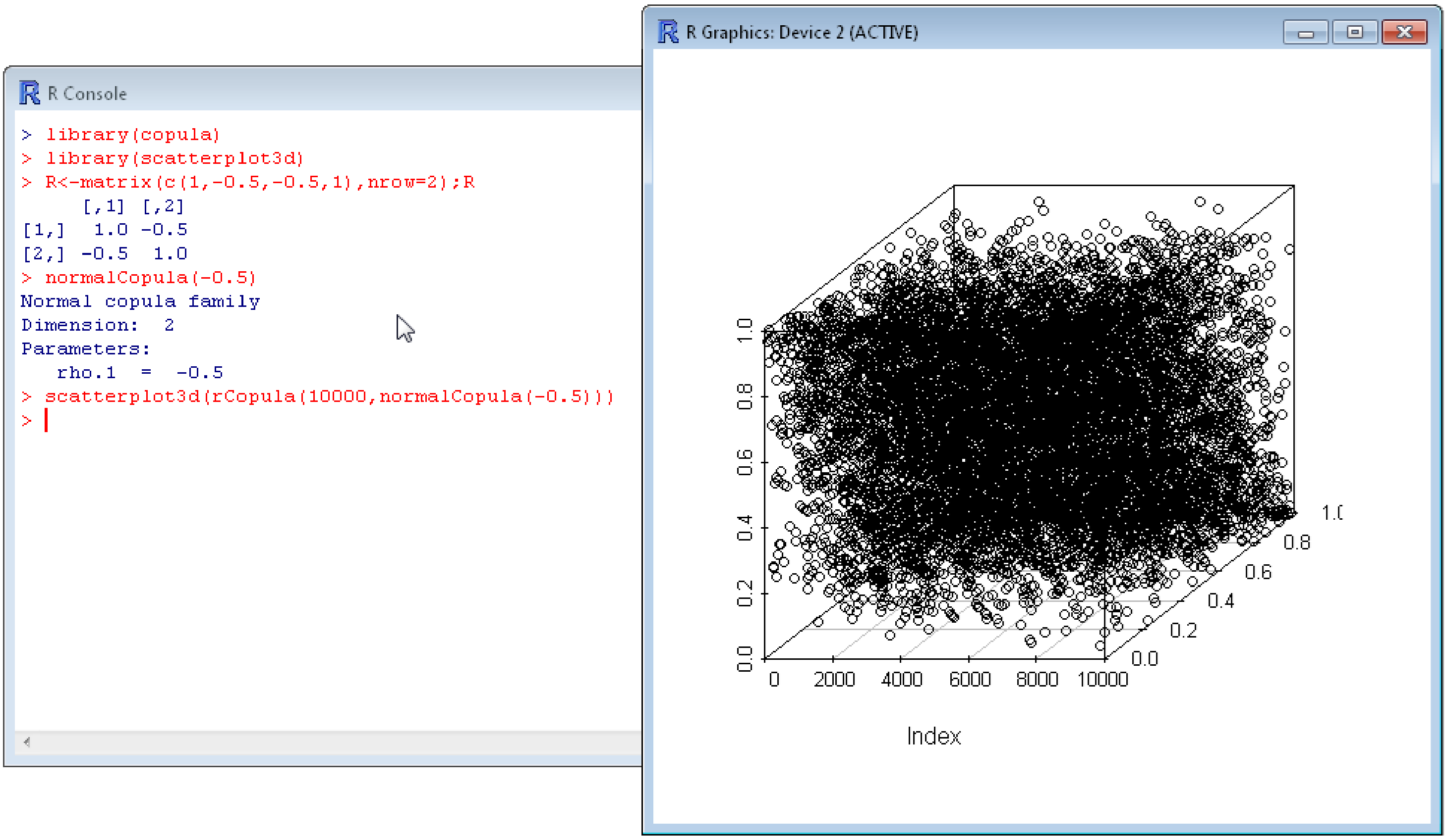
R 3.1.1

Nous avons étudié dans le cours théorique les distributions bivariées de la loi Normale et mentionné les différentes variantes des distributions de Student.

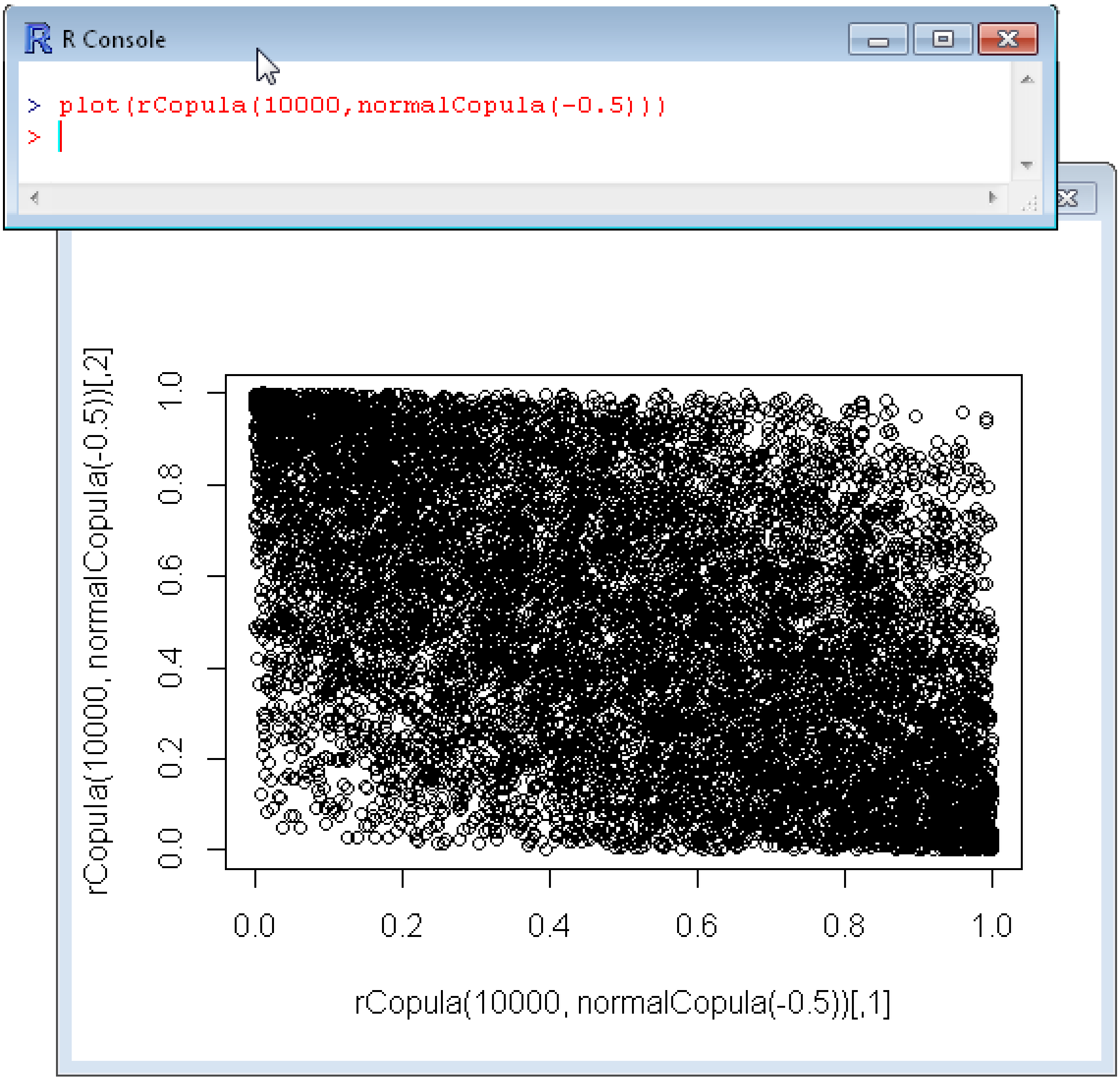
Le but ici va être de générer des variables aléatoires Normales bivariées dans un premier temps et de calculer ensuite le meilleur ajustement.

Dans un premier temps nous faisons comme dans le cours théorique en simulant les réalisations 10'000 variables aléatoires uniformes qui sont donc l'inverse d'une distribution Normale centrée réduite bivariée dont le coefficient de corrélation est -0.5 avec la fonction **normalCopula( )** du package **copula**.

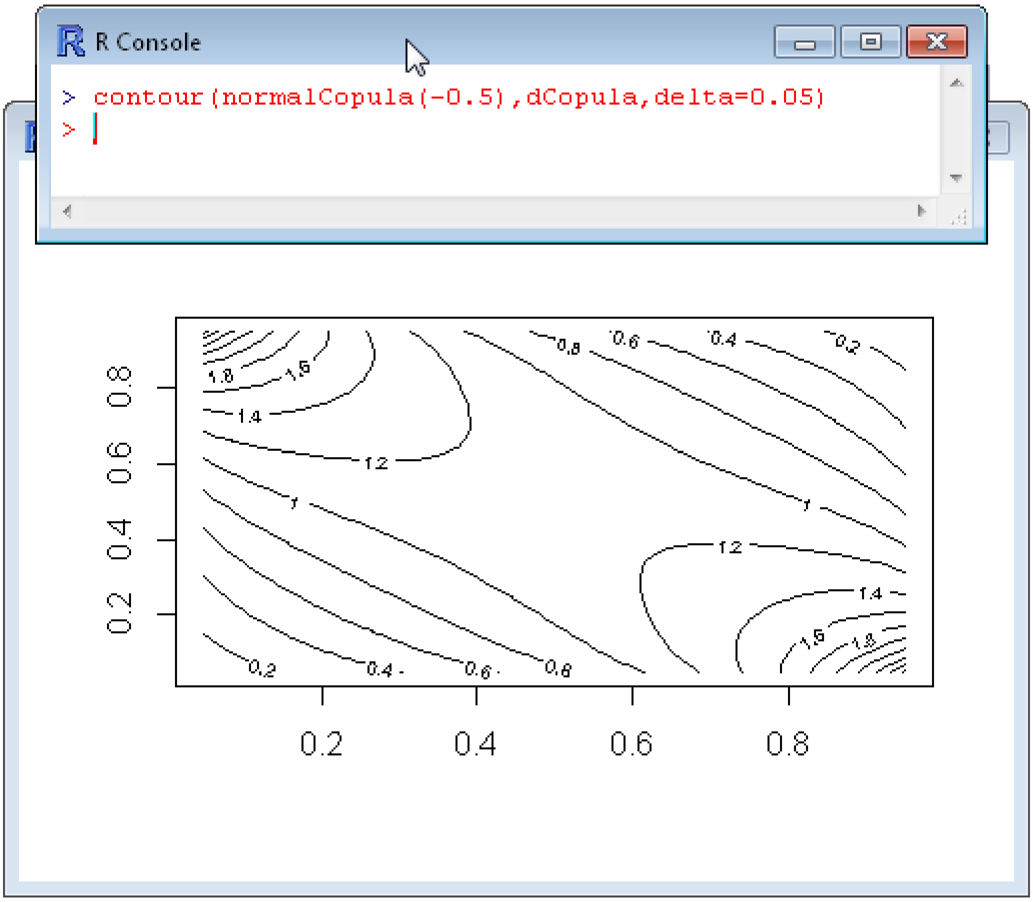
Voici une première représentation pas forcément très utile (opinion très personnelle):



Et à la page suivante une représentation beaucoup plus intéressante et remarquez que nous retrouvons ce que nous avons construit à la main dans le cours théorique avec le tableur Microsoft Excel:

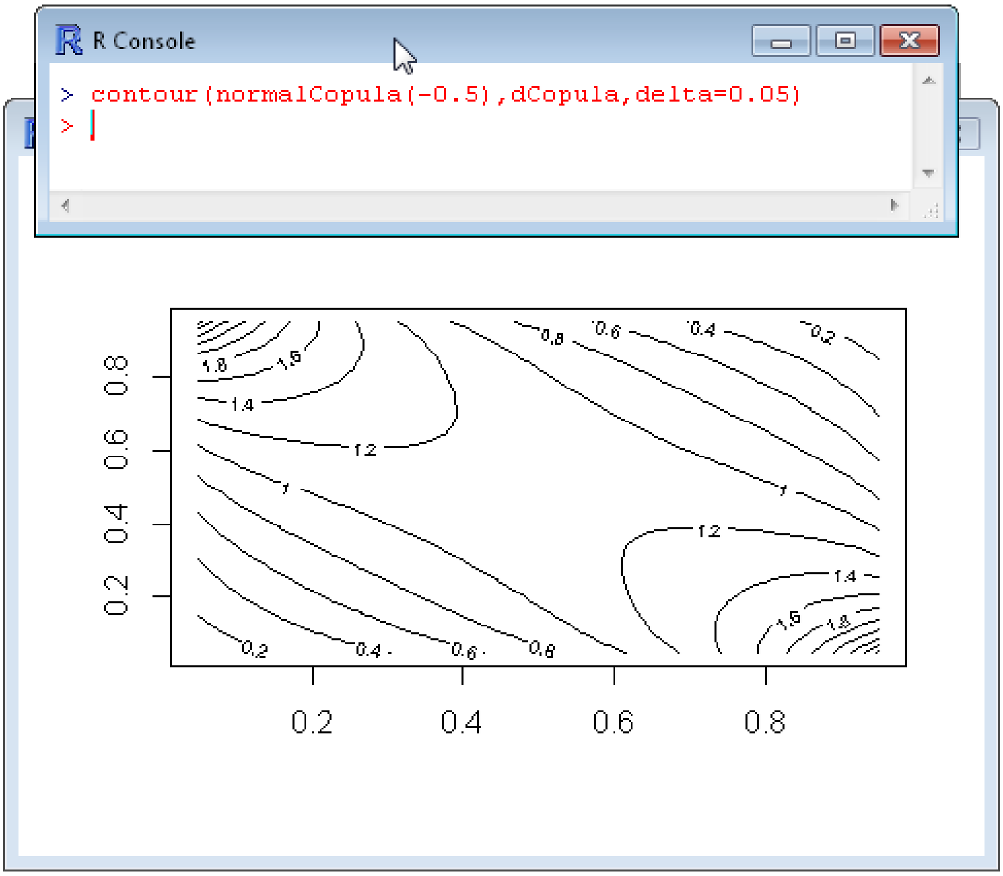


Nous pouvons afficher les courbes d'iso-densité:

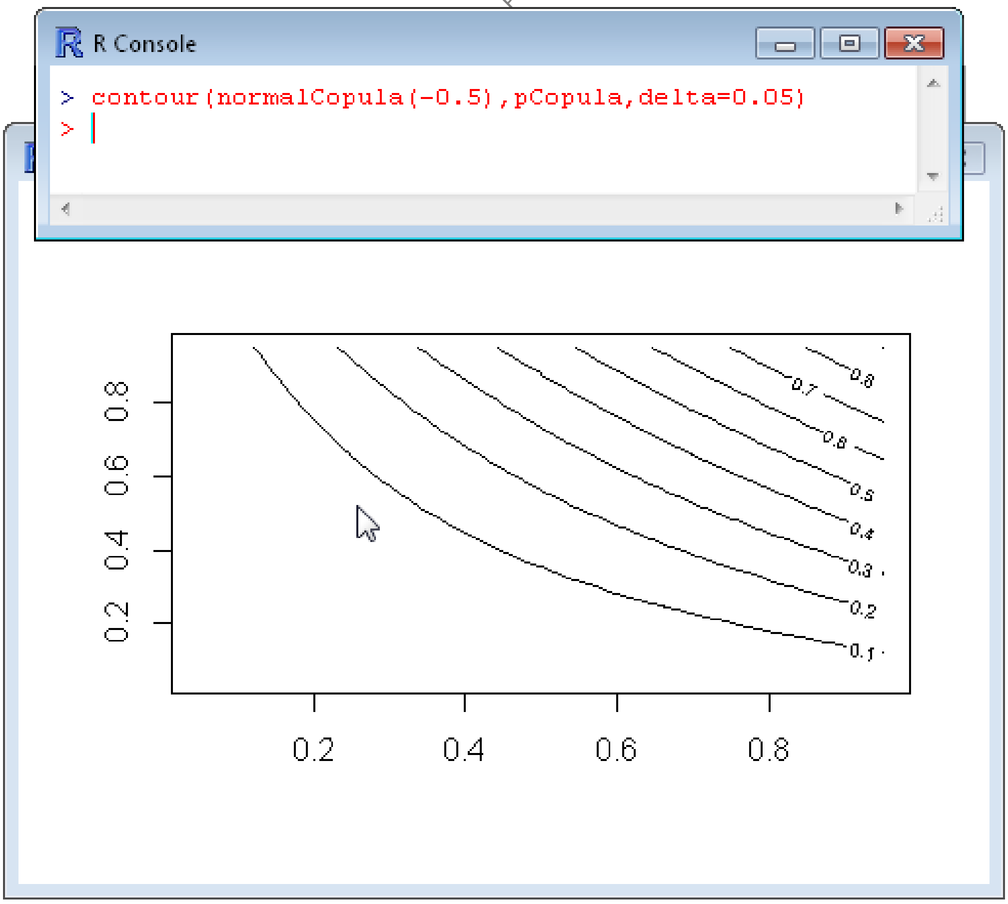


Ensuite, "amusons-nous" à faire de l'ingénierie inverse où depuis les points simulés et en imposant que nous avons connaissance que c'est un copula Normal centré réduit de dimension 2 (bivarié) quel est la valeur du coefficient du corrélation estimé à l'aide de la fonction **fitCopula( )** :

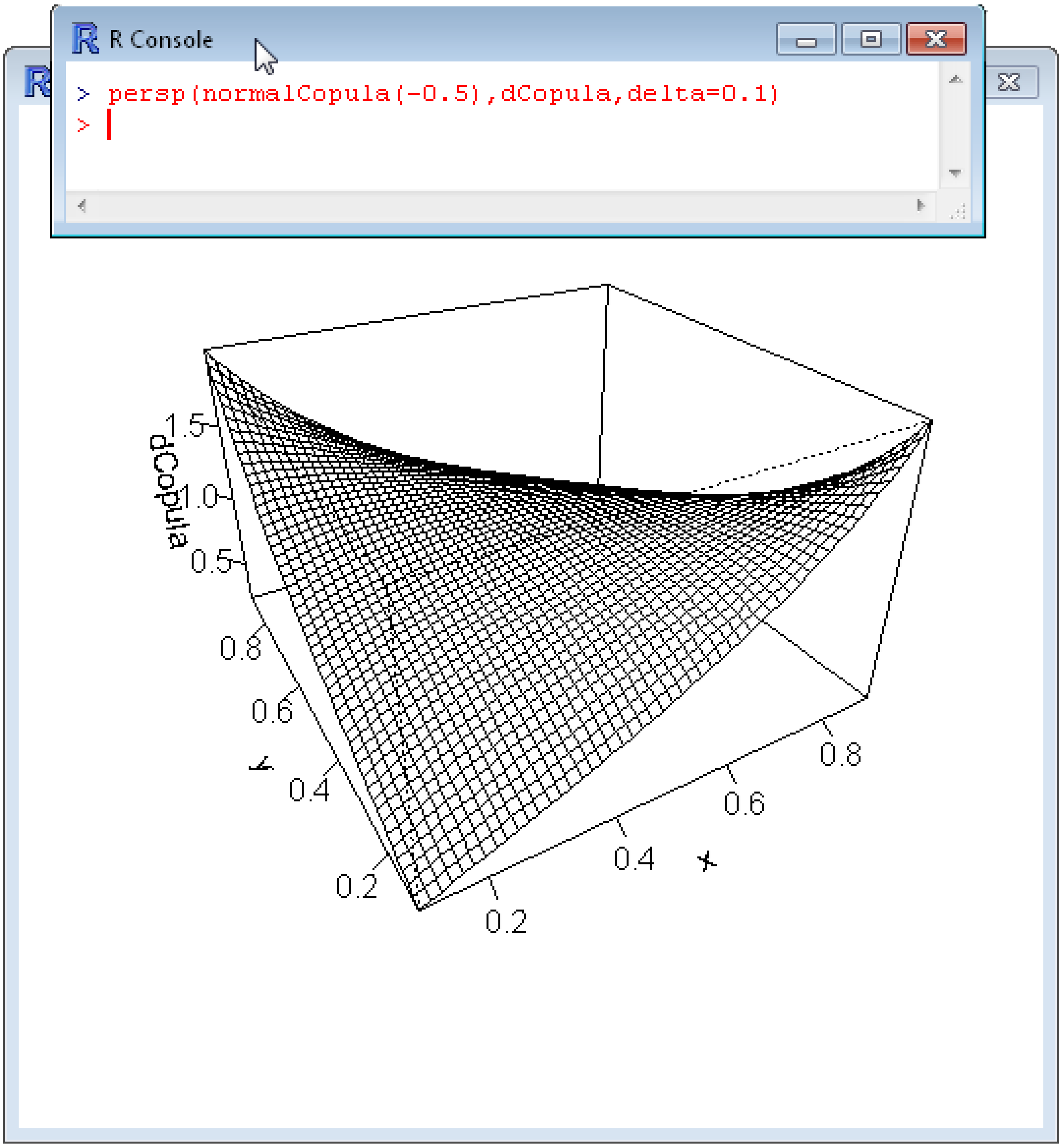

```
> rGauC<-rCopula(10000,normalCopula(-0.5))
> norm.cop<-normalCopula(dim=2,dispstr="un")
> norm.fit<-fitCopula(rGauC,copula=norm.cop)
> norm.fit
fitCopula() estimation based on 'maximum pseudo-likelihood'
and a sample of size 10000.
       Estimate Std. Error z value Pr(>|z|)
rho.1 -0.495579   0.007593  -65.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
The maximized loglikelihood is  1399
Optimization converged
Number of loglikelihood evaluations:
function gradient
      18        3
>
```



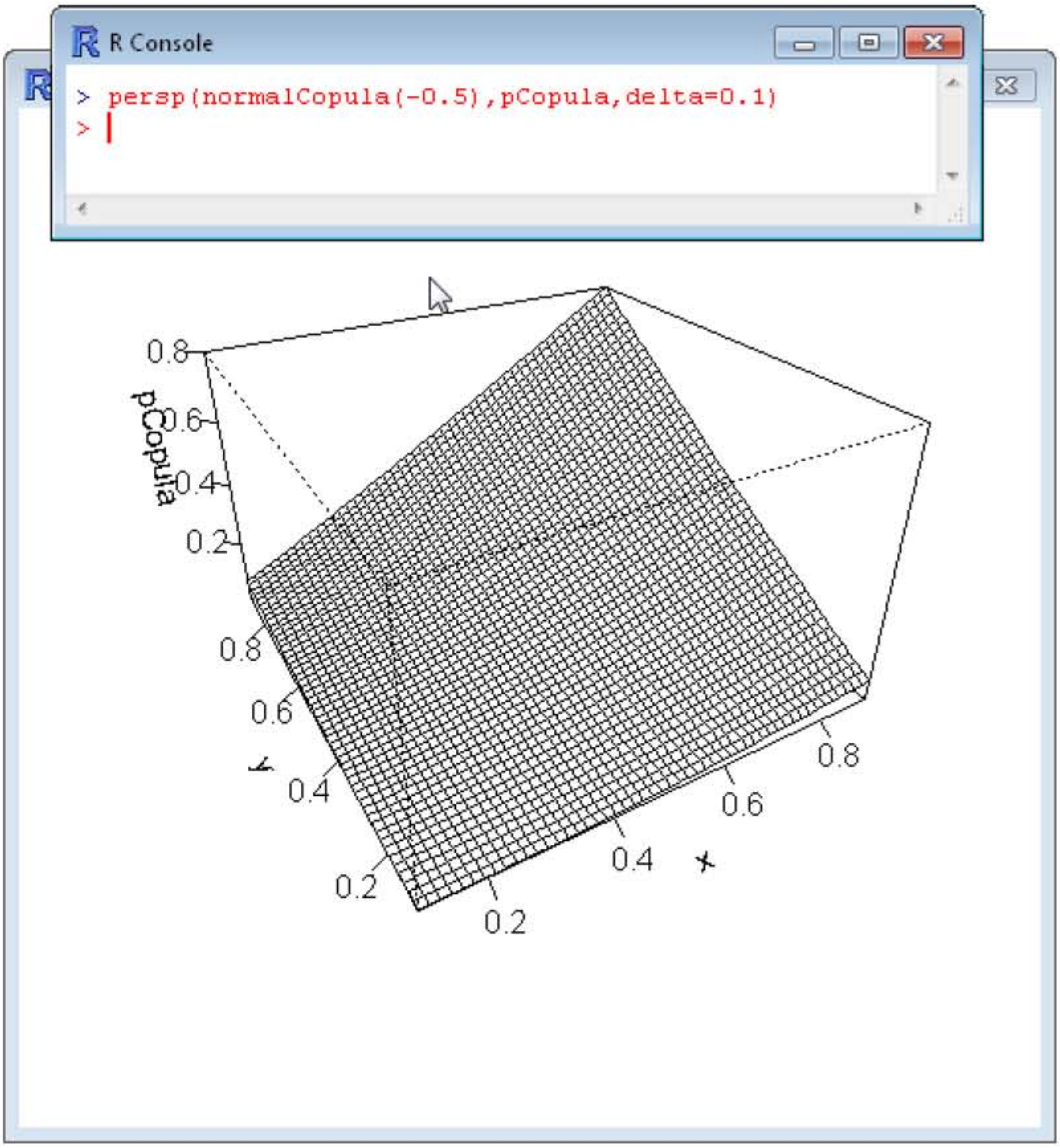
Ou en termes de probabilité cumulée:



Ou la densité sous forme de perspective 3D:



Ou la probabilité cumulée sous forme de perspective 3D:


R Console
> persp(normalCopula(-0.5),pCopula,delta=0.1)
>
pCopula
0.8
0.6
0.4
0.2
y
x

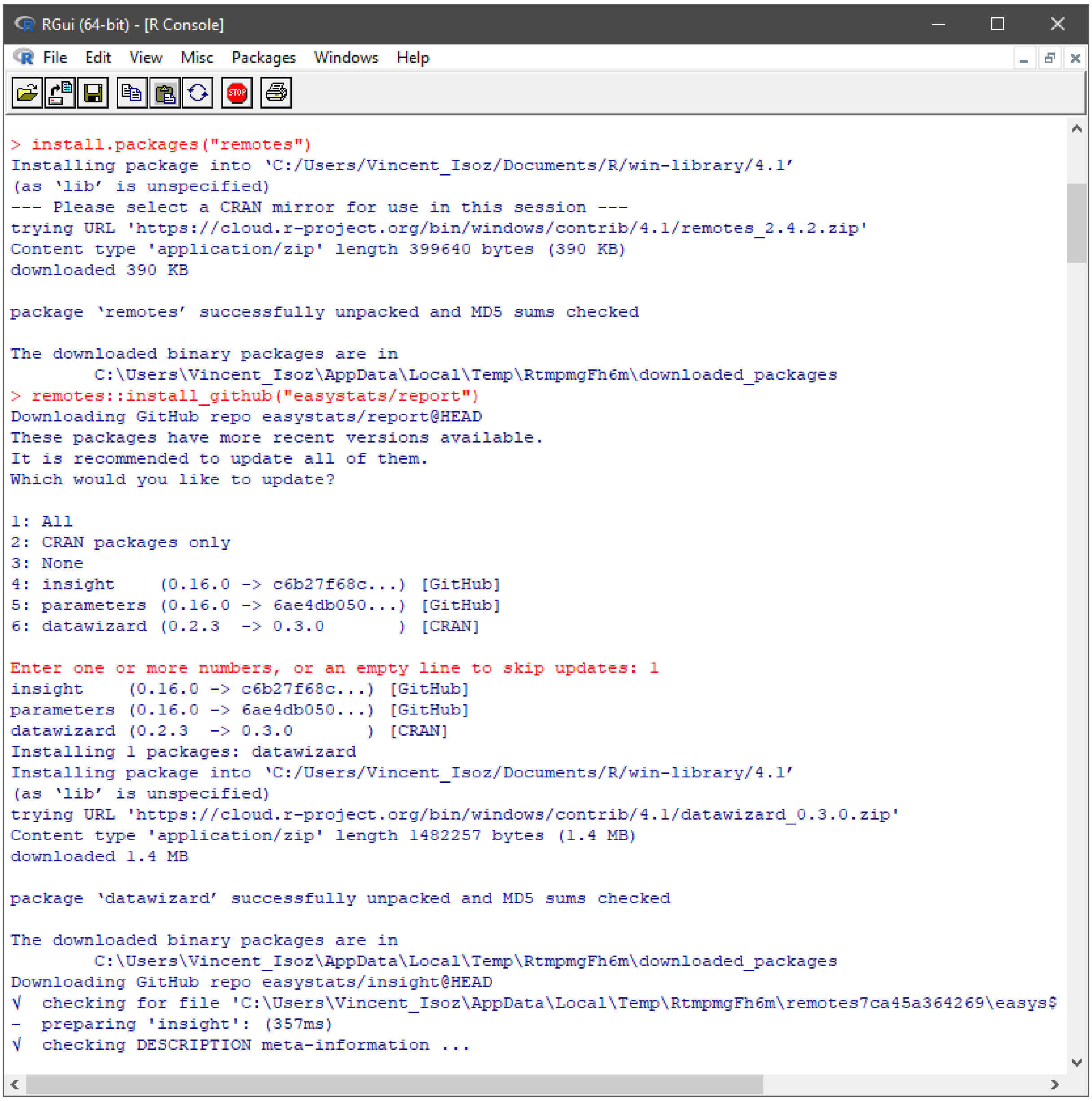
Exemple 221.: Reporter les conclusions et analyses automatiquement

R 3.1.1

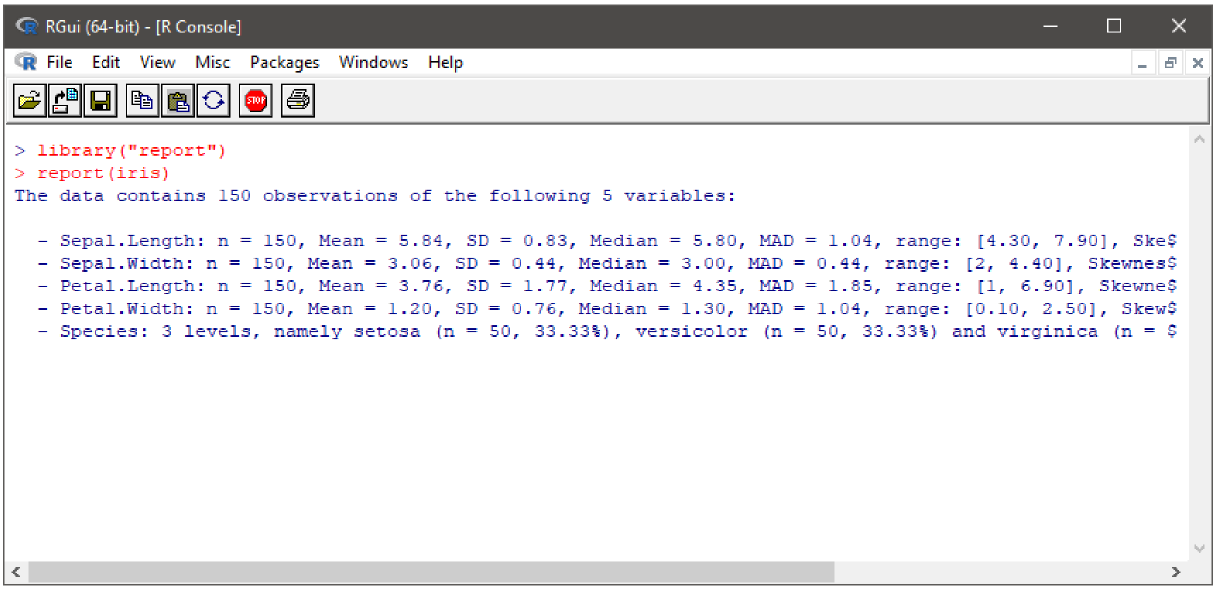
Comme nous l'avons mentionné au début de ce chapitre, il existe donc un package R qui génère automatiquement en anglais le phrases de conclucions et d'analyses des tests statistiques et jeux de données. Voyons donc quelques exemples tirés directement de sa vignette disponible ici:

https://easystats.github.io/report/

D'abord on l'installe de manière un peu particulière car au jour où nous écrivons ces lignes il n'est pas disponible sur le serveur CRAN:



Une fois ceci fait, voyons déjà comment il analyse un bête jeu de données:



Ensuite nous pouvons avoir avec quelques exemples de tests:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Console Terminal Jobs

R 4.1.1 · C:/tmp/

```
> library("report")
> report(cor.test(iris$Sepal.Length, iris$Sepal.Width) )


The Pearson's product-moment correlation between iris$Sepal.Length and iris$Sepal.Width is negative, statistically not s
ignificant, and small (r = -0.12, 95% CI [-0.27, 0.04], t(148) = -1.44, p = 0.152)
> report(t.test(mtcars$mpg ~ mtcars$am))


The Welch Two Sample t-test testing the difference of mtcars$mpg by mtcars$am (mean in group 0 = 17.15, mean in group 1
 = 24.39) suggests that the effect is negative, statistically significant, and large (difference = -7.24, 95% CI [-11.2
8, -3.21], t(18.33) = -3.77, p = 0.001; Cohen's d = -1.41, 95% CI [-2.26, -0.53])
> report(aov(Sepal.Length ~ Species, data = iris))
For one-way between subjects designs, partial eta squared is equivalent to eta squared.
Returning eta squared.
The ANOVA (formula: Sepal.Length ~ Species) suggests that:

  - The main effect of Species is statistically significant and large (F(2, 147) = 119.26, p < .001; Eta2 = 0.62, 95% CI
 [0.54, 1.00])

> report(glm(vs ~ mpg * drat, data = mtcars, family = "binomial"))
We fitted a logistic model (estimated using ML) to predict vs with mpg and drat (formula: vs ~ mpg * drat). The model's
 explanatory power is substantial (Tjur's R2 = 0.51). The model's intercept, corresponding to mpg = 0 and drat = 0, is a
t -33.43 (95% CI [-77.90, 3.25], p = 0.083). Within this model:

  - The effect of mpg is statistically non-significant and positive (beta = 1.79, 95% CI [-0.10, 4.05], p = 0.066; Std.
 beta = 3.63, 95% CI [1.36, 7.50])
  - The effect of drat is statistically non-significant and positive (beta = 5.96, 95% CI [-3.75, 16.26], p = 0.205; St
d. beta = -0.36, 95% CI [-1.96, 0.98])
  - The interaction effect of drat on mpg is statistically non-significant and negative (beta = -0.33, 95% CI [-0.83, 0.
15], p = 0.141; Std. beta = -1.07, 95% CI [-2.66, 0.48])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Inte
rvals (CIs) and p values were computed using
> report(lme4::lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris))
We fitted a linear mixed model (estimated using REML and nloptwrap optimizer) to predict Sepal.Length with Petal.Length
 (formula: Sepal.Length ~ Petal.Length). The model included Species as random effect (formula: ~1 | Species). The mode
l's total explanatory power is substantial (conditional R2 = 0.97) and the part related to the fixed effects alone (marg
inal R2) is of 0.66. The model's intercept, corresponding to Petal.Length = 0, is at 2.50 (95% CI [1.19, 3.82], t(146) =
 3.75, p < .001). Within this model:

  - The effect of Petal Length is statistically significant and positive (beta = 0.89, 95% CI [0.76, 1.01], t(146) = 13.
93, p < .001; Std. beta = 1.89, 95% CI [1.63, 2.16])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Inte
rvals (CIs) and p-values were computed using
> report(lm(Sepal.Length ~ Species, data = iris))
We fitted a linear model (estimated using OLS) to predict Sepal.Length with Species (formula: Sepal.Length ~ Species). T
he model explains a statistically significant and substantial proportion of variance (R2 = 0.62, F(2, 147) = 119.26, p <
 .001, adj. R2 = 0.61). The model's intercept, corresponding to Species = setosa, is at 5.01 (95% CI [4.86, 5.15], t(14
7) = 68.76, p < .001). Within this model:

  - The effect of Species [versicolor] is statistically significant and positive (beta = 0.93, 95% CI [0.73, 1.13], t(14
7) = 9.03, p < .001; Std. beta = 1.12, 95% CI [0.88, 1.37])
  - The effect of Species [virginica] is statistically significant and positive (beta = 1.58, 95% CI [1.38, 1.79], t(14
7) = 15.37, p < .001; Std. beta = 1.91, 95% CI [1.66, 2.16])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Inte
rvals (CIs) and p-values were computed using the Wald approximation.
> 
```

14. Statistiques robustes et non paramétriques

Nous allons donc voir dans ce chapitre si nous retrouvons les résultats calculés à la main dans le cours théorique pour voir si les résultats correspondent aux démonstrations mathématiques détaillées et aussi vérifier s'il y a correspondance avec Minitab, SPSS et MS Excel.

Pour que les choses soient claires, nous ne reviendrons pas dans ce chapitre ni dans celui des Statistiques non paramétriques sur le problème de la *p*-value et de ses utilisations fallacieuses que nous avons longuement développement lors des séances théoriques.

Ce chapitre contient actuellement seulement:

21 tests non-paramétriques sur un peu plus de 409 connus

Comme nous le savons sur un total de plus de 800 tests paramétriques et non-paramétriques!!!

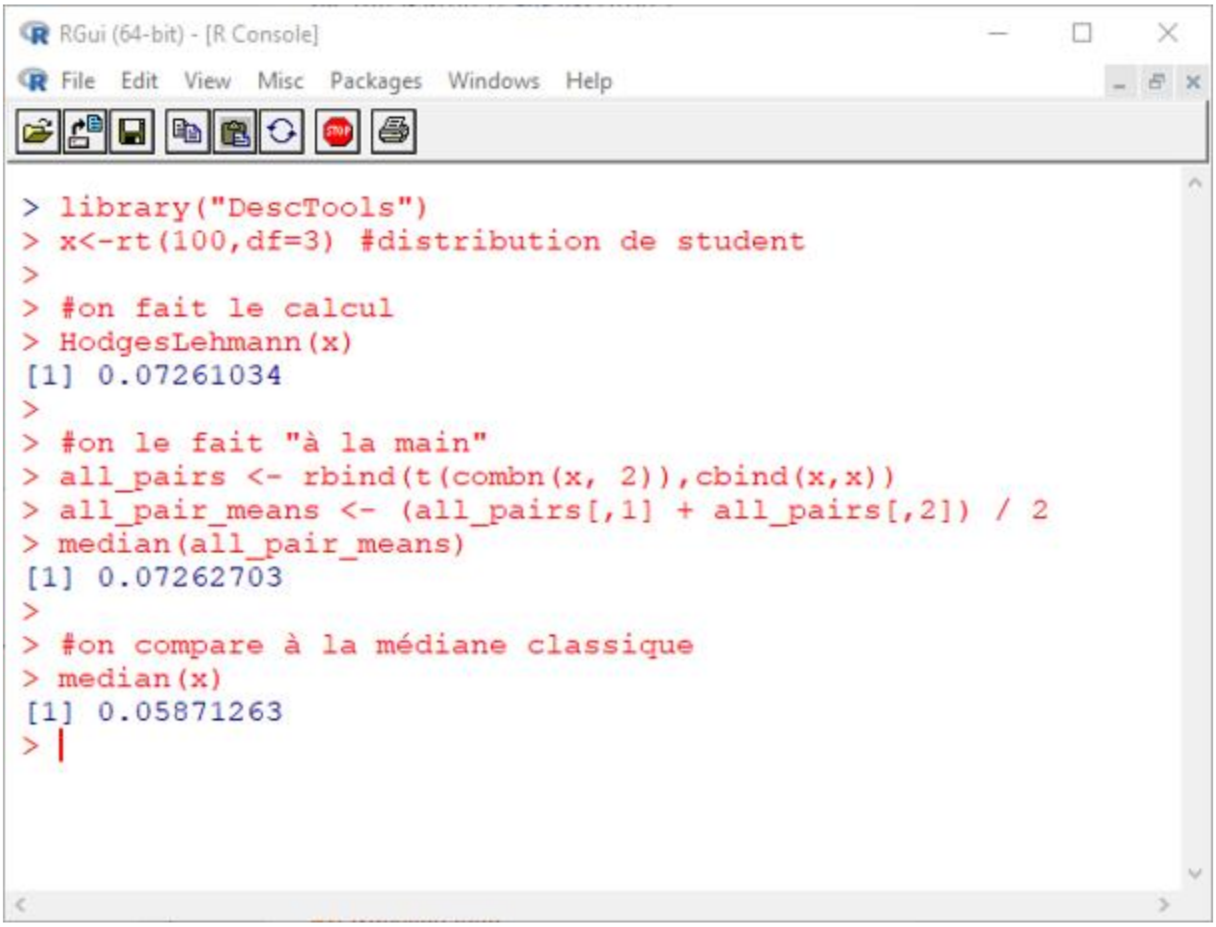
Soit un total à ce jour d'environ **55 tests paramétriques et non paramétriques** présentés dans ce livre (sans compter les tests relatifs aux méthodes de régressions!) sur un total de 510 tests paramétriques et non-paramétriques (voir la liste dans le cours théorique).

Exemple 222.: M-estimateurs et pseudomédiane (estimateur de Hodges-Lehmann)

R 3.0.2

Comme c'est élémentaire et empirique, nous avons déjà traité certains estimateurs robustes et M-estimateur à la page 224

Concernant la pseudomédiane qui est plus robuste que la médiane pour certaines fonctions asymétriques, nous avons la fonction **HodgesLehmann( )** dans le package **DescTools**:

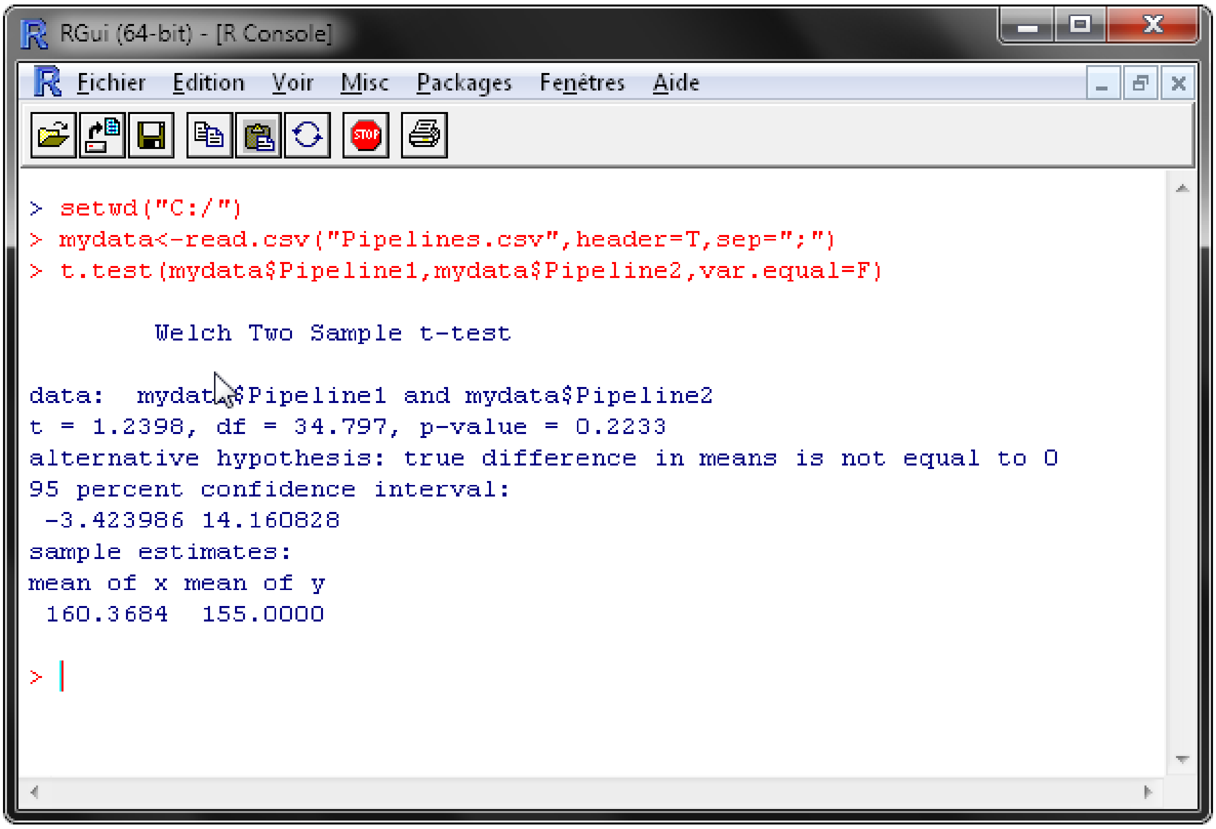


Exemple 223.: Test de Yuen-Welch test

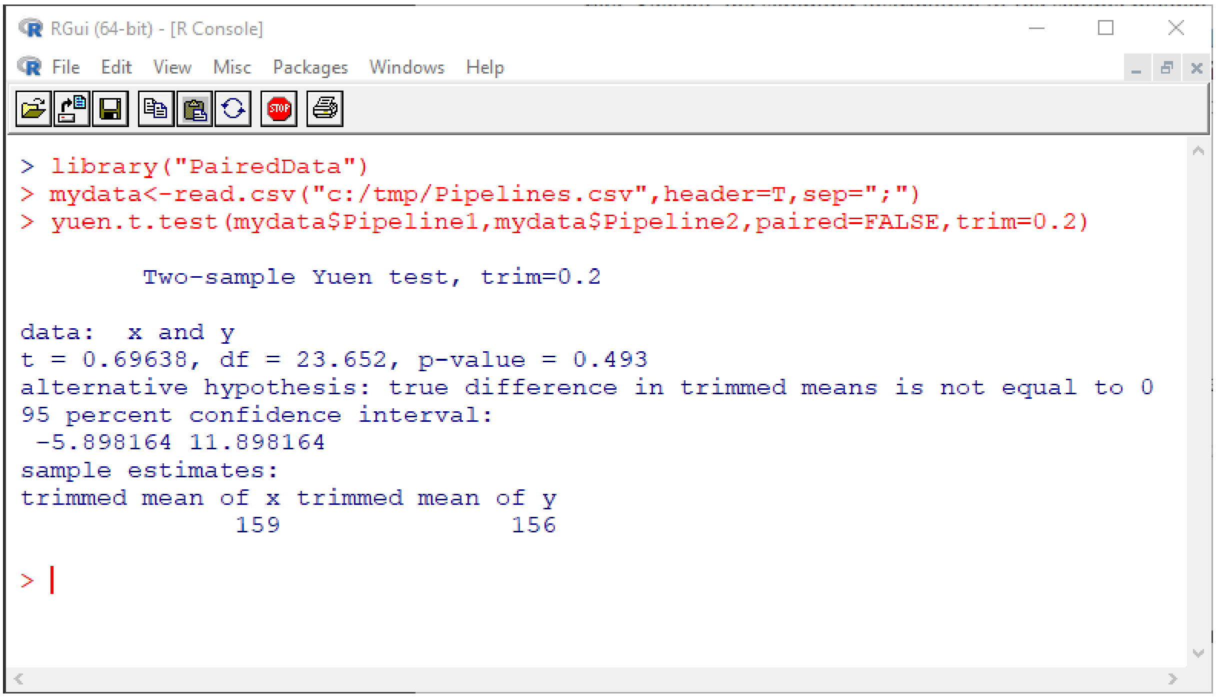
R 3.6.3

Nous avons présenté ce test, aussi appelé "test-T de Student des moyennes élaguées à deux échantillons avec variances winzorisées", dans le cours théorique, et comme nous l'avons vu d'après la lecture de l'article original, il est complétement empirique et construit sur la base de simulation de Monte-Carlo.

Voyons comment l'utiliser en reprenant les donnés du test de Welch qui nous avait donné pour rappel:

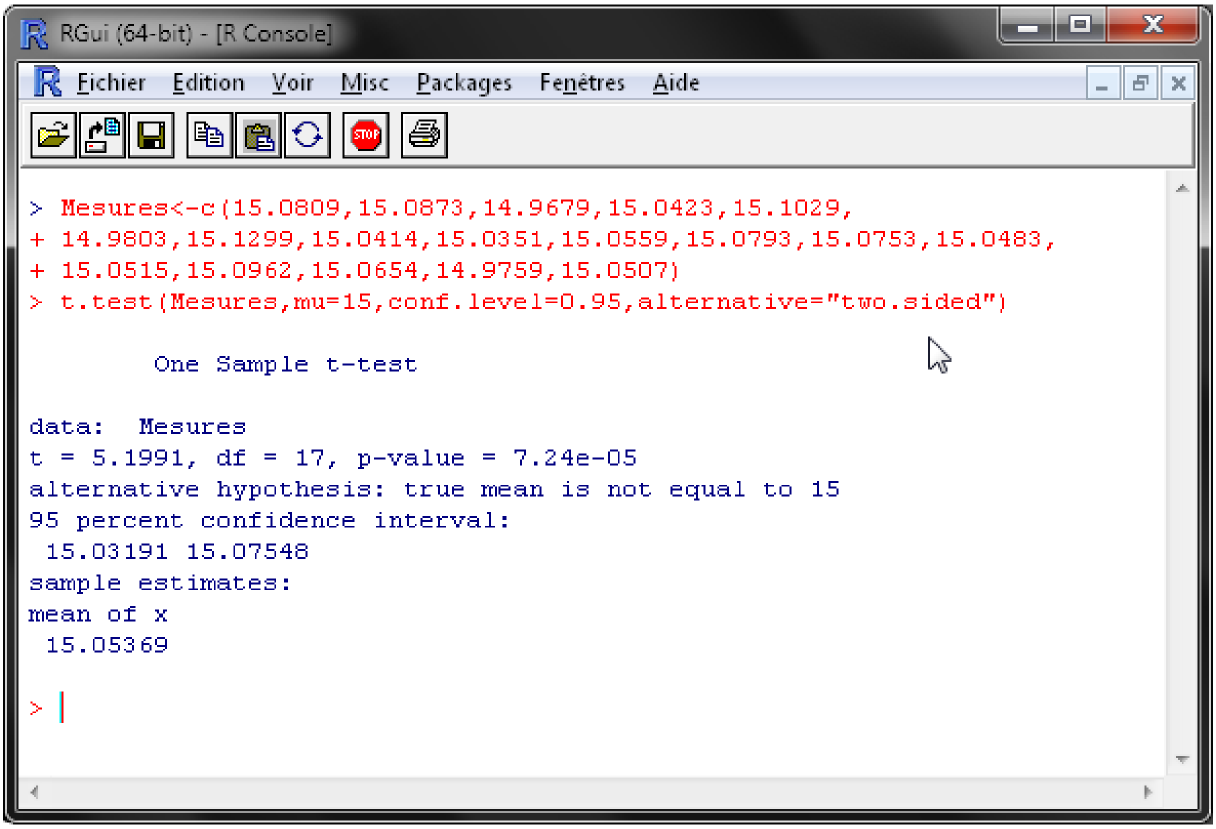


Comparons avec le test de Yuen-Welch:

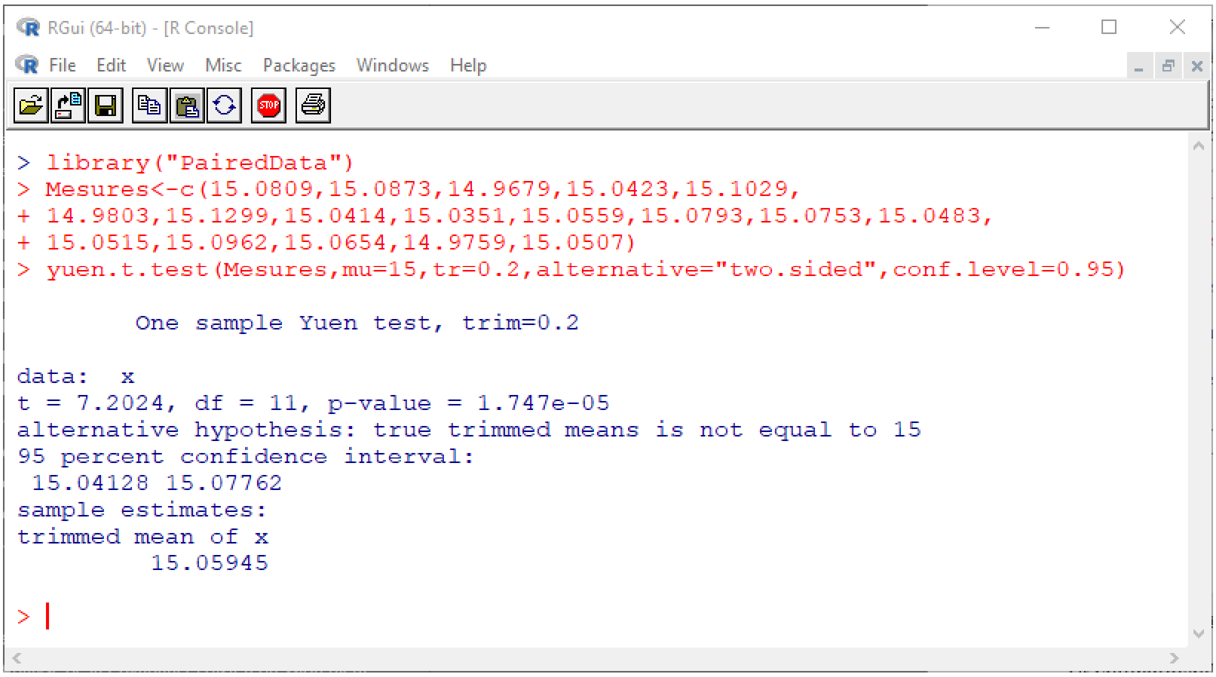


Donc cela ne change pas la conclusion.

On peut aussi faire un test à un seul échantillon. Rappelons le test-T de Student à 1 échantillons que nous avions fait:



Et comparaison avec la version de Yuen-Welch:



```
> library("PairedData")
> Mesures<-c(15.0809,15.0873,14.9679,15.0423,15.1029,
+ 14.9803,15.1299,15.0414,15.0351,15.0559,15.0793,15.0753,15.0483,
+ 15.0515,15.0962,15.0654,14.9759,15.0507)
> yuen.t.test(Mesures,mu=15,tr=0.2,alternative="two.sided",conf.level=0.95)

        One sample Yuen test, trim=0.2

data:  x
t = 7.2024, df = 11, p-value = 1.747e-05
alternative hypothesis: true trimmed means is not equal to 15
95 percent confidence interval:
 15.04128 15.07762
sample estimates:
trimmed mean of x
         15.05945

> 
```

Encore une fois nous voyons que cela ne change pas la conclusion.

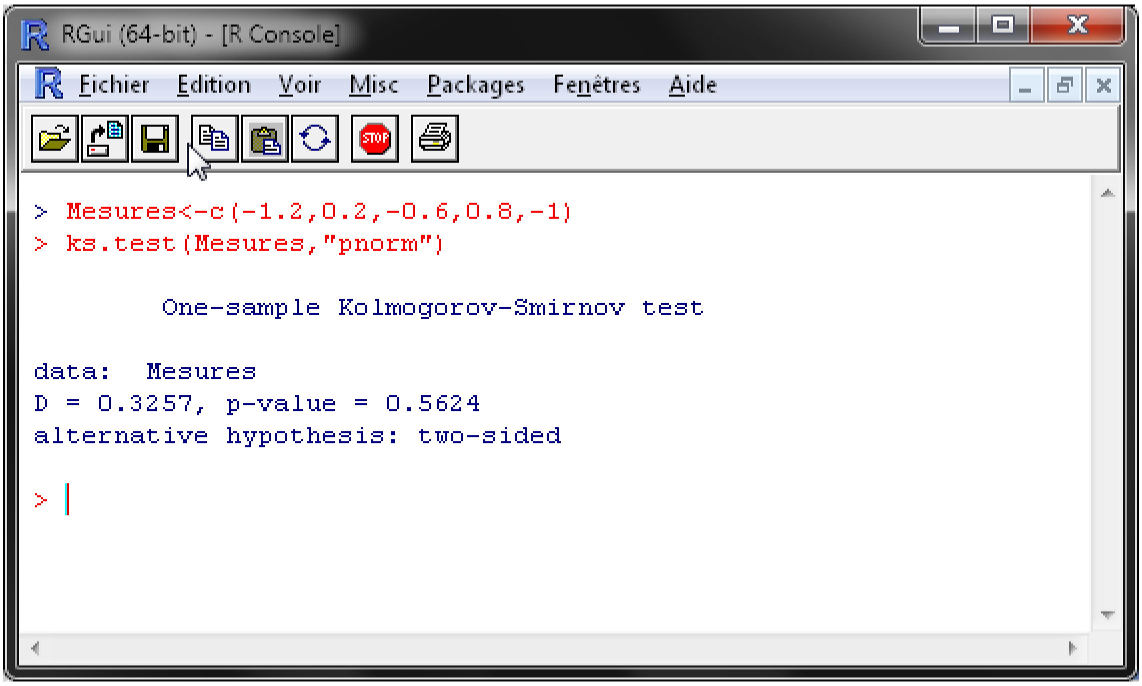
Exemple 224.: Test d'ajustement de Kolmogorov-Smirnov

R 3.0.2

Cas à 1 échantillon

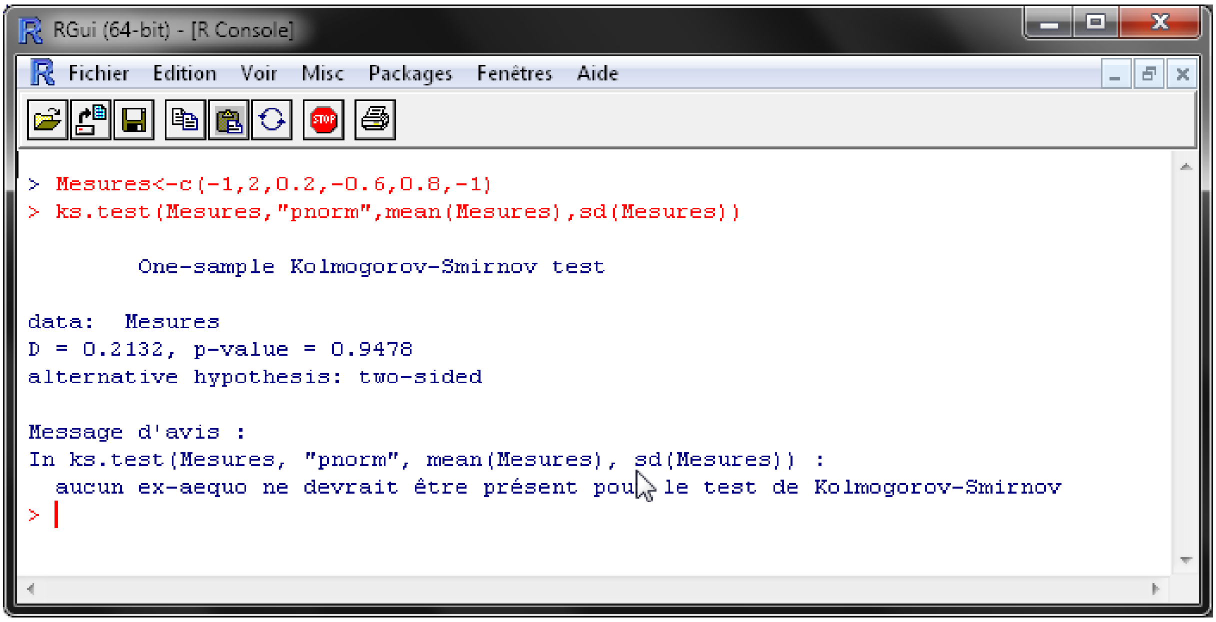
Nous souhaitons ici simplement vérifier que nous obtenons les mêmes résultats que ceux obtenus dans Microsoft Excel avec la méthode Monte Carlo (et in extenso vérifier les résultats obtenus lors de l'étude théorique du test de Kolmogorov-Smirnov) et Minitab.

En utilisant la commande **ks.test( )** et avec les mêmes données brutes que dans le cours théorique, nous avons pour le test de Kolmogorov-Smirnov dans le cas particulier d'un test de Normalité:



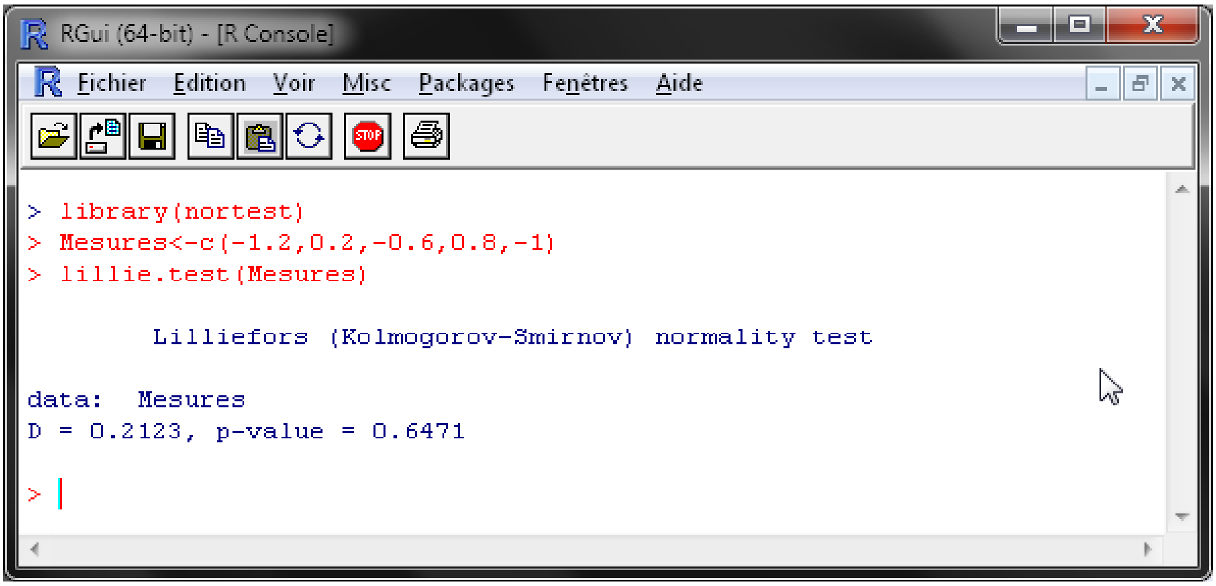
Donc ici nous voyons un énorme avantage par rapport à Minitab déjà: nous pouvons choisir la distribution selon nos désirs!!

Nous n'avons pas la même valeur de *D* que Minitab (qui était 0.212) par contre elle correspond à celle calculée dans le cours théorique à la main (idem pour la *p*-value). En réalité avec R il y a un petit piège pour retomber sur ce que fait Minitab, il faut écrire (dès lors cela équivaut à la variante dite pour rappel de "test d'ajustement de Lilliefor"):



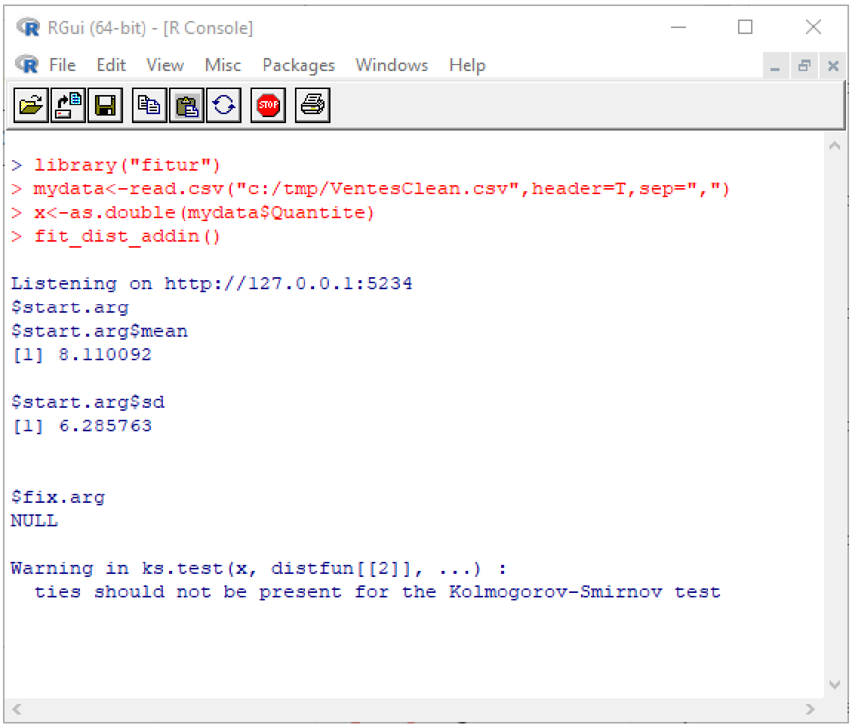
et là tout est bon!

Nous pouvons faire une double vérification en utilisant le test de Lilliefor explicitement via le package **nortest** et la commande **lillie.test( )** :

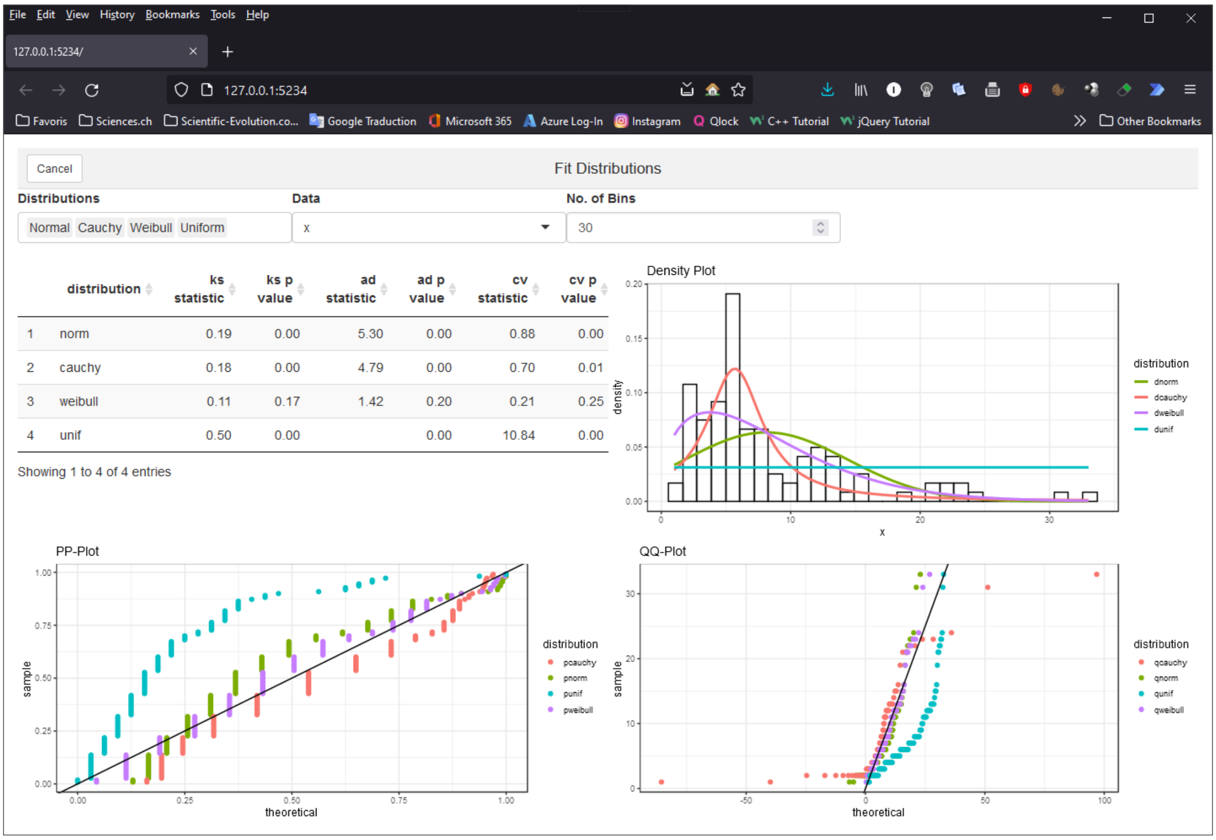


Bien que la valeur *D* soit la même, la *p*-value est très différente! Il faudrait investiguer pour savoir d'où vient cette différence. Peut-être dans la manière de calculer l'écart-type??? A suivre...

Pour ceux qui veulent jouer dynamiquement avec des ajustements basés sur des multiples distributions (parfois utile lors de réunions clients), il existe le package **fitur** et sa fonction **fit_dist_addin()**:

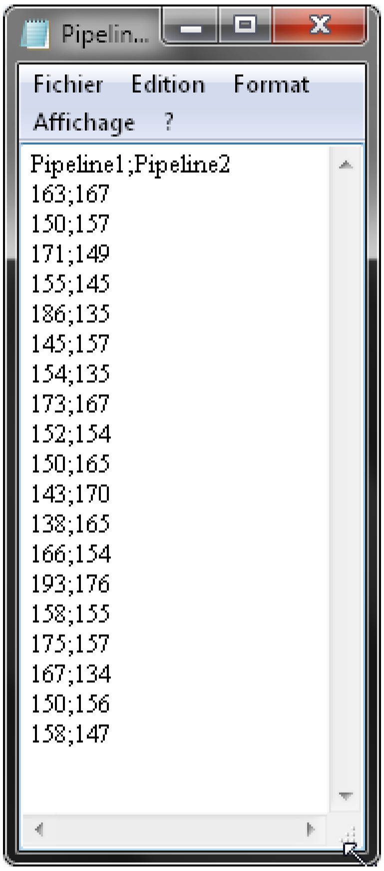


Ce qui donnera:



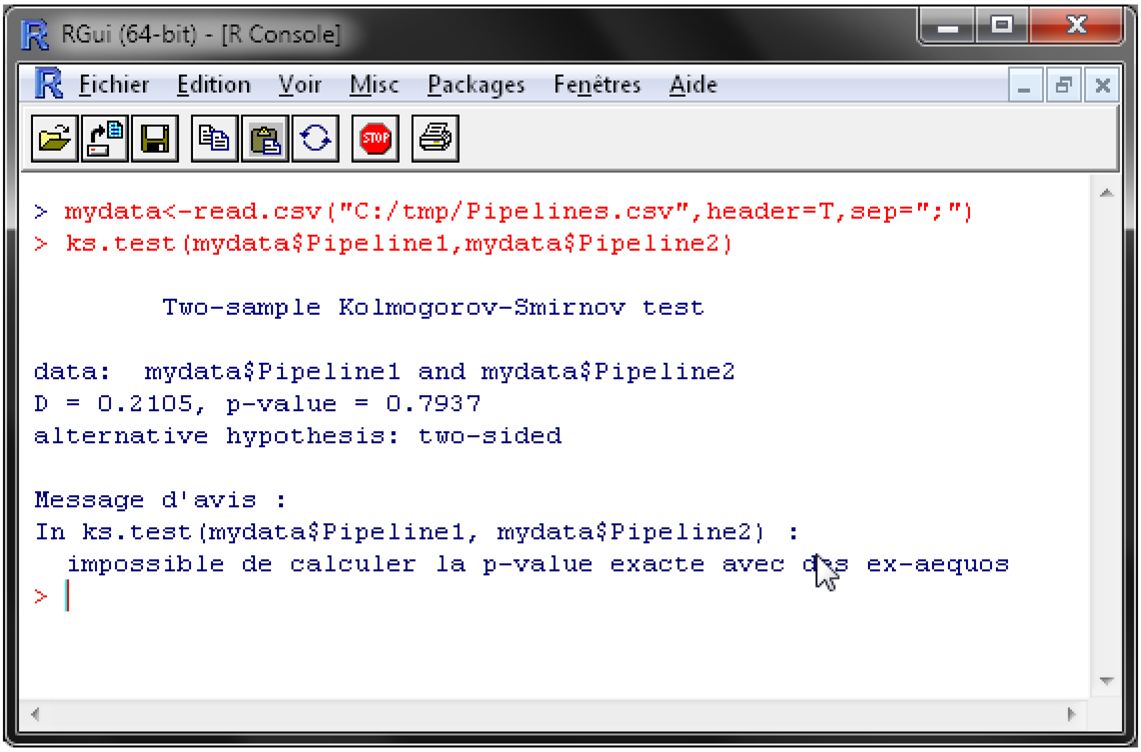
Cas à 2 échantillons

Nous avons mentionné dans le cours théorique l'extension triviale de pouvoir généraliser le test de Kolmogorov-Smirnov à deux échantillons empiriques. Voyons si R propose cela (nous n'avais pas fait de calcul à la main). Considérons le jeu de données suivant:




```
Pipeline1;Pipeline2
163;167
150;157
171;149
155;145
186;135
145;157
154;135
173;167
152;154
150;165
143;170
138;165
166;154
193;176
158;155
175;157
167;134
150;156
158;147
```

Nous souhaitons savoir si leur distribution sont significativement différentes:



Il semblerait donc que non!

Exemple 225.: Meilleur ajustement au sens de la maximisation de la log-vraisemblance

R 3.0.2

Cela n'a rien à voir avec du non-paramétrique mais il nous semble que c'est le bon moment pour donner l'information…

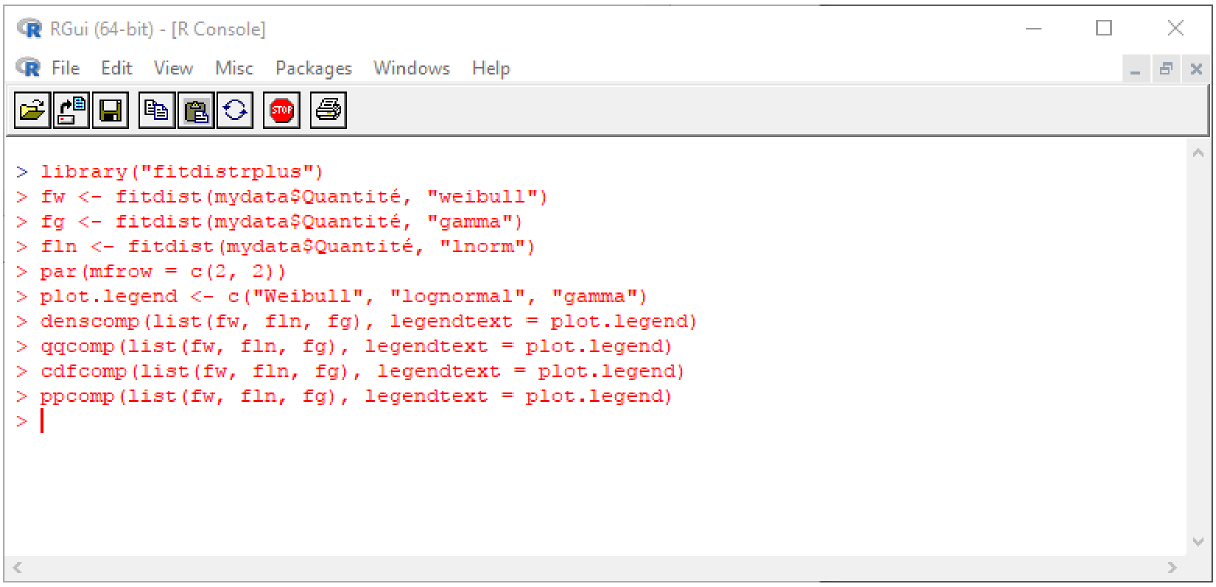
Dans le package **MASS** il y a une commande **fitdistr( )** et qui va chercher avec les techniques classiques d'optimisation non linéaire à maximiser la log-vraisemblance. Voyons cela avec les quantités de notre fichier des ventes d'articles en utilisant un script qui exécutera les optimisations de plusieurs distributions d'un seul coup:

```
> library(MASS)
> mydata<-read.csv("C:/VentesClean.csv",header=T,sep=";")
> x<-mydata$Quantite
> distributions=c("normal","exponential","negative binomial","gamma","exponential")
> for (dist in distributions){
+ print( paste("fitting parameters for ",dist))
+ params=fitdistr(x,dist)
+ print(params)
+ print(params$loglik)
+ }
[1] "fitting parameters for  normal"
      mean          sd
  8.1100917    6.2857629
 (0.6020669)  (0.4257256)
[1] -355.0376
[1] "fitting parameters for  exponential"
      rate
  0.1233032
 (0.0118103)
[1] -337.1489
[1] "fitting parameters for  negative binomial"
      size          mu
  2.6509137    8.1100903
 (0.4612122)  (0.5495764)
[1] -327.8388
[1] "fitting parameters for  gamma"
      shape          rate
  2.01390399    0.24831066
 (0.25344963)  (0.03545991)
[1] -324.2914
[1] "fitting parameters for  exponential"
      rate
  0.1233032
 (0.0118103)
[1] -337.1489
> 
```

Ou quand on doit manipuler des fonctions moins évidentes, il faudra spécifier certaines de leurs paramètres:



Sinon, il y a aussi la fonction **fitdist( )** du package **fitdistrplus** qui permet d'avoir rapidement un visuel sur l'ajustement:



Ce qui donne:

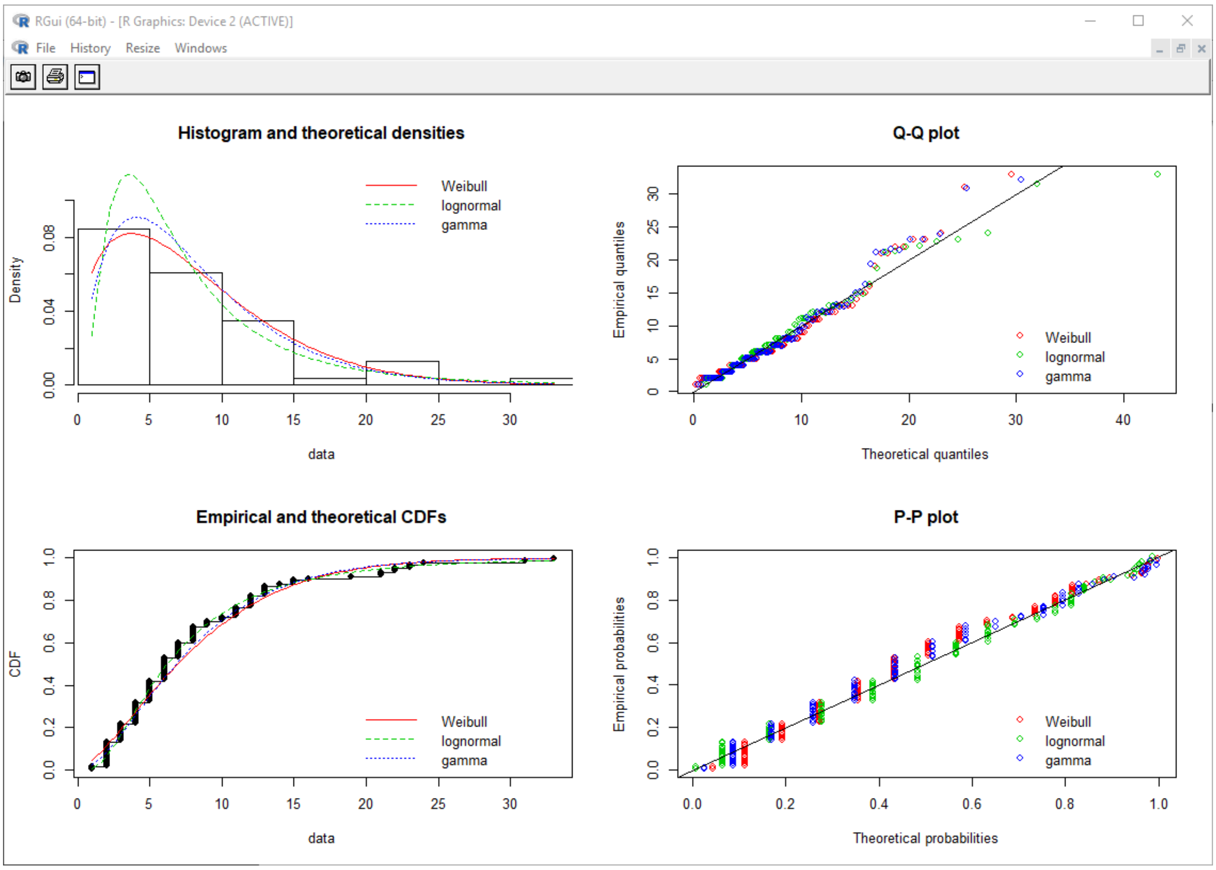


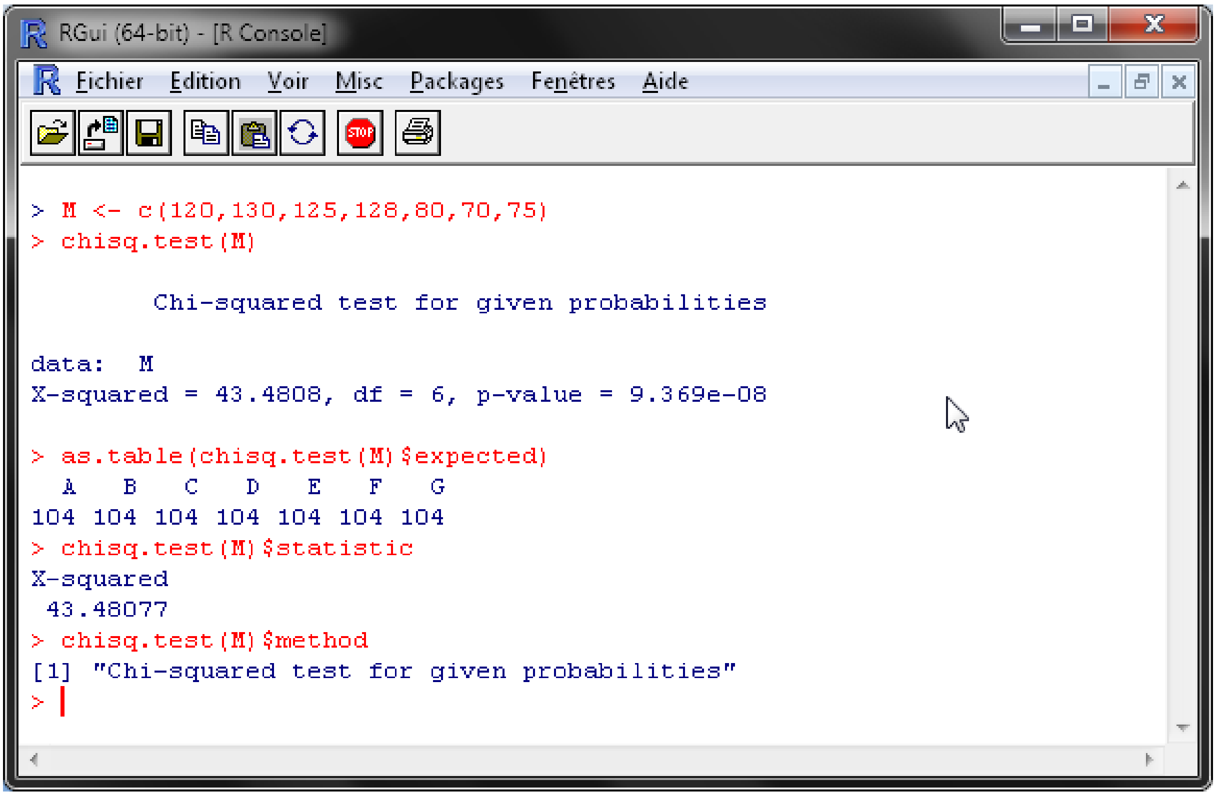
Figure 234 Graph de diagnostique d'ajustements multiples de fonctions de distributions

Exemple 226.: Test d'ajustement (indépendance) du khi-2

R 3.0.2

Là encore nous allons vérifier la conformité des résultats avec toujours les mêmes logiciels et les calculs effectués à la main suite à la démonstration mathématique faite dans le cours théorique.

Nous utilisons donc la commande **chisq.test( )** :



Nous obtenons donc la même chose que les calculs effectués avec Minitab, Microsoft Excel et à la main.

De même si nous spécifions le vecteur des attendus (c'est pas intuitif dans R le fait qu'il faille écrire ainsi dans ce cas particulier1):

```
> O<-c(120,130,125,128,80,70,75)
> E<-c(104,104,104,104,104,104,104)
> chisq.test(O,p=E/sum(E))

        Chi-squared test for given probabilities

data:  O
X-squared = 43.48076923077, df = 6, p-value = 9.36861855241e-08

>
> #on retrouve bien la p-valeur à la main
> 1-pchisq(sum((O-E)^2/E),6)
[1] 9.36861854761162e-08
>
```

Notons que si nous faisons le calcul ainsi:

```
> M <- as.table(rbind(c(120,130,125,128,80,70,75), c(104,104,104,104,104,104,104)))
> dimnames(M) <- list(type = c("Observé", "Attendu"),
+                    jour = c("Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"))
> M
         jour
type      Lundi Mardi Mercredi Jeudi Vendredi Samedi Dimanche
  Observé   120   130      125   128       80     70       75
  Attendu   104   104      104   104      104    104      104
> Xsq <- chisq.test(M)
> Xsq

        Pearson's Chi-squared test

data:  M
X-squared = 22.91270581045, df = 6, p-value = 0.000826249061073

>
```

Même si la conclusion reste la même dans ce cas particulier le résultat diffère significativement. La raison est que ce ne sont pas les mêmes tests qui sont effectués!

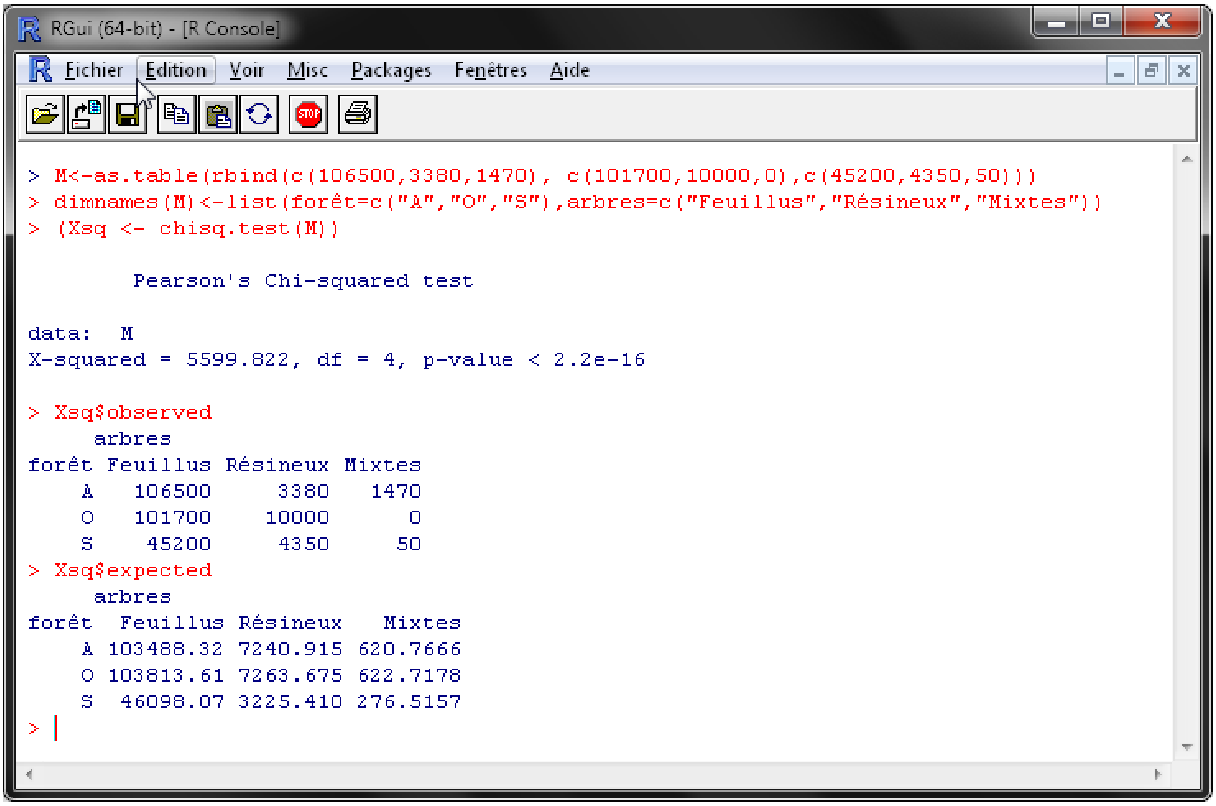
Le premier effectue un test du chi-2 à un facteur, l'autre effectue un test du chi-deux des correspondances de facteurs (C2CFA) comme nous allons le voir de suite au prochain exemple!

Exemple 227.: Test d'indépendance du Khi-deux d'une table de contingence (C2CFA)

R 3.0.2

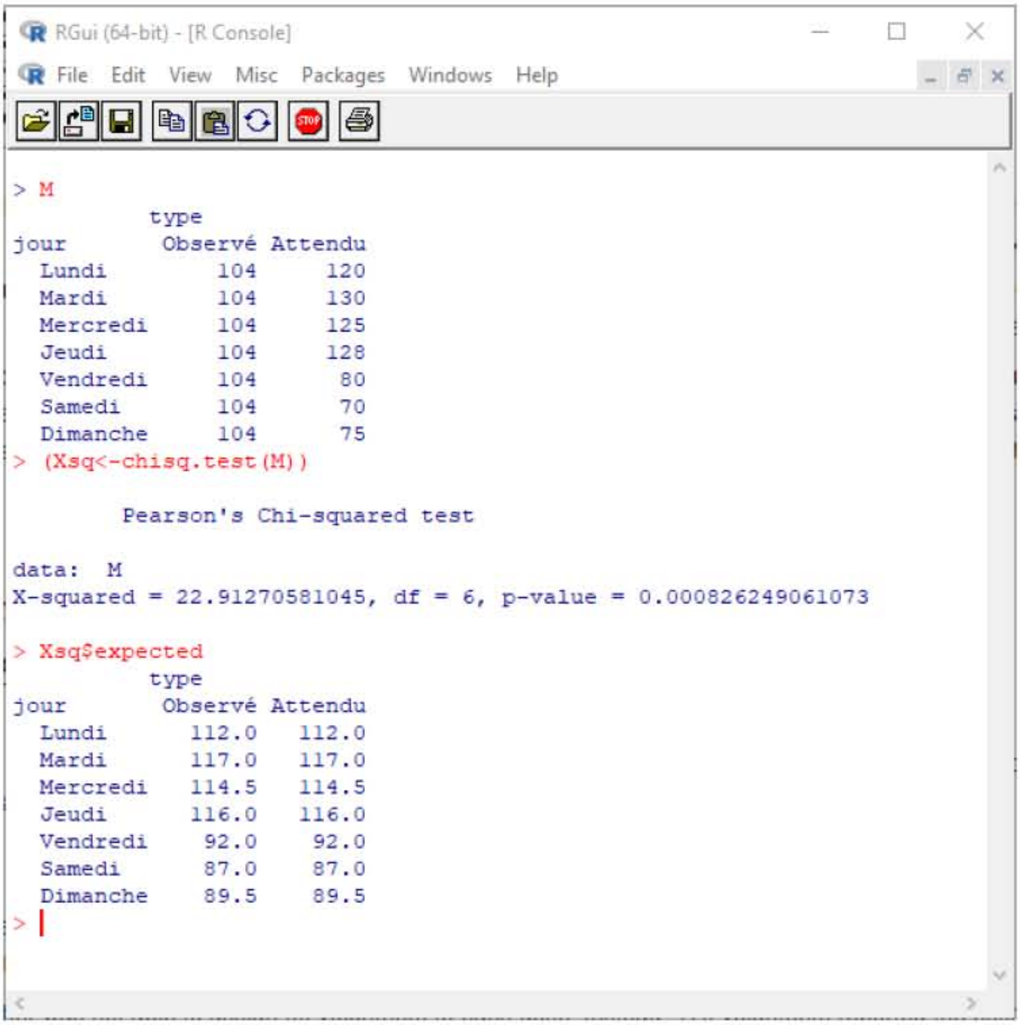
Là encore nous allons vérifier la conformité des résultats avec toujours les mêmes logiciels et les calculs effectués à la main suite à la démonstration mathématique faite dans le cours théorique.

Nous utilisons donc toujours la commande **chisq.test( )** :



Les résultats sont donc conformes à la différence que la *p*-value est extrêmement précise dans R...

Si nous faisons de même avec les valeurs attendues du précédent exemple, tout s'explique pour le coup:



```
> M
          type
jour       Observé Attendu
  Lundi        104     120
  Mardi        104     130
  Mercredi     104     125
  Jeudi        104     128
  Vendredi     104      80
  Samedi       104      70
  Dimanche     104      75
> (Xsq<-chisq.test(M))

        Pearson's Chi-squared test

data:  M
X-squared = 22.91270581045, df = 6, p-value = 0.000826249061073

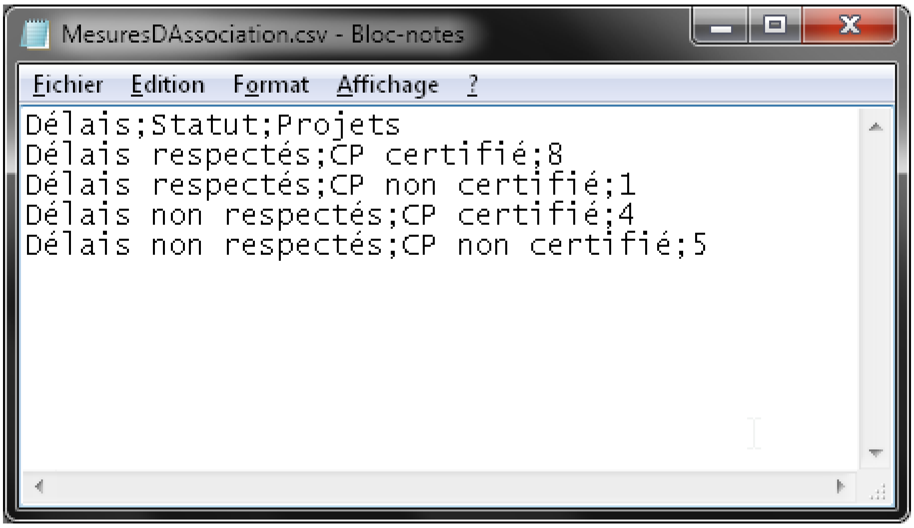
> Xsq$expected
          type
jour       Observé Attendu
  Lundi      112.0   112.0
  Mardi      117.0   117.0
  Mercredi   114.5   114.5
  Jeudi      116.0   116.0
  Vendredi    92.0    92.0
  Samedi      87.0    87.0
  Dimanche    89.5    89.5
>
```

Exemple 228.: Test exact de Fisher

R 3.0.2

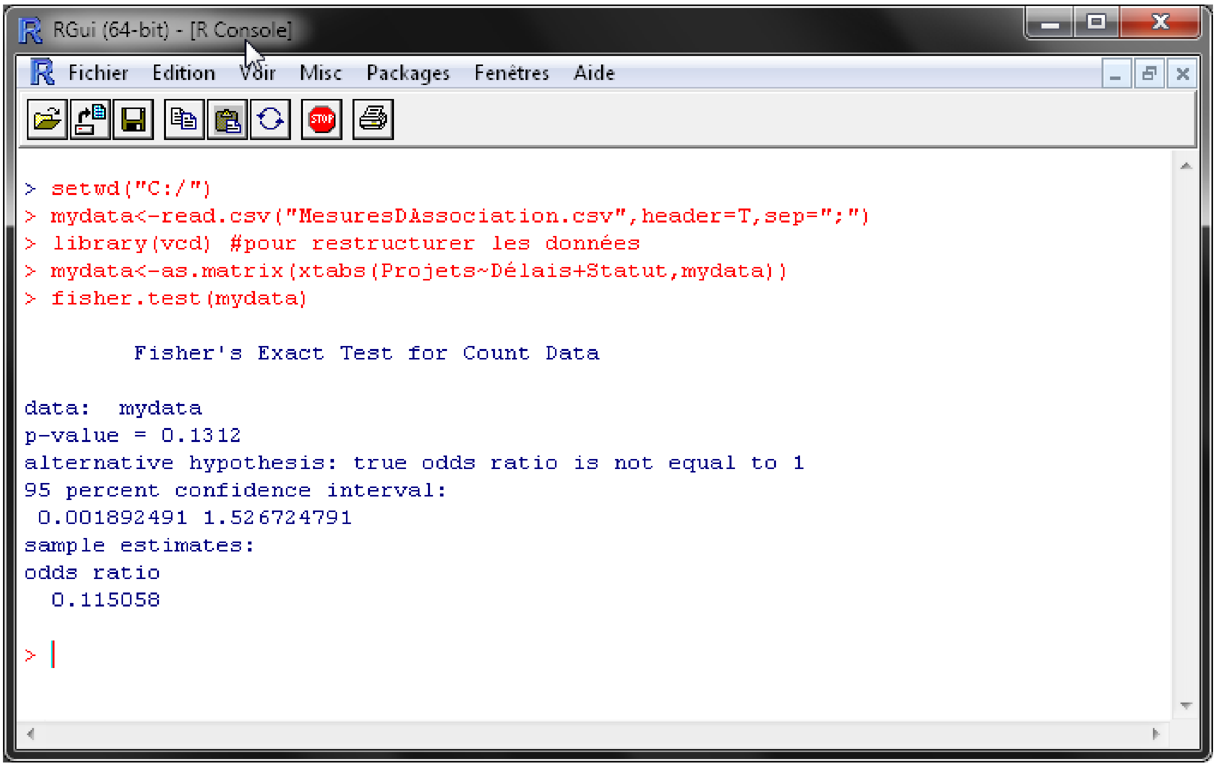
Même si nous l'avons vu précédemment, nous allons toutefois refaire l'exemple vu dans le cours théorique et que nous avions contrôlé à la main avec Microsoft Excel et Minitab.

Nous prenons les données suivantes (les mêmes que dans le cours théorique) avec la commande **fisher.test( )** :




```
Délais;Statut;Projets
Délais respectés;CP certifié;8
Délais respectés;CP non certifié;1
Délais non respectés;CP certifié;4
Délais non respectés;CP non certifié;5
```

Nous avons alors:




```
> setwd("C:/")
> mydata<-read.csv("MesuresDAssociation.csv",header=T,sep=";")
> library(vcd) #pour restructurer les données
> mydata<-as.matrix(xtabs(Projets~Délais+Statut,mydata))
> fisher.test(mydata)

        Fisher's Exact Test for Count Data

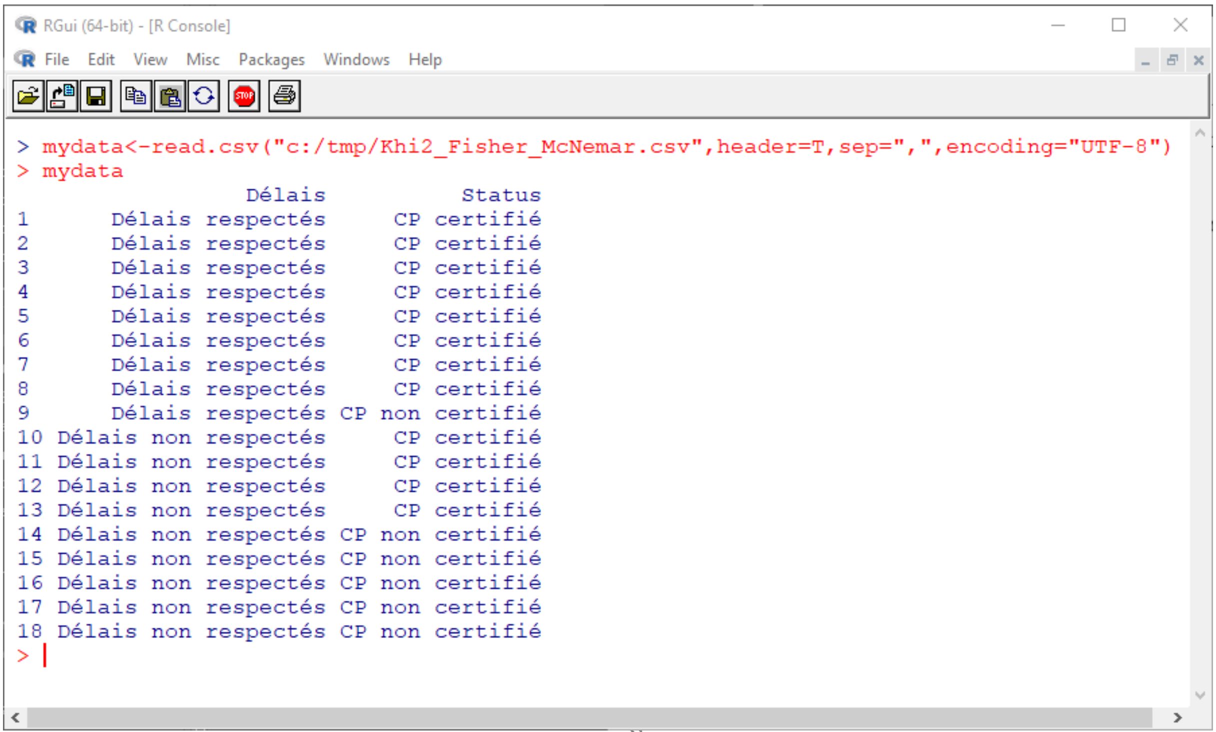
data:  mydata
p-value = 0.1312
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.001892491 1.526724791
sample estimates:
odds ratio
  0.115058

> 
```

Nous nous retrouvons avec la même p-value que dans Minitab et Microsoft Excel à la différence que les autres informations n'étaient pas données par Minitab.

Le test exact de Fisher peut bien évidemment être utilisé en machine learning pour mesurer la performance de classification binaire!

Sinon avec le package **gmodels** et la fonction **CrossTable( )**, nous avons avec le jeu de données suivant:



Le résultat statistique suivant:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> CrossTable(mydata$Délais,mydata$Status,fisher=TRUE)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  18


                     | mydata$Status
       mydata$Délais |     CP certifié | CP non certifié |       Row Total |
---------------------|-----------------|-----------------|-----------------|
Délais non respectés |               4 |               5 |               9 |
                     |           0.667 |           1.333 |                 |
                     |           0.444 |           0.556 |           0.500 |
                     |           0.333 |           0.833 |                 |
                     |           0.222 |           0.278 |                 |
---------------------|-----------------|-----------------|-----------------|
    Délais respectés |               8 |               1 |               9 |
                     |           0.667 |           1.333 |                 |
                     |           0.889 |           0.111 |           0.500 |
                     |           0.667 |           0.167 |                 |
                     |           0.444 |           0.056 |                 |
---------------------|-----------------|-----------------|-----------------|
        Column Total |              12 |               6 |              18 |
                     |           0.667 |           0.333 |                 |
---------------------|-----------------|-----------------|-----------------|

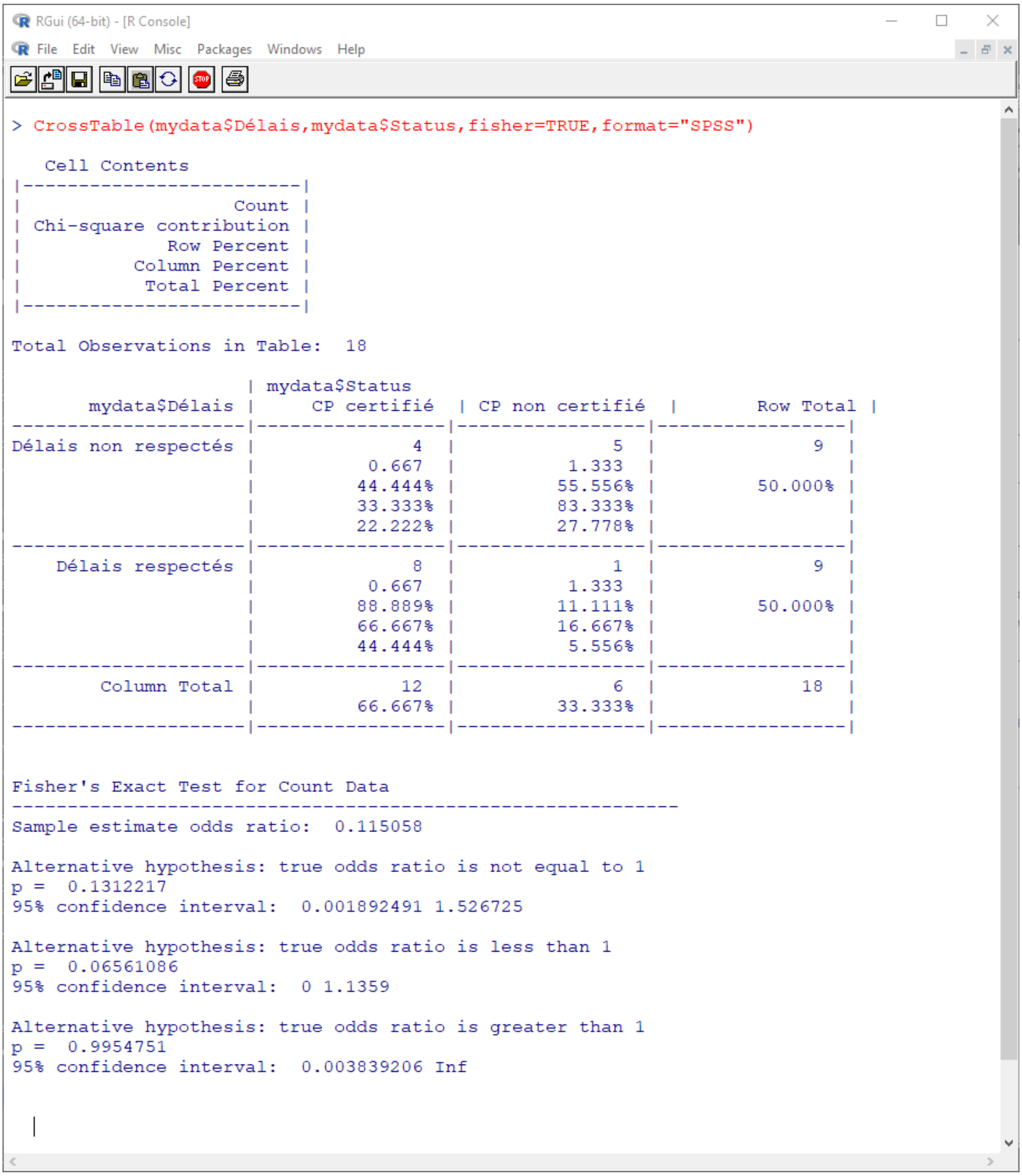

Fisher's Exact Test for Count Data
------------------------------------------------------------
Sample estimate odds ratio:  0.115058

Alternative hypothesis: true odds ratio is not equal to 1
p =  0.1312217
95% confidence interval:  0.001892491 1.526725

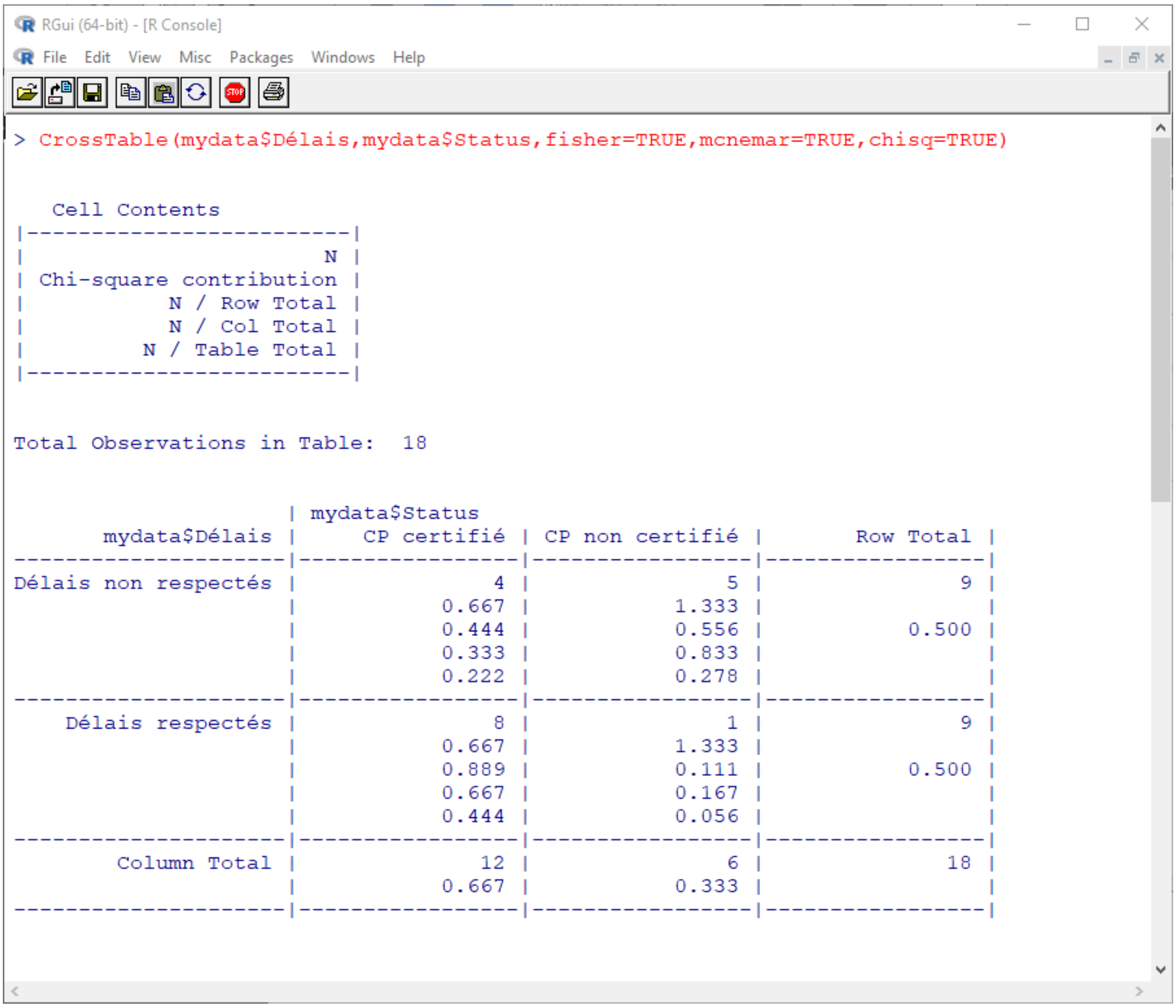
Alternative hypothesis: true odds ratio is less than 1
p =  0.06561086
95% confidence interval:  0 1.1359

Alternative hypothesis: true odds ratio is greater than 1
p =  0.9954751
95% confidence interval:  0.003839206 Inf
```

On peut avoir un format SPSS aussi… :



On peut aussi exécuter trois tests en même temps, ce qui donne d'abord la première partie classique:



```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

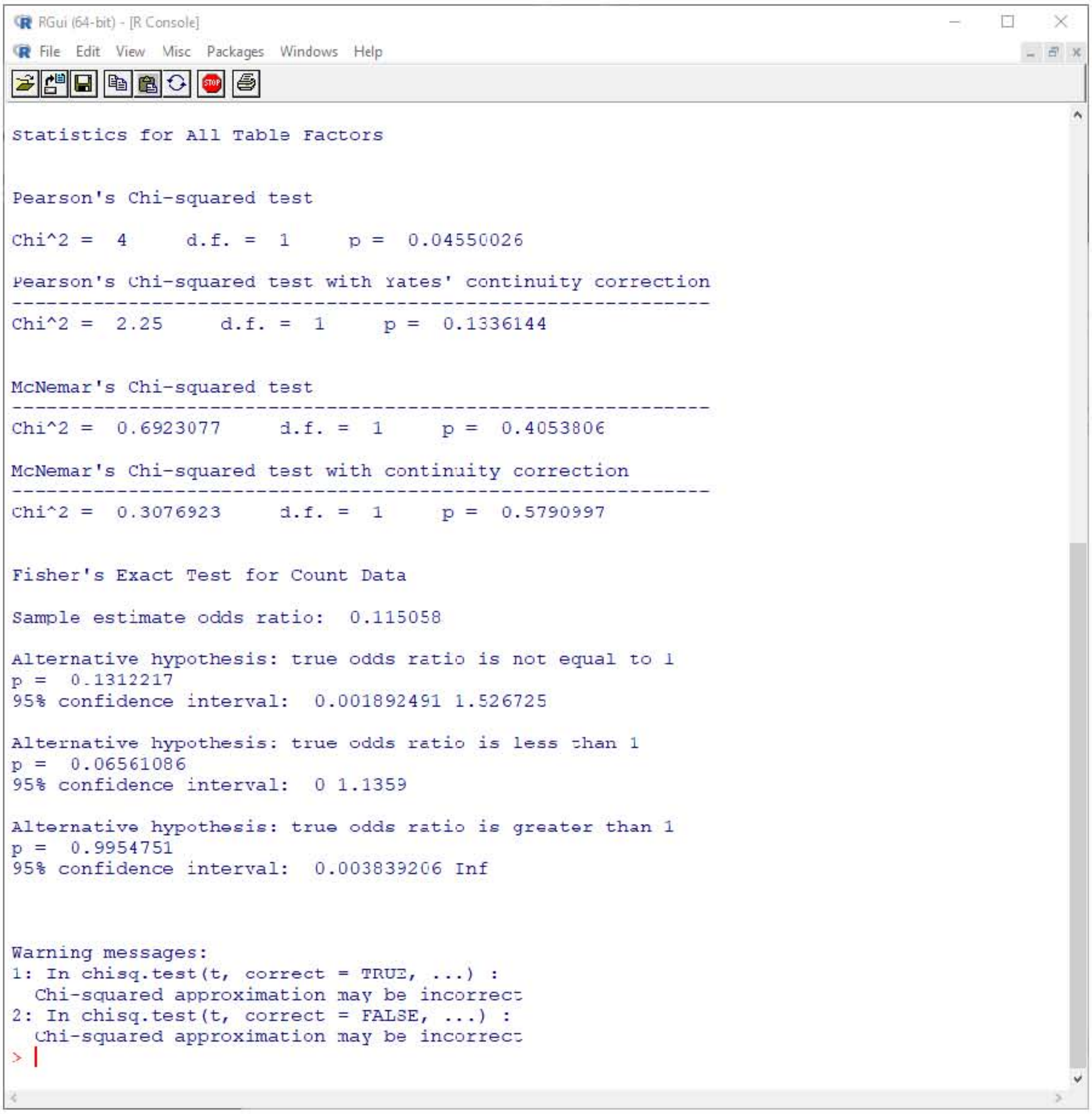
> CrossTable(mydata$Délais,mydata$Status,fisher=TRUE,mcnemar=TRUE,chisq=TRUE)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  18


                     | mydata$Status
       mydata$Délais |     CP certifié | CP non certifié |       Row Total |
---------------------|-----------------|-----------------|-----------------|
Délais non respectés |               4 |               5 |               9 |
                     |           0.667 |           1.333 |                 |
                     |           0.444 |           0.556 |           0.500 |
                     |           0.333 |           0.833 |                 |
                     |           0.222 |           0.278 |                 |
---------------------|-----------------|-----------------|-----------------|
    Délais respectés |               8 |               1 |               9 |
                     |           0.667 |           1.333 |                 |
                     |           0.889 |           0.111 |           0.500 |
                     |           0.667 |           0.167 |                 |
                     |           0.444 |           0.056 |                 |
---------------------|-----------------|-----------------|-----------------|
        Column Total |              12 |               6 |              18 |
                     |           0.667 |           0.333 |                 |
---------------------|-----------------|-----------------|-----------------|
```

Et enfin la partie spécifique d'intérêt!




```
Statistics for All Table Factors


Pearson's Chi-squared test

Chi^2 =  4      d.f. =  1      p =  0.04550026

Pearson's Chi-squared test with Yates' continuity correction
------------------------------------------------------------
Chi^2 =  2.25      d.f. =  1      p =  0.1336144


McNemar's Chi-squared test
------------------------------------------------------------
Chi^2 =  0.6923077      d.f. =  1      p =  0.4053806

McNemar's Chi-squared test with continuity correction
------------------------------------------------------------
Chi^2 =  0.3076923      d.f. =  1      p =  0.5790997


Fisher's Exact Test for Count Data

Sample estimate odds ratio:  0.115058

Alternative hypothesis: true odds ratio is not equal to 1
p =  0.1312217
95% confidence interval:  0.001892491 1.526725

Alternative hypothesis: true odds ratio is less than 1
p =  0.06561086
95% confidence interval:  0 1.1359

Alternative hypothesis: true odds ratio is greater than 1
p =  0.9954751
95% confidence interval:  0.003839206 Inf



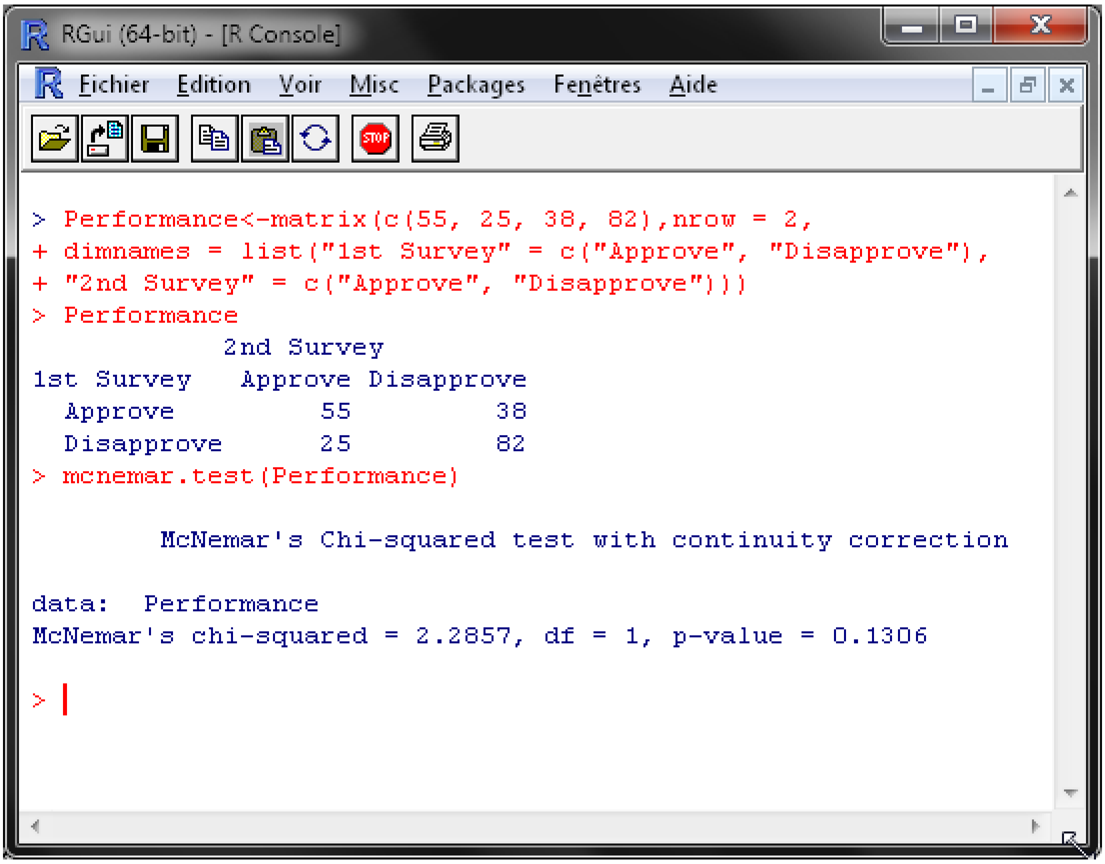
Warning messages:
1: In chisq.test(t, correct = TRUE, ...) :
  Chi-squared approximation may be incorrect
2: In chisq.test(t, correct = FALSE, ...) :
  Chi-squared approximation may be incorrect
>
```

Exemple 229.: Test de McNemar

R 3.0.2

Nous allons ici encore une fois recalculer l'exemple vu dans le cours théorique et que nous avions contrôlé à la main avec Microsoft Excel et Minitab.

Nous utilisons alors la commande native **mcnemar.test( )** de R:



Nous pouvons faire quelques graphes dont je ne suis vraiment, mais alors vraiment pas un fan!

Comme le four fold avec la fonction **fourfoldplot( )** qui est assez curieusement dans le package de base **graphics** de R:

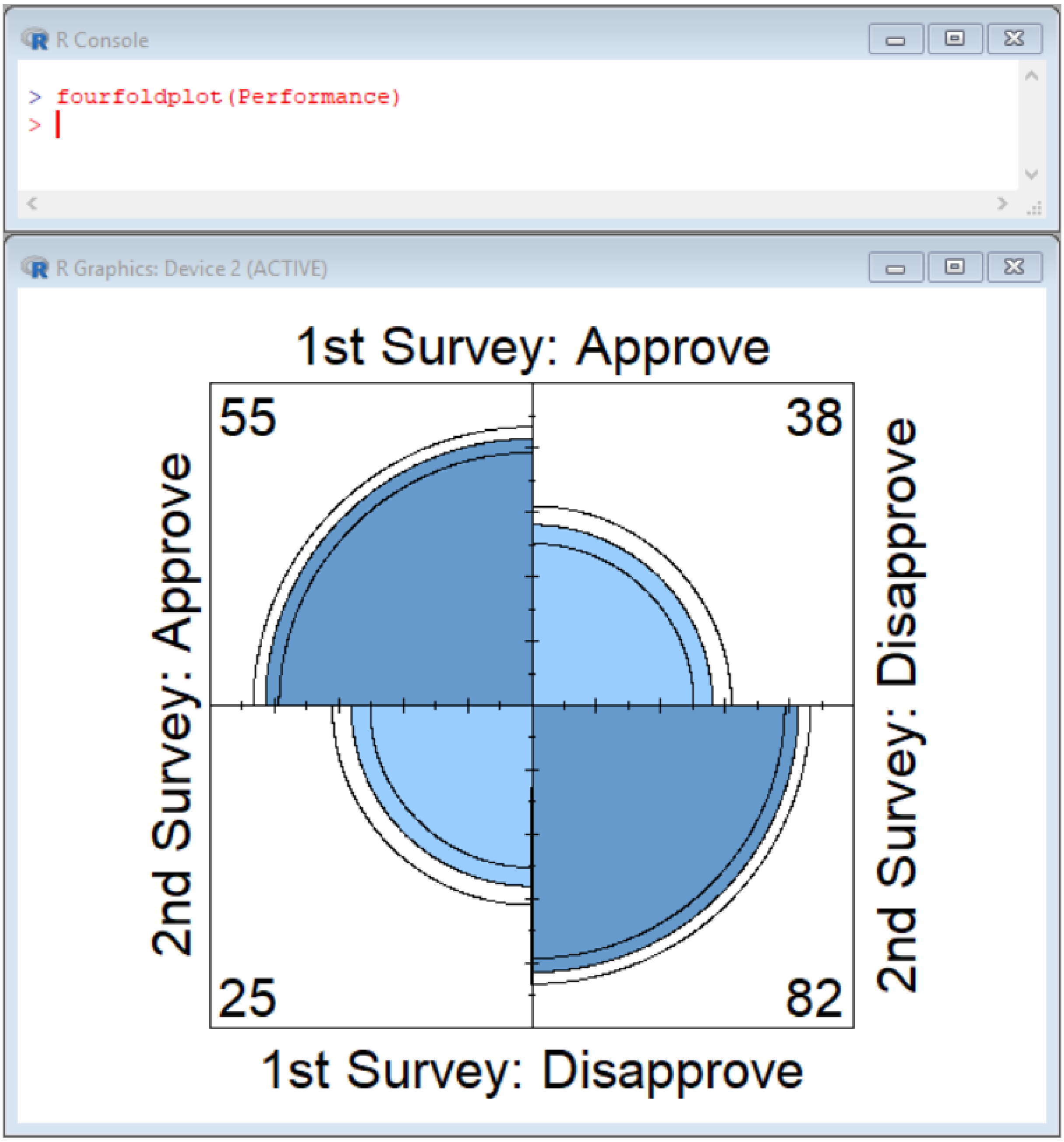


Figure 235 Graph four-fold

ou le plot d'agrément avec la fonction **agreementplot( )** du package **vcd**:

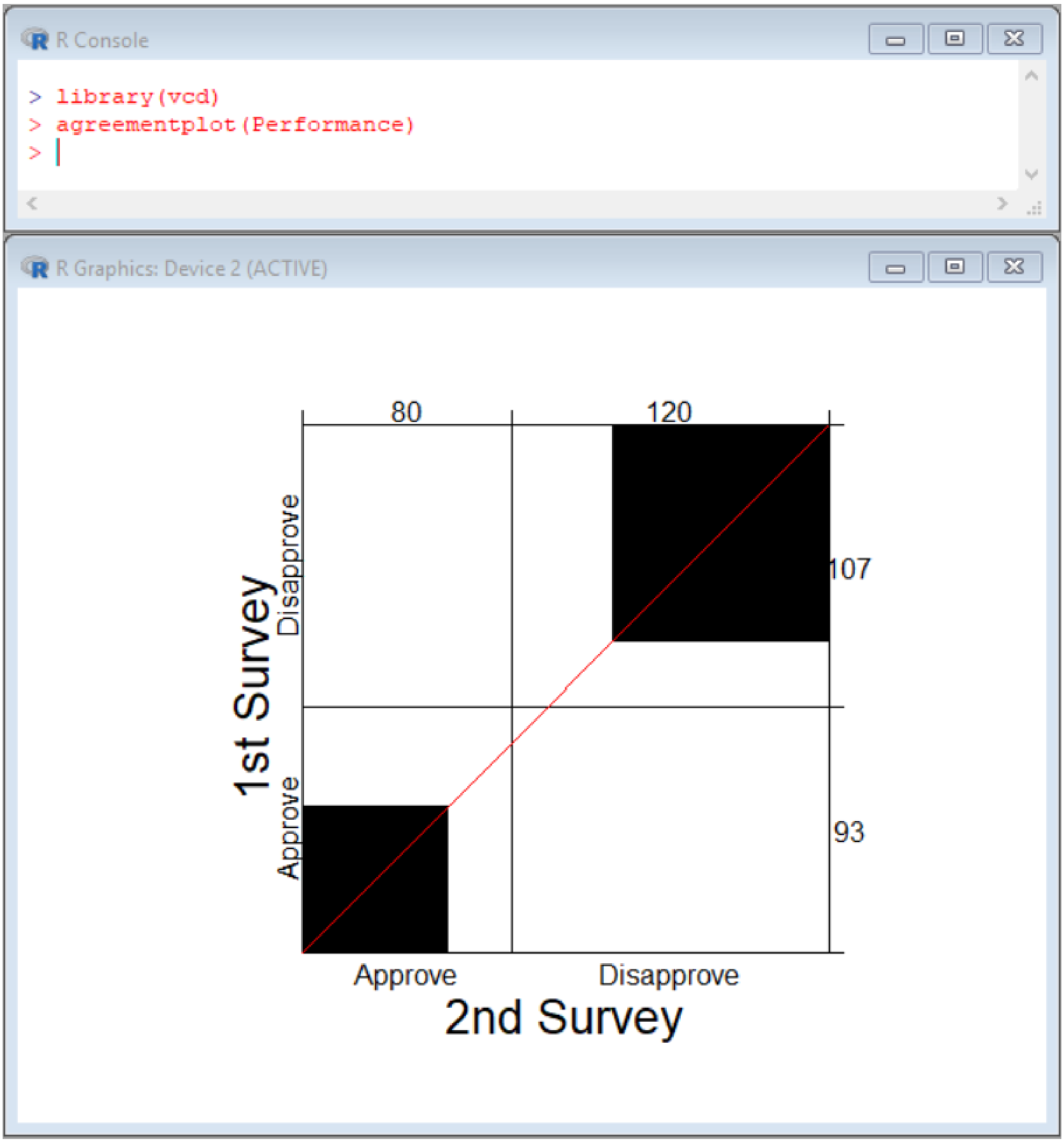


Figure 236 Graph d'agrément

On peut utiliser ces deux graphiques dans le cadre des matrices de confusion en Machine Learning bien évidemment!

Exemple 230.: Test d'ajustement du khi-deux avec correction de Yates

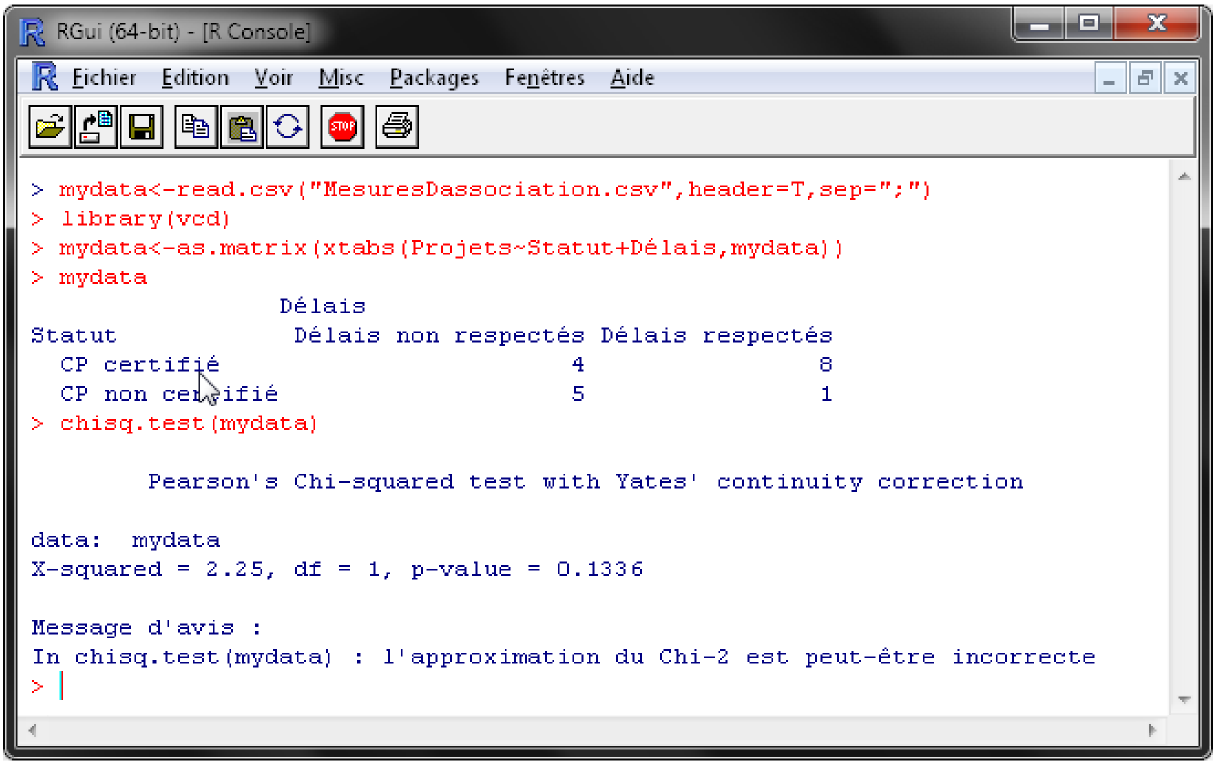
R 3.0.2

Bon nous n'allons nous pas nous étendre sur cette technique empirique qui est sujet à débat et que de plus nous n'avons pas validé avec les techniques de Monte-Carlo dans le cours théorique et qui n'est pas disponible dans Minitab.

Rappelons que cette technique ne serait approximativement valable que pour des tableaux de contingences 2×2 dont certaines observations sont inférieures aux 5 unités et l'ensemble des observations inférieures à 20 unités (d'après les tests qui auraient été faits par plusieurs personnes de la communauté des praticiens de la statistique).

Pour cet exemple nous allons reprendre la petite table utilisée pour le test de Fisher. Évidemment c'est erroné de la faire puisque les différentes cases de ce tableau de contingence sont dépendantes entre elles. Mais bon... c'est juste pour montrer que R va voir qu'il y a un faible nombre d'observations et va donc in extenso appliquer la correction de Yates.

Nous utilisons donc à nouveau la commande **chisq.test( )** :



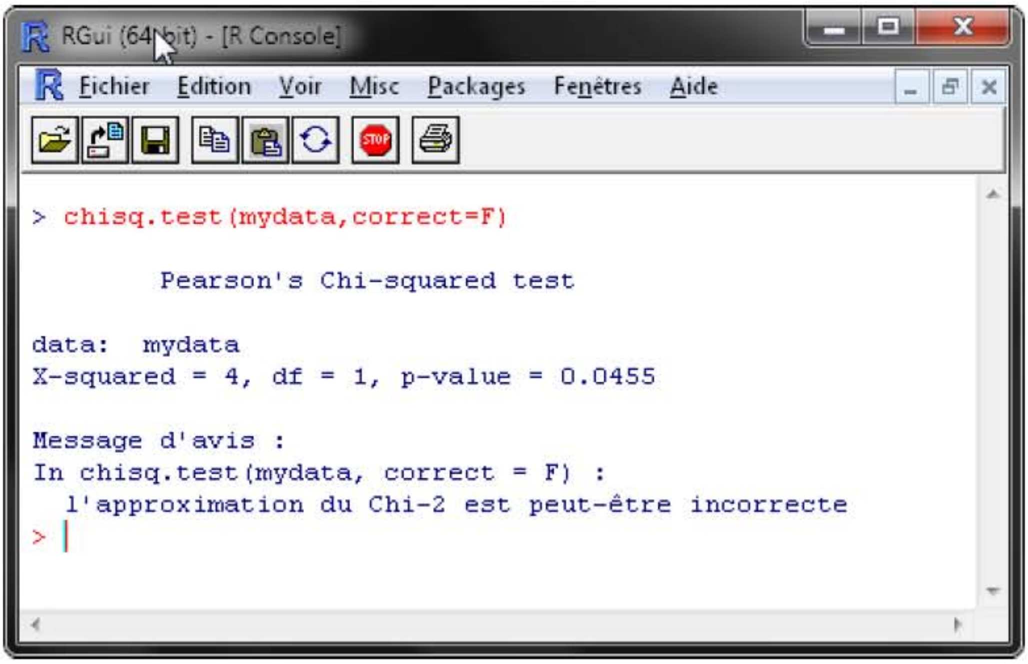
```
> mydata<-read.csv("MesuresDassociation.csv",header=T,sep=";")
> library(vcd)
> mydata<-as.matrix(xtabs(Projets~Statut+Délais,mydata))
> mydata
                 Délais
Statut             Délais non respectés Délais respectés
  CP certifié                         4                8
  CP non certifié                     5                1
> chisq.test(mydata)

        Pearson's Chi-squared test with Yates' continuity correction

data:  mydata
X-squared = 2.25, df = 1, p-value = 0.1336

Message d'avis :
In chisq.test(mydata) : l'approximation du Chi-2 est peut-être incorrecte
> 
```

Nous pouvons forcer l'interdiction d'utiliser la correction de Yates:

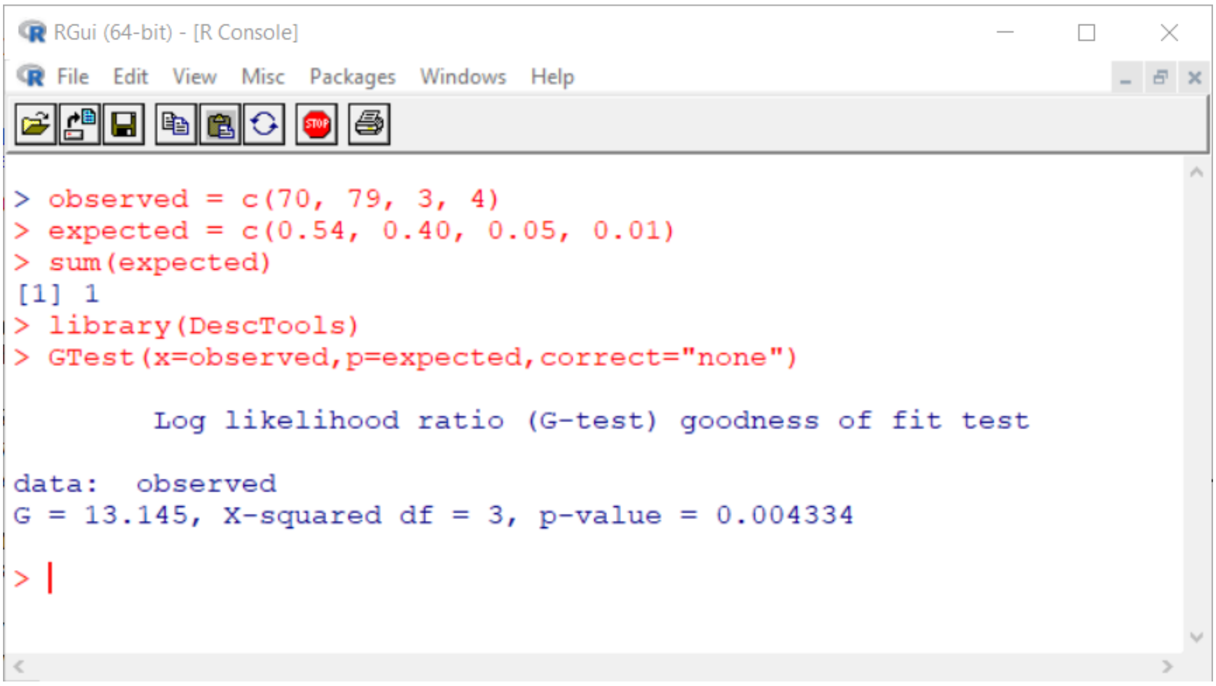

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> chisq.test(mydata,correct=F)
Pearson's Chi-squared test
data: mydata
X-squared = 4, df = 1, p-value = 0.0455
Message d'avis :
In chisq.test(mydata, correct = F) :
l'approximation du Chi-2 est peut-être incorrecte
>

Exemple 231.: G-Test d'ajustement sans correction de Yates

R 3.4.3

Le but ici est de vérifier que le test dont nous avons démontré l'origine mathématique dans le cours théorique existe dans R.

Pour exécuter ce test nous allons utiliser la fonction **GTest( )** du package **DescTools**:



Exemple 232.: Coefficient de corrélation de Spearmann

R 3.2.1

Le lecteur pourra se référer pour ce dernier à la page 1576.

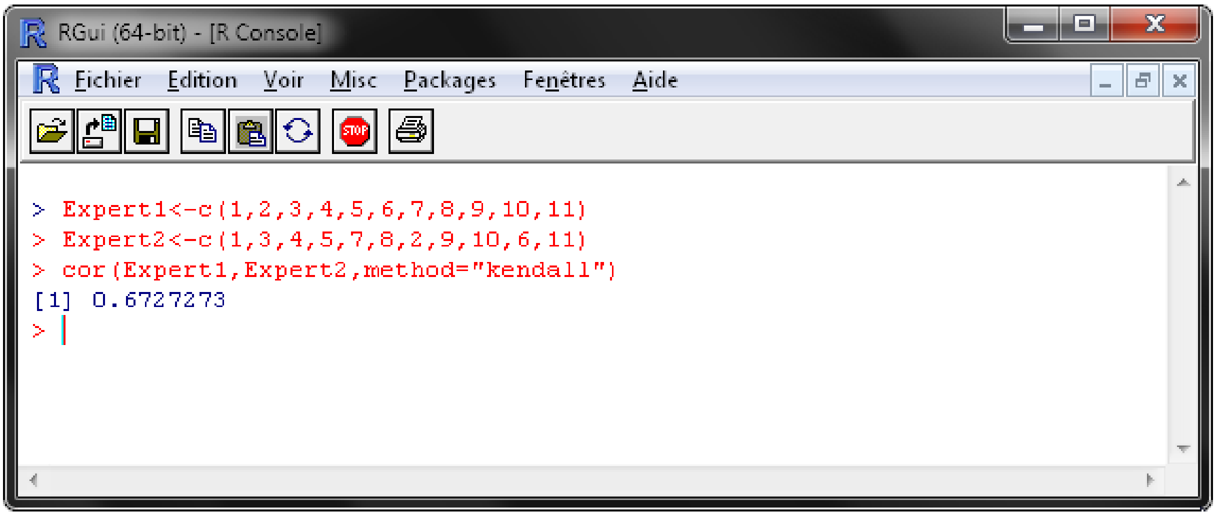
Exemple 233.: Mesures d'associations

Coefficient de corrélation de concordances des rangs de Kendall-tau

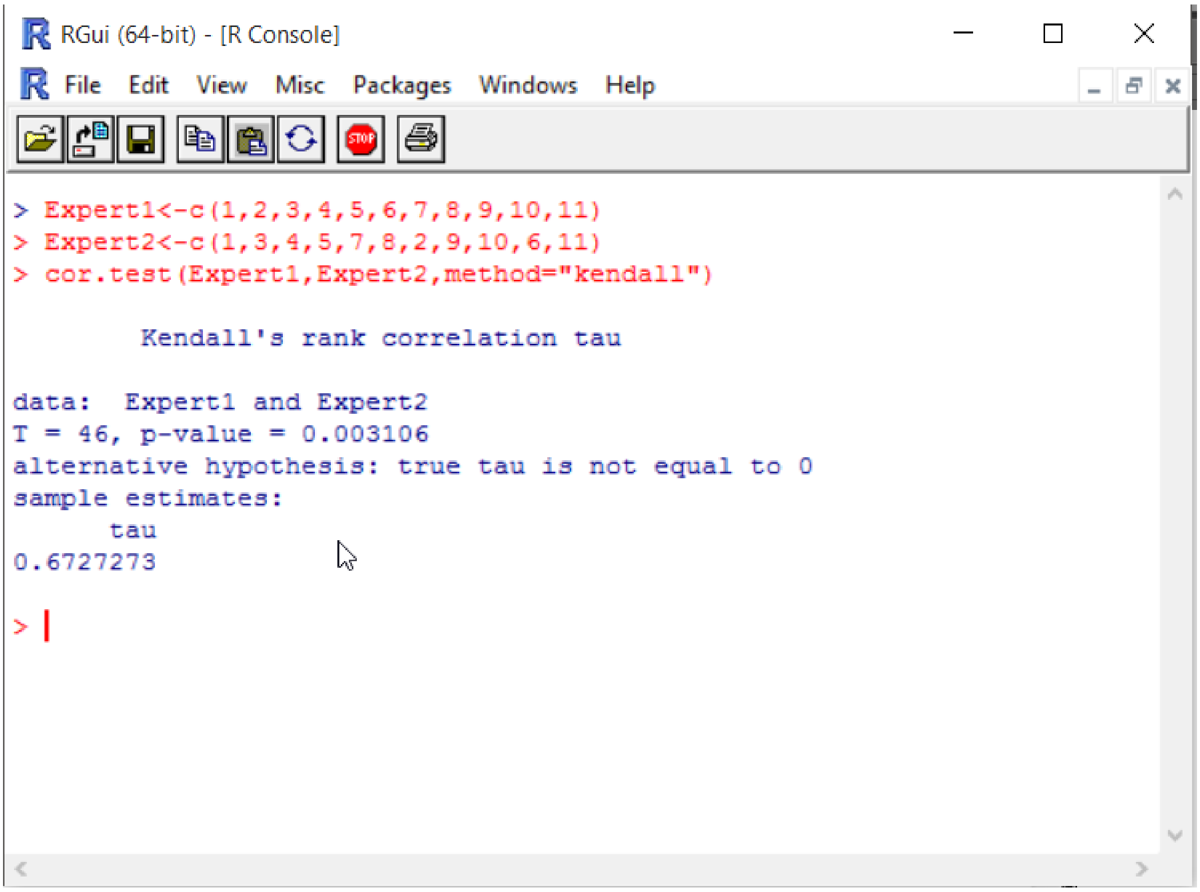
R 3.0.2

Nous avons donc deux experts qui ont notés des vins. Les notes sont rangées dans l'ordre croissant des notes et selon le numéro d'étiquette d'identifiant du vin (bref le même exemple que nous avons utilisé après avoir fait les démonstrations mathématiques de cet indicateur dans le cours théorique).

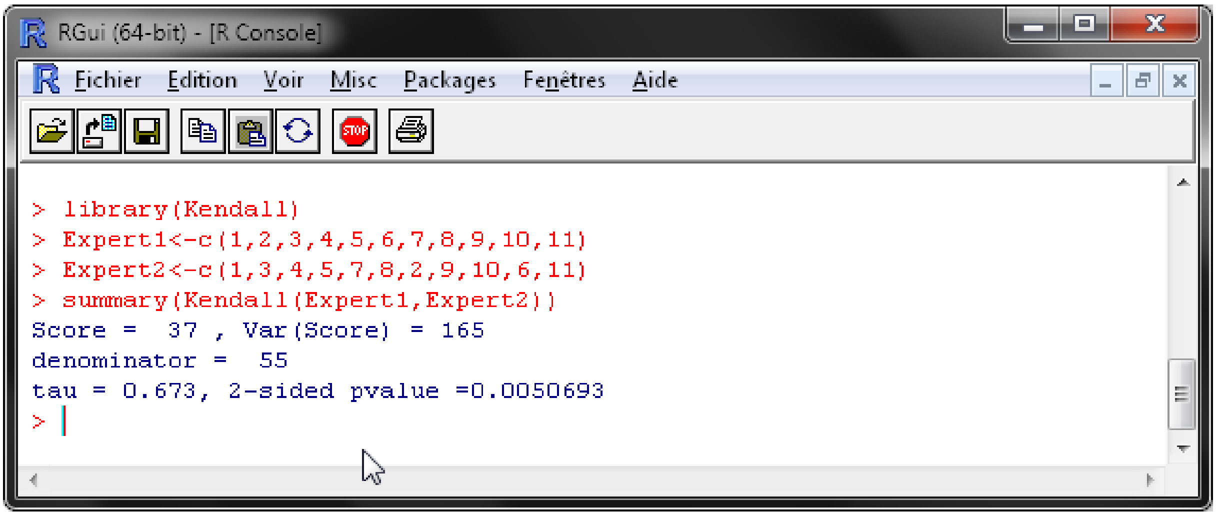
Nous avons alors en utilisant la commande **cor( )** avec le paramètre **kendall**:



Le résultat est juste mais c'est dommage qu'il n'y ait pas d'intervalle de confiance ni de *p*-value. Pour cela nous utilisons la fonction native:

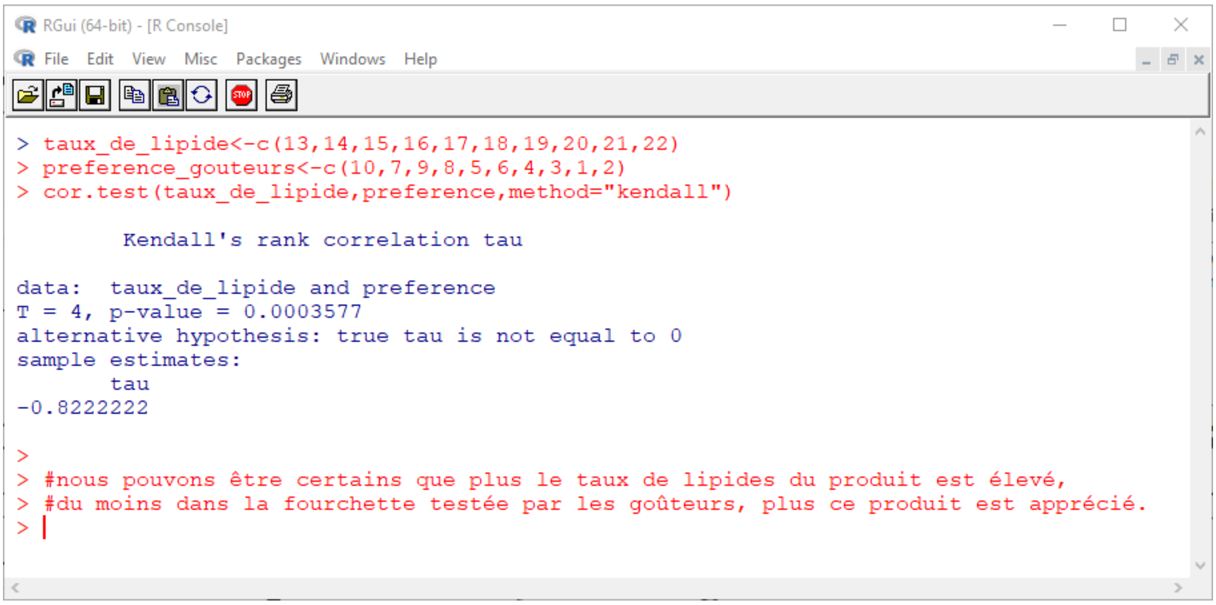


ou avec la commande **Kendall( )** du package **Kendall**:



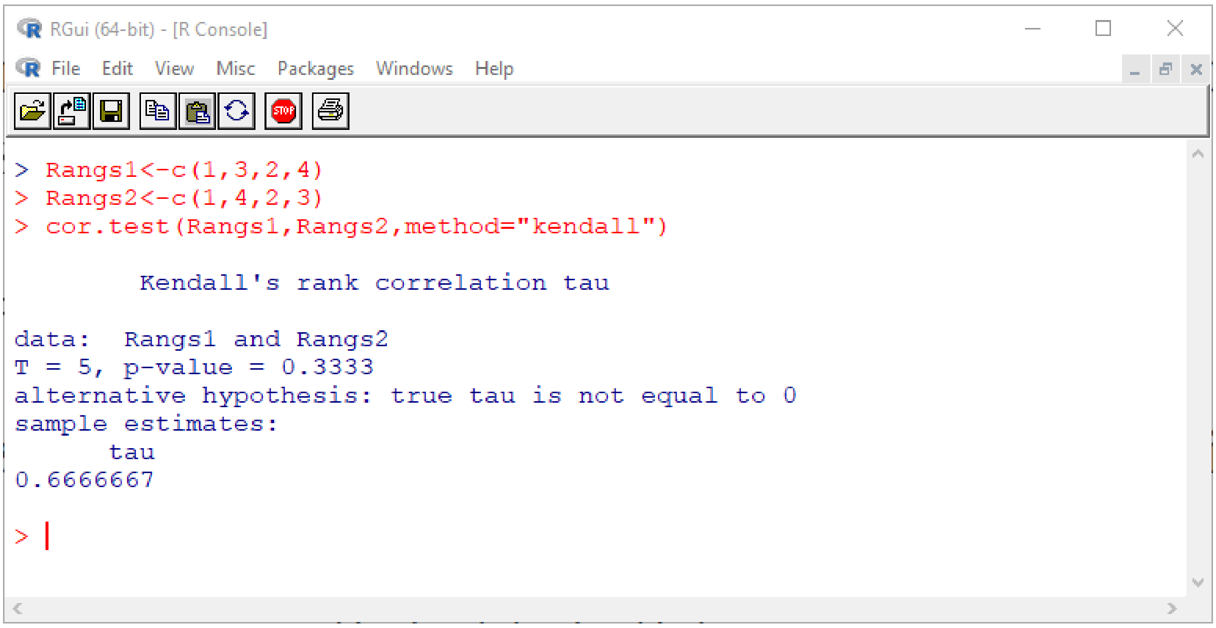
Par contre pour la p-value nous sommes loin du compte (même si la conclusion reste la même...).

Vérifions avec un autre exemple:



Nous obtenons donc bien le même résultat qu'avec le logiciel XLStat.

Sinon pour refaire l'exemple que nous avons fait à la main dans le cours théorique:

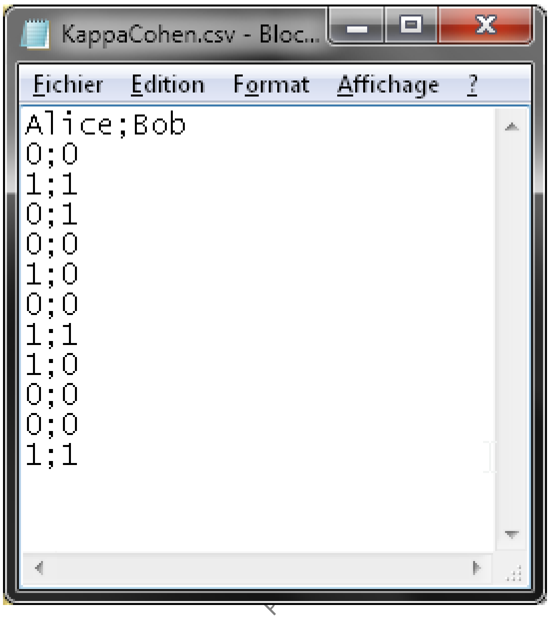


Kappa de Cohen

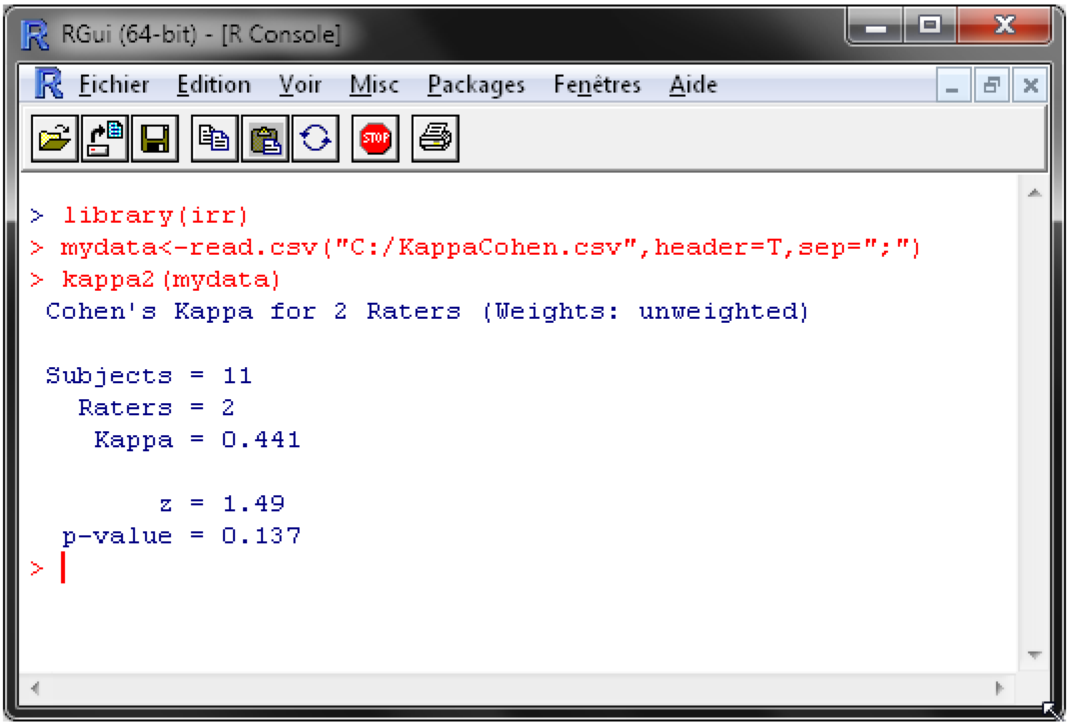
R 3.0.2

Encore une fois, le but est d'appliquer les calculs et démonstrations faits à la main pendant le cours théorique de méthodes numériques et de vérifier que nous retrouvons les mêmes valeurs avec R (et comparer par la même occasion avec Minitab).

Nous utilisons les mêmes données que dans le cours théorique:



Et nous utilisons la commande **kappa2( )** du package **irr**:



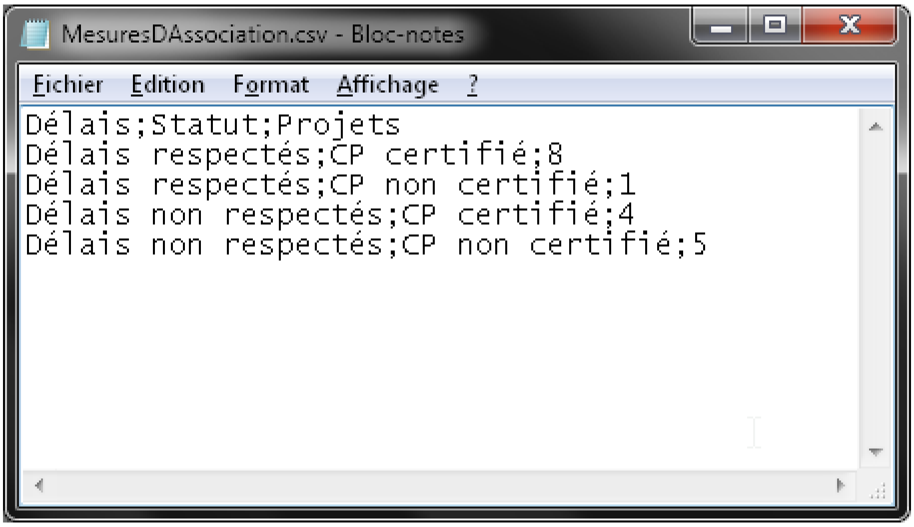
Nous retrouvons donc la même valeur de Kappa et de Z que ce que nous avons calculé à la main et obtenu avec Minitab. Par contre la *p*-value est doublée dans R par rapport à celle de Minitab. Cela signifie simplement que Minitab fait le test en bilatéral alors que R le fait en unilatéral (ce dernier choix me semble personnellement plus pertinent).

Le Kappa de Cohen peut bien évidemment être utilisé en machine learning pour mesurer la performance de classification binaire!

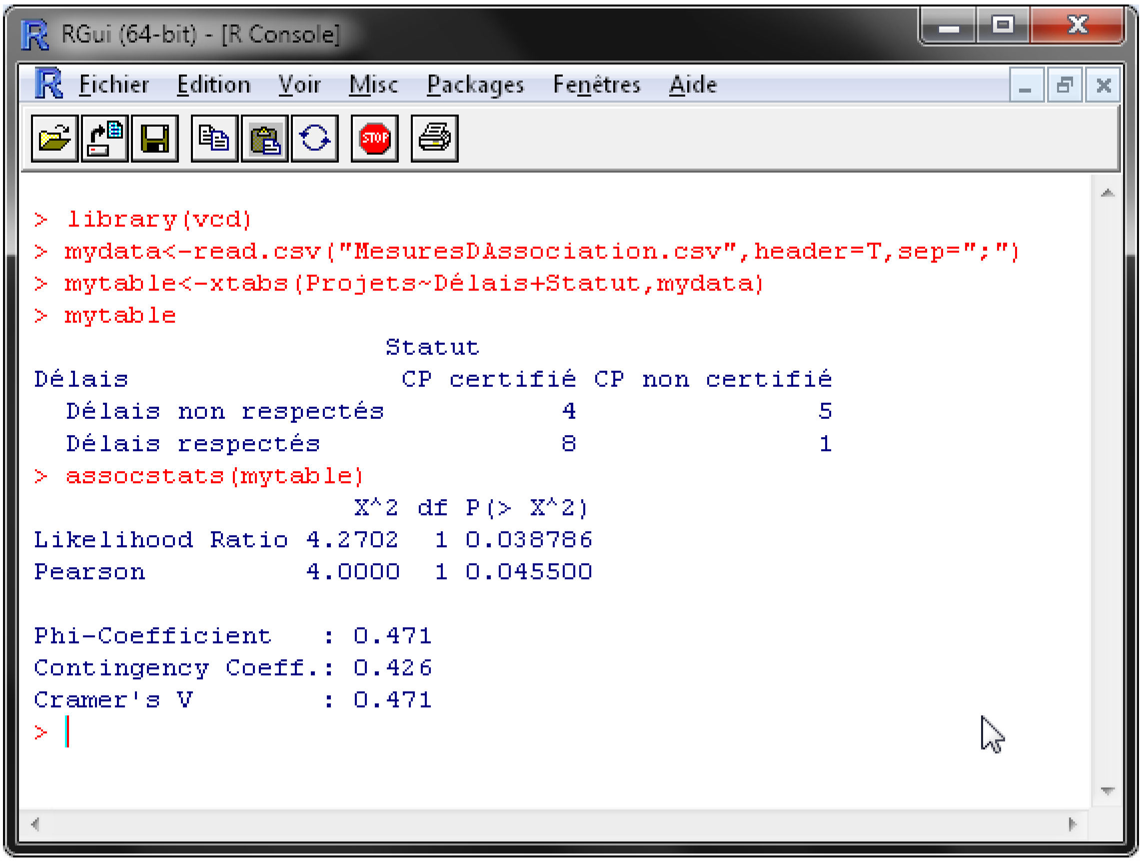
V de Cramér (mesure d'association) et coefficient de corrélation phi de Pearson

R 3.0.2

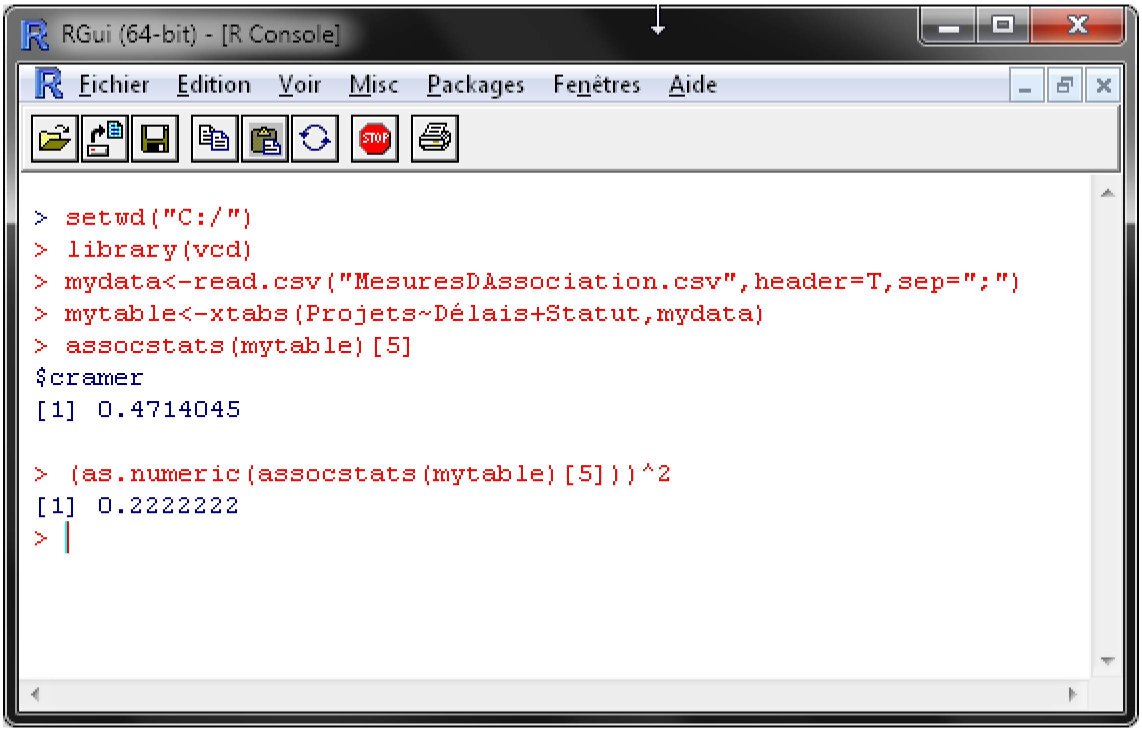
Nous partons des mêmes données que dans le cours théorique:



et nous utilisons la commande **assoctats( )** du package **vcd**:



Et nous n'obtenons pas du tout le même résultat que dans le cours théorique! Mais la même chose qu'avec Minitab mais c'est normal. Effectivement pour réobtenir la même chose que Minitab, il nous faudra écrire:



et voilà! Idem pour le phi de Pearson.

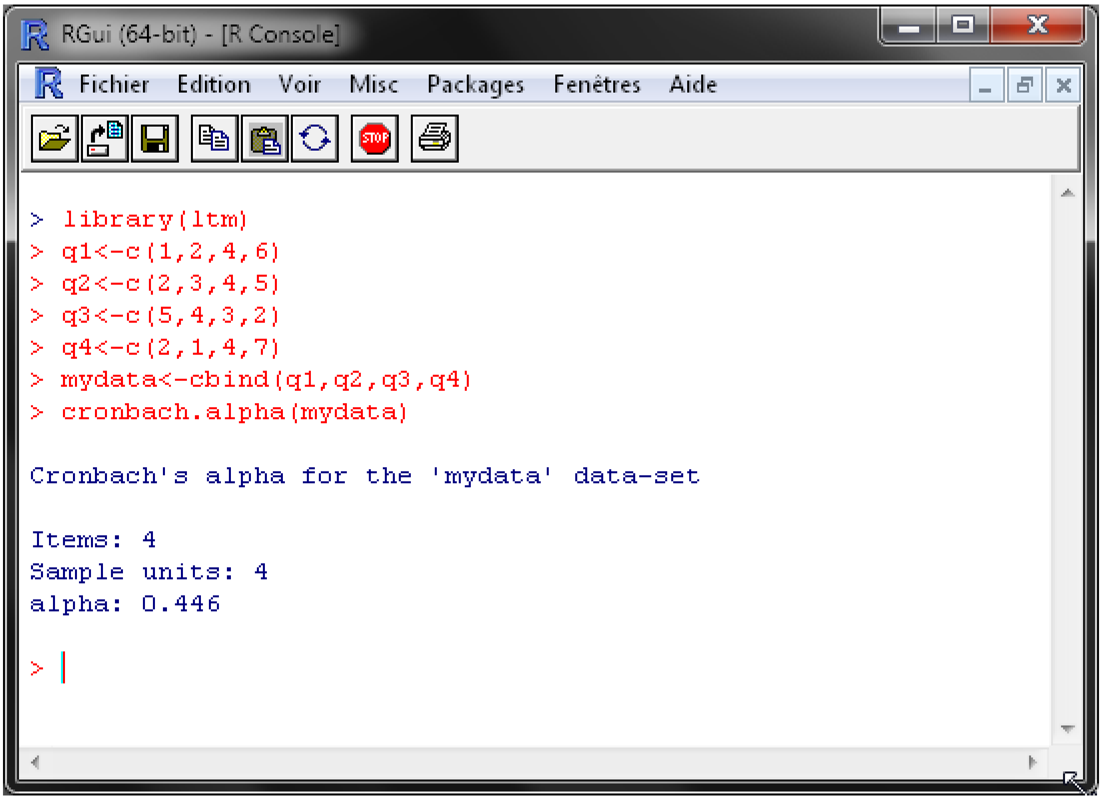
Le V de Cramér peut bien évidemment être utilisé en machine learning pour mesurer la performance de classification binaire!

Étude de la cohérence avec l'alpha de Cronbach

R 3.0.2

Comme à l'habitude, le but va être de vérifier ici que nous retombons sur les calculs faits à la main dans le cours théorique suite à la démonstration mathématique de l'origine de l'alpha de Cronbach.

Nous allons pour cela utiliser la commande **cronbach.alpha( )** du package **ltm**:



Donc nous retrouvons bien les mêmes résultats que ceux calculés à la main et que Minitab.

Étude de la cohérence avec le coefficient de Goodman-Kruskal tau

R 3.4.1

Le but ici va être de vérifier que nous retombons bien sur les calculs faits à la mains (surtout pour le coefficient en ligne). Nous partons donc du même jeu de données que dans le cours théorique et nous obtenons à l'aide de la fonction **GoodmanKruskalTau( )** du package **DescTools**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

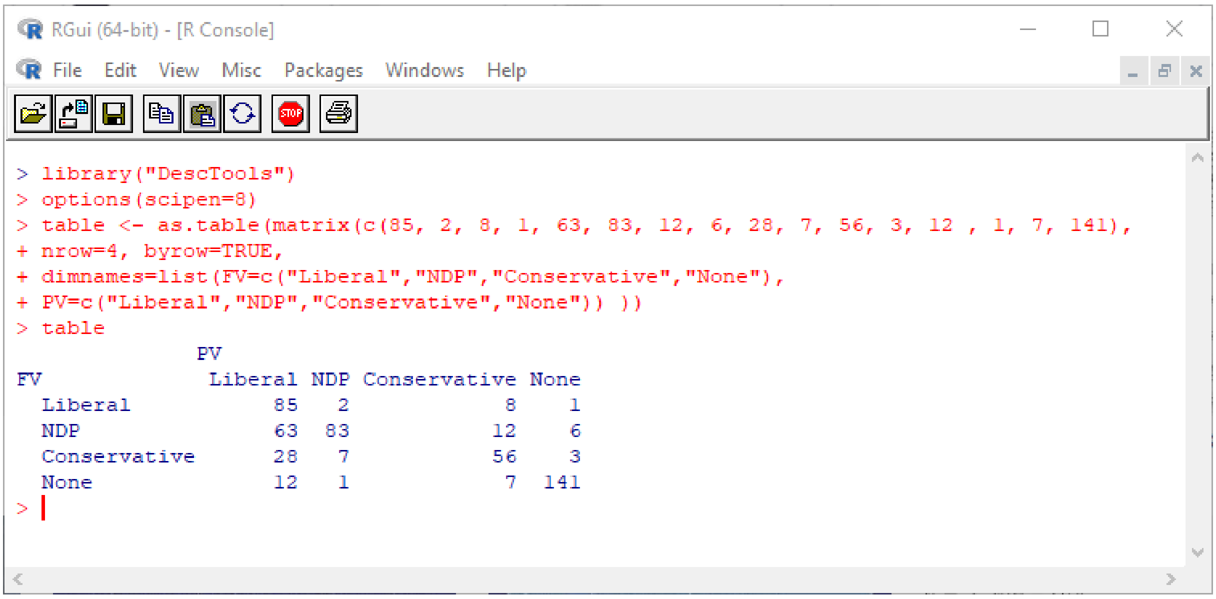
> library("DescTools")
> options(scipen=8)
> table <- as.table(matrix(c(85, 2, 8, 1, 63, 83, 12, 6, 28, 7, 56, 3, 12 , 1, 7, 141),
+ nrow=4, byrow=TRUE,
+ dimnames=list(FV=c("Liberal","NDP","Conservative","None"),
+ PV=c("Liberal","NDP","Conservative","None")) ))
> table
              PV
FV             Liberal NDP Conservative None
  Liberal           85   2            8    1
  NDP               63  83           12    6
  Conservative      28   7           56    3
  None              12   1            7  141
> GoodmanKruskalTau(table, direction="row", conf.level=0.95)
     tauA    lwr.ci    upr.ci
0.4620451 0.4070072 0.5170830
> GoodmanKruskalTau(table, direction="column", conf.level=0.95)
     tauA    lwr.ci    upr.ci
0.4496022 0.3938795 0.5053249
> 
```

Nous retombons bien sur la même valeur de 0.462 que dans le cours théorique!

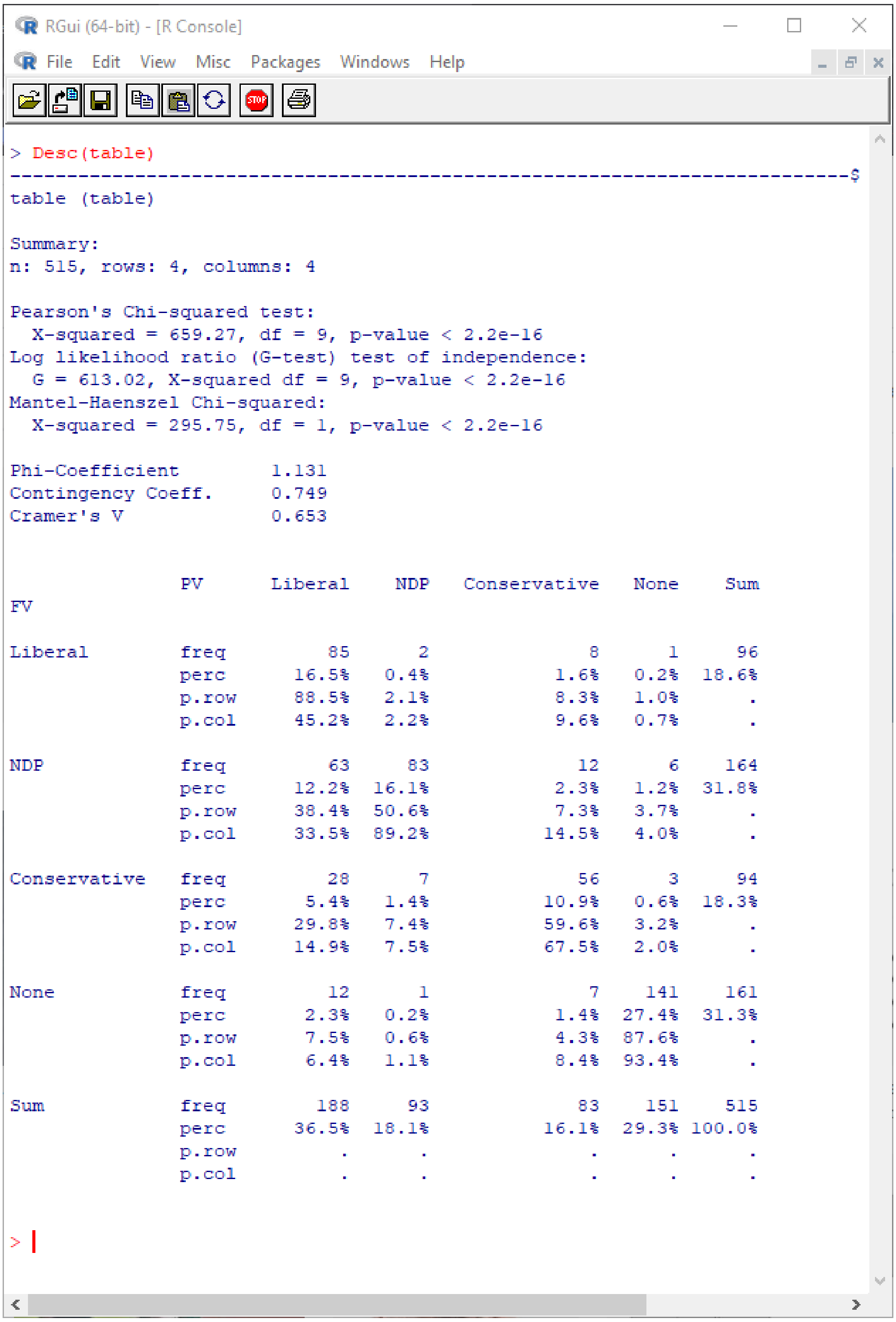
Étude de la cohérence avec le coefficient Lambda

R 3.4.1

Le but ici va être de vérifier que nous retombons bien sur les calculs faits à la mains (surtout pour le coefficient en ligne). Nous partons donc du même jeu de données que dans le cours théorique:



Faisons un petit résumé statistique et graphique avec la fonction **Desc( )**:




```
> Desc(table)
------------------------------------------------------------------------$
table (table)

Summary:
n: 515, rows: 4, columns: 4

Pearson's Chi-squared test:
  X-squared = 659.27, df = 9, p-value < 2.2e-16
Log likelihood ratio (G-test) test of independence:
  G = 613.02, X-squared df = 9, p-value < 2.2e-16
Mantel-Haenszel Chi-squared:
  X-squared = 295.75, df = 1, p-value < 2.2e-16

Phi-Coefficient        1.131
Contingency Coeff.     0.749
Cramer's V             0.653


               PV      Liberal    NDP  Conservative   None    Sum
FV

Liberal        freq         85      2             8      1     96
               perc      16.5%   0.4%          1.6%   0.2%  18.6%
               p.row     88.5%   2.1%          8.3%   1.0%      .
               p.col     45.2%   2.2%          9.6%   0.7%      .

NDP            freq         63     83            12      6    164
               perc      12.2%  16.1%          2.3%   1.2%  31.8%
               p.row     38.4%  50.6%          7.3%   3.7%      .
               p.col     33.5%  89.2%         14.5%   4.0%      .

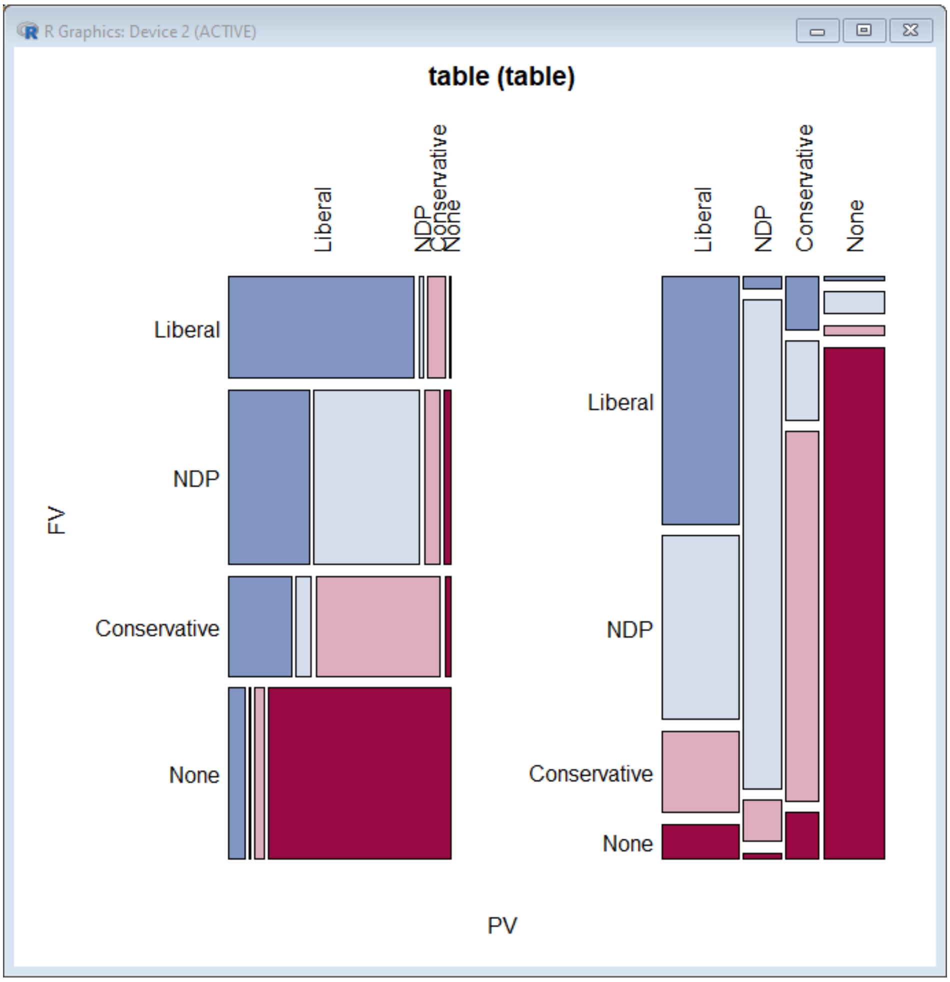
Conservative   freq         28      7            56      3     94
               perc       5.4%   1.4%         10.9%   0.6%  18.3%
               p.row     29.8%   7.4%         59.6%   3.2%      .
               p.col     14.9%   7.5%         67.5%   2.0%      .

None           freq         12      1             7    141    161
               perc       2.3%   0.2%          1.4%  27.4%  31.3%
               p.row      7.5%   0.6%          4.3%  87.6%      .
               p.col      6.4%   1.1%          8.4%  93.4%      .

Sum            freq        188     93            83    151    515
               perc      36.5%  18.1%         16.1%  29.3% 100.0%
               p.row         .      .             .      .      .
               p.col         .      .             .      .      .


>
```

Avec le graphique correspondant:



et nous obtenons à l'aide de la fonction **Assocs( )** du package **DescTools**:

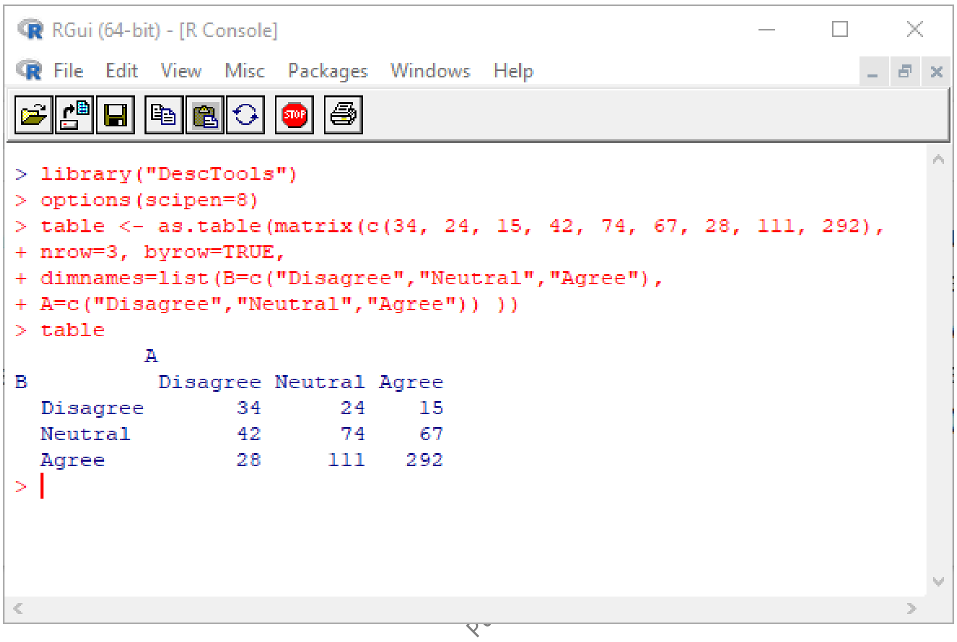
```
> Assocs(table)
                              estimate  lwr.ci  upr.ci
Phi Coeff.                      1.1314       -       -
Contingency Coeff.              0.7493       -       -
Cramer V                        0.6532  0.5992  0.6993
Goodman Kruskal Gamma           0.8170  0.7658  0.8682
Kendall Tau-b                   0.6904  0.6361  0.7448
Stuart Tau-c                    0.6697  0.6165  0.7229
Somers D C|R                    0.6854  0.6303  0.7405
Somers D R|C                    0.6955  0.6358  0.7552
Pearson Correlation             0.7585  0.7193  0.7930
Spearman Correlation            0.7484  0.7078  0.7841
Lambda C|R                      0.5413  0.4618  0.6207
Lambda R|C                      0.5726  0.5031  0.6422
Lambda sym                      0.5575  0.4858  0.6293
Uncertainty Coeff. C|R          0.4472  0.3928  0.5016
Uncertainty Coeff. R|C          0.4404  0.3859  0.4948
Uncertainty Coeff. sym          0.4438  0.3897  0.4979
Mutual Information              0.8586       -       -
>
```

Nous retombons bien sur la même valeur de 0.5726 et 0.5413 que dans le cours théorique!

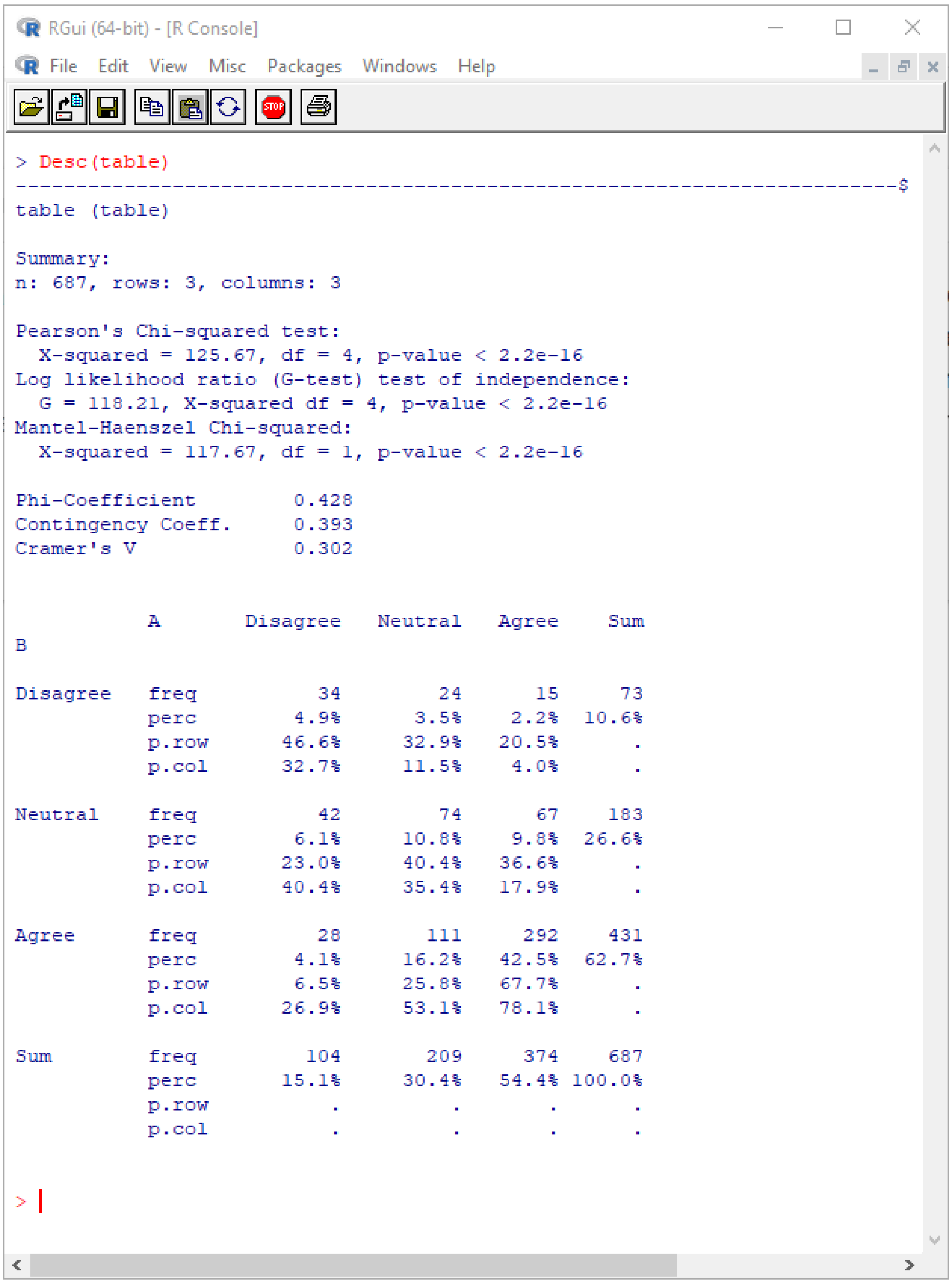
Étude de la cohérence avec le coefficient de Goodman-Kruskal gamma (et autres)

R 3.4.1

Le but ici va être de vérifier que nous retombons bien sur les calculs faits à la mains (surtout pour le coefficient en ligne). Nous partons donc du même jeu de données que dans le cours théorique:



Et faisons un petit résumé descriptif avec la fonction **Desc( )**:




```
> Desc(table)
------------------------------------------------------------------------$
table (table)

Summary:
n: 687, rows: 3, columns: 3

Pearson's Chi-squared test:
  X-squared = 125.67, df = 4, p-value < 2.2e-16
Log likelihood ratio (G-test) test of independence:
  G = 118.21, X-squared df = 4, p-value < 2.2e-16
Mantel-Haenszel Chi-squared:
  X-squared = 117.67, df = 1, p-value < 2.2e-16

Phi-Coefficient        0.428
Contingency Coeff.     0.393
Cramer's V             0.302


           A      Disagree  Neutral   Agree    Sum
B

Disagree   freq         34       24      15     73
           perc       4.9%     3.5%    2.2%  10.6%
           p.row     46.6%    32.9%   20.5%      .
           p.col     32.7%    11.5%    4.0%      .

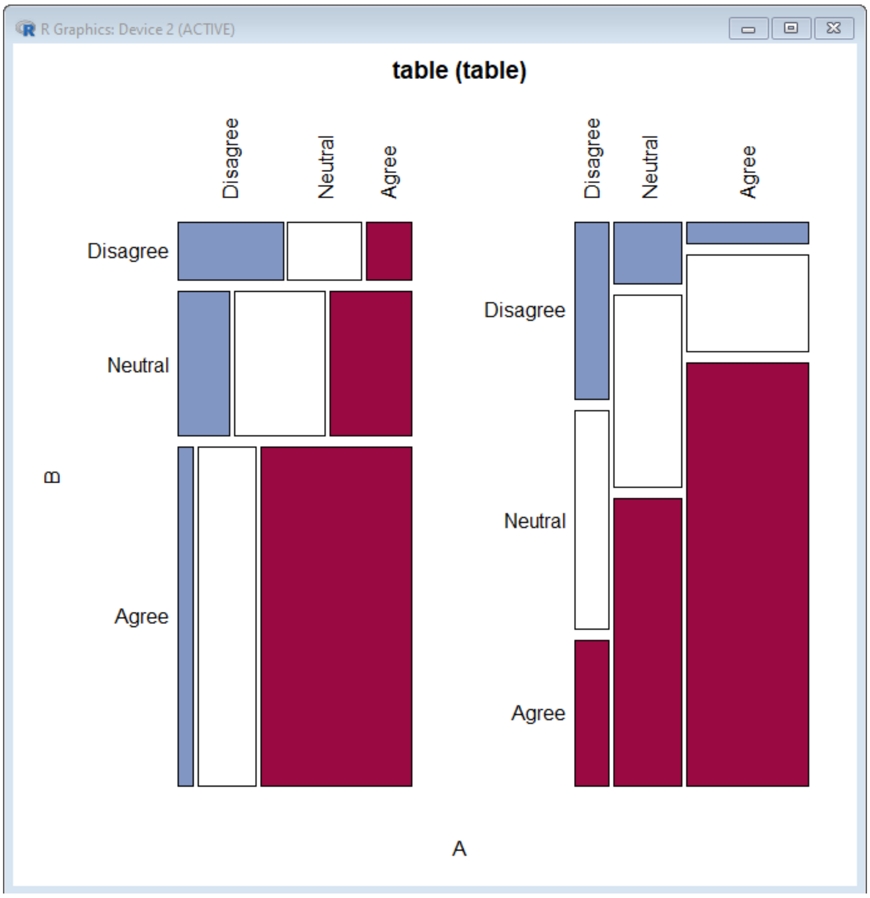
Neutral    freq         42       74      67    183
           perc       6.1%    10.8%    9.8%  26.6%
           p.row     23.0%    40.4%   36.6%      .
           p.col     40.4%    35.4%   17.9%      .

Agree      freq         28      111     292    431
           perc       4.1%    16.2%   42.5%  62.7%
           p.row      6.5%    25.8%   67.7%      .
           p.col     26.9%    53.1%   78.1%      .

Sum        freq        104      209     374    687
           perc      15.1%    30.4%   54.4% 100.0%
           p.row         .        .       .      .
           p.col         .        .       .      .


>
```

Avec le graphique correspondant:



et nous obtenons encore une fois à l'aide de la fonction **Assocs( )** du package **DescTools**:


```
> Assocs(table)
                          estimate  lwr.ci  upr.ci
Phi Coeff.                  0.4277       -       -
Contingency Coeff.          0.3932       -       -
Cramer V                    0.3024  0.2456  0.3520
Goodman Kruskal Gamma       0.5907  0.5070  0.6744
Kendall Tau-b               0.3721  0.3071  0.4371
Stuart Tau-c                0.3099  0.2530  0.3668
Somers D C|R                0.3942  0.3254  0.4629
Somers D R|C                0.3513  0.2897  0.4129
Pearson Correlation         0.4142  0.3502  0.4743
Spearman Correlation        0.3997  0.3349  0.4607
Lambda C|R                  0.0831  0.0004  0.1657
Lambda R|C                  0.0547  0.0000  0.1170
Lambda sym                  0.0703  0.0172  0.1234
Uncertainty Coeff. C|R      0.0879  0.0573  0.1185
Uncertainty Coeff. R|C      0.0974  0.0636  0.1312
Uncertainty Coeff. sym      0.0924  0.0603  0.1245
Mutual Information          0.1241       -       -
>
```

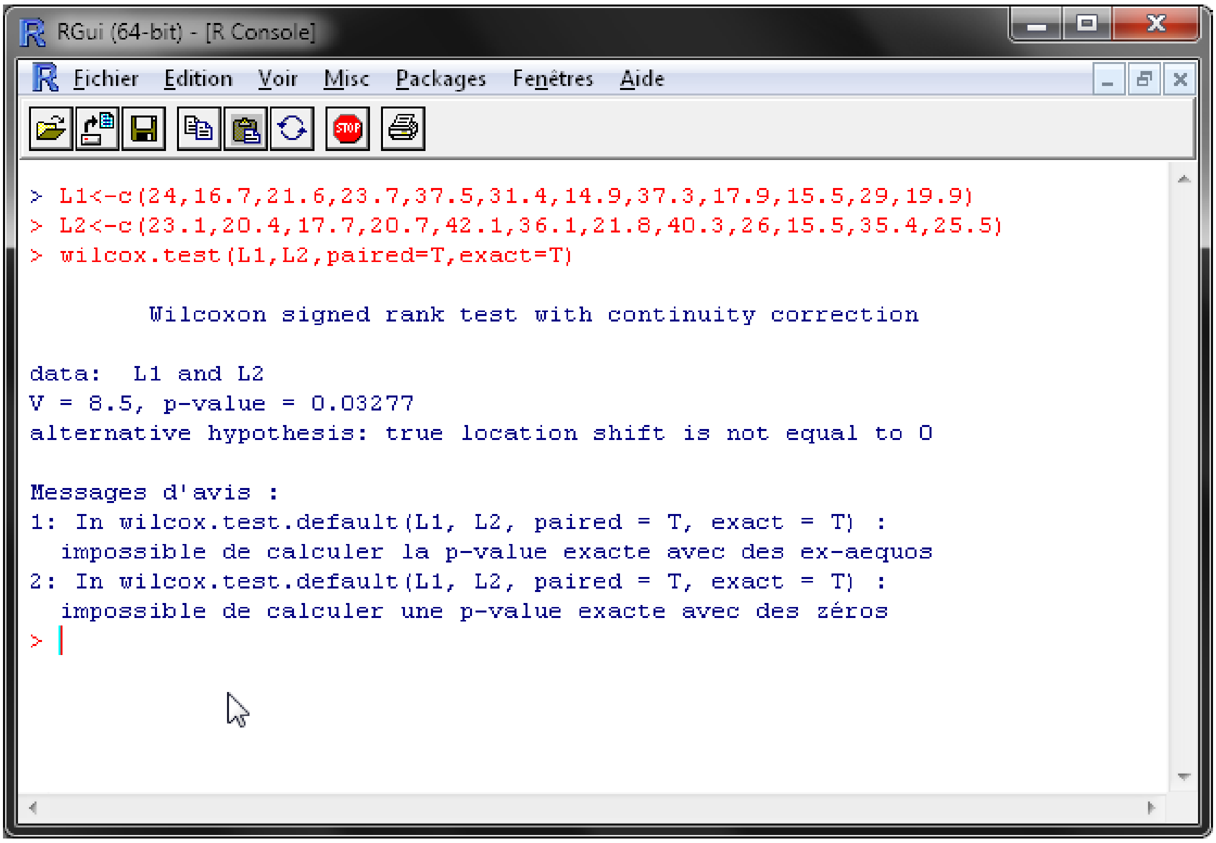
Nous retombons bien sur toutes les mêmes valeurs que dans le cours théorique!

Exemple 234.: Test de la somme des rangs signés de Wilcoxon pour 2 échantillons appariés

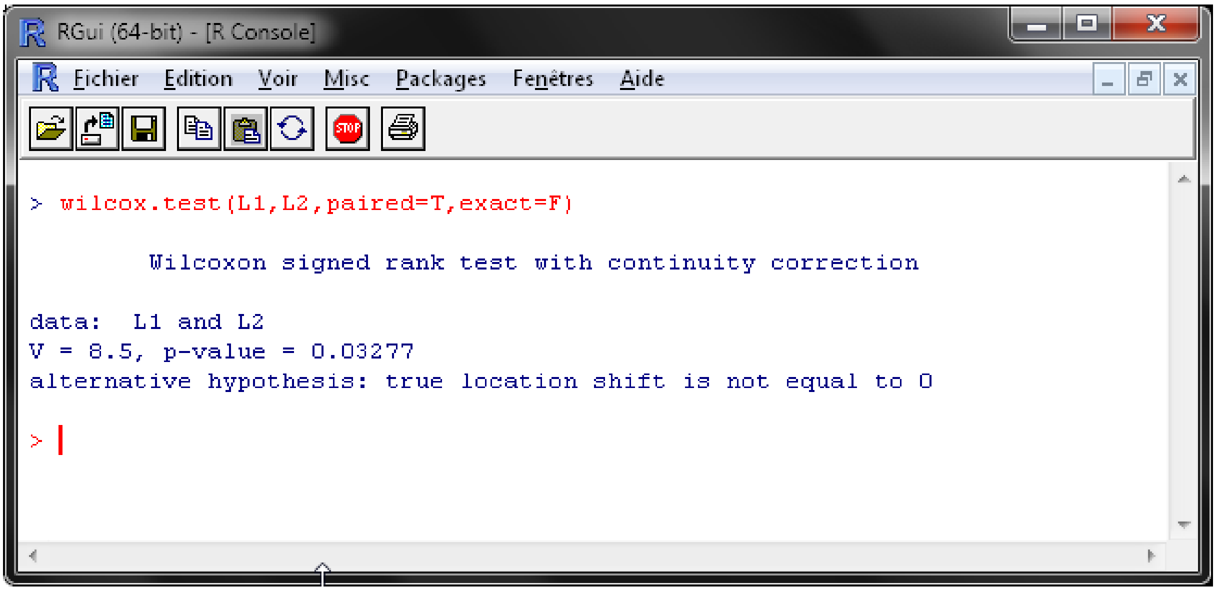
R 3.0.2

Nous voulons appliquer le test des rangs signés de Wilcoxon pour 2 échantillons appariés comme démontré dans le cours de statistique théorique bien que celui-ci ne soit pas disponible explicitement dans Minitab et ce afin de comparer au résultat calculé avec Microsoft Excel dans le cadre de l'approximation par une loi Normale.

Nous utiliserons pour cela les mêmes données brutes que dans le cours théorique avec la commande **wilcox.test( )** et d'abord sans approximation:



Nous retrouvons donc exactement les mêmes valeurs que dans Minitab (messages d'avertissements exceptés) mais qui comme nous le savons ne correspondant pas à ce que nous avons calculé à la main. Donc faisons avec l'approximation Normale:



Nous voyons que cela ne change rien. Certes la valeur *V* est la même que dans le cours théorique mais toujours pas la *p*-value.

Certes la conclusion est toutefois la même que dans le cours théorique.

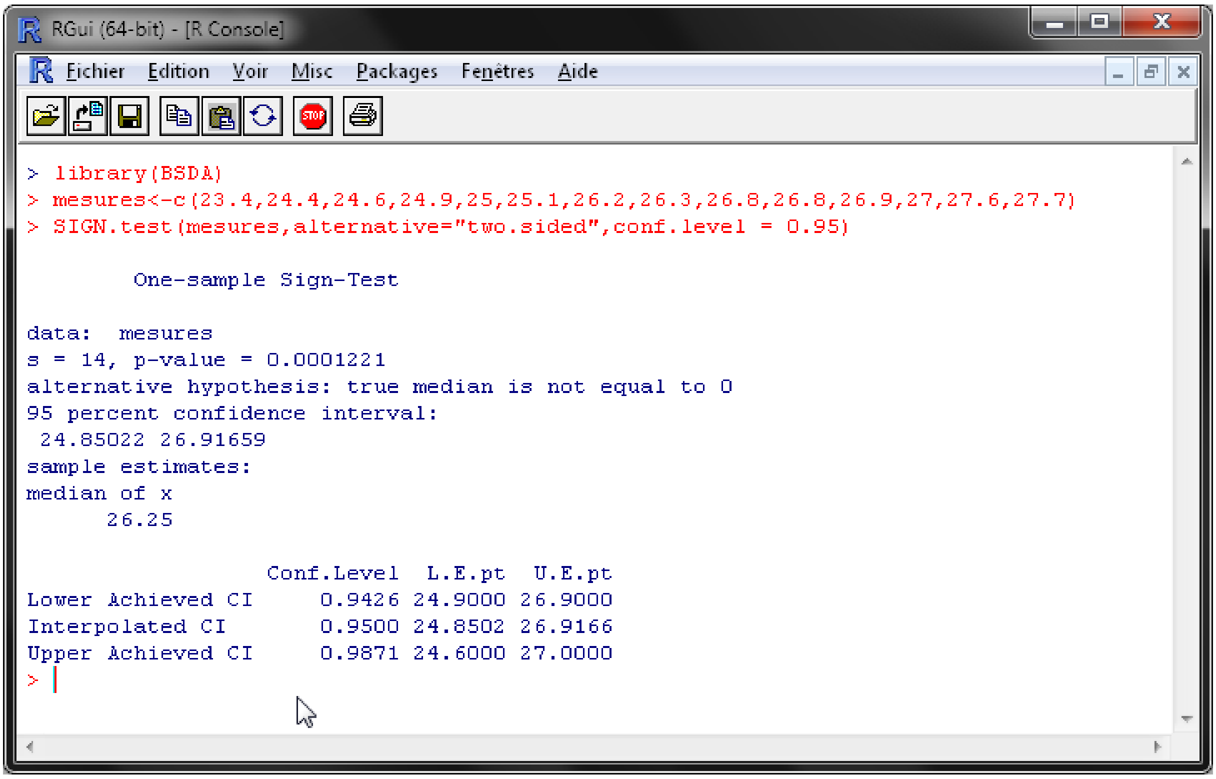
Exemple 235.: Intervalle de confiance de la médiane (via test du signe à 1 échantillon)

R 3.0.2

Nous avons effectué une série de mesures de valeurs continues.

Nous souhaiterions simplement connaître l'intervalle de confiance de la médiane, le but étant aussi de contrôler que R utilise bien la méthode que nous avions mentionnée dans le cours de statistique théorique et que nous avions mis en pratique à la main, dans Microsoft Excel et dans Minitab.

Nous utilisons la commande **SIGN.test( )** du package **BSDA** où la valeur test par défaut du paramètre md est nul si pas spécifié mais la seule chose qui nous intéresse ici est l'intervalle!:



Nous retrouvons donc exactement la même sortie que Minitab.

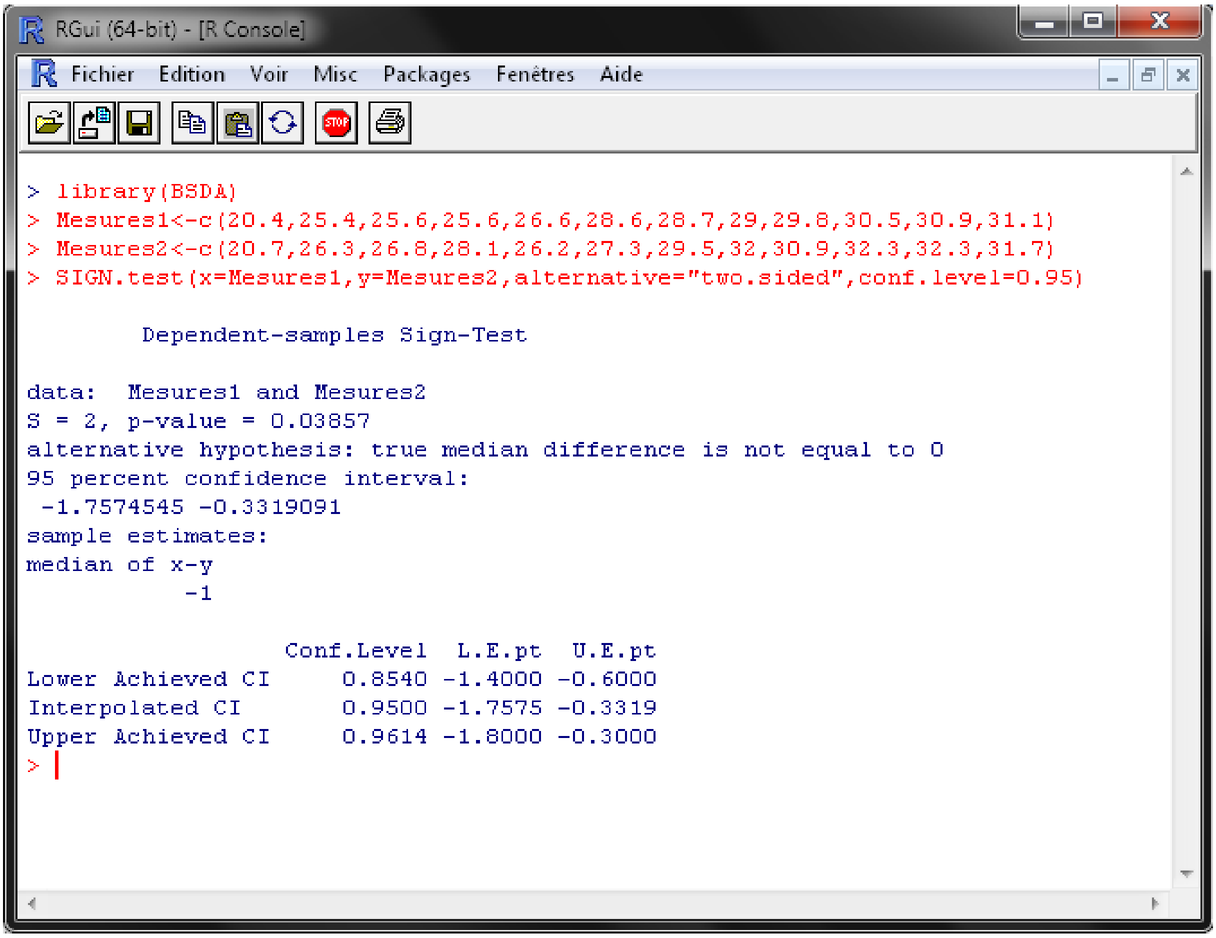
Exemple 236.: Test du signe binomial (dixit: test de la médiane pour 2 échantillons appariés)

R 3.0.2

Nous avons effectué deux séries de mesures avec deux méthodes différentes.

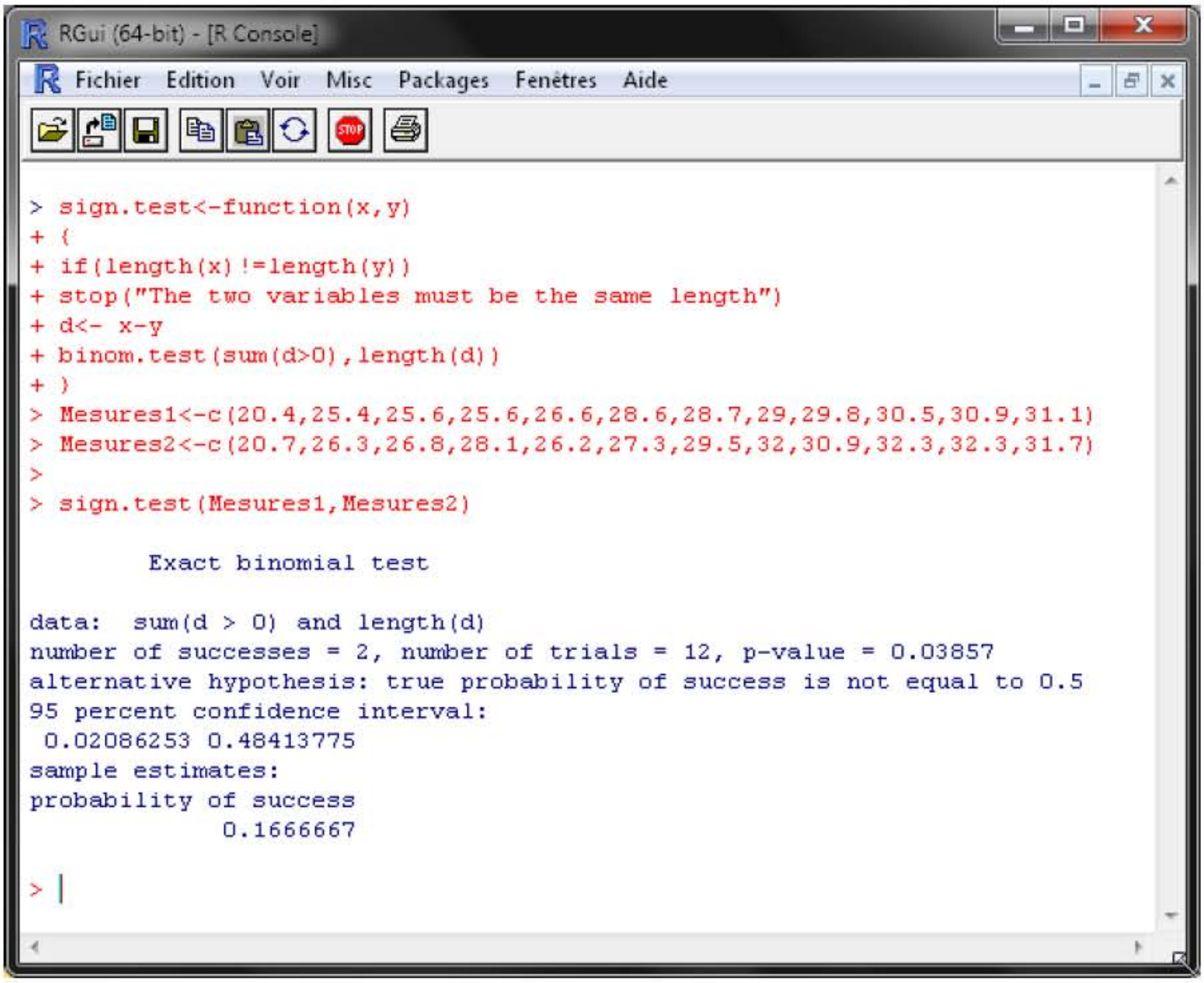
Nous souhaiterions savoir la différence sont significatives ou pas. Le but étant aussi de contrôler si nous avons un résultat différent de celui calculé à la main en cours et dans MS Excel.

Nous utilisons la commande **SIGN.test( )** du package **BSDA** où la valeur test par défaut du paramètre md est nul si pas spécifié:



Nous retrouvons toutes les valeurs vues dans le cours de statistique théorique et que dans le cours Minitab. Nous rejetons donc au vu de la *p*-value (inférieure à 5%) que la différence n'est pas significative.

Nous pouvons aussi construire l'équivalent à partir du test binomial exact mais en focalisant uniquement sur la p-value du résultat renvoyé:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> sign.test<-function(x,y)
+ {
+ if(length(x)!=length(y))
+ stop("The two variables must be the same length")
+ d<- x-y
+ binom.test(sum(d>0),length(d))
+ }
> Mesures1<-c(20.4,25.4,25.6,25.6,26.6,28.6,28.7,29,29.8,30.5,30.9,31.1)
> Mesures2<-c(20.7,26.3,26.8,28.1,26.2,27.3,29.5,32,30.9,32.3,32.3,31.7)
>
> sign.test(Mesures1,Mesures2)
Exact binomial test
data: sum(d > 0) and length(d)
number of successes = 2, number of trials = 12, p-value = 0.03857
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
0.02086253 0.48413775
sample estimates:
probability of success
0.1666667
>

Exemple 237.: Test de Mood (test des médianes)

R 3.0.2

Le but du test de Mood n'est pas de contrôler si des données appariées peuvent être considérées comme égales ou non en se basant sur leur médiane (test des signes), mais simplement de vérifier sur la base d'un tableau de contingence du Khi-deux, si le nombre de valeurs au-dessus ou en-dessous de la médiane de deux échantillons est significativement différente et proviennent donc de deux populations différentes.

Voyons comment exécuter le test de Mood avec respectivement le même résultat que dans le cours théorique et Minitab avec le package **RVAideMemoire** et sa fonction **median_test( )** dans un premier temps, et ensuite de façon similaire à SPSS avec le package **coin** et sa fonction **mood.medtest( )**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> Mesures2<-c(22.5,22.9,23.7,24,24.4,24.5,25.3,26,26.2,26.4,26.7,26.9,27.4)
> ToutesMesures<-c(Mesures1,Mesures2)
> Categories<-gl(2,13,labels=LETTERS[1:2])
> Categories
 [1] A A A A A A A A A A A A A B B B B B B B B B B B B B
Levels: A B
>
> #pour réobtenir le résultat de Minitab et calculé à la main dans le cours théorique
> library("RVAideMemoire")
> median_test(ToutesMesures~Categories,conf.int=TRUE,conf.level=0.95)

        Asymptotic Two-Sample Brown-Mood Median Test

data:  ToutesMesures by Categories (A, B)
Z = 1.1538, p-value = 0.2486
alternative hypothesis: true mu is not equal to 0
95 percent confidence interval:
 -1.5  2.9
sample estimates:
difference in location
                     1

>
> #pour réobtenir le résultat de SPSS
> library("coin")
> mood.medtest(ToutesMesures~Categories)

        Mood's median test

data:  ToutesMesures by Categories
p-value = 0.4338

> |
```

Nous voyons que bien que la conclusion soit la même, le test fait une approximation par la loi Normale.

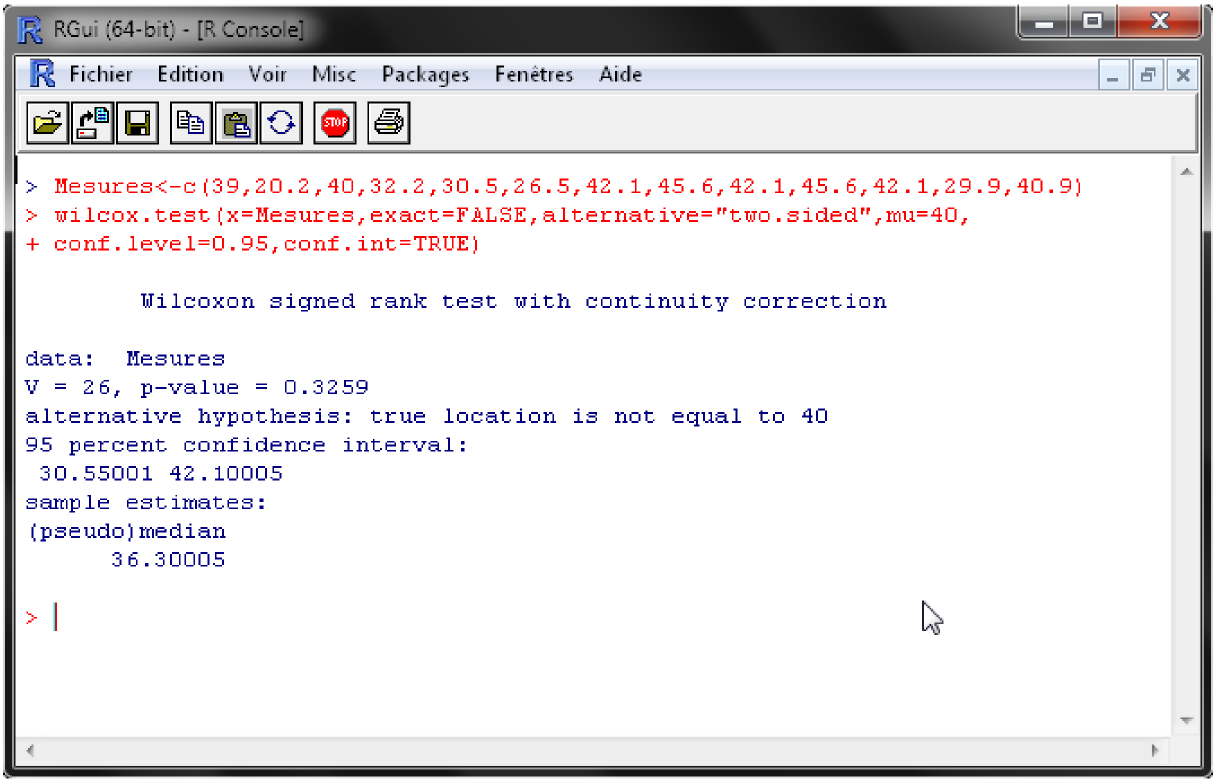
Il y a cependant un petit avantage, nous avons automatiquement un intervalle de confiance et nous constatons que ce dernier n'est pas très éloigné de celui renvoyé par Minitab.

Exemple 238.: Test de la somme des rangs signés de Wilcoxon pour 1 échantillon

R 3.0.2

Encore une fois, nous souhaitons comparer les démonstrations faites dans le cours de statistiques théorique avec l'application numérique approximée dans Microsoft Excel et exact avec Minitab pour comparer avec le résultat obtenu avec R.

Nous allons utiliser la commande native **wilcox.test( )** de R:



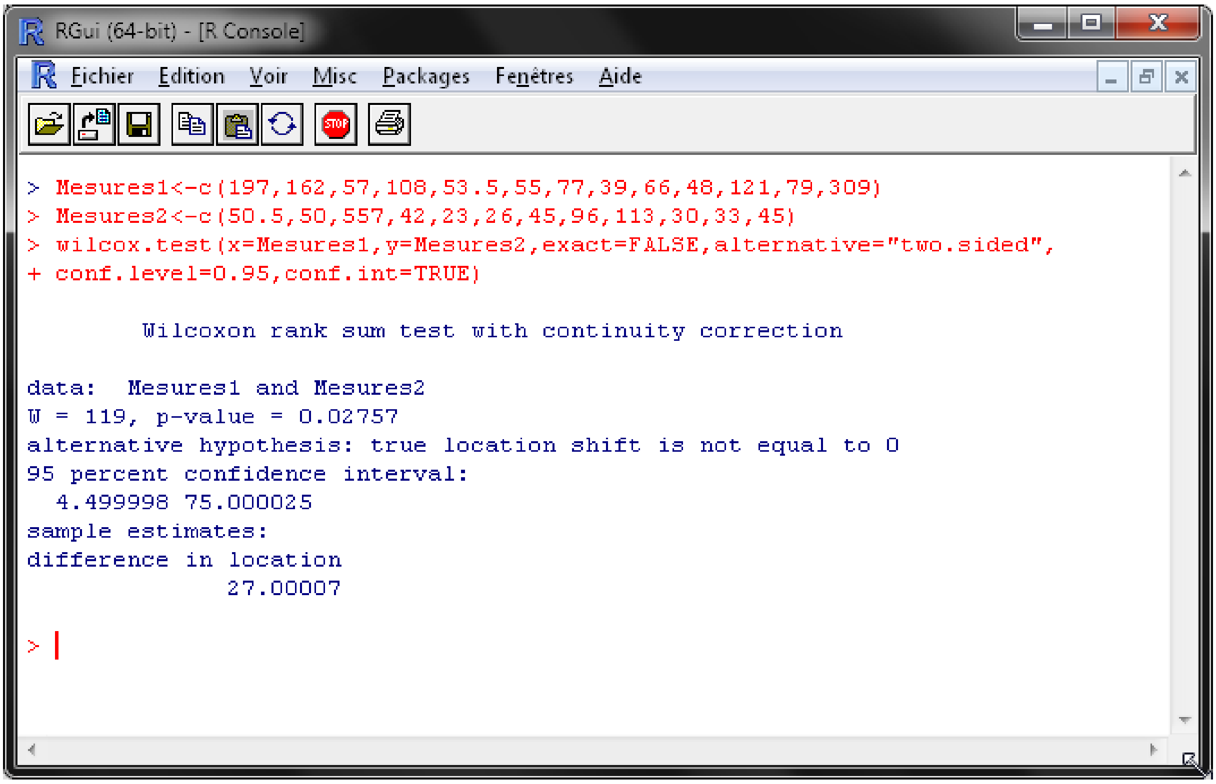
Nous retrouvons bien la valeur V égale à 26 que Minitab ne nous donnait pas directement. La *p*-value est sensiblement différente de celle de Minitab et de celle calculée dans le cours théorique (et rappelons que celle calculée à la main dans cours théorique était aussi sensiblement différente que celle donnée par Minitab). Contrairement à Minitab, nous avons cependant un intervalle de confiance ce qui est bien pratique. Concernant l'estimation ponctuelle de la médiane, nous retrouvons bien la même valeur.

Exemple 239.: Test de la somme des rangs de (Wilcoxon)Mann-Withney pour deux échantillons indépendants

R 3.0.2

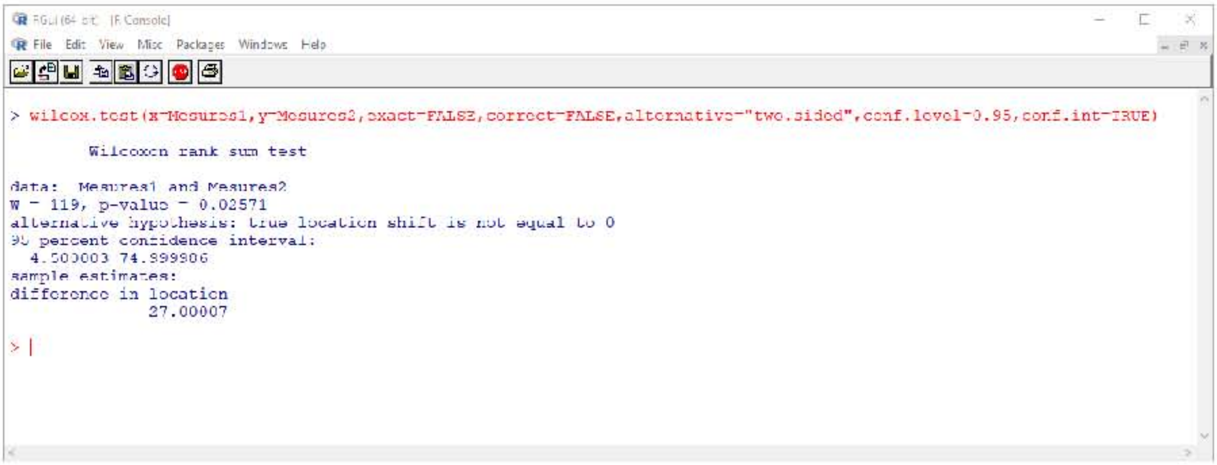
Comme d'haaaabitudeeee (en chantant...) le but est de voir si nous retrouvons les résultats numériques calculés avec Microsoft Excel en utilisant les relations démontrées dans le cours de statistique théorique.

Par contre nous n'allons par reprendre les mêmes valeurs qu'en dans le cours théorique car l'exemple y était trop petit (pour des raisons pédagogiques). Nous allons encore une fois utiliser la commande native **wilcox.test( )** de R:



Nous retrouvons la même *p*-value, le même intervalle de confiance et la même différence ponctuelle que dans le cours théorique et Minitab. Par contre, nous remarquons que contrairement au cours théorique et à Minitab la somme W n'est pas la même. Ceci n'est pas grave car nous avons vu dans le cours théorique que nous pouvions prendre la variable que nous voulions comme base de calcul. Cependant dans le cours théorique, nous avions calculé respectivement W comme valant 210 pour Y et 114 pour X alors que R donne 119. Cela est tout à fait normal puisque dans le cours théorique nous avions décidé de ne pas faire de correction empirique concernant les valeurs doubles (oui... il y a deux fois la valeur 45 dans les mesures 2).

Sinon sans la correction de continuité cela donne (même résultat qu'avec Oracle 11g comme vu dans le cours SQL pour Data Scientists):

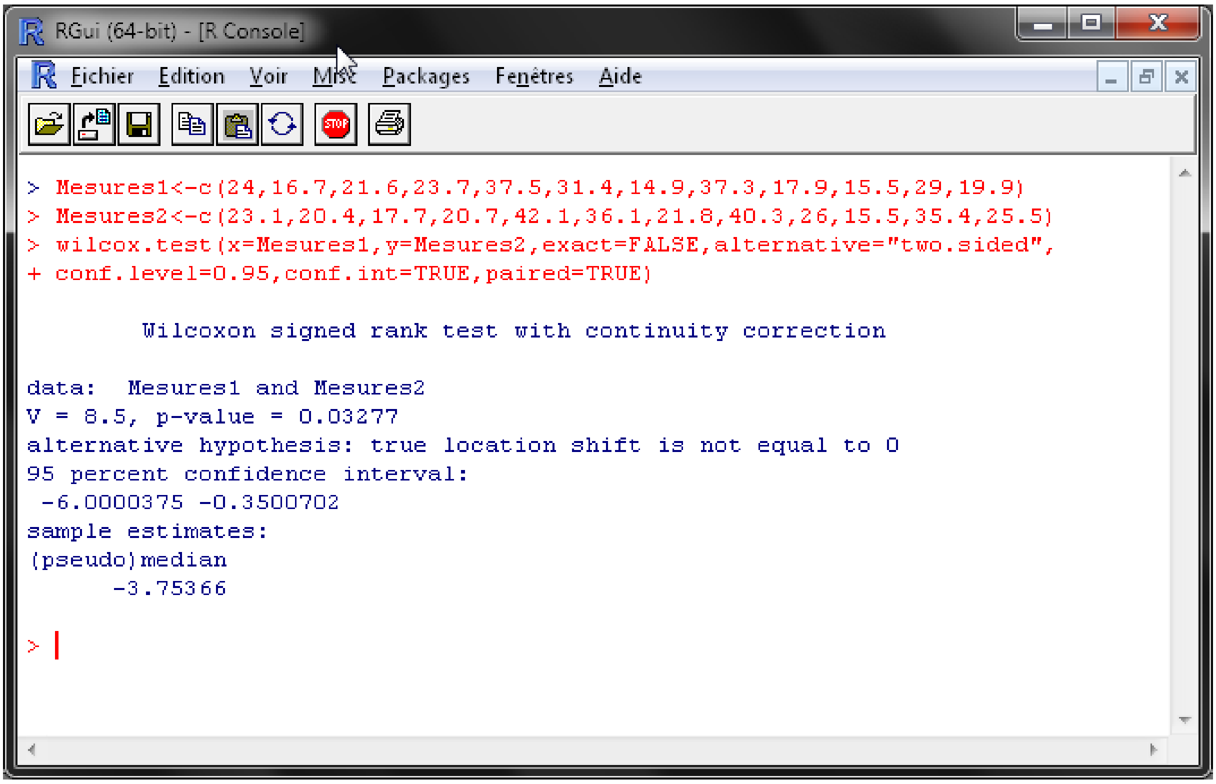

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> wilcox.test(x=Mesures1,y=Mesures2,exact=FALSE,correct=FALSE,alternative="two.sided",conf.level=0.95,conf.int=TRUE)
Wilcoxon rank sum test
data: Mesures1 and Mesures2
W = 119, p-value = 0.02571
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
4.500003 74.999906
sample estimates:
difference in location
27.00007
>

Exemple 240.: Test de la somme des rangs signés de Wilcoxon pour 2 échantillons appariés

R 3.0.2

Nous voulons appliquer le test des rangs signés de Wilcoxon pour 2 échantillons appariés comme démontré dans le cours de statistique théorique. Rappelons que celui-ci ne n'était pas disponible explicitement dans Minitab mais que faisant plusieurs manipulations nous avons tout de même pu l'obtenir et le comparer au résultat calculé avec Microsoft Excel dans le cadre de l'approximation par une loi Normale. Nous allons ici vérifier la correspondance avec R.

Pour cela, nous allons encore une fois utiliser la commande native **wilcox.test( )** de R:



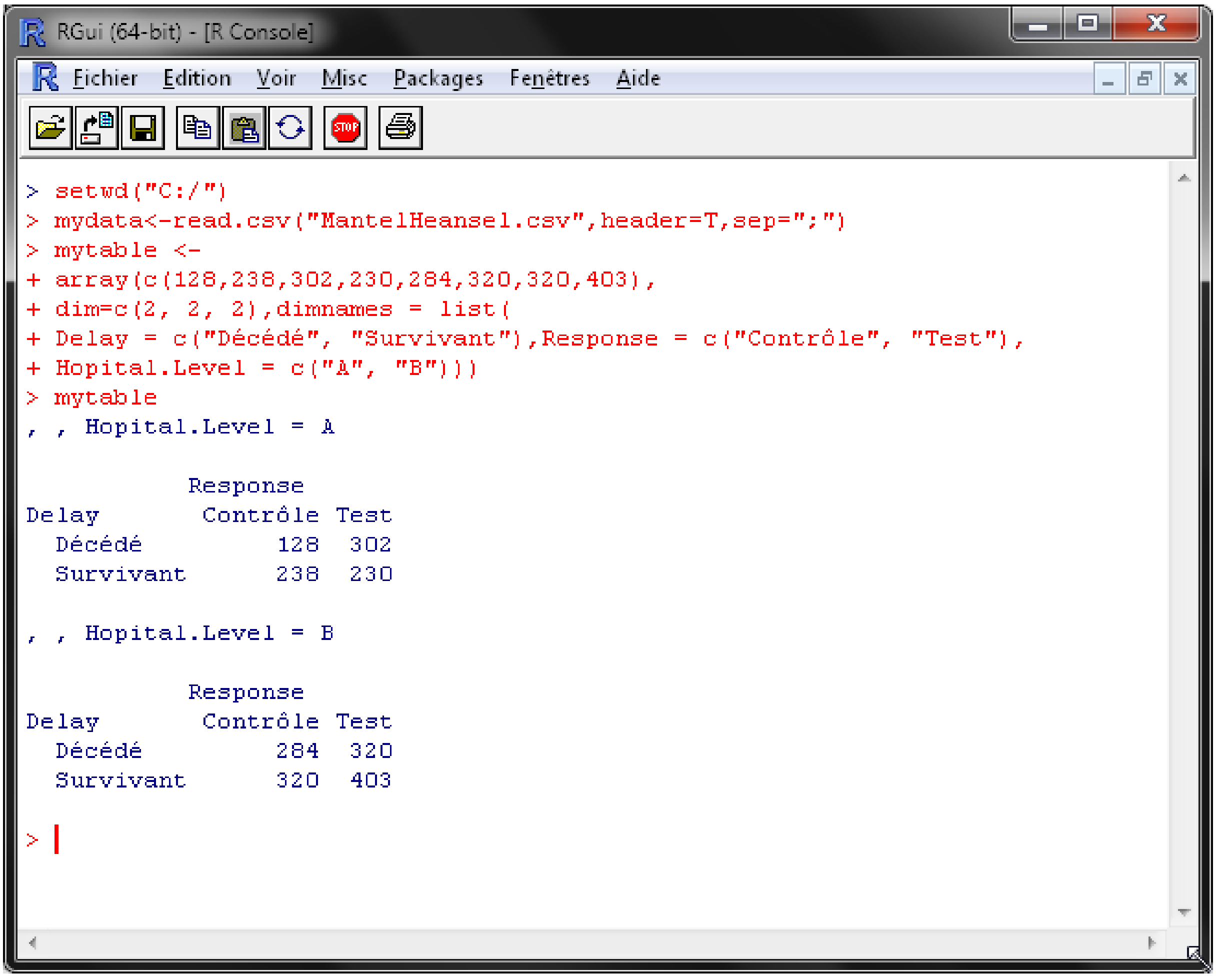
Donc à part l'intervalle de confiance que Minitab ne donnait pas, nous retrouvons les mêmes valeurs que dans ce dernier. La seule différence réside toujours dans la *p*-value qui est significativement différente que celle obtenue à la main dans le cours théorique.

Exemple 241.: Test de Mantel-Haenszel-Cochran

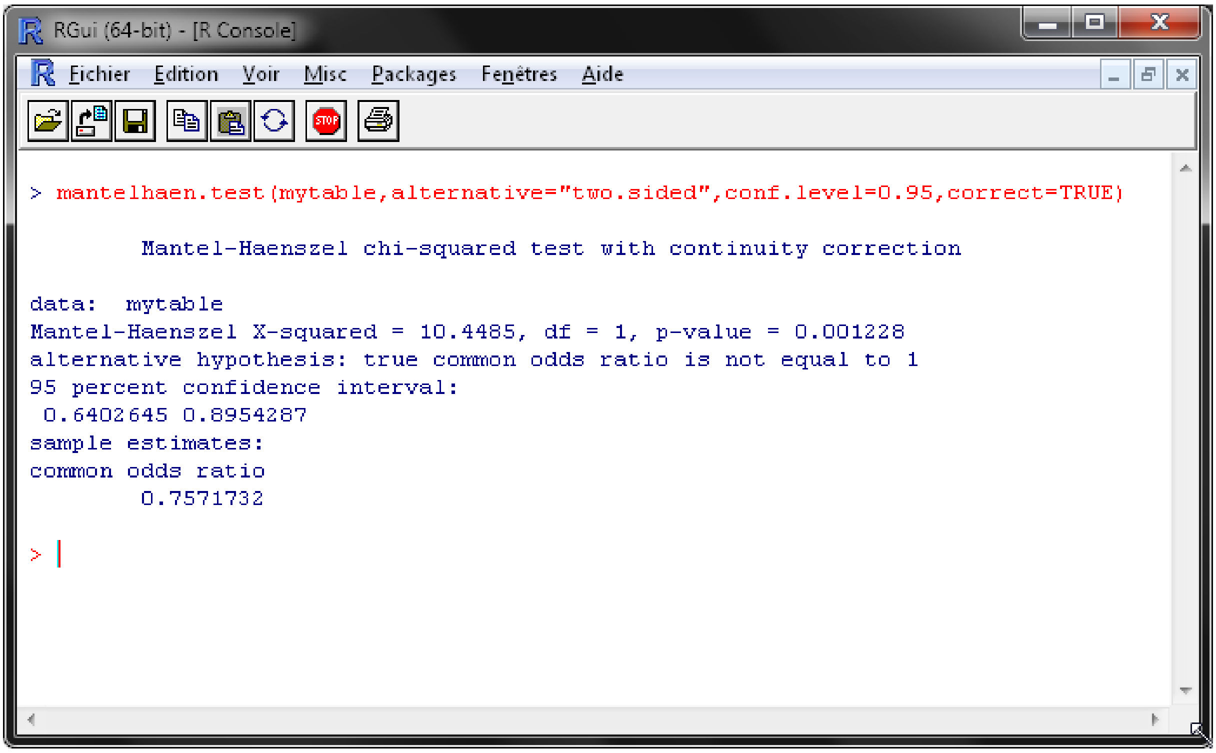
R 3.0.2

Encore une fois nous allons vérifier que nous retrouvons bien le même résultat que ce que nous avons obtenu dans le cours théorique lors des calculs à la main ou dans Minitab.

Nous partons des données suivantes:



Ensuite, nous utilisons la commande **mantelhaen.test( )** :



Nous retrouvons donc les mêmes valeurs que celles calculés dans le cours théorique ainsi que dans Minitab et donc avec les mêmes conclusions!

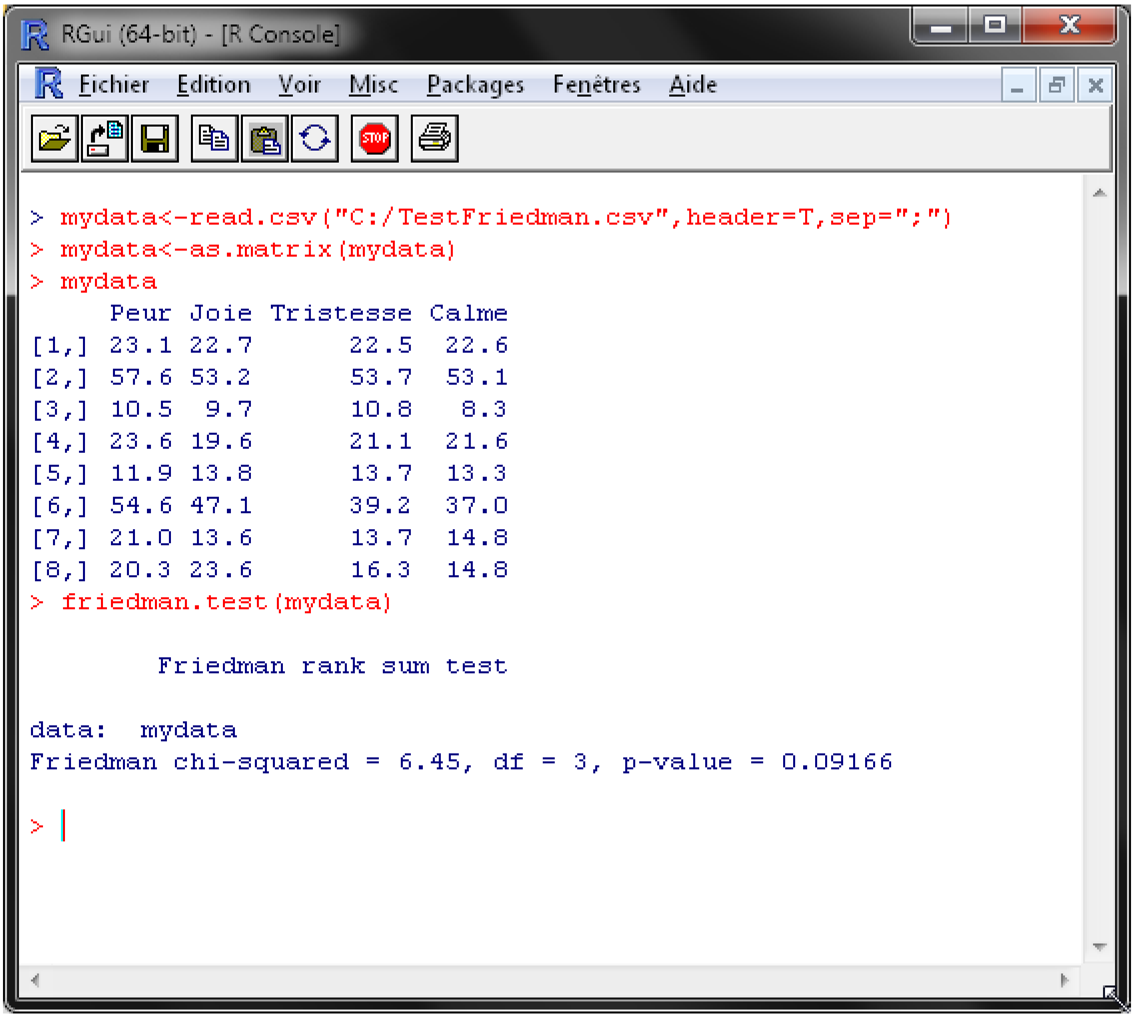
Au niveau de la conclusion, la *p*-value étant beaucoup plus petite que les valeurs traditionnelles critiques (10%, 5%, 1%) nous mettons donc en évidence le fait qu'il y a une différence significative entre le groupe de contrôle et de test à travers les différentes strates.

Exemple 242.: Test de Friedman (ANOVA canonique non paramétrique par les rangs)

R 3.0.2

Nous allons nous baser sur l'exemple pratique fait aussi en cours pour appliquer la pseudo-démonstration (guère convaincante) que nous avons étudiée concernant ce test. Nous allons comparer le résultat obtenu aussi par rapport à Minitab.

Nous utilisons pour cela la commande **friedman.test( )** native à R:




```
> mydata<-read.csv("C:/TestFriedman.csv",header=T,sep=";")
> mydata<-as.matrix(mydata)
> mydata
     Peur Joie Tristesse Calme
[1,] 23.1 22.7      22.5  22.6
[2,] 57.6 53.2      53.7  53.1
[3,] 10.5  9.7      10.8   8.3
[4,] 23.6 19.6      21.1  21.6
[5,] 11.9 13.8      13.7  13.3
[6,] 54.6 47.1      39.2  37.0
[7,] 21.0 13.6      13.7  14.8
[8,] 20.3 23.6      16.3  14.8
> friedman.test(mydata)

        Friedman rank sum test

data:  mydata
Friedman chi-squared = 6.45, df = 3, p-value = 0.09166

> 
```

Nous retrouvons donc exactement les mêmes résultats avec les mêmes conclusions que dans le cours théorique.

Exemple 243.: Test de Kruskal-Wallis (ANOVA canonique à 1 facteur non paramétrique)

R 3.0.2

Au même titre que dans le cours théorique et Minitab nous allons prendre par hommage l'excellent exemple original de l'article de Kruskal-Wallis:

	Standard		Modification 1		Modification 2	
	Output	Rank	Output	Rank	Output	Rank	
	340	5	339	4	347	10	
	345	9	333	2	343	7	
	330	1	344	8	349	11	
	342	6			355	12	
	338	3					
							Sum
n		5		3		4	12
R		24		14		40	78
R^2/n		115.2		65.333		400.	580.533

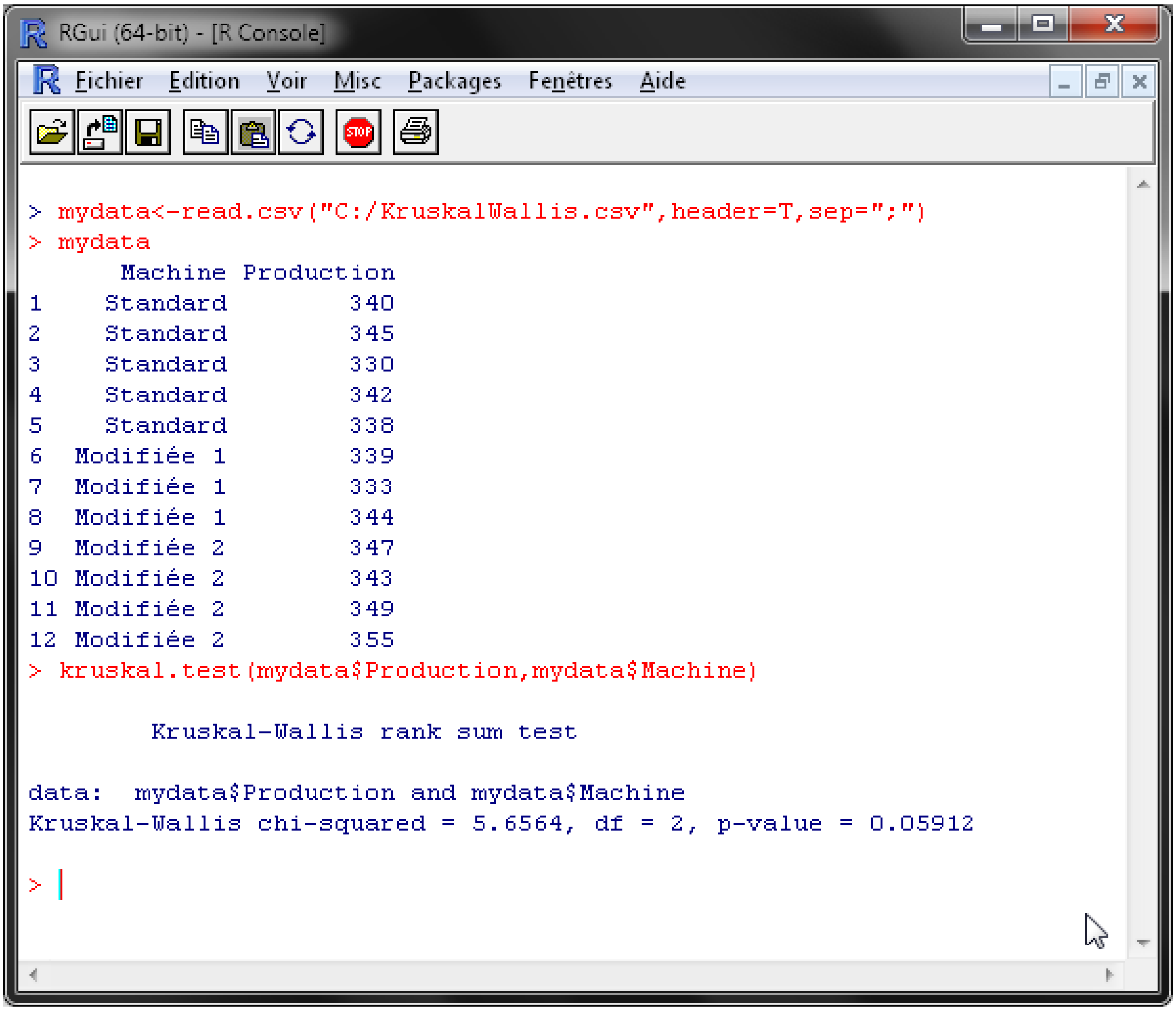
Checks : $\sum n = N = 12$ $\sum R = \frac{1}{2}N(N+1) = 78$

$$H = \frac{12 \times 580.533}{12 \times 13} - 3 \times 13 = 5.656 \simeq \chi^2(2) \qquad \textit{from (1.2)}$$

$\Pr[\chi^2(2) \geqq 5.656] = 0.059$ *from [9] or [13]*

$\Pr[H(5, 4, 3) \geqq 5.656] = 0.049$ *from Table 6.1*

Nous utilisons alors la commande **kruskal.test( )** native de R:



Donc à part le faire que nous avons moins d'informations que dans le cours théorique et qu'avec Minitab, les résultats donnés sont en parfait adéquation.

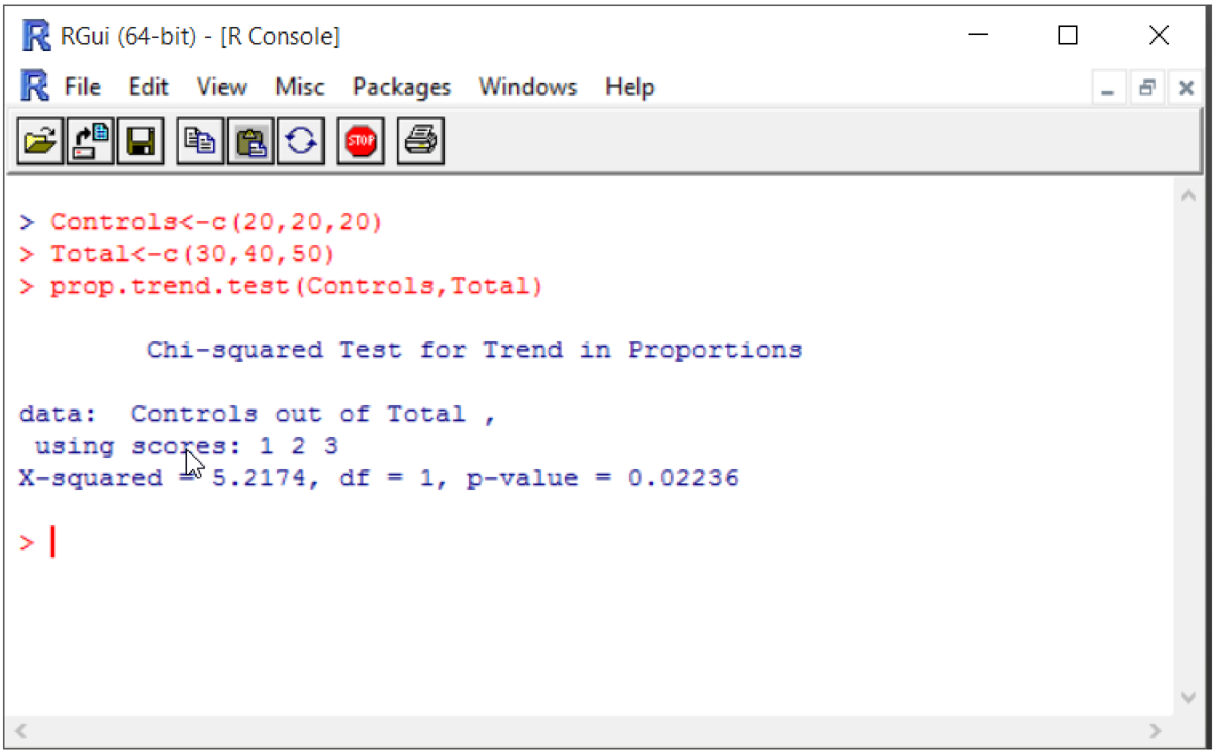
Exemple 244.: Test de Cochran-Armitage (test du chi-2 pour tendances dans les proportions)

R 3.2.1

Nous partons de la même table (en dimensions et en contenu!) que celle utilisée pour la démonstration dans le cours théorique (exemple reprise de Wikipédia):

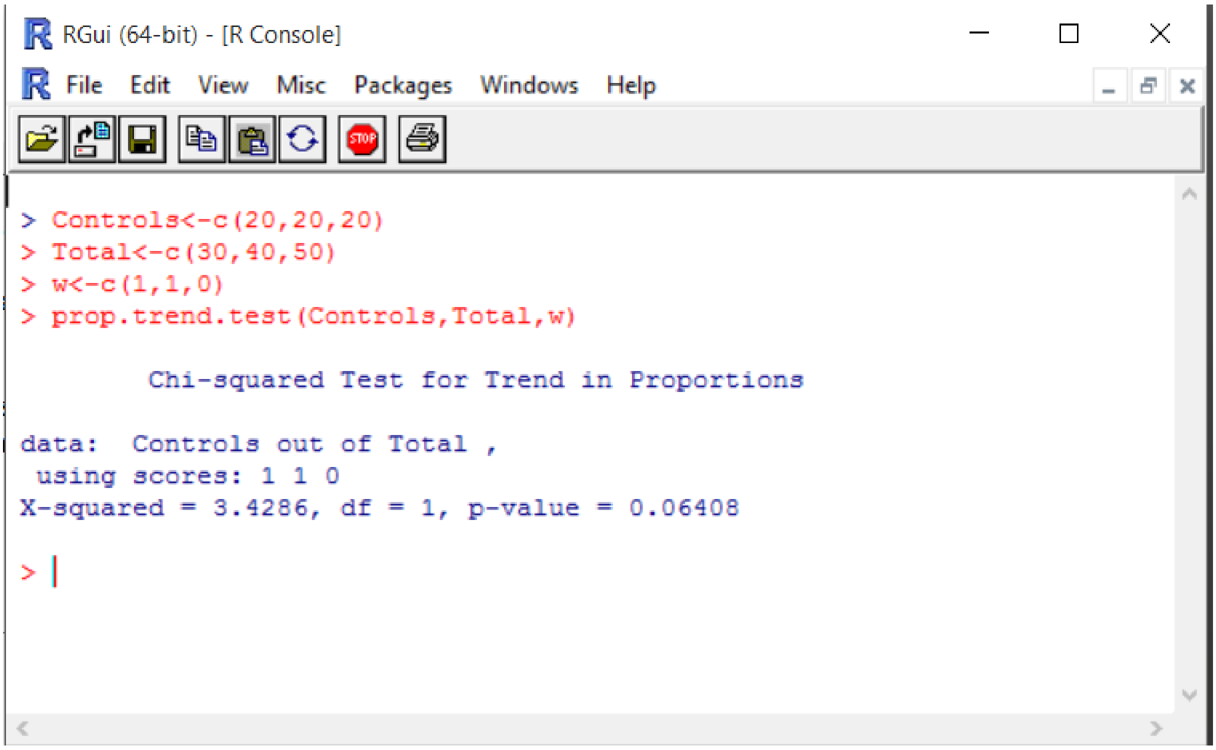
	Genotype aa	Genotype Aa	Genotype AA	Somme
Contrôles	20	20	20	60
Cas	10	20	30	60
Somme	30	40	50	120

L'utilisation de la fonction native **prop.trend.test( )** donne:



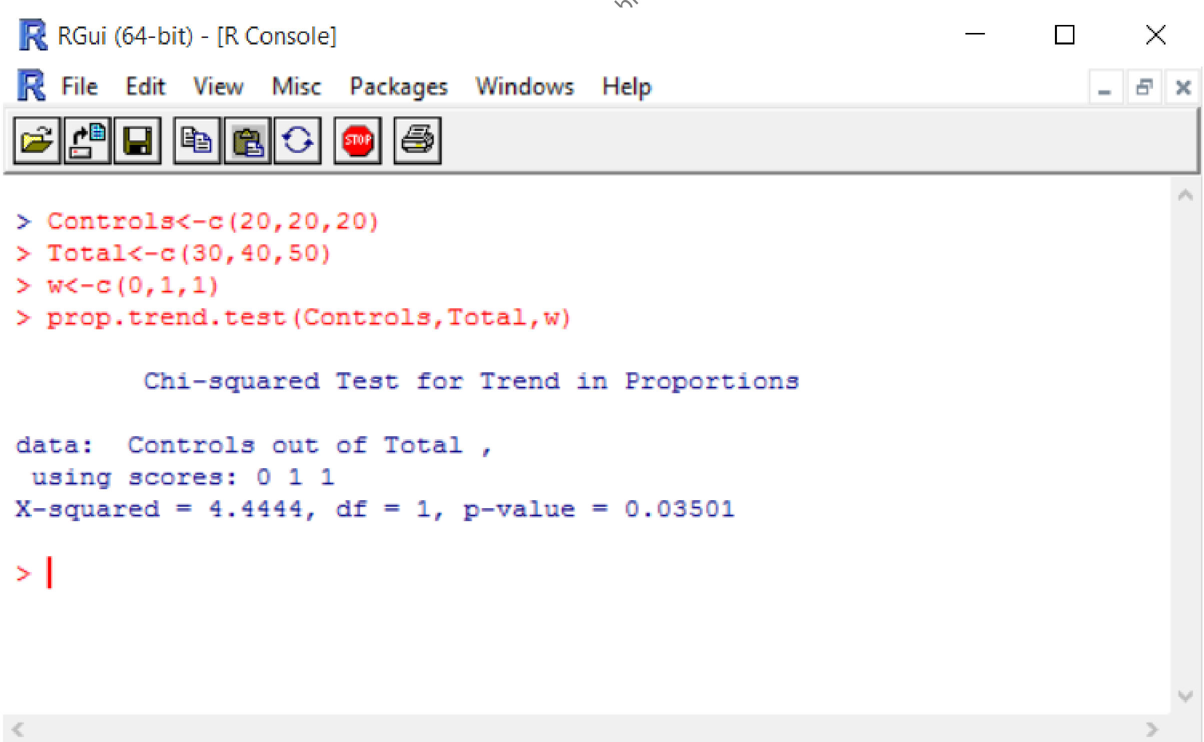
Nous voyons que les poids par défaut sont discutables… Dans le cas ci-dessus nous rejetons l'hypothèse nulle comme quoi toutes les entrées de la table son proportionnelles en faveur de l'hypothèse alternative comme quoi il y a une influence de tendance entre les deux variables catégorielles.

Dans le cadre de la génétique, les poids sont sélectionnés en conformité avec le mode d'héritage. Par exemple, pour tester si l'allèle *a* est dominante sur l'allèle *A*, nous choisirons alors les poids tels que:



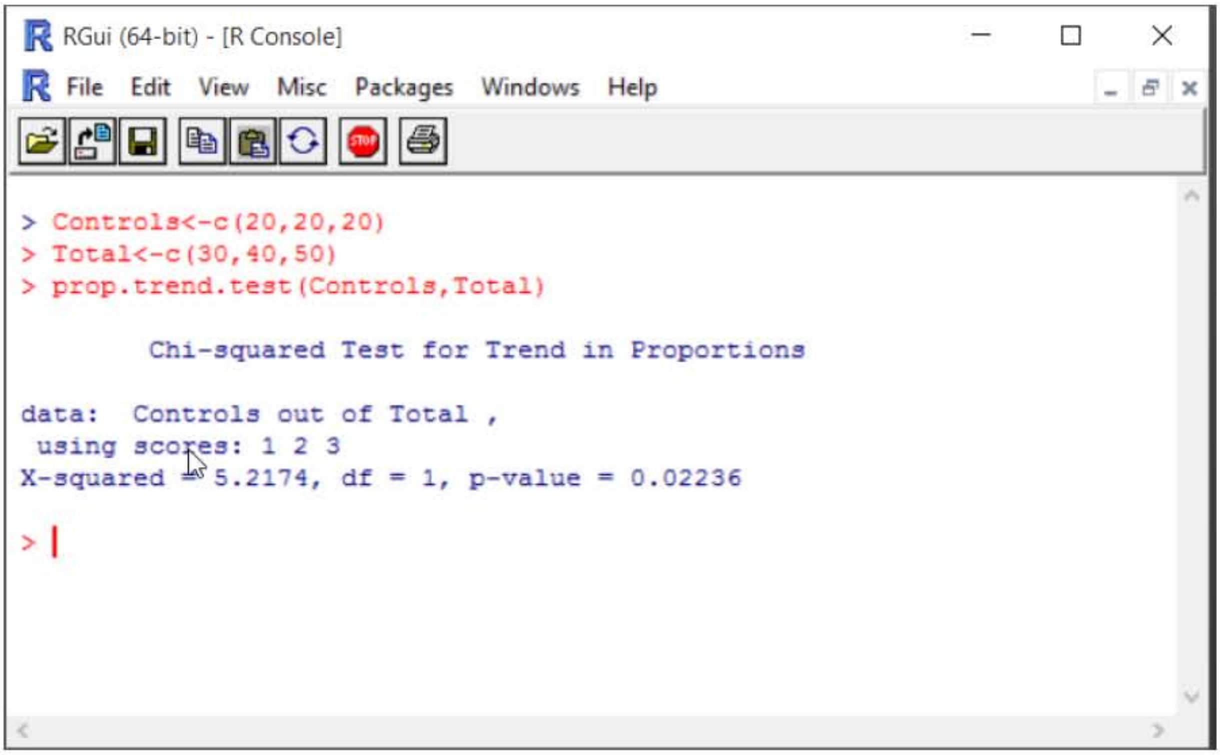
Là il n'y a donc pas d'association significative entre le nombre de cas et le type d'allèle sous l'hypothèse que l'allèle *a* est prédominante.

Si l'allèle *a* est récessive, nous choisissons alors les poids tels que:



Là il y a donc une association significative entre le nombre de cas et le type d'allèle sous l'hypothèse que l'allèle *a* est effectivement récessive.

Si les allèles *a* et *A* sont codominants alors ce sont les poids défaut de la fonction R qui s'appliquent et qui comme nous l'avons vu sont donc:

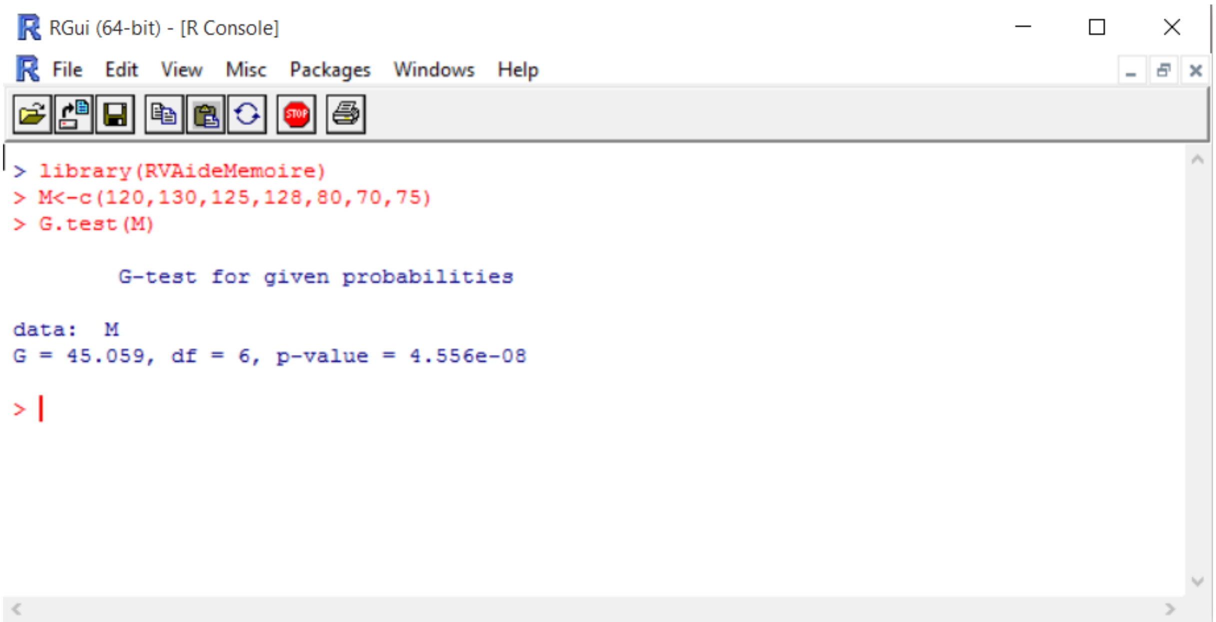


Exemple 245.: Test G^2

R 3.2.2

Nous avons vu dans le cours théorique que le test G^2 est un petit bijou du mélange entre la théorie de l'information, les développements limités et les statistiques! Il s'agissait d'une première excursion sur le fait que la théorie de l'information (entropie) peut engendrer à elle seule une grande partie de nos connaissances statistiques.

Pour vérifier si nous obtenons les mêmes calculs que ceux effectués à la main nous utilisons le même jeu que pour le test d'ajustement du khi-2. Pour cela, nous utilisons la fonction **G.test( )** du package **RVAideMemoire**:

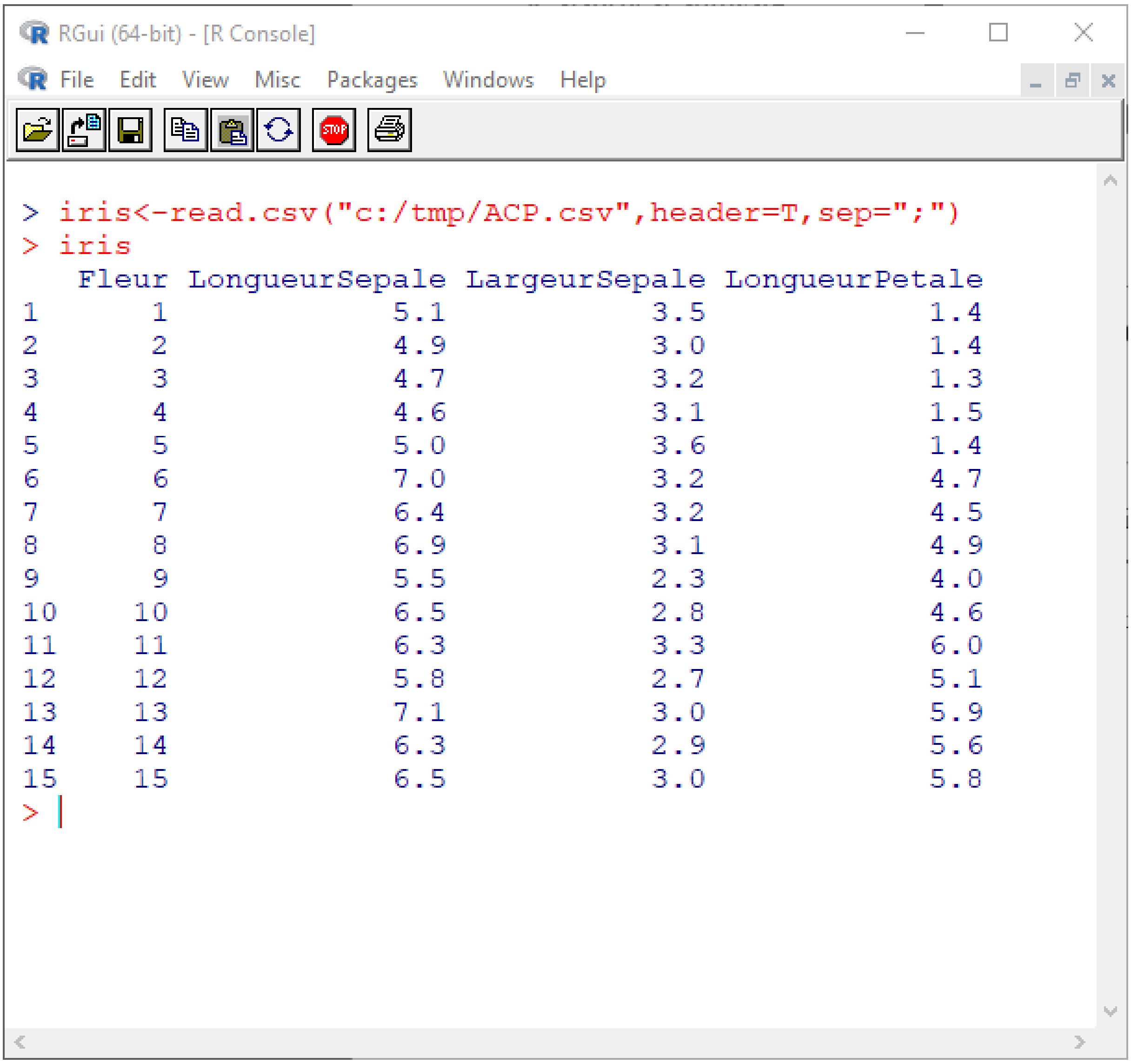


Exemple 246.: ACP (Analyses en Composantes Principales) non-paramétrique

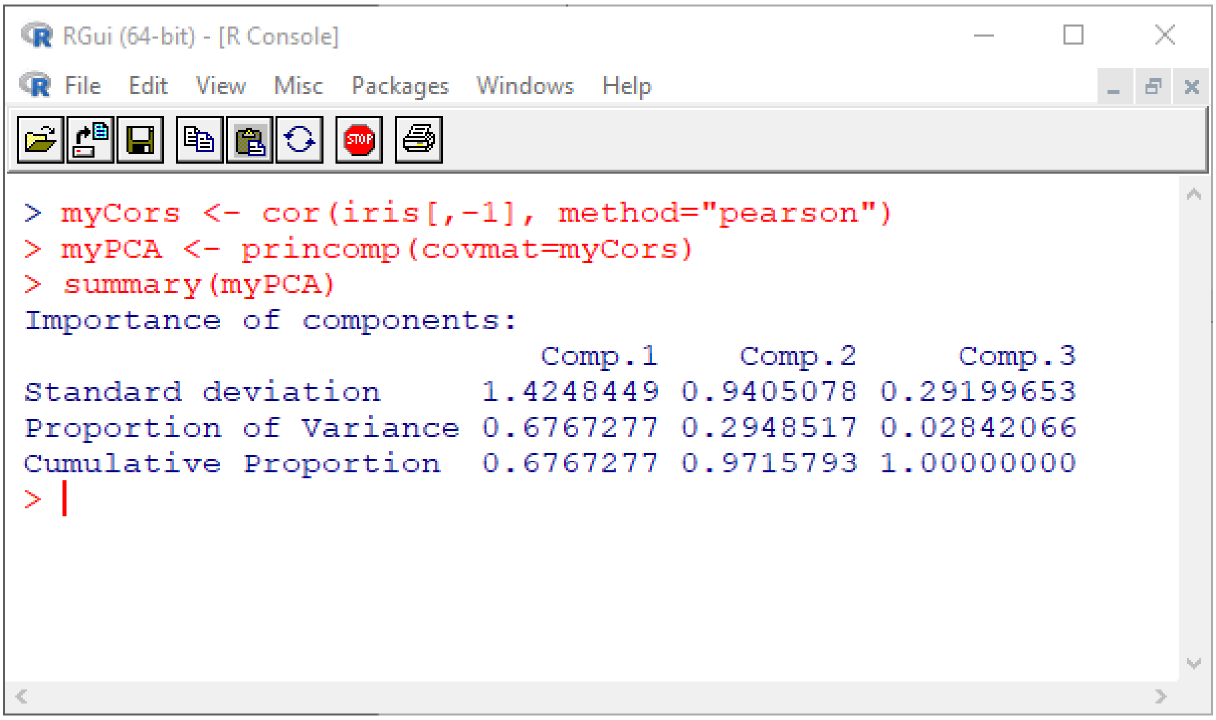
R 3.6.0

Nous avons mentionné dans le cours théorique qu'il était tout à fait possible de faire une ACP par les rangs. Voyons si nous pouvons faire cela avec R.

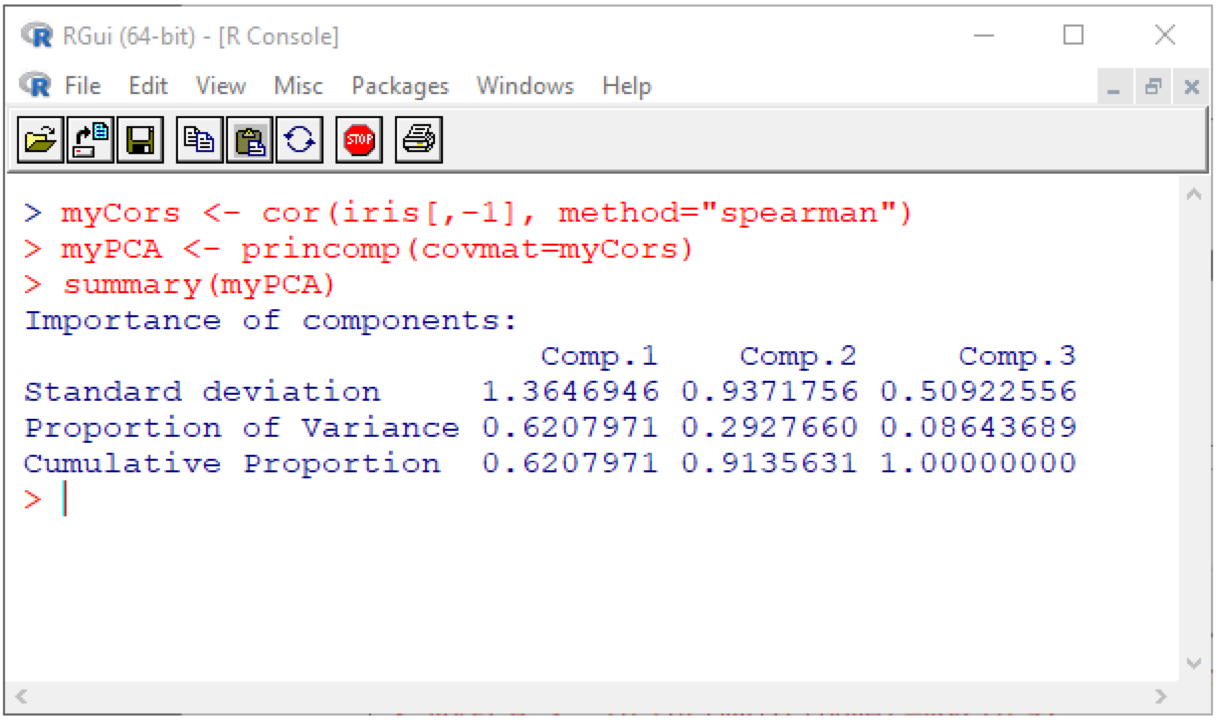
Rappelons d'abord que pour la version paramétrique, avec le jeu de données suivant:



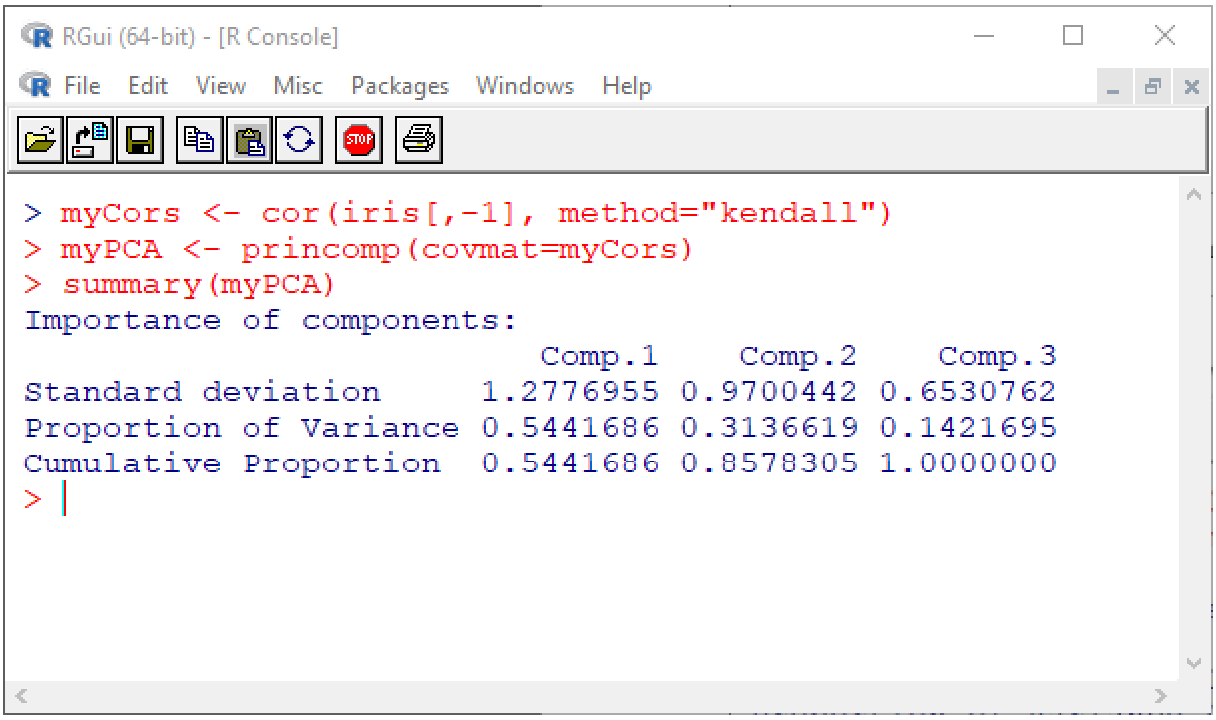
nous avions obtenu:



Voyons avec la corrélation de Spearman:



Et avec la corrélation de Kendall:



Donc nous voyons que tous les modèles amènent dans ce cas particulier, à la même conclusion!

15. Régressions (paramétriques et non paramétriques)

Ce chapitre est dédié aux différentes techniques de régressions. Rappelons que nous avons vu dans le cours théorique qu'il y en un relativement grand nombre. Nous allons passer ici en revue que celle dont nous avons étudiés et démontrés les développements mathématiques pendant le cours théorique.

Avant de commencer juste en petit rappel sur la notation dans R des modèles de régression s'avère après expérience indispensable:

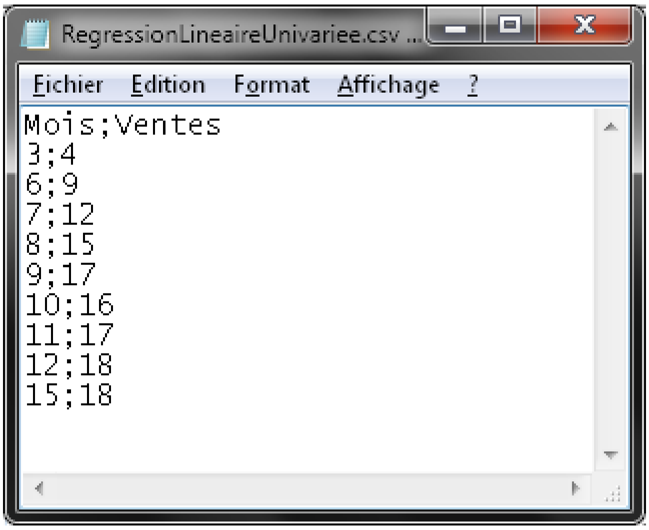
Syntaxe		Lecture
$Y \sim A$	$Y=\beta_0+\beta_1 A$	Droite avec constante. Une écriture équivalente est: $Y \sim 1+A$
$Y \sim -1+A$	$Y=\beta_1 A$	Droite affine. Une écriture équivalente est: $Y \sim 0+A$
$Y \sim A+I(A \wedge 2)$	$Y=\beta_0+\beta_1 A+\beta_2 A^2$	Régression quadratique (ou autre en changeant l'exposant)
$Y \sim A+B$	$Y=\beta_0+\beta_1 A+\beta_2 B$	Modèle multilinéaire sans interactions. Une écriture équivalente est: $Y \sim .$
$Y \sim A:B$	$Y=\beta_0+\beta_1 AB$	Modèle unique avec interaction
$Y \sim A*B$	$Y=\beta_0+\beta_1 A+\beta_2 B+\beta_3 AB$	Modèle non quadratique avec interactions
$Y \sim A \mid B$	Une saloperie à écrire avec Microsoft Word…	Modèle linéaire mixte de *A* avec tous les *B* possibles.

Tableau 3 Syntaxe des différentes régressions

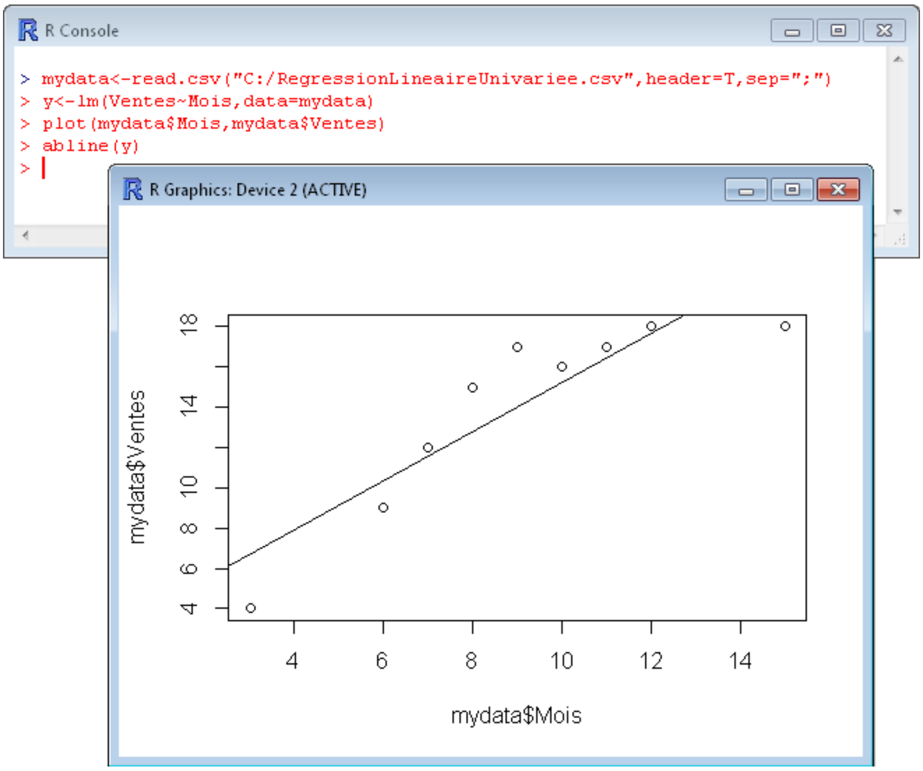
Exemple 247.: Régression linéaire univariée par moindres carrés ordinaires (modèle Gaussien)

R 3.0.2

Pour ce nous allons d'abord utiliser les mêmes données[27] que dans le cours Microsoft Excel et Minitab (nous allons aussi passer sous silence la problématique des valeurs manquantes):



Sans package dans un premier temps, nous avons en utilisant la fonction **lm( )** :



---

[27] Rappel!!! Normalement pour pouvoir appliquer le modèle Gaussien de régression linéaire il faut pour chaque point d'abscisse plusieurs mesures en ordonnées (au moins 10).

À noter que nous aurions aussi pu écrire:

- **y<-lm(Ventes ~ ., data=mydata)** où le point permet d'indiquer que nous prenons toutes les autres variables restantes.
- ou encore **y<-lm(Ventes ~ 0+Mois, data=mydata)** pour forcer l'ordonnée à l'origine

Avec un grand classique en plus au niveau de la représentation visuelle:

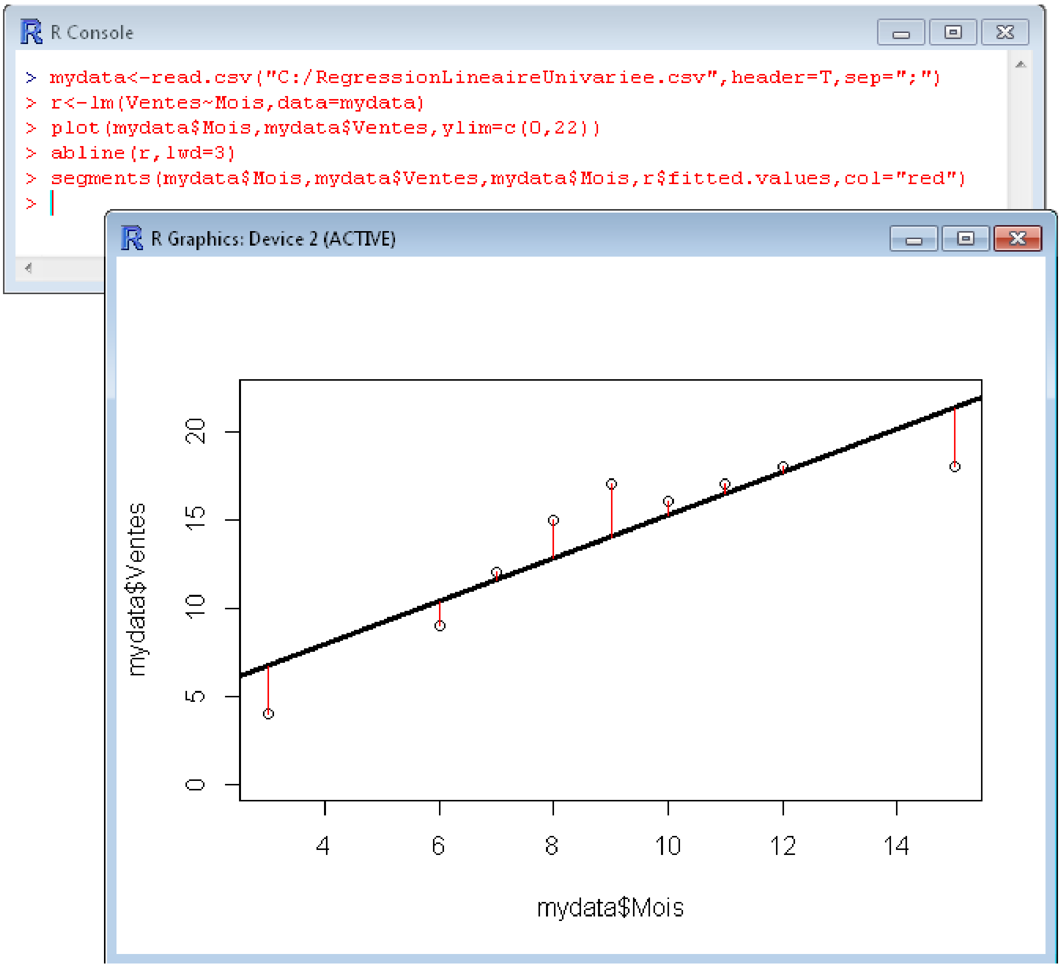
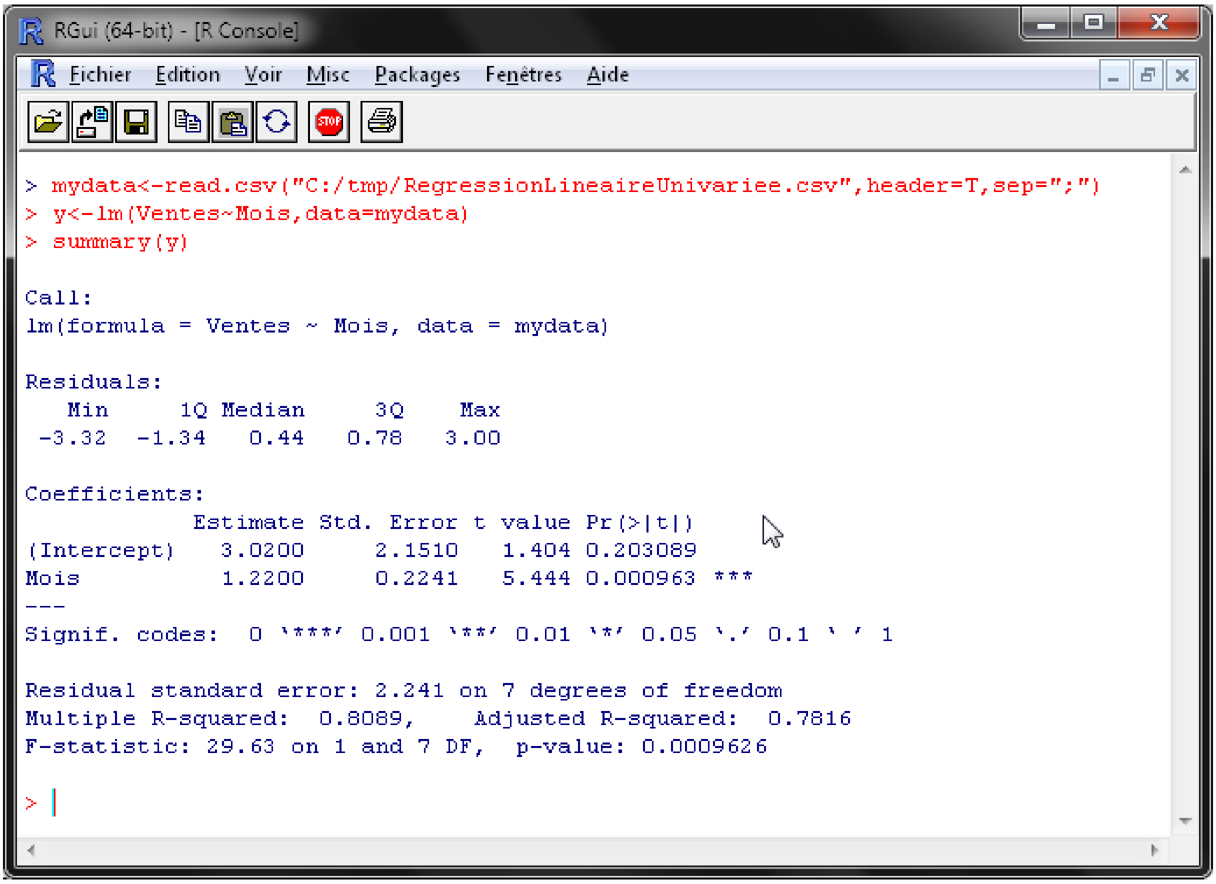
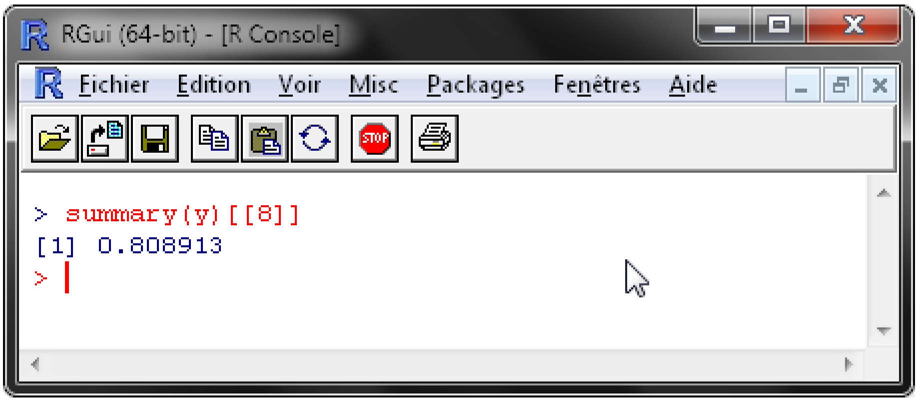


Figure 237 Graph régression avec erreurs verticales

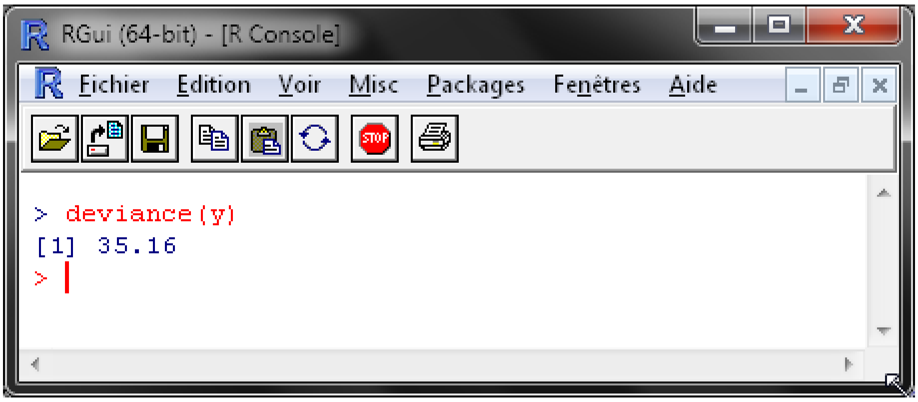
Au niveau des statistiques de la régression nous avons simplement:



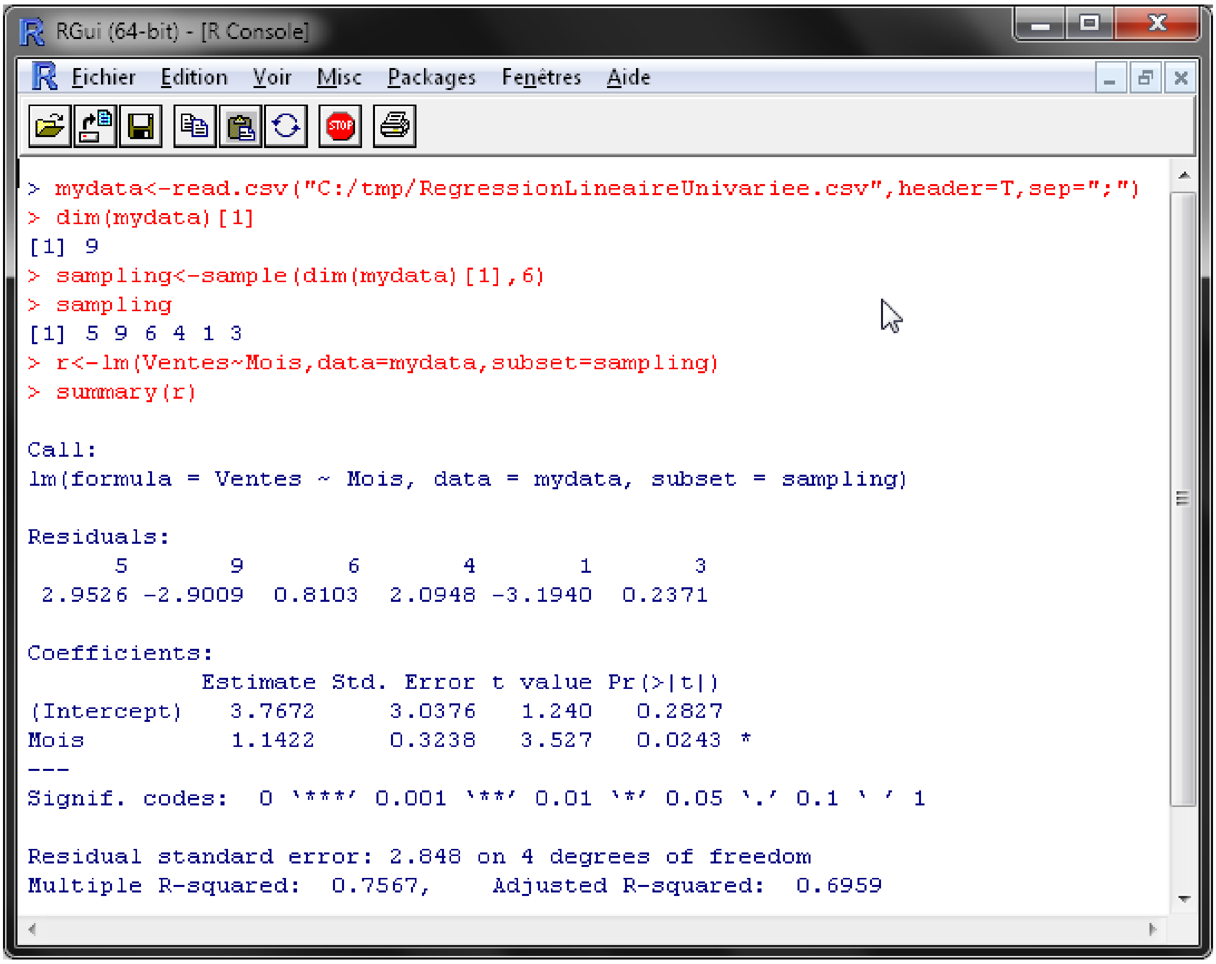
Pour extraire le coefficient de corrélation seul il suffit d'écrire (nous verrons un peu plus loin comment extraire d'autres informations):



Pour obtenir la somme des carrés des erreurs il suffit d'utiliser la fonction **deviance( )** :



Nous pouvons également directement échantillonner les données d'origines pour faire une régression sur un sous-ensemble:



Et avec la package **ggplot2** nous traçons d'abord la meilleure droite de régression selon la méthode des moindres carrés avec l'intervalle de confiance à 95% en utilisant la commande **stat_smooth( )** du package ggplot2:

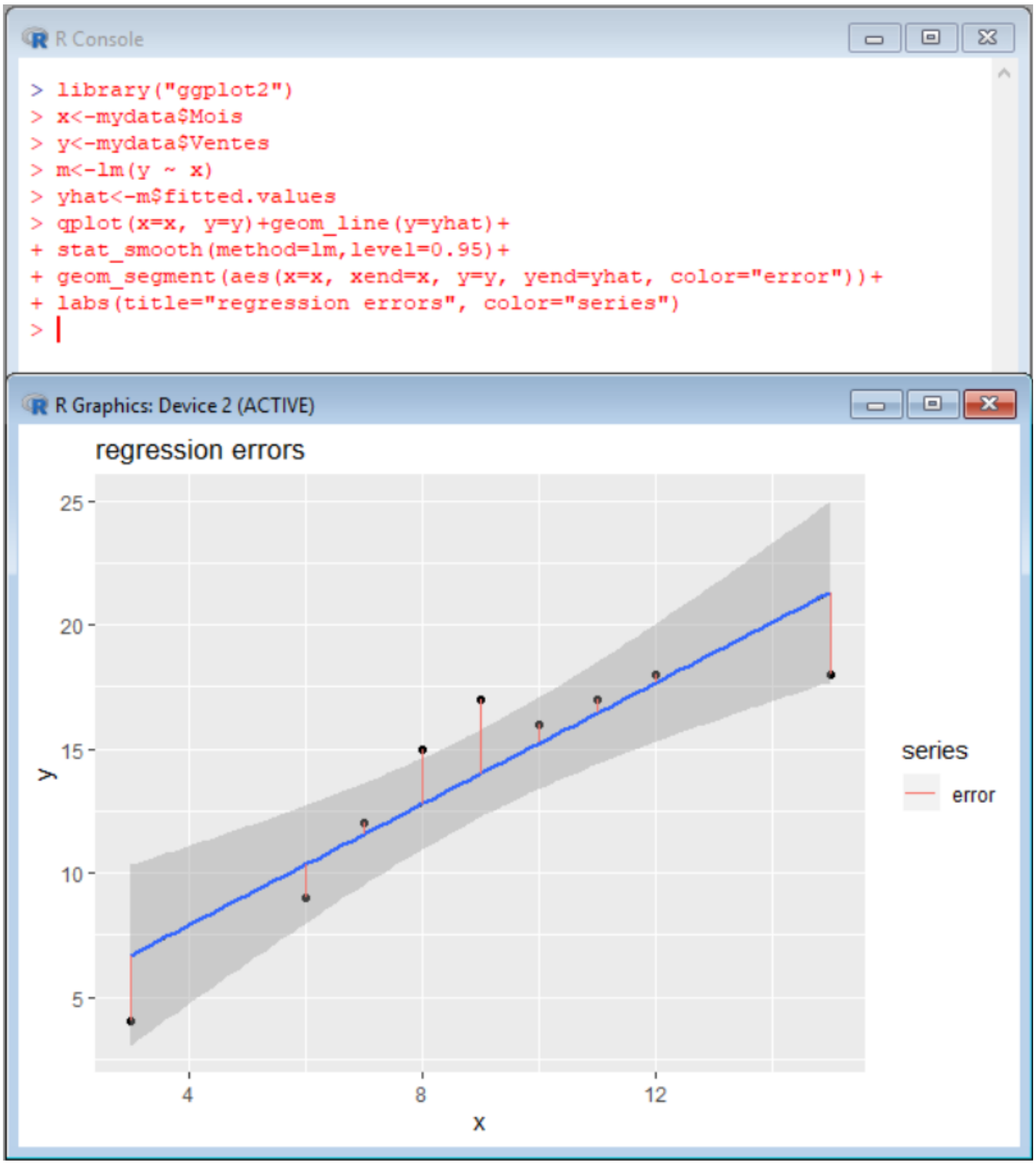
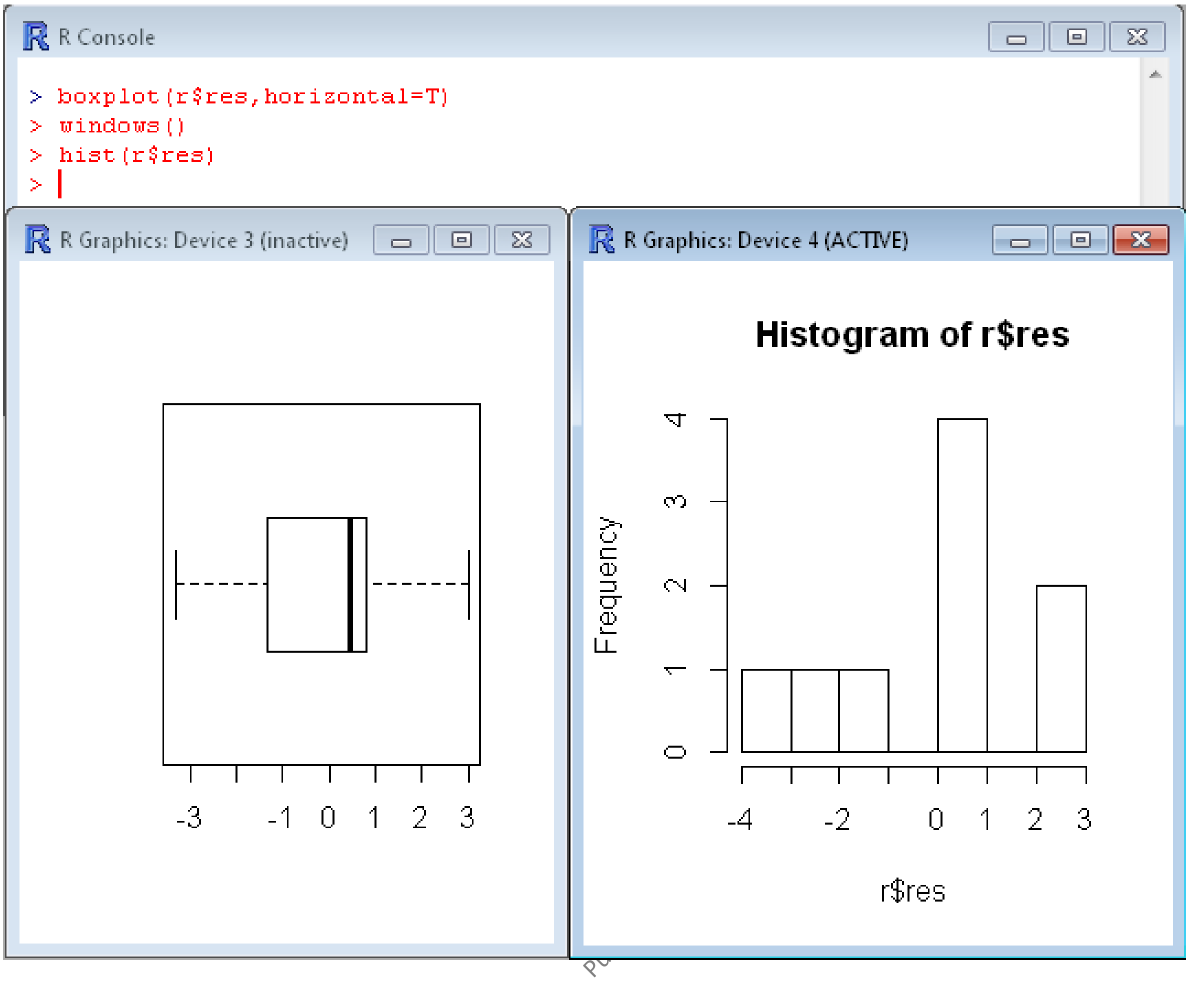


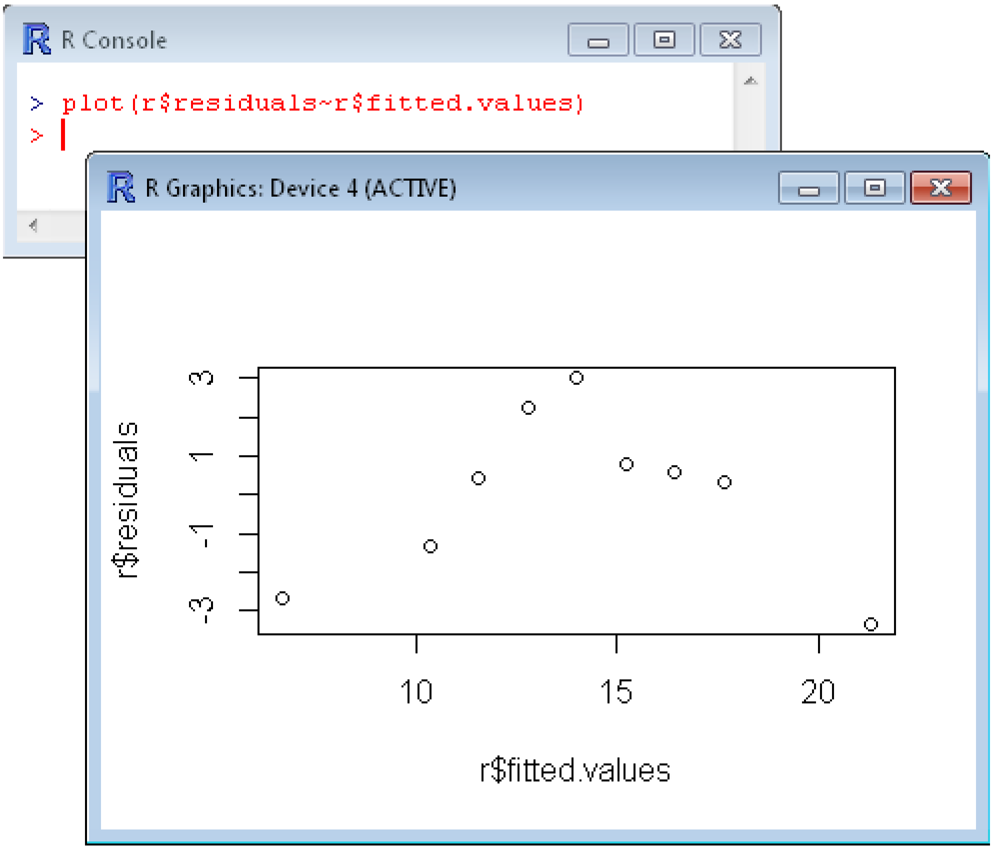
Figure 238 Graph de régression avec intervalle de confiance (ggplot)

Nous retrouvons bien la même chose que dans le cours Microsoft Excel et Minitab.

Nous pouvons aussi ploter les résidus en utilisant l'attribut **res**:

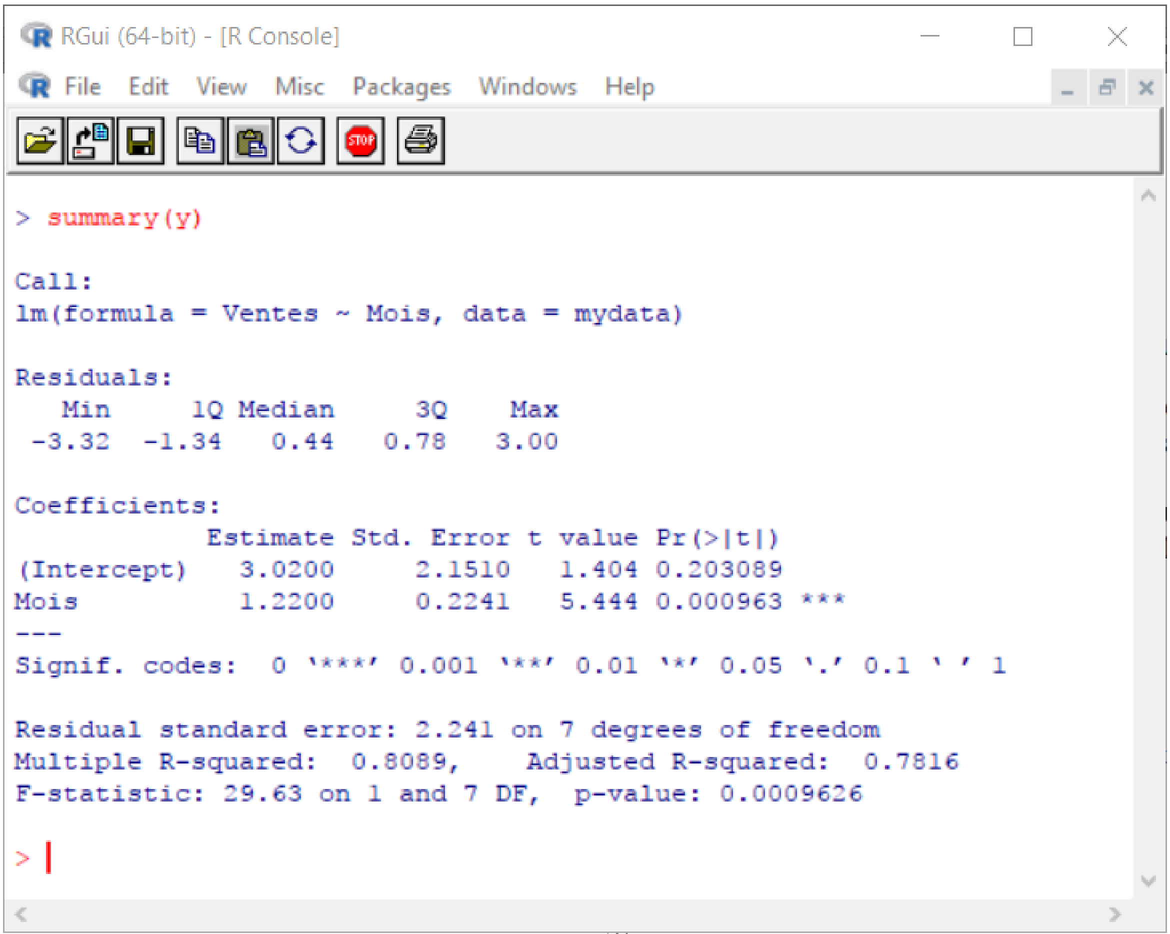


Ce qui est intéressant aussi et très important c'est de vérifier l'homoscédasticité des résidus. Nous pouvons faire cela visuellement avec la commande suivante:

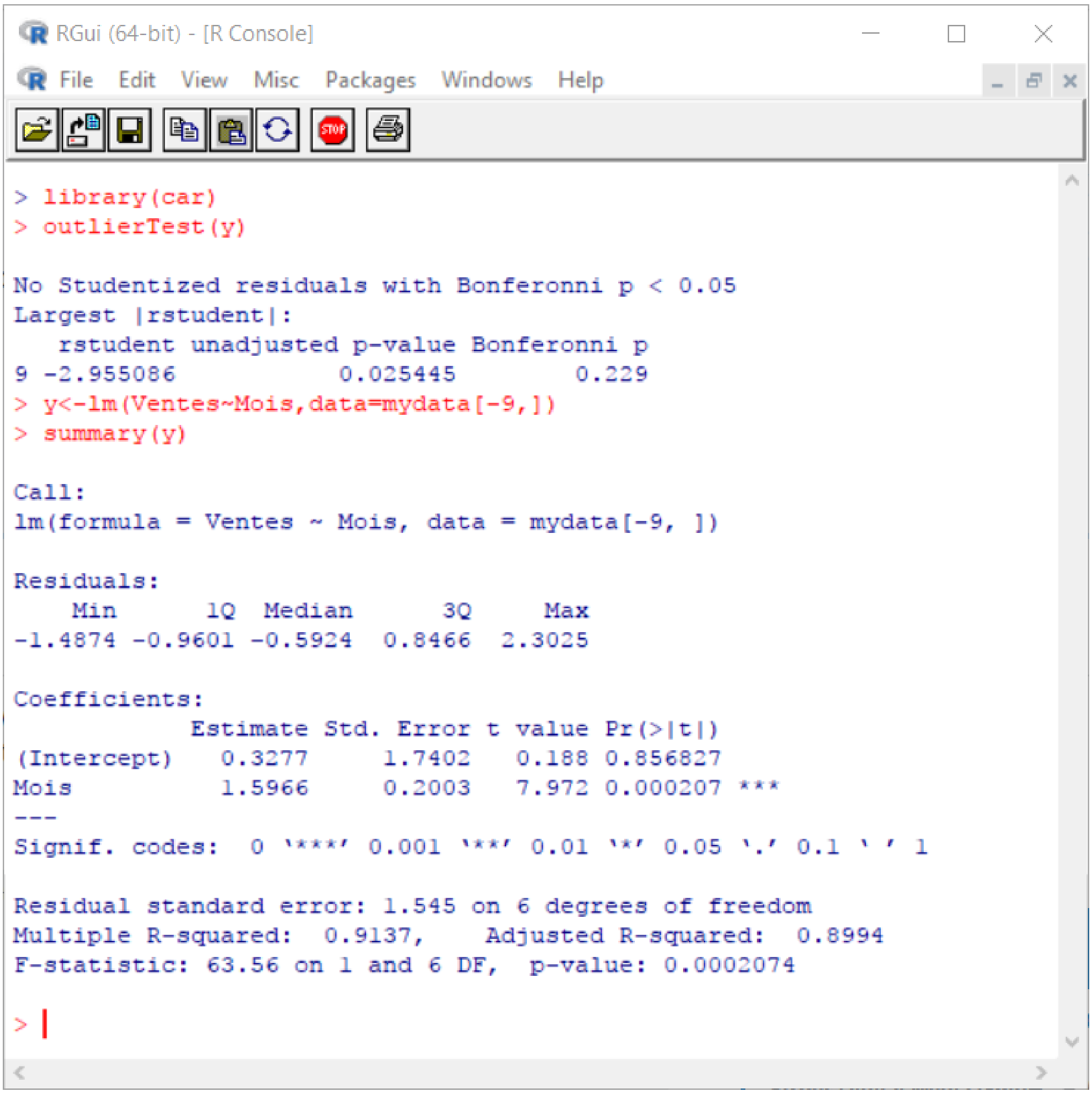


Bon évidemment... dans le cas présent comme nous n'avons qu'une seule mesure par point d'abscisse il est presque impossible de dire quoi que ce soit de qualitativement sérieux!

Avec la fonction **outlierTest( )** du package car nous pouvons identifier les valeurs "aberrantes" et au besoin les éliminer. Redonnons d'abord le modèle avec tous les points:



Maintenant allons-y pour l'élimination des valeurs aberrantes et refaire la régression:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library(car)
> outlierTest(y)

No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferonni p
9 -2.955086           0.025445        0.229
> y<-lm(Ventes~Mois,data=mydata[-9,])
> summary(y)

Call:
lm(formula = Ventes ~ Mois, data = mydata[-9, ])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.4874 -0.9601 -0.5924  0.8466  2.3025 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   0.3277     1.7402   0.188 0.856827    
Mois          1.5966     0.2003   7.972 0.000207 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.545 on 6 degrees of freedom
Multiple R-squared:  0.9137,	Adjusted R-squared:  0.8994 
F-statistic: 63.56 on 1 and 6 DF,  p-value: 0.0002074

> 
```

Le lecteur peut réitérer la procédure avec le même jeu de données et verra alors qu'éliminer les valeurs aberrantes… est scientifiquement… parfois aberrant!

Avec la commande **predict( )** , nous pouvons nous faire des projections avec intervalle de confiance pour une ou plusieurs valeurs:

RGui (64-bit) - [R Console]

Fichier Edition Voir Misc Packages Fenêtres Aide

```
> mydata<-read.csv("C:/RegressionLineaireUnivariee.csv",header=T,sep=";")
> lm_fit<-lm(Ventes~Mois,data=mydata)
> prevision<-17
> predict(lm_fit,data.frame(Mois=prevision),interval="prediction")
    fit      lwr      upr
1 23.76 16.74715 30.77285
>
```

Nous pouvons faire mieux en associant intervalle de confiance et de prédiction comme nous l'avions fait dans le cours Microsoft Excel et Minitab:

R Console

```
> library(ggplot2)
> setwd("C:/")
> mydata<-read.csv("RegressionLineaireUnivariee.csv",header=T,sep=";")
>
> lm_fit=lm(Ventes~Mois,data=mydata)
> mypredictions=data.frame(mydata,predict(lm_fit,interval="prediction"))
Message d'avis :
In predict.lm(lm_fit, interval = "prediction") :
  predictions on current data refer to _future_ responses

> mypredictions
  Mois Ventes   fit        lwr      upr
1    3      4  6.68  0.2522306 13.10777
2    6      9 10.34  4.5319661 16.14803
3    7     12 11.56  5.8741398 17.24586
4    8     15 12.78  7.1687208 18.39128
5    9     17 14.00  8.4138023 19.58620
6   10     16 15.22  9.6087208 20.83128
7   11     17 16.44 10.7541398 22.12586
8   12     18 17.66 11.8519661 23.46803
9   15     18 21.32 14.8922306 27.74777
>
> ggplot(mypredictions,aes(x=Mois,y=Ventes))+geom_point()+
+ geom_smooth(method="lm",aes(fill="confidence"),alpha=0.5,level=0.95)+
+ geom_ribbon(aes(y=fit,ymin=lwr,ymax=upr,fill="prediction"),
+ alpha = 0.2)+ scale_fill_manual("Interval",values=c("green","blue"))+
+ theme(legend.position=c(0.20,0.85))
>
```

Ce qui donne:

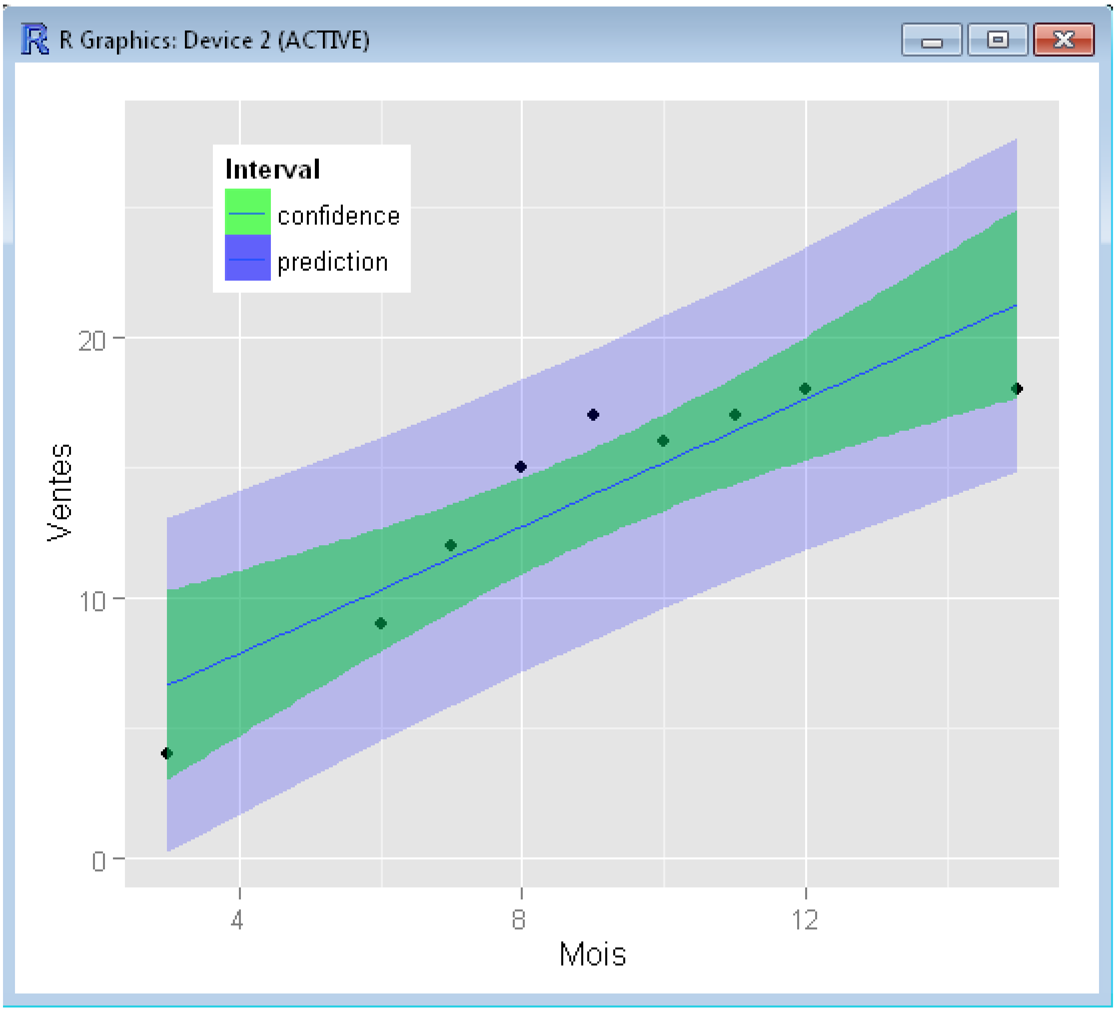


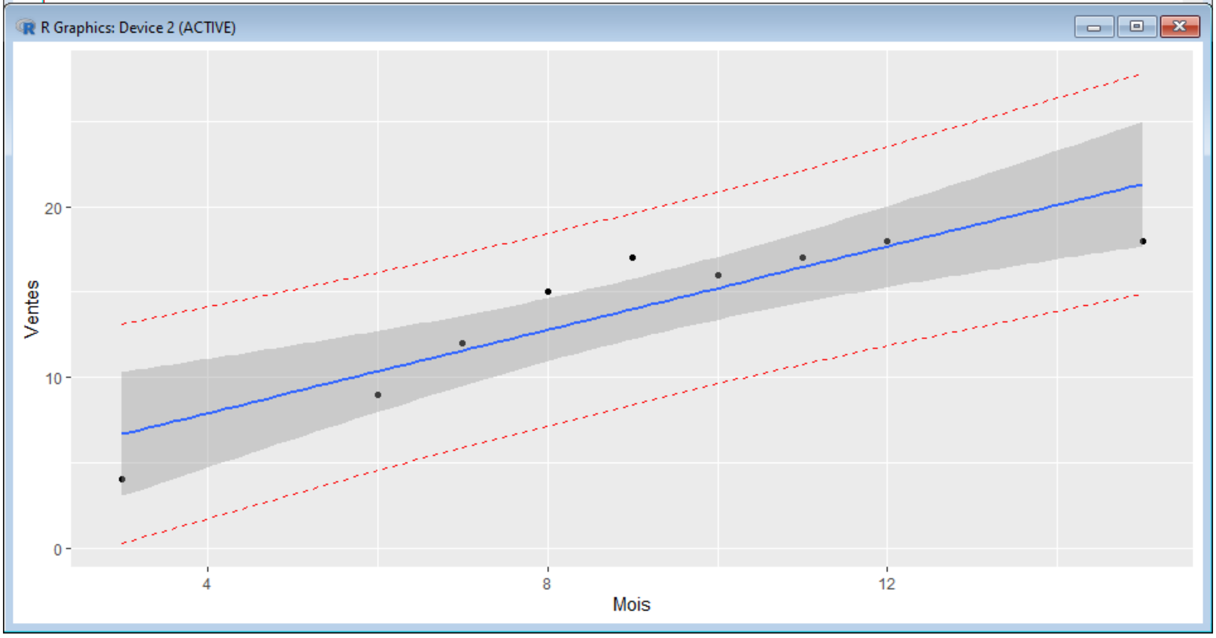
Figure 239 Graph régression avec intervalle de confiance et de prédiction (ggplot)

Et donc nous retrouvons bien la même chose que dans les cours Microsoft Excel et Minitab.

Ou différemment:

```
R Console
> library("ggplot2")
> lm_fit<-lm(Ventes~Mois,data=dataset)
> pred.int <- predict(lm_fit, interval = "prediction")
Warning message:
In predict.lm(lm_fit, interval = "prediction") :
  predictions on current data refer to _future_ responses

> (mydata <- cbind(dataset, pred.int))
  Mois Ventes   fit        lwr      upr
1    3      4  6.68  0.2522306 13.10777
2    6      9 10.34  4.5319661 16.14803
3    7     12 11.56  5.8741398 17.24586
4    8     15 12.78  7.1687208 18.39128
5    9     17 14.00  8.4138023 19.58620
6   10     16 15.22  9.6087208 20.83128
7   11     17 16.44 10.7541398 22.12586
8   12     18 17.66 11.8519661 23.46803
9   15     18 21.32 14.8922306 27.74777
> p <- ggplot(mydata, aes(Mois, Ventes)) +
+   geom_point() +
+   stat_smooth(method = lm)
> # Add prediction intervals
> p + geom_line(aes(y = lwr), color = "red", linetype = "dashed")+
+     geom_line(aes(y = upr), color = "red", linetype = "dashed")
`geom_smooth()` using formula = 'y ~ x'
> |
```

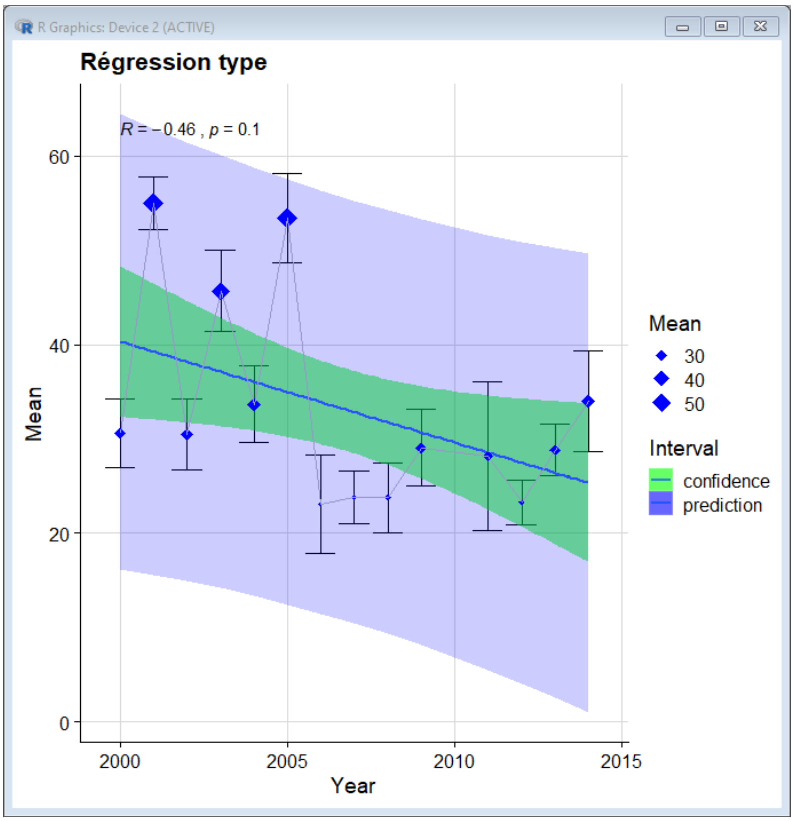


Nous pouvons cependant faire mieux et en conformité avec la tradition scientifique (on change de jeu de données pour cet exemple!!!):

```
> library("tidyverse")
> library("ggplot2")
> library("ggpubr")
> library("cowplot")
>
> mydata <- data_frame(
+     Year = c(2000L, 2001L, 2002L, 2003L, 2004L, 2005L, 2006L, 2007L, 2008L,
+ 2009L, 2011L, 2012L, 2013L, 2014L),
+     Mean = c(30.62089116, 54.99066324, 30.48859003, 45.70410611, 33.68256601,
+ 53.43496462, 23.08799875, 23.79166667, 23.75925926, 29.05555556, 28.13333333,
+ 23.3, 28.83333333, 34.01666667),
+     SE = c(3.672624771, 2.822405771, 3.748388489, 4.284244405, 4.030964172,
+ 4.687042033, 5.17753488, 2.837795432, 3.688185081, 4.08597798, 7.877429079,
+ 2.383624691, 2.750959429, 5.340999698)
+ )
>
> model <- lm(Mean ~ Year, data = mydata )
> pred.int <- predict(model, interval = "prediction")
Warning message:
In predict.lm(model, interval = "prediction") :
  predictions on current data refer to _future_ responses

> mydata <- cbind(mydata , pred.int)
>
> ggplot(mydata, aes(Year, Mean)) +
+     geom_errorbar(aes(ymin = Mean - SE, ymax = Mean + SE)) +
+     geom_smooth(data = mydata,method = "gam", aes(fill="confidence"),alpha=0.5,level=0.9) +
+     geom_point(shape=18, color="blue",aes(size = Mean))+
+     geom_line(color="gray")+
+     geom_ribbon(aes(ymin=lwr,ymax=upr,fill="prediction"),alpha=0.2)+
+     scale_fill_manual("Interval",values=c("green","blue"))+
+     labs(title="Régression type")+
+     stat_cor(method = "pearson")+
+     theme_half_open() +
+     background_grid()
>
```

Ce qui donne:



Maintenant passons à l'aspect un peu plus technique en utilisant la commande **lm( )** :

```
R Console
> setwd("C:/")
> mydata<-read.csv("RegressionLineaireUnivariee.csv",header=T,sep=";")
> summary(ols<-lm(formula=Ventes~Mois,data=mydata))

Call:
lm(formula = Ventes ~ Mois, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max 
 -3.32  -1.34   0.44   0.78   3.00 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3.0200     2.1510   1.404 0.203089    
Mois          1.2200     0.2241   5.444 0.000963 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.241 on 7 degrees of freedom
Multiple R-squared:  0.8089,    Adjusted R-squared:  0.7816 
F-statistic: 29.63 on 1 and 7 DF,  p-value: 0.0009626

> 
```

Nous retrouvons la même chose que dans Microsoft Excel et Minitab à la différence qu'il y a des informations manquantes avec R qui sont intéressantes dans la pratique et que la mise en page n'est pas des meilleures...

Nous pouvons cependant faire mieux pour récupérer les valeurs manquantes:

```
R Console
> setwd("C:/")
> mydata<-read.csv("RegressionLineaireUnivariee.csv",header=T,sep=";")
> ols<-lm(formula=Ventes~Mois,data=mydata)
> attributes(ols) #liste des paramètres qu'on peut récuperer
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"

> ols$coefficients #récuper la liste des coefficients seuls
(Intercept)        Mois
       3.02        1.22
> confint(ols)  #intervalle de confiance des coefficients
                 2.5 %   97.5 %
(Intercept) -2.0662015 8.106202
Mois         0.6900468 1.749953
> anova(ols)
Analysis of Variance Table

Response: Ventes
          Df Sum Sq Mean Sq F value    Pr(>F)
Mois       1 148.84 148.840  29.633 0.0009626 ***
Residuals  7  35.16   5.023
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

Nous pouvons aussi afficher un graphique un peu similaire à celui de Minitab mais plus technique incluant le leverage (niveau de levier) et la distance de Cook:

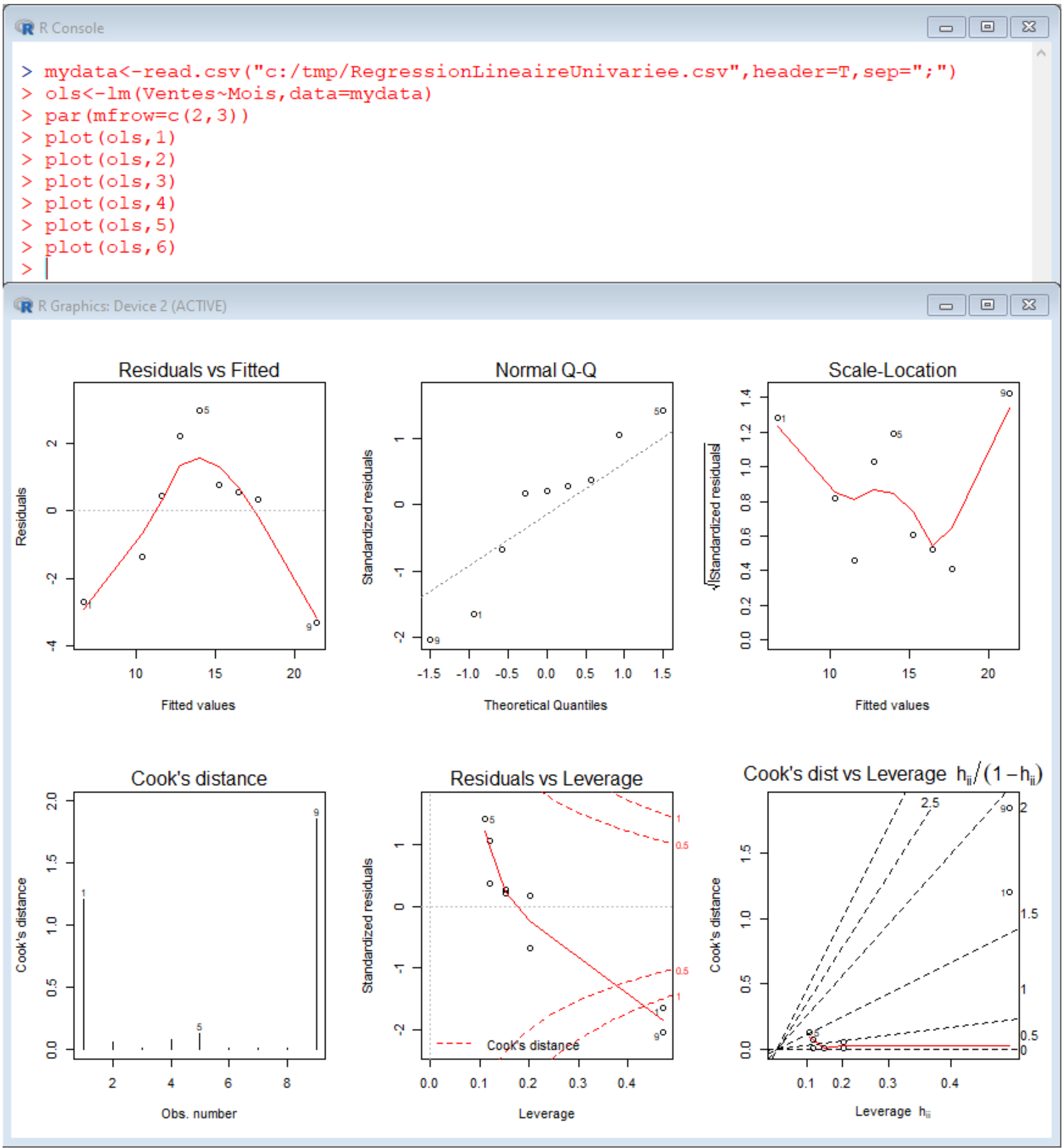
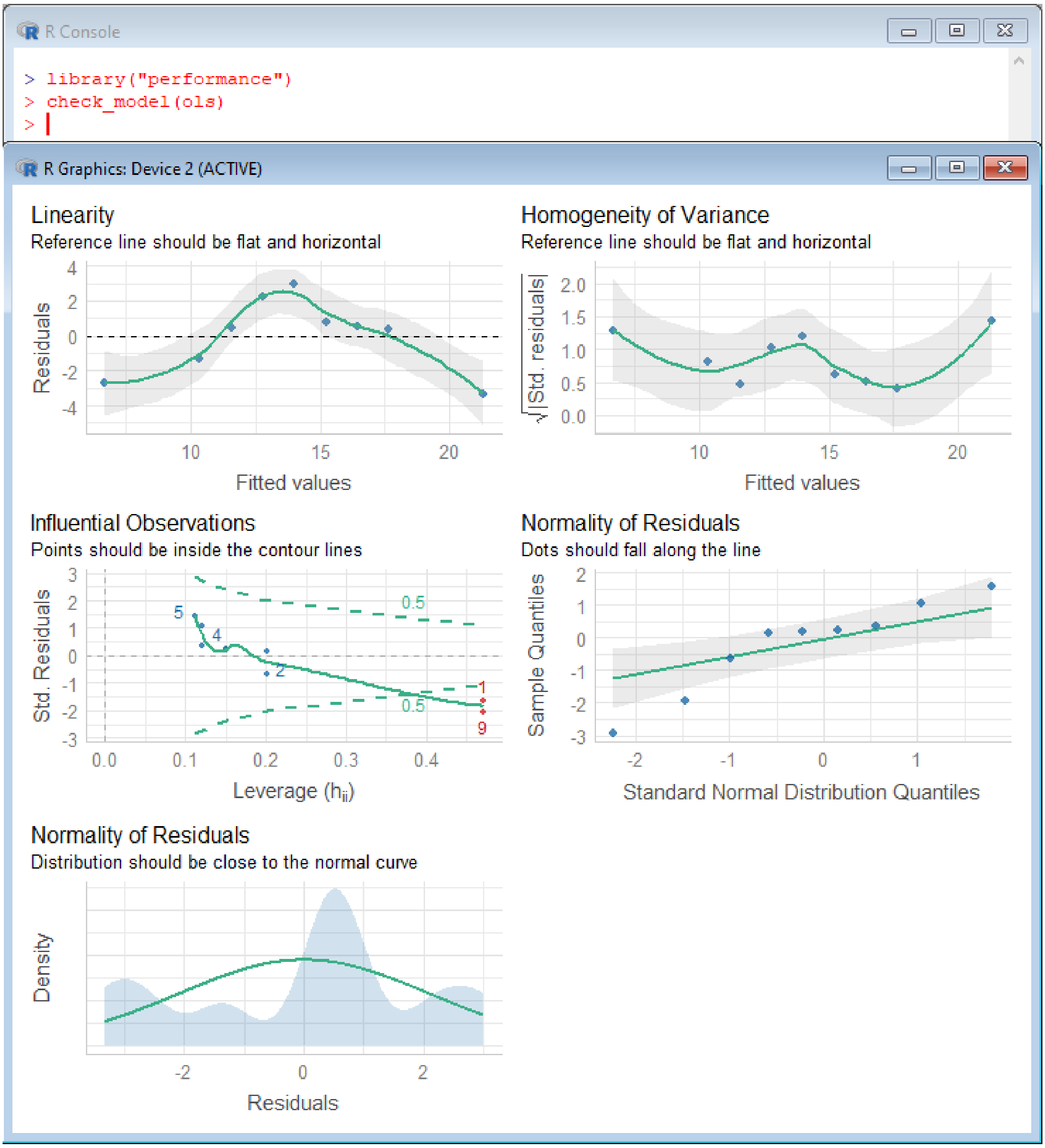


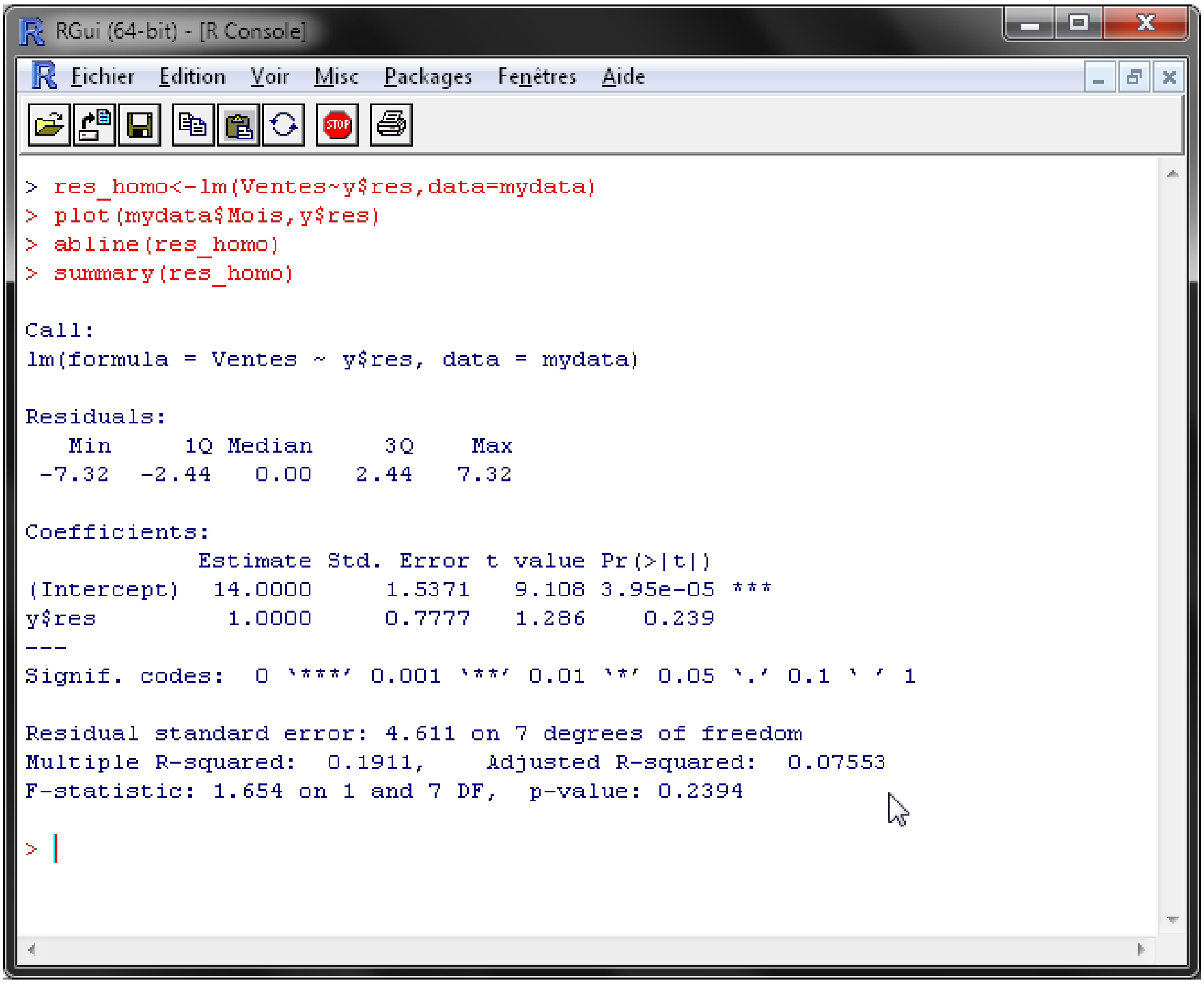
Figure 240 Graph de diagnostique de la régression

Sinon de manière plus courte et plus élégante avec le package **performance** et sa fonction **check_model( )**:



où on dépasse largement la qualité et la pertinence de sortie de JMP ou Minitab!

Nous pouvons faire un test de l'homoscédasticité sur la régression des résidus puisque si ces derniers satisfont les hypothèses, la droite passant par les résidus devrait avoir une pente nulle (ou non significative) ainsi qu'une ordonnée à l'origine nulle. Or, dans le cas présent:



```
> res_homo<-lm(Ventes~y$res,data=mydata)
> plot(mydata$Mois,y$res)
> abline(res_homo)
> summary(res_homo)

Call:
lm(formula = Ventes ~ y$res, data = mydata)

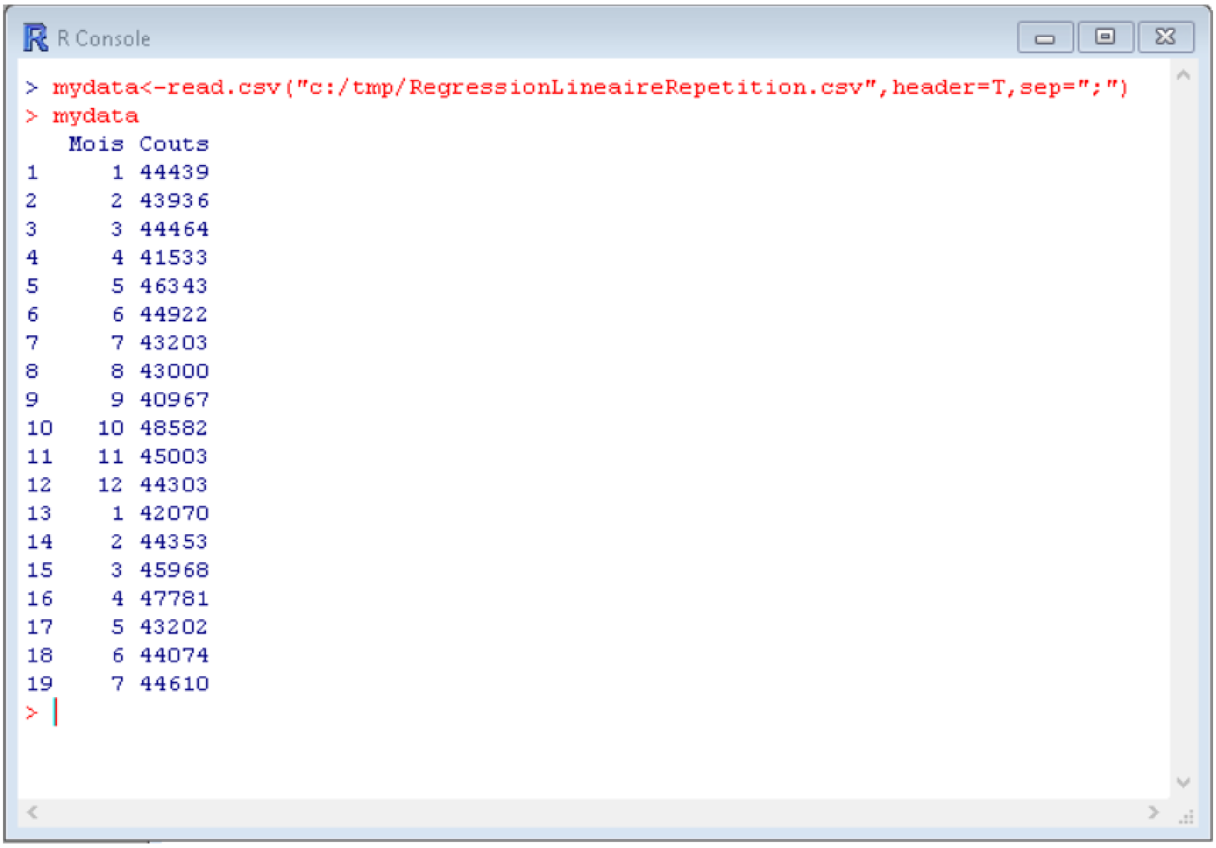
Residuals:
   Min     1Q Median     3Q    Max 
 -7.32  -2.44   0.00   2.44   7.32 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  14.0000     1.5371   9.108 3.95e-05 ***
y$res         1.0000     0.7777   1.286    0.239    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.611 on 7 degrees of freedom
Multiple R-squared:  0.1911,	Adjusted R-squared:  0.07553 
F-statistic: 1.654 on 1 and 7 DF,  p-value: 0.2394

> 
```

Revenons maintenant à du ludique. Si nous avons un jeu de données du type suivant où certaines valeurs d'abscisse se répètent:



Nous pouvons alors de façon élégante représenter l'amplitude de variation de chaque point dans le cadre d'une régression loess+linéaire avec l'ellipse de confiance avec le script suivant où nous avons mis en rouge les paramètres avec lesquels on joue normalement en fonction du jeu de données:

```
do.it <- function(df, type="confidence", ...) {
  require(ellipse)
  lm0 <- lm(y ~ x, data=df)
  xc <- with(df, xyTable(x, y,digit=1))
  df.new <- data.frame(x=seq(min(df$x), max(df$x), 1))
  pred.ulb <- predict(lm0, df.new, interval=type)
  pred.lo <- predict(loess(y ~ x, data=df), df.new)
  plot(df$x, df$y, cex=xc$number*2, xlab="x", ylab="y", ...)
  abline(lm0, col="red")
  lines(df.new$x, pred.lo, col="green", lwd=1.5)
  lines(df.new$x, pred.ulb[,"lwr"], lty=2, col="red")
  lines(df.new$x, pred.ulb[,"upr"], lty=2, col="red")
  lines(ellipse(cor(df$x, df$y), scale=c(sd(df$x),sd(df$y)),
        centre=c(mean(df$x),mean(df$y))), lwd=1.5, col="green")
  invisible(lm0)
}

mydata<-
read.csv("c:/tmp/RegressionLineaireRepetition.csv",header=T,sep=";")
df<-data.frame(x=mydata$Mois,y=mydata$Couts)
do.it(df, pch=19, col=rgb(0,0,.7,.5))
```

Ce qui donne:

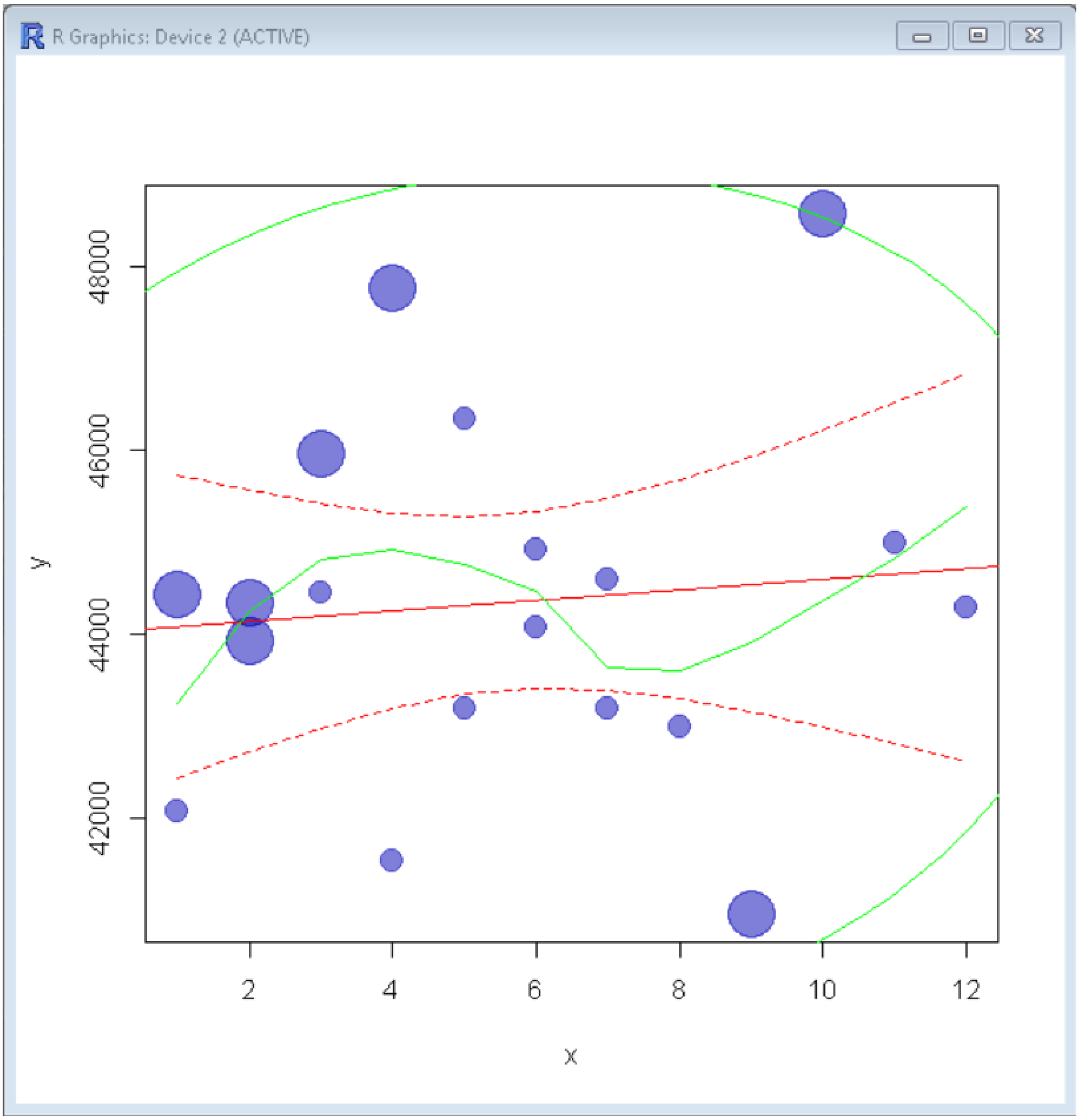


Figure 241 Graph de régression avec intervalle de confiance et prédiction et poids des points

Et en prenant la moyenne des points (attention!!! Il n'est pas possible d'avoir le vrai rayon des cercles pour l'écart-type dès le moment où les deux axes n'ont pas les mêmes échelles):

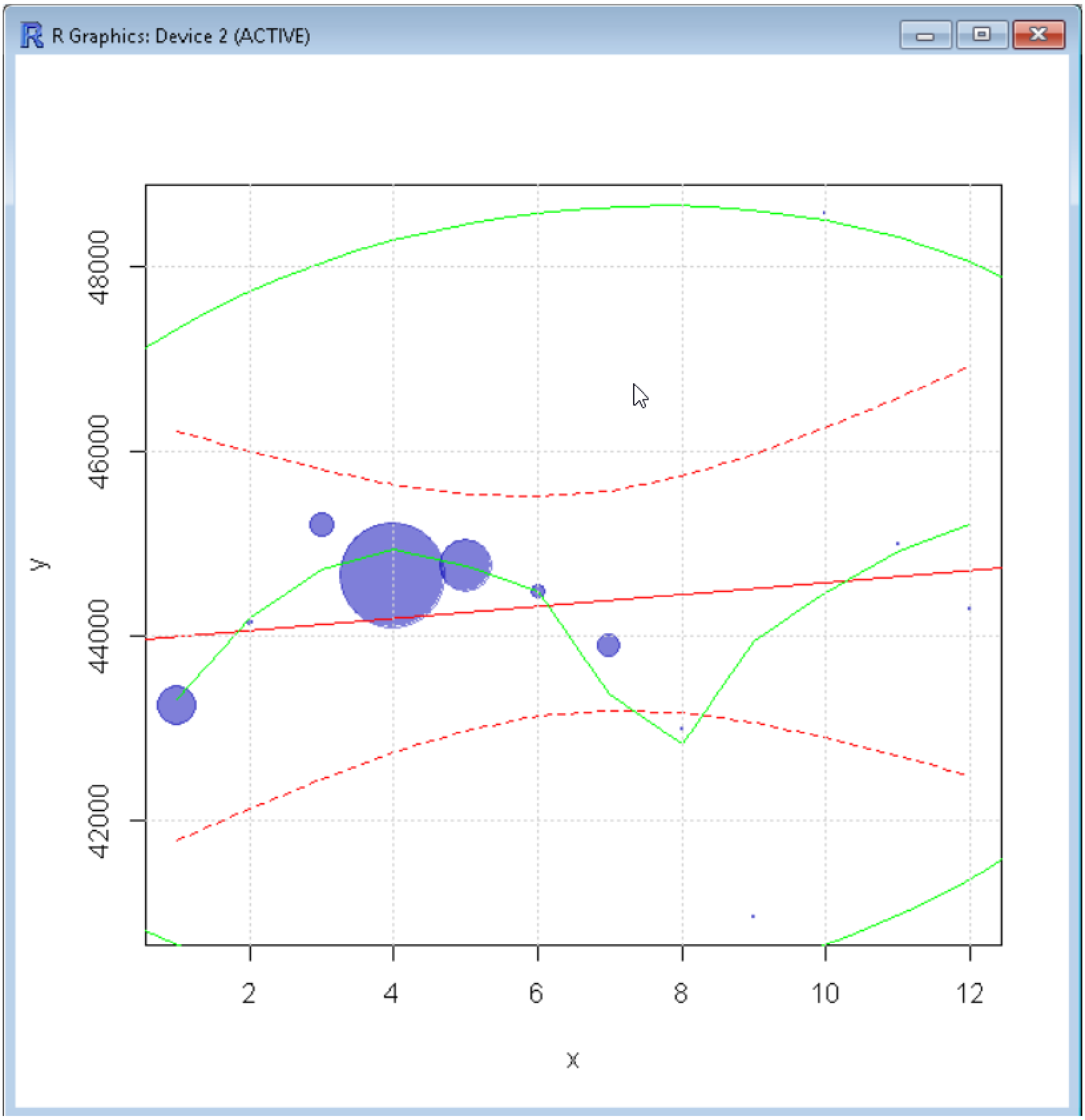
```
mydata<-
read.csv("C:/tmp/RegressionLineaireRepetition.csv",header=T,sep=";")
mydata.mean<-aggregate(mydata[2],by=list(mydata$Mois),FUN=mean)
mydata.sd<-aggregate(mydata[2],by=list(mydata$Mois),FUN=sd)
mydata<-merge(mydata.mean,mydata.sd,by="Group.1")
colnames(mydata)<-c("Mois","Moyenne","StDev")
mydata[is.na(mydata)]<-1
mydata

do.it <- function(df, type="confidence", ...) {
  require(ellipse)
  lm0 <- lm(y ~ x, data=df)
  df.new <- data.frame(x=seq(min(df$x), max(df$x), 1))
  pred.ulb <- predict(lm0, df.new, interval=type)
```

```
  pred.lo <- predict(loess(y ~ x, data=df), df.new)
  #Attention essayer de faire des disques de vrais rayons n'a pas de sens
  #dès le moment que l'axe X et Y n'ont pas la même échelle!!!
  plot(df$x, df$y, cex=df$stdev/500, xlab="x", ylab="y", ...)
  abline(lm0, col="red")
  lines(df.new$x, pred.lo, col="green", lwd=1.5)
  lines(df.new$x, pred.ulb[,"lwr"], lty=2, col="red")
  lines(df.new$x, pred.ulb[,"upr"], lty=2, col="red")
  lines(ellipse(cor(df$x, df$y), scale=c(sd(df$x),sd(df$y)),
  centre=c(mean(df$x),mean(df$y))), lwd=1.5, col="green")
  invisible(lm0)
  grid()
}

df<-data.frame(x=mydata$Mois,y=mydata$Moyenne,stdev=mydata$StDev)
do.it(df, pch=19, col=rgb(0,0,.7,.5))
```

Ce qui donne:



Effet de levier (leverage) et distance de Cook

Dans le cours théorique nous avons mentionné qu'il existait de nombreuses techniques empiriques pour mesurer l'influence d'un point. R propose quelques classiques à ce propos et dans le cadre de ce présent support de cours nous allons nous concentrer sur le seul dont nous avons démontré l'origine mathématique détaillée dans le cours théorique et dont de toute façon la Distance de Cook et le DFITS découlent: l'effet de levier H_i.

D'abord voyons comment afficher le graph de l'effet de levier seul:

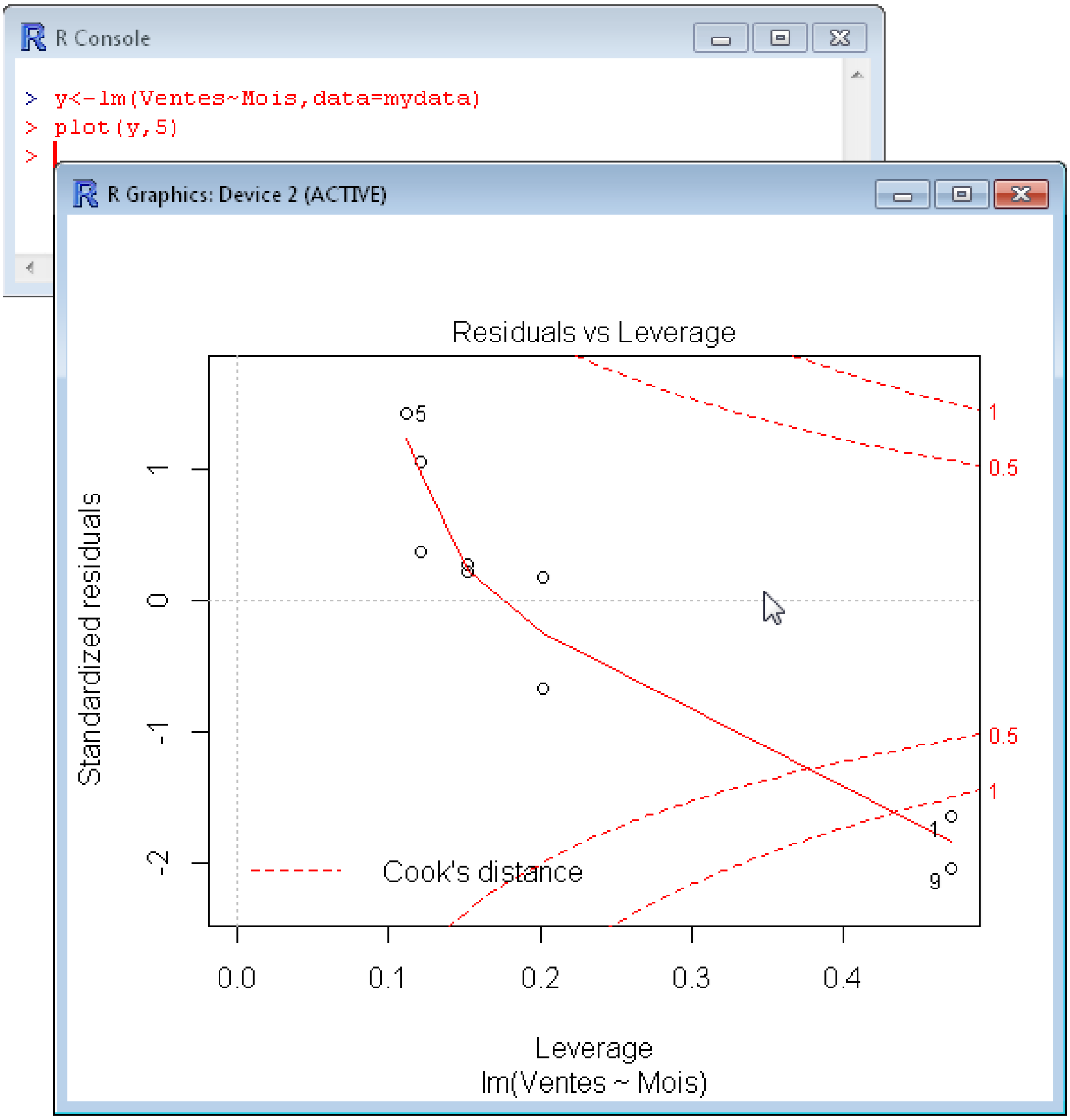
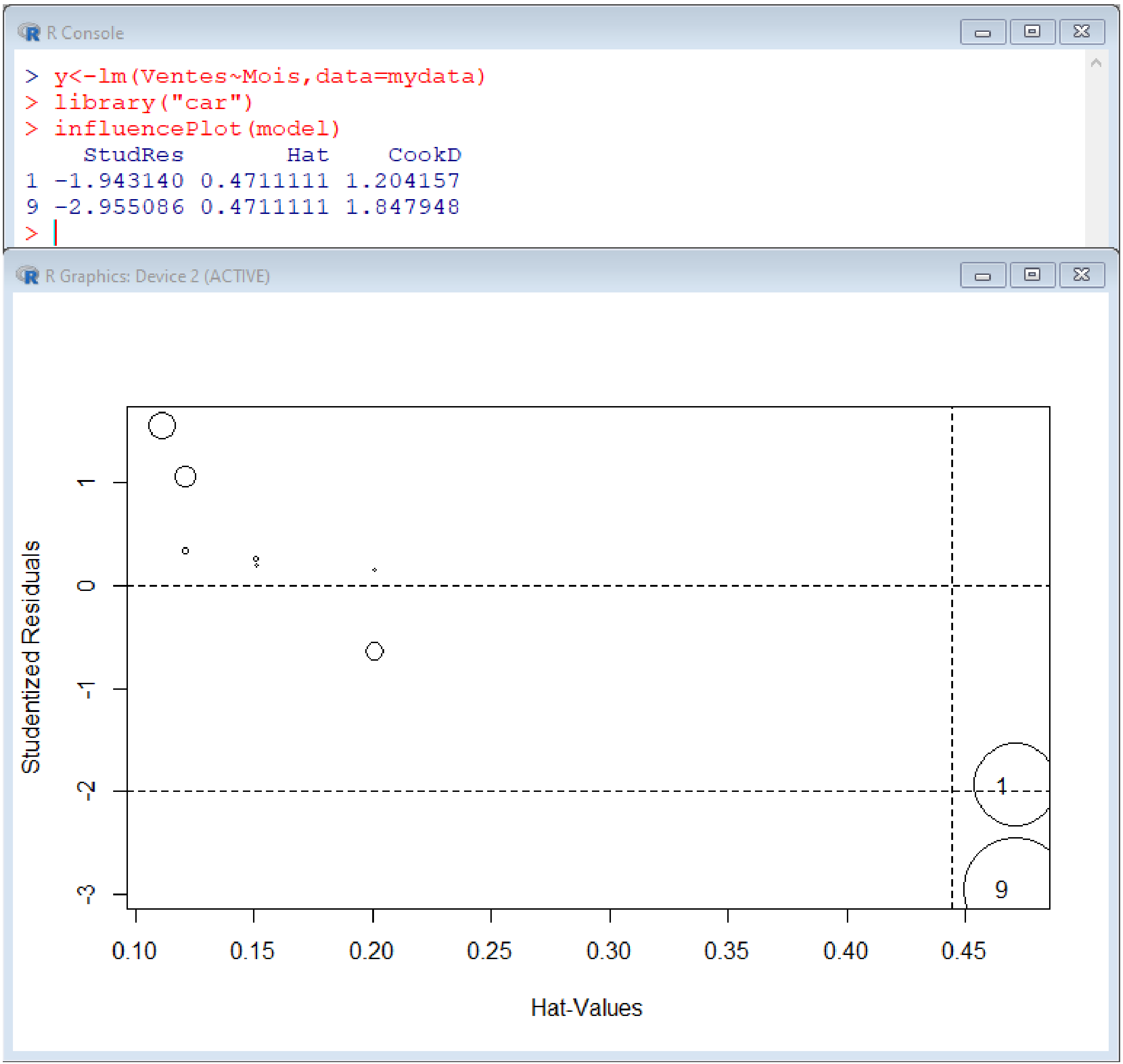
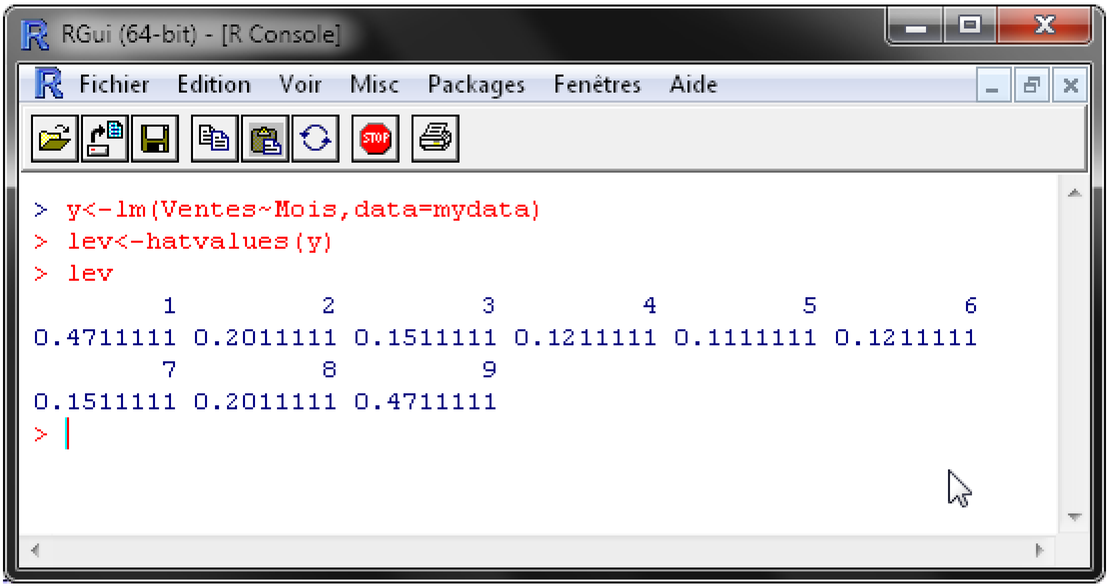


Figure 242 Graph de diagnostique de effets de levier des résidus

Nous pouvons obtenir une version complémentaire de ce dernier graph avec la fonction du **influencePlot( )** package car:



Nous pouvons aussi obtenir directement l'effet de levier avec la commande **hatvalues( )** :

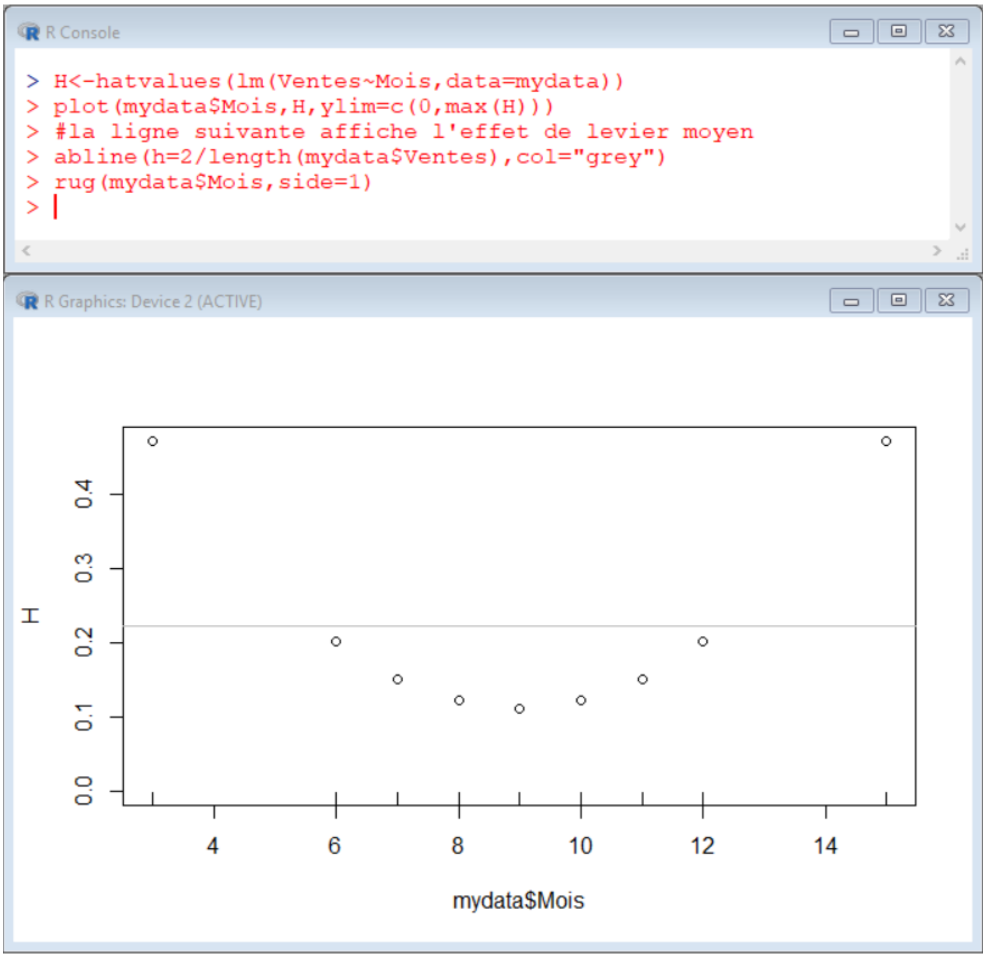


On peut constater que la somme des effets de levier vaut bien:

$$p + 1=2$$

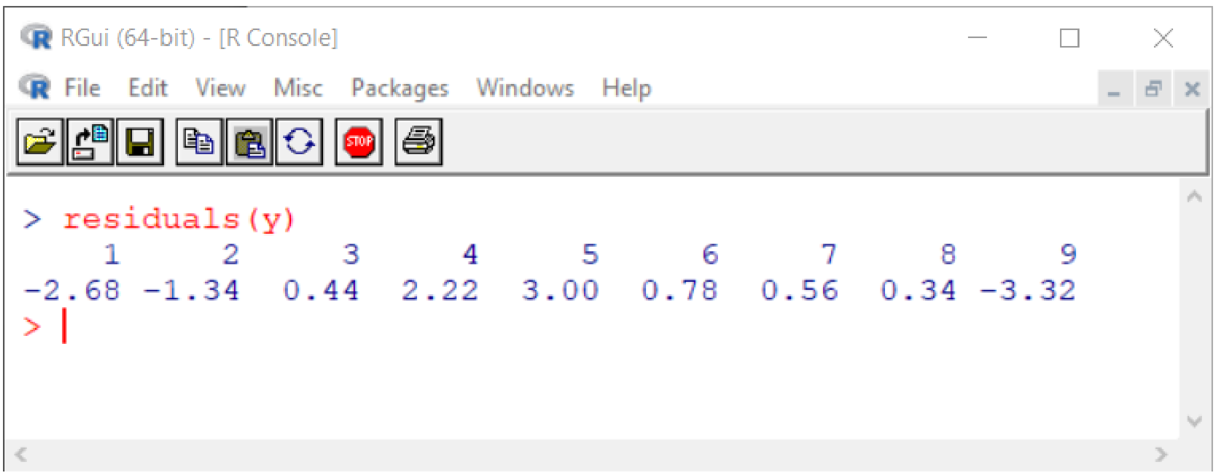
où p est pour rappel le nombre de variables explicatives du modèle.

On peut aussi afficher les effets de levier de manière visuelle pour chacun des points:



Résidus, résidus standardisés et résidus studentisés

Les résidus sont obtenus simplement avec la fonction **residuals( )** :



Les résidus standardisés (normalisés) peuvent être obtenus avec la fonction **rstandard( )** :

```
> rstandard(y)
         1          2          3          4          5 
-1.6442852 -0.6689385  0.2130845  1.0565999  1.4197834 
         6          7          8          9 
 0.3712378  0.2711984  0.1697307 -2.0369504 
> #ou respectivement
> residuals(y)/(summary(y)$sigma*sqrt(1-hatvalues(y)))
         1          2          3          4          5 
-1.6442852 -0.6689385  0.2130845  1.0565999  1.4197834 
         6          7          8          9 
 0.3712378  0.2711984  0.1697307 -2.0369504 
> 
```

Donc pour l'instant tout est conforme à ce que nous avons vu dans le cours théorique!

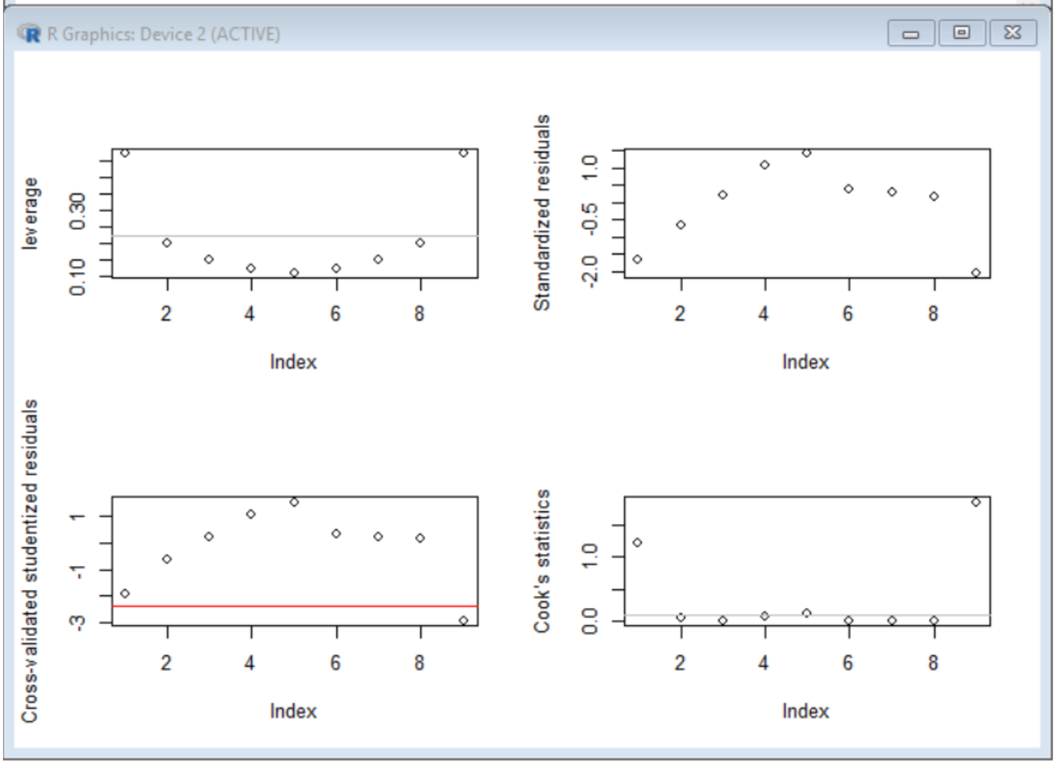
Maintenant voyons comment obtenir les résidus studentisés:


```
> rstudent(y)
         1          2          3          4          5 
-1.9431403 -0.6401144  0.1979208  1.0670008  1.5577557 
         6          7          8          9 
 0.3471336  0.2524105  0.1574644 -2.9550863 
> 
```

Ces derniers étant pénibles à obtenir "à la main" on s'abstiendra de montrer une approche alternative!

On peut résumer un peu tout cela sous forme visuelle:

```
> par(mfrow=c(2,2))
> y<-lm(Ventes~Mois,data=mydata)
> plot(hatvalues(y),ylab="leverage")
> abline(h=2/length(mydata$Ventes),col="grey")
> plot(rstandard(y),ylab="Standardized residuals")
> plot(rstudent(y),ylab="Cross-validated studentized residuals")
> abline(h=qt(0.025,df=length(mydata$Ventes)-2),col="red")
> abline(h=qt(1-0.025,df=length(mydata$Ventes)-2),col="red")
> plot(cooks.distance(y),ylab="Cook's statistics")
> abline(h=qchisq(0.1,2)/2,col="grey")
>
```



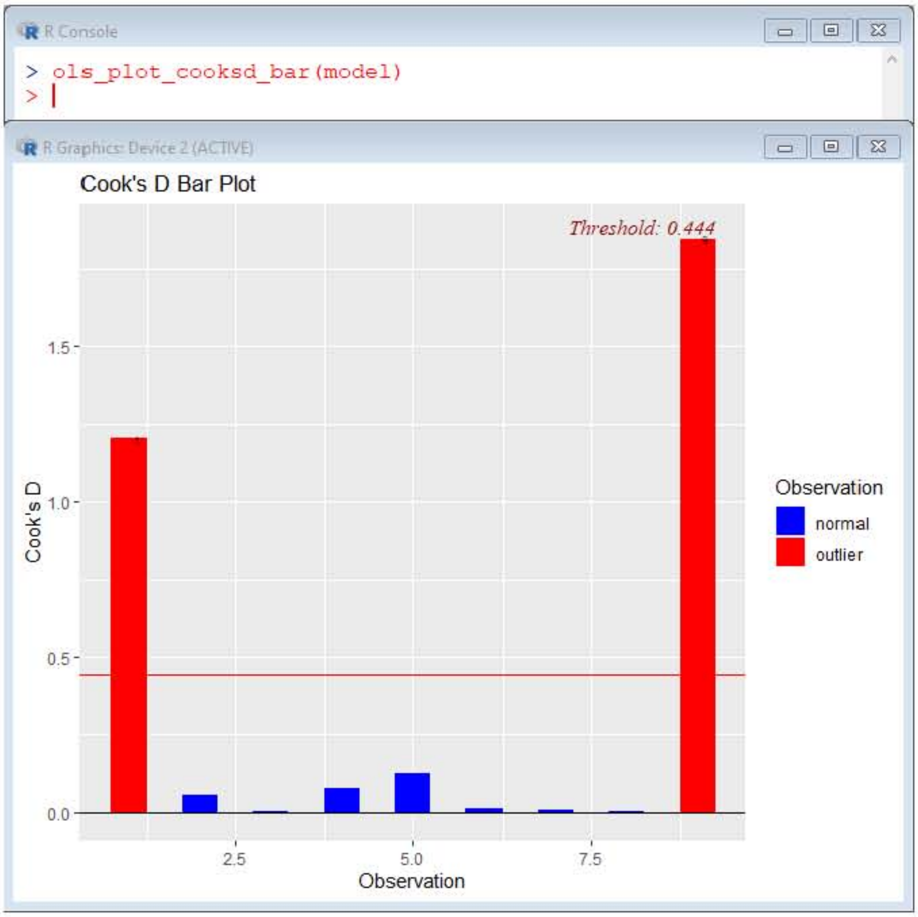
Analyse des influenceurs

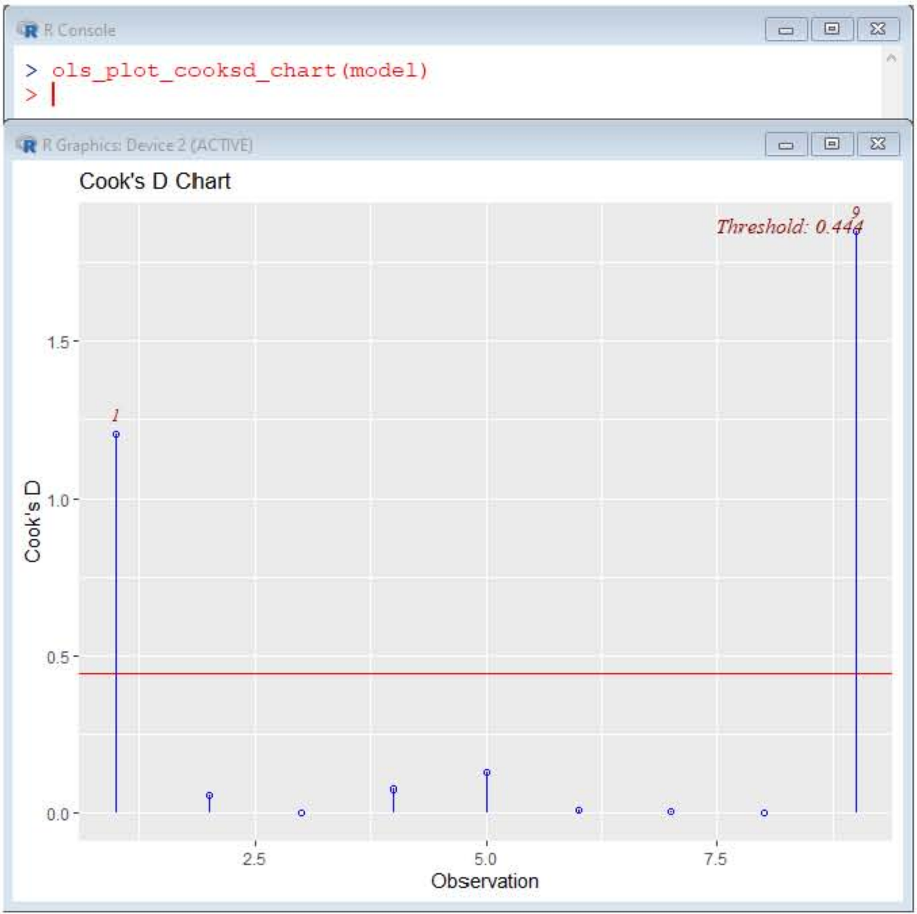
Il existe un formidable package qui reprend les éléments vus précédemment et qui en améliorent grandement l'aspect visuel. Il s'agit du package OLSRR.

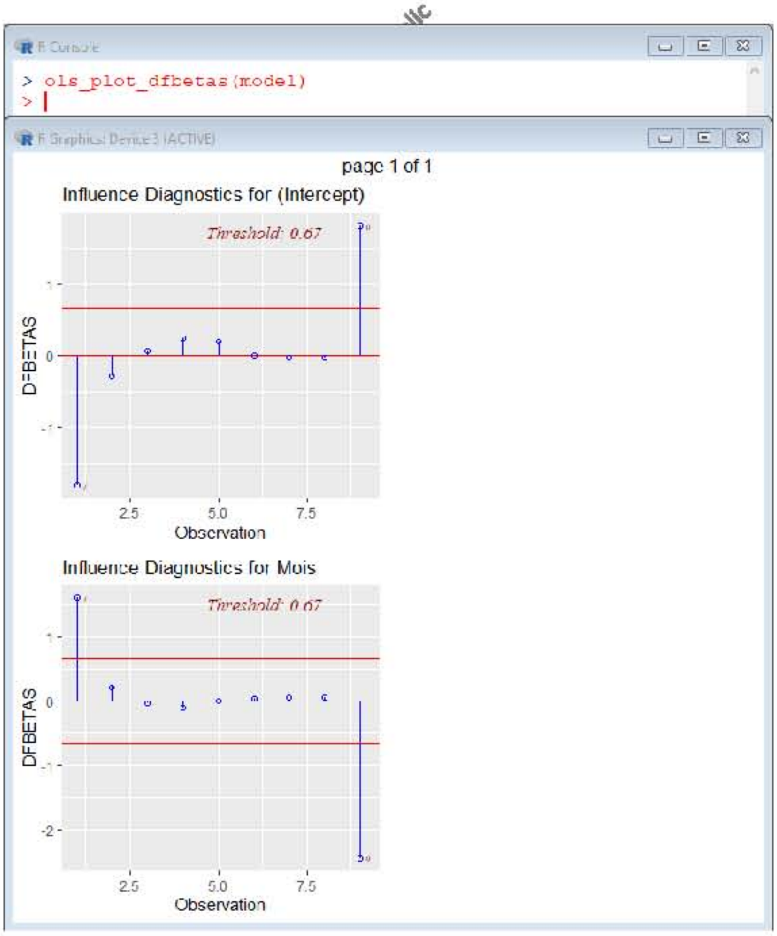
Voyons de quels visuels il s'agit en chargeant d'aborde la package avec notre modèle habituel:

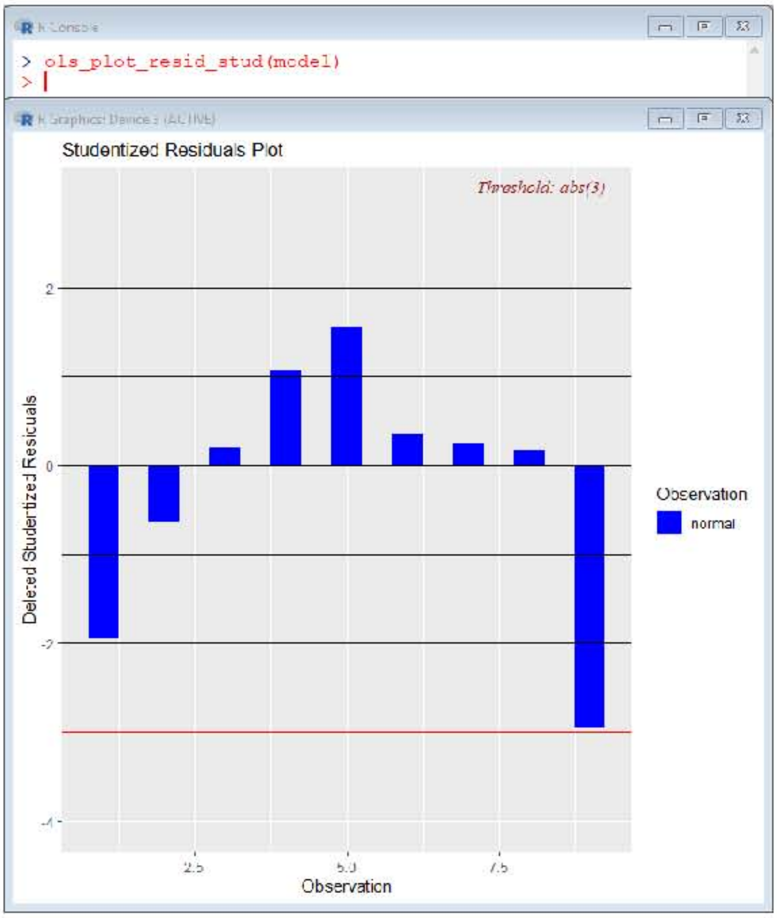
```
> mydata<-read.csv("c:/tmp/RegressionLineaireUnivariee.csv",header=T,sep=";")
> model<-lm(Ventes~Mois,data=mydata)
> library("olsrr")
>
```

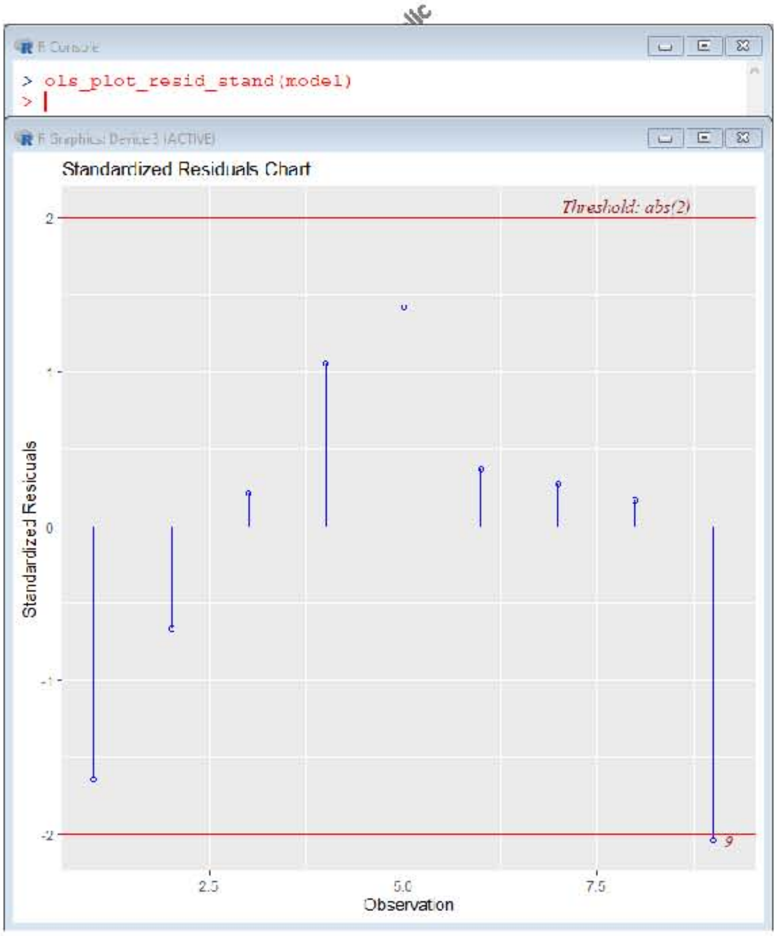
Maintenant voyons les différents graphiques proposés par ce package (il est inutile de les commenter car le titre du graphique suffit à lui seul pour ceux qui ont suivi le cours théorique):

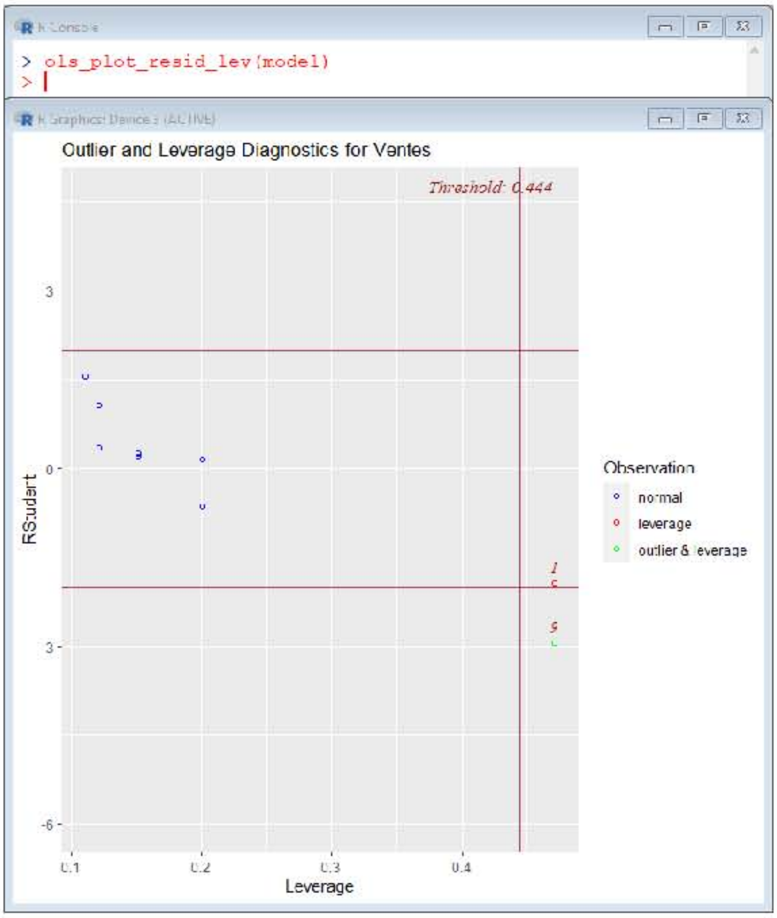


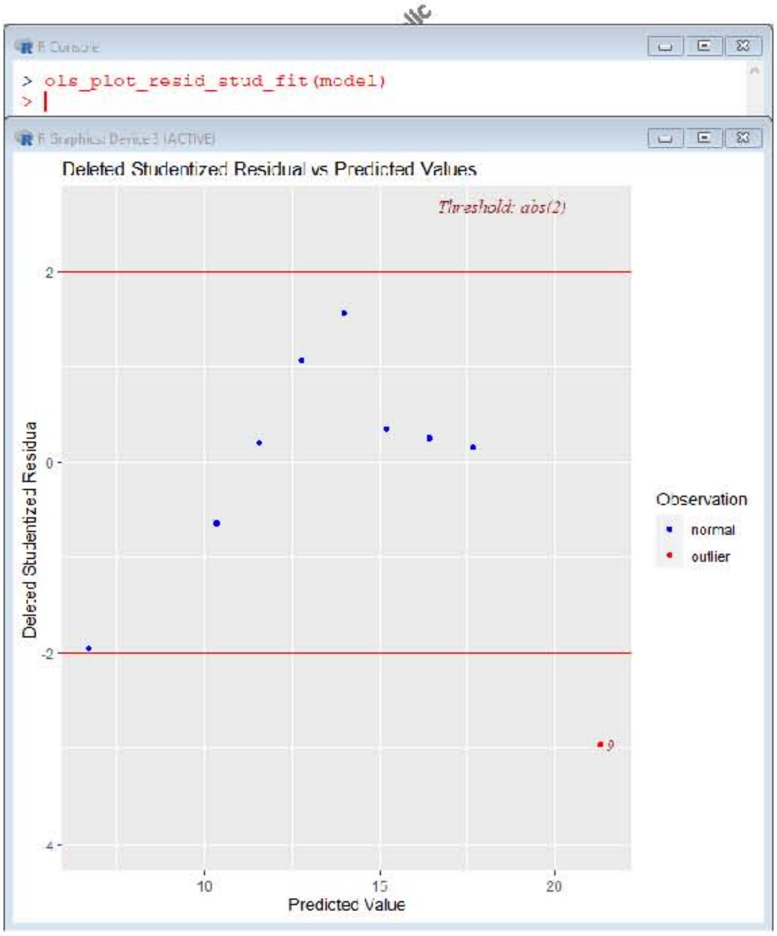

> ols_plot_cooksd_chart(model)
Cook's D Chart
Threshold: 0.444
Cook's D
Observation


> ols_plot_dfbetas(model)
page 1 of 1
Influence Diagnostics for (Intercept)
Threshold: 0.67
DFBETAS
Observation
Influence Diagnostics for Mois
Threshold: 0.67
DFBETAS
Observation


R Console
> ols_plot_resid_stud(model)
>
R Graphics: Device 2 (ACTIVE)
Studentized Residuals Plot
Threshold: abs(3)
Deleted Studentized Residuals
Observation
Observation
normal
2
0
-2
-4
2.5
5.0
7.5


R Console
> ols_plot_resid_stand(model)
>
R Graphics: Device 3 (ACTIVE)
Standardized Residuals Chart
Threshold: abs(2)
Standardized Residuals
Observation
2
1
0
-1
-2
9
2.5
5.0
7.5


> ols_plot_resid_lev(model)
Outlier and Leverage Diagnostics for Ventes
Threshold: 0.444
RStudent
Leverage
Observation
normal
leverage
outlier & leverage

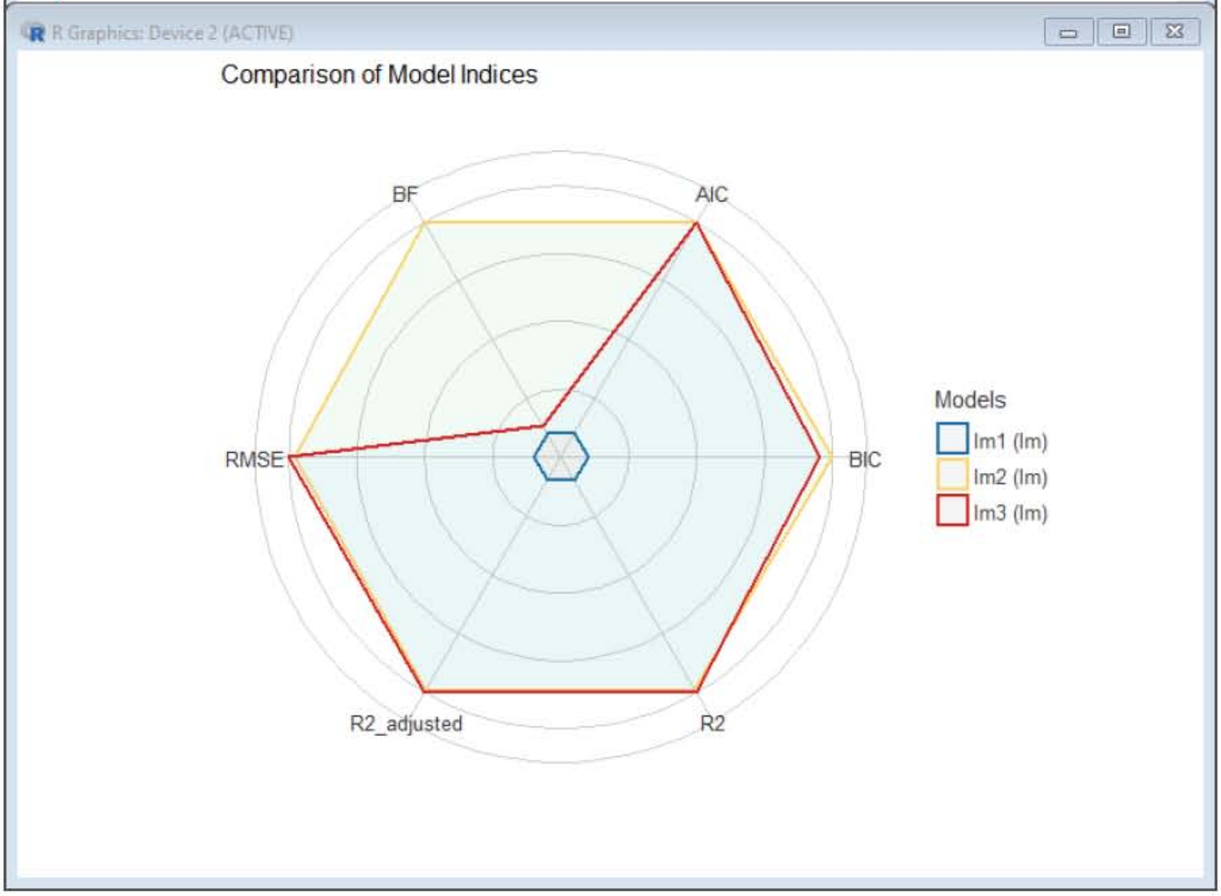

> ols_plot_resid_stud_fit(model)
Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

Analyse de la performance

Il existe un package nommé **performance** et une fonction **compare_performance( )** associée à un package nommé **see** qui permet de faire une analyse pertinente et pédagogique de différents modèles et pas que des modèles linéaires. Voyons cela avec un exemple:

```
> library("performance")
> library("see")
> data(iris)
> lm1<-lm(Sepal.Length~Species,data=iris)
> lm2<-lm(Sepal.Length~Species+Petal.Length,data=iris)
> lm3<-lm(Sepal.Length~Species*Petal.Length,data=iris)
> (result<-compare_performance(lm1,lm2,lm3))
# Comparison of Model Performance Indices

Model | Type |    AIC |    BIC |   R2 | R2_adjusted | RMSE |       BF
--------------------------------------------------------------------
lm1   |   lm | 231.45 | 243.49 | 0.62 |        0.61 | 0.51 |
lm2   |   lm | 106.23 | 121.29 | 0.84 |        0.83 | 0.33 | 3.45e+26
lm3   |   lm | 106.77 | 127.84 | 0.84 |        0.83 | 0.33 | 1.30e+25
> plot(result)
>
```

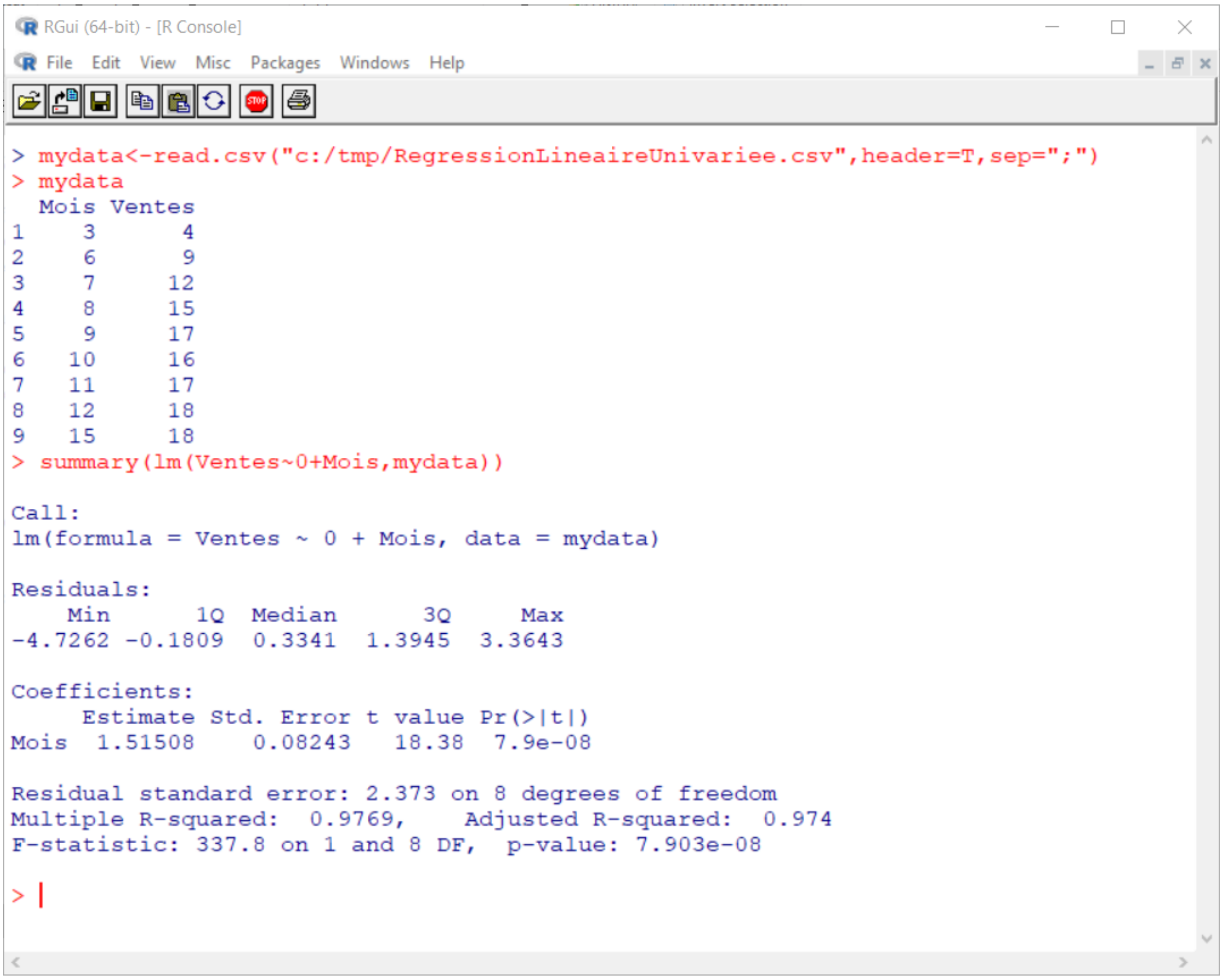


Exemple 248.: Régression linéaire univariée par moindres carrés ordinaires forcée à l'origine

R 3.4.3

Le fait de forcer la régression à l'origine est très important dans la pratique mais comme nous l'avons étudié dans le cours théorique, c'est également un sujet à controverses. Nous allons vérifier ici si R calcule le coefficient de corrélation conformément à la définition alternative de ce dernier dans le cas où la régression est forcée à l'origine.

Toute l'analyse se résume alors simplement à la série de commandes suivantes:



À comparer avec la sortie donnée par Minitab 18.1:

Analyse de régression : Ventes en fonction de Mois

Analyse de variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
Régression	1	1902.94	1902.94	337.84	0.000
Mois	1	1902.94	1902.94	337.84	0.000
Erreur	8	45.06	5.63		
Total	9	1948.00			

Récapitulatif du modèle

S	R carré	R carré (ajust)	R carré (prév)
2.37333	97.69%	97.40%	96.44%

Coefficients

Terme	Coeff	Coef ErT	Valeur de T	Valeur de p	FIV
Mois	1.5151	0.0824	18.38	0.000	1.00

Equation de régression

Ventes = 1.5151 Mois

À comparer avec la sortie donnée par Microsoft Excel 14.0:

	A	B	C	D	E	F	G	H	I
1	RAPPORT DÉTAILLÉ								
2									
3	*Statistiques de la régression*								
4	Coefficient de détermination multiple	0.98836623							
5	Coefficient de détermination R^2	0.9768678							
6	Coefficient de détermination R^2	0.8518678							
7	Erreur-type	2.37332888							
8	Observations	9							
9									
10	ANALYSE DE VARIANCE								
11		*Degré de libert*	*mme des carr*	*yenne des car*	*F*	*eur critique de F*			
12	Régression	1	1902.93848	1902.93848	337.838312	3.4961E-07			
13	Résidus	8	45.0615199	5.63268999					
14	Total	9	1948						
15									
16		*Coefficients*	*Erreur-type*	*Statistique t*	*Probabilité*	*pour seuil de*	*pour seuil de*	*our seuil de c*	*our seuil de confiance = 95.0%*
17	Constante	0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
18	Mois	1.51507841	0.08242912	18.3803785	7.9029E-08	1.32499652	1.7051603	1.32499652	1.7051603
19									

Et dans SPSS 25:

Model Summary

Model	R	R Square[b]	Adjusted R Square	Std. Error of the Estimate
1	.988[a]	.977	.974	2.37333

a. Predictors: Mois

b. For regression through the origin (the no-intercept model), R Square measures the proportion of the variability in the dependent variable about the origin explained by regression. This CANNOT be compared to R Square for models which include an intercept.

ANOVA[a,b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	1902.938	1	1902.938	337.838	.000[c]
	Residual	45.062	8	5.633		
	Total	1948.000[d]	9			

a. Dependent Variable: Ventes

b. Linear Regression through the Origin

c. Predictors: Mois

d. This total sum of squares is not corrected for the constant because the constant is zero for regression through the origin.

Coefficients[a,b]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	Mois	1.515	.082	.988	18.380	.000

a. Dependent Variable: Ventes

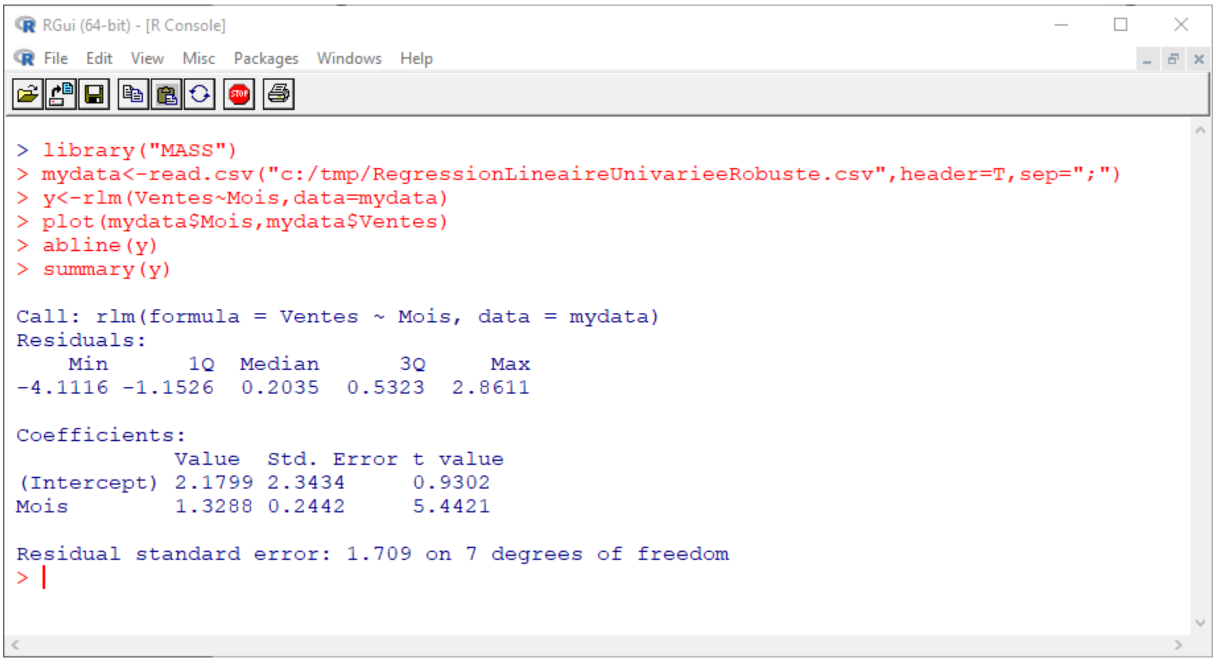
b. Linear Regression through the Origin

Donc nous voyons que tous les logiciels utilisent la définition alternative du coefficient de corrélation alternatif puisque nous retombons sur la valeur calculée à la main dans le cours théorique de 0.9769!

Exemple 249.: Régression linéaire univariée par moindres carrés ordinaires avec M-estimateurs

R 3.1.2

Nous avons étudié dans le cours théorique quelques M-estimateurs avec un regard très critique sur le choix de ces derniers. Voyons comme il est possible avec R de faire une régression robuste basée par défaut sur le M-estimateur de Humber à l'aide de la fonction **rlm( )** du package MASS en reprenant les mêmes données que l'exemple précédent:



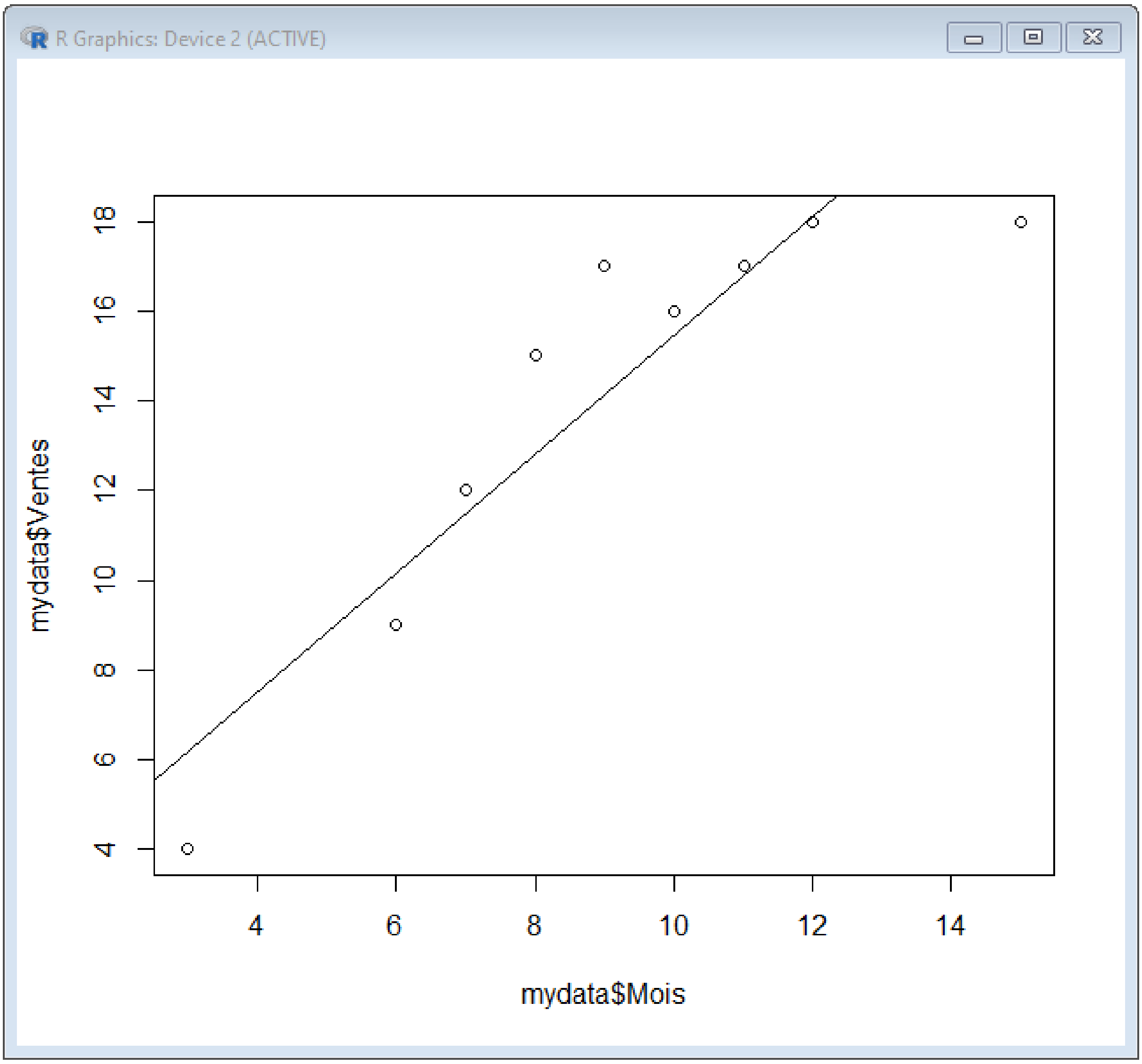
```
> library("MASS")
> mydata<-read.csv("c:/tmp/RegressionLineaireUnivarieeRobuste.csv",header=T,sep=";")
> y<-rlm(Ventes~Mois,data=mydata)
> plot(mydata$Mois,mydata$Ventes)
> abline(y)
> summary(y)

Call: rlm(formula = Ventes ~ Mois, data = mydata)
Residuals:
    Min      1Q  Median      3Q     Max
-4.1116 -1.1526  0.2035  0.5323  2.8611

Coefficients:
            Value  Std. Error t value
(Intercept) 2.1799 2.3434     0.9302
Mois        1.3288 0.2442     5.4421

Residual standard error: 1.709 on 7 degrees of freedom
>
```

Avec le graphique correspondant:



Comparons maintenant le modèle avec M-estimateur de Huber et de Tukey (pour information il y aussi la méthode des estimateurs MM!!!):

```
> model_huber<-rlm(Ventes~Mois,method="M", psi=psi.huber,data=mydata)
> summary(model_huber)

Call: rlm(formula = Ventes ~ Mois, data = mydata, psi = psi.huber,
    method = "M")
Residuals:
    Min      1Q  Median      3Q     Max
-4.1116 -1.1526  0.2035  0.5323  2.8611

Coefficients:
            Value  Std. Error t value
(Intercept) 2.1799 2.3434     0.9302
Mois        1.3288 0.2442     5.4421

Residual standard error: 1.709 on 7 degrees of freedom
> model_tukey<-rlm(Ventes~Mois,method="M", psi=psi.bisquare,data=mydata)
> summary(model_tukey)

Call: rlm(formula = Ventes ~ Mois, data = mydata, psi = psi.bisquare,
    method = "M")
Residuals:
    Min      1Q  Median      3Q     Max
-3.7032 -1.2706  0.3780  0.6483  2.9185

Coefficients:
            Value  Std. Error t value
(Intercept) 2.6488 2.4469     1.0825
Mois        1.2703 0.2550     4.9824

Residual standard error: 1.884 on 7 degrees of freedom
>
```

à comparer avec le modèle non robuste où nous avions obtenu pour rappel…:

```
> model_ols<-lm(Ventes~Mois,data=mydata)
> summary(model_ols)

Call:
lm(formula = Ventes ~ Mois, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max
 -3.32  -1.34   0.44   0.78   3.00

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0200     2.1510   1.404 0.203089
Mois          1.2200     0.2241   5.444 0.000963

Residual standard error: 2.241 on 7 degrees of freedom
Multiple R-squared:  0.8089,    Adjusted R-squared:  0.7816
F-statistic: 29.63 on 1 and 7 DF,  p-value: 0.0009626

>
```

Exemple 250.: Régression par densité de noyaux

R 3.0.2

Nous avons vu dans le cours théorique les détails mathématiques des deux cas classiques de régression par densité de noyaux et régression polynomial par régression de noyaux.

D'abord commençons par poser le contexte en faisant une simple régression linéaire par moindres carrés ordinaires:

```
> mydata<-read.csv("c:/tmp/RegressionLineaireUnivariee.csv",header=T,sep=";")
> mydata
  Mois Ventes
1    3      4
2    6      9
3    7     12
4    8     15
5    9     17
6   10     16
7   11     17
8   12     18
9   15     18
> fit.lm<-lm(Ventes~Mois,data=mydata)
> summary(fit.lm)

Call:
lm(formula = Ventes ~ Mois, data = mydata)

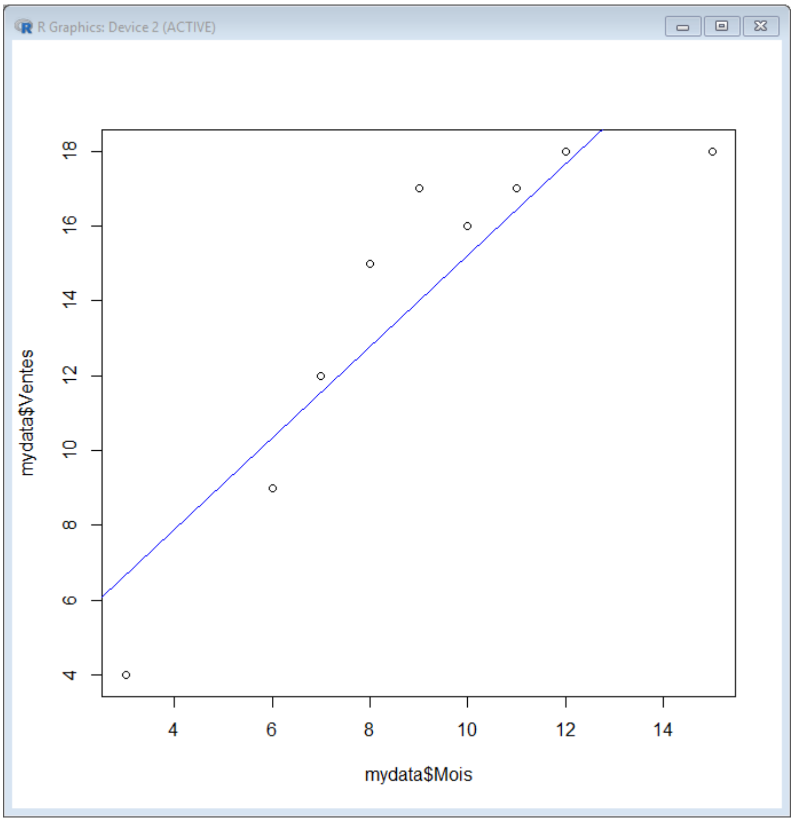
Residuals:
   Min     1Q Median     3Q    Max 
 -3.32  -1.34   0.44   0.78   3.00 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0200     2.1510   1.404 0.203089
Mois          1.2200     0.2241   5.444 0.000963

Residual standard error: 2.241 on 7 degrees of freedom
Multiple R-squared:  0.8089,	Adjusted R-squared:  0.7816 
F-statistic: 29.63 on 1 and 7 DF,  p-value: 0.0009626

> plot(mydata$Mois,mydata$Ventes)
> abline(fit.lm,col="blue")
> 
```

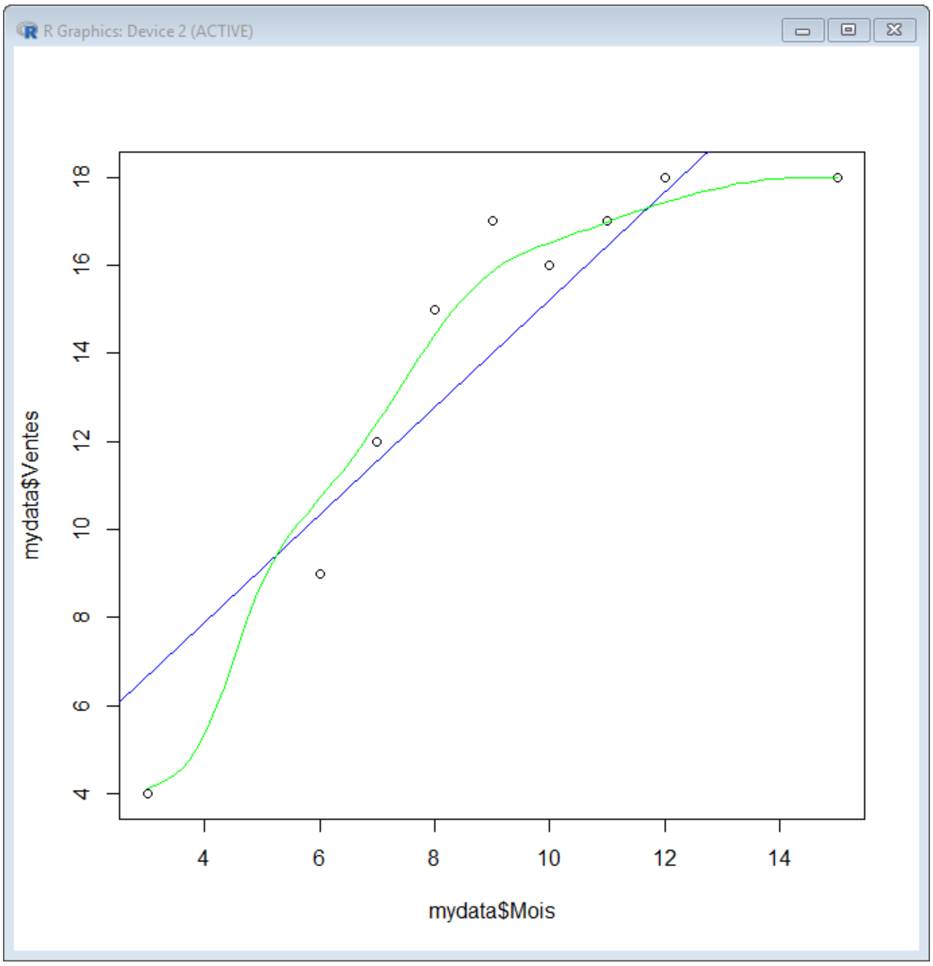
Ce qui donne visuellement:



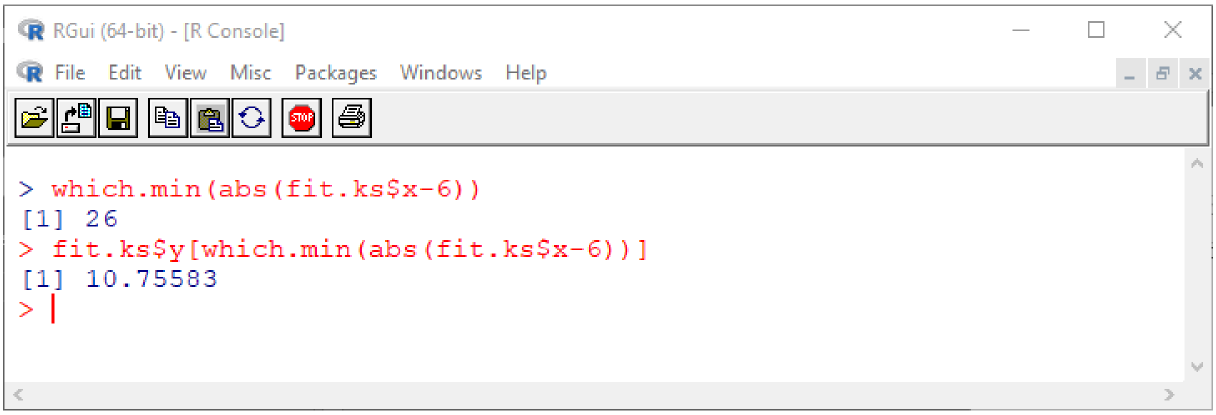
Voyons maintenant la régression par densité de noyaux en choisissant un noyau gaussien via la fonction native **ksmooth( )**:

```
> fit.ks<-ksmooth(mydata$Mois, mydata$Ventes, bandwidth = 3, kernel = "normal",n.points = 100)
>
> plot(mydata$Mois,mydata$Ventes)
> abline(fit.lm,col="blue")
> lines(fit.ks,col="green")
>
> fit.ks$x
  [1]  3.000000  3.121212  3.242424  3.363636  3.484848  3.606061  3.727273  3.848485
  [9]  3.969697  4.090909  4.212121  4.333333  4.454545  4.575758  4.696970  4.818182
 [17]  4.939394  5.060606  5.181818  5.303030  5.424242  5.545455  5.666667  5.787879
 [25]  5.909091  6.030303  6.151515  6.272727  6.393939  6.515152  6.636364  6.757576
 [33]  6.878788  7.000000  7.121212  7.242424  7.363636  7.484848  7.606061  7.727273
 [41]  7.848485  7.969697  8.090909  8.212121  8.333333  8.454545  8.575758  8.696970
 [49]  8.818182  8.939394  9.060606  9.181818  9.303030  9.424242  9.545455  9.666667
 [57]  9.787879  9.909091 10.030303 10.151515 10.272727 10.393939 10.515152 10.636364
 [65] 10.757576 10.878788 11.000000 11.121212 11.242424 11.363636 11.484848 11.606061
 [73] 11.727273 11.848485 11.969697 12.090909 12.212121 12.333333 12.454545 12.575758
 [81] 12.696970 12.818182 12.939394 13.060606 13.181818 13.303030 13.424242 13.545455
 [89] 13.666667 13.787879 13.909091 14.030303 14.151515 14.272727 14.393939 14.515152
 [97] 14.636364 14.757576 14.878788 15.000000
> fit.ks$y
  [1]  4.139831  4.187533  4.250917  4.334612  4.444195  4.590427  4.773757  5.003715
  [9]  5.285432  5.620750  6.006504  6.433465  6.886672  7.351381  7.802459  8.226824
 [17]  8.614130  8.960062  9.265489  9.534861  9.774532  9.991428 10.193975 10.385099
 [25] 10.571051 10.755829 10.942546 11.133531 11.330434 11.534311 11.746693 11.966073
 [33] 12.192831 12.426274 12.665377 12.908812 13.155000 13.403175 13.649650 13.892882
 [41] 14.131259 14.362792 14.585617 14.798068 14.998725 15.186464 15.360492 15.520371
 [49] 15.666024 15.797729 15.916094 16.022025 16.116675 16.201386 16.277625 16.346912
 [57] 16.410756 16.470588 16.527706 16.583233 16.638078 16.692931 16.749004 16.804969
 [65] 16.861677 16.919130 16.977082 17.035208 17.093135 17.150483 17.207545 17.262453
 [73] 17.315881 17.367597 17.417448 17.465355 17.511308 17.555363 17.598182 17.638606
 [81] 17.677544 17.715087 17.751255 17.785953 17.818946 17.849852 17.878188 17.903706
 [89] 17.925464 17.943633 17.958286 17.969718 17.978377 17.984768 17.989387 17.993196
 [97] 17.995299 17.996770 17.997790 17.998493
>
```

Ce qui donne visuellement:



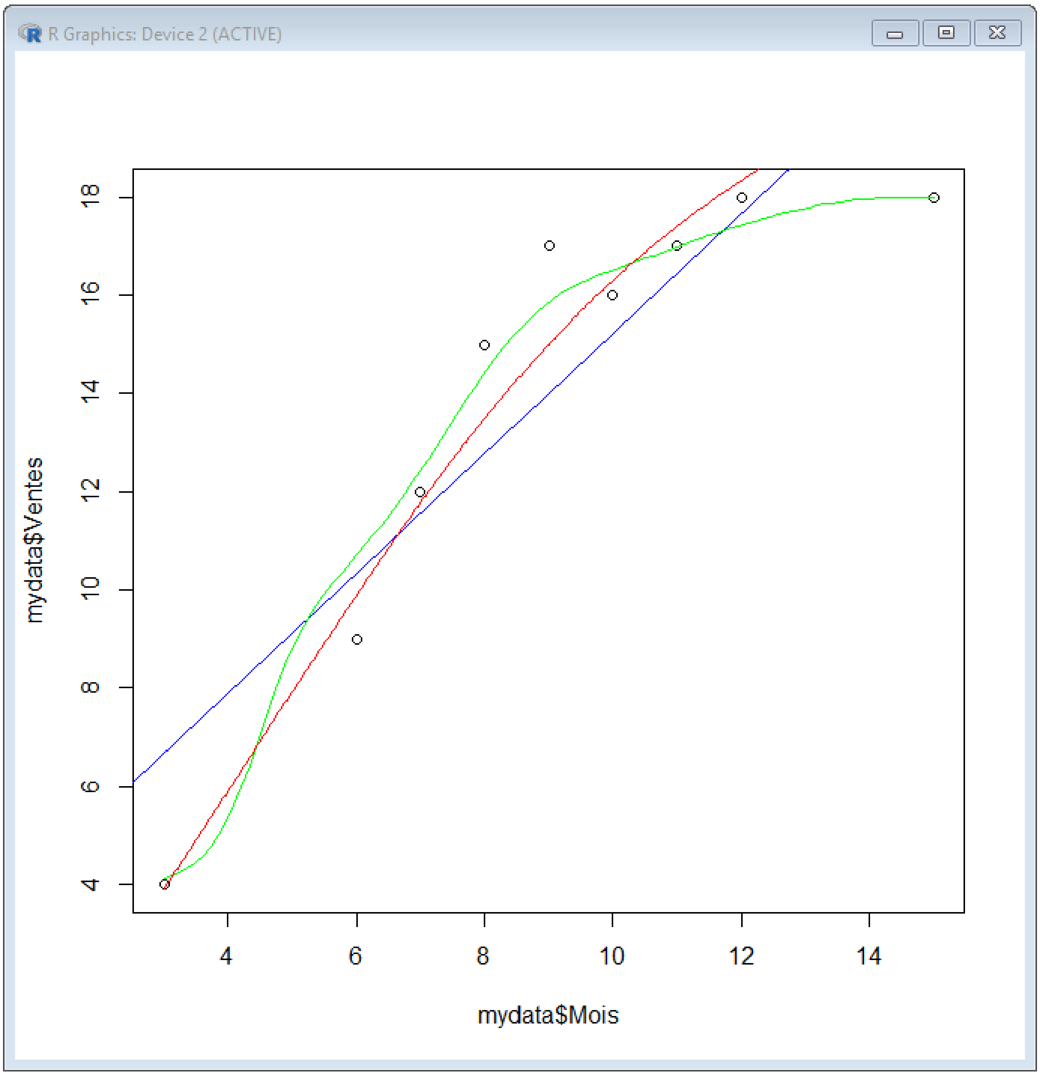
Pour déterminer la valeur de l'ordonnée correspondant à (exemple pris au hasard) de $x = 6$ on utilisera:



On peut procéder de manière similaire avec la fonction **locpoly( )** du package pour la KernSmooth régression polynomial par densité de noyaux:

```
> library(KernSmooth)
> fit.pks<-locpoly(mydata$Mois, mydata$Ventes, bandwidth = 3, kernel = "normal")
> plot(mydata$Mois,mydata$Ventes)
> abline(fit.lm,col="blue")
> lines(fit.ks,col="green")
> lines(fit.pks,col="red")
>
> #une prédiction
> which.min(abs(fit.pks$x-6))
[1] 101
> fit.pks$y[which.min(abs(fit.pks$x-6))]
[1] 9.888969
>
```

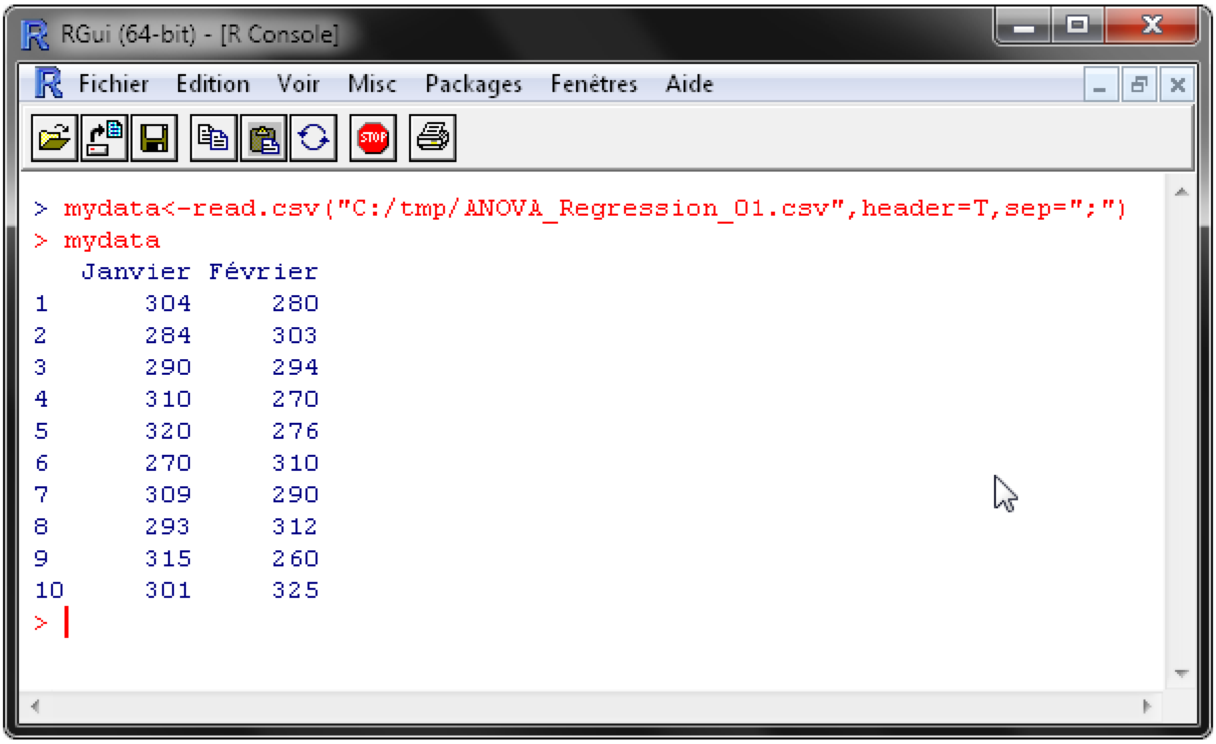
Ce qui donne visuellement:



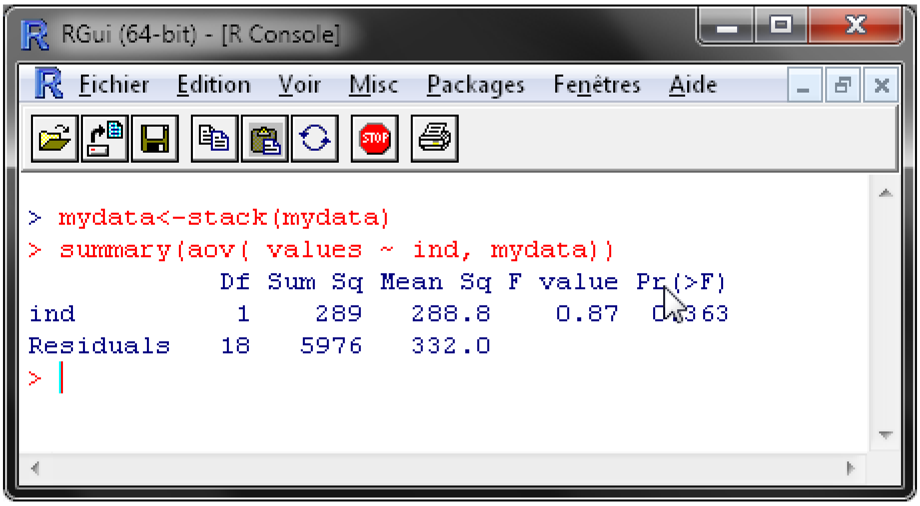
Exemple 251.: ANOVA de la régression linéaire (bivariée) à facteurs fixes

R 3.0.2

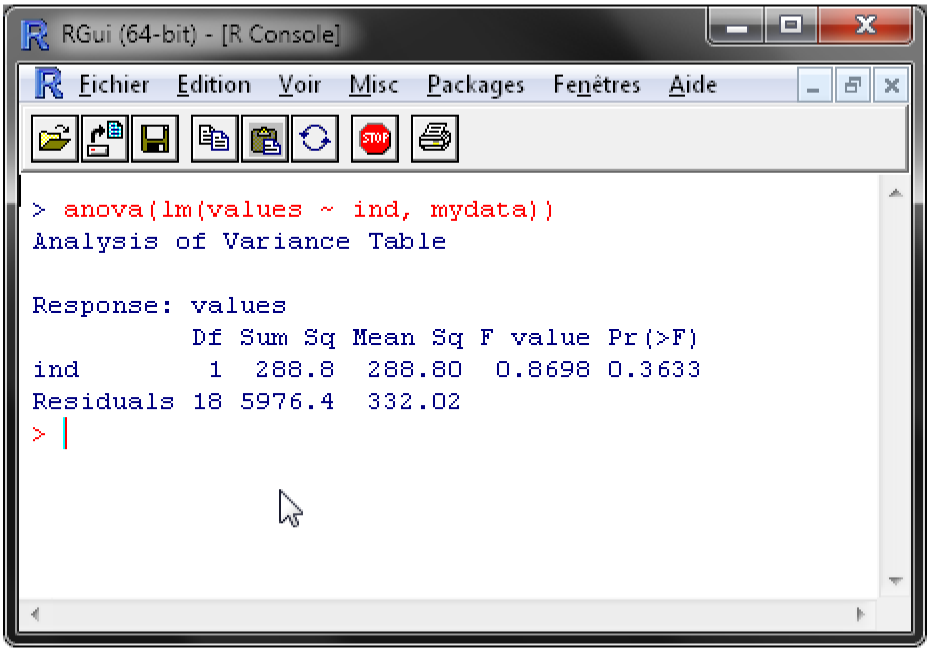
Considérons les données suivantes du cours théorique:



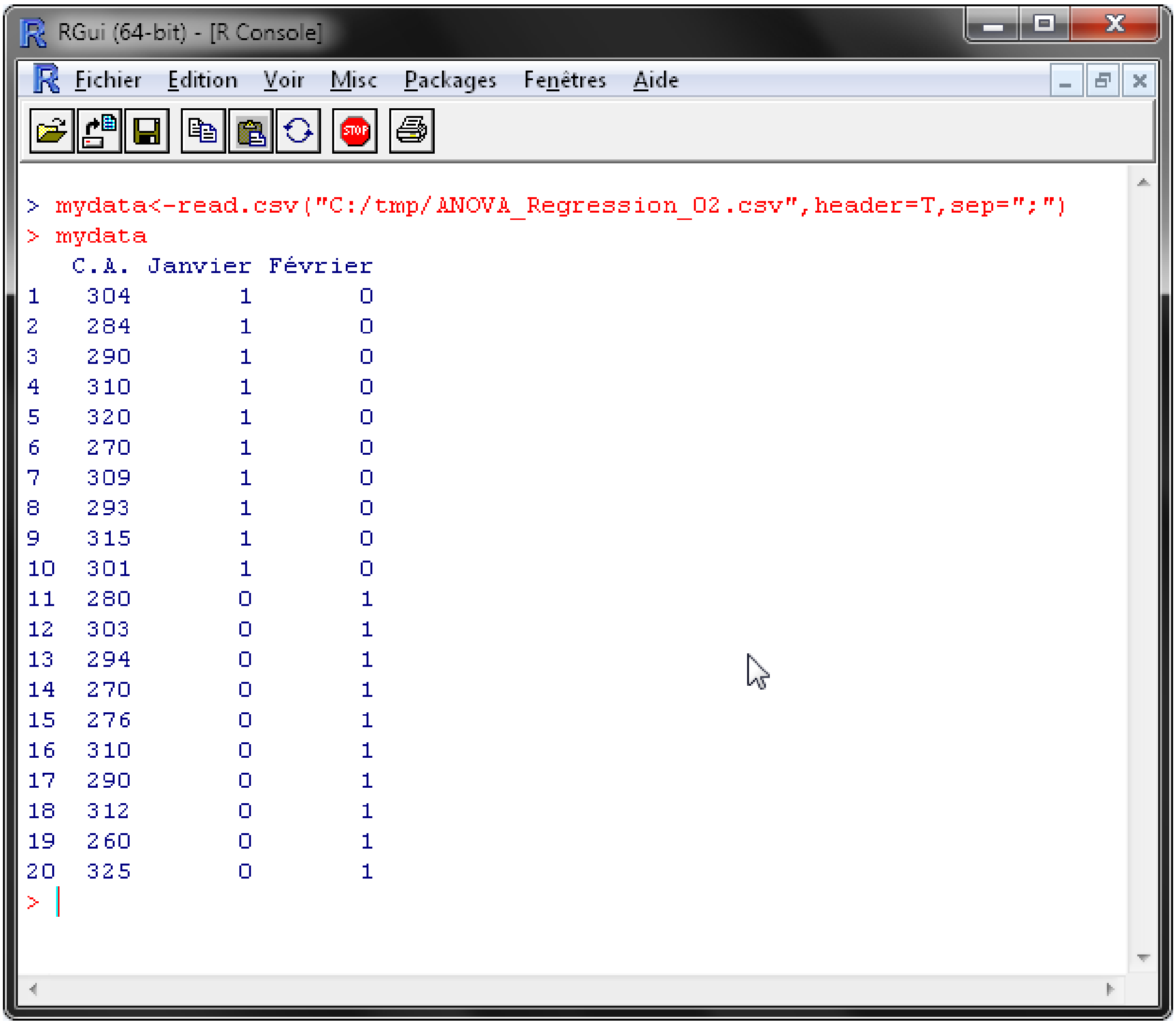
Et procédons à une ANOVA à 1 facteur fixe:



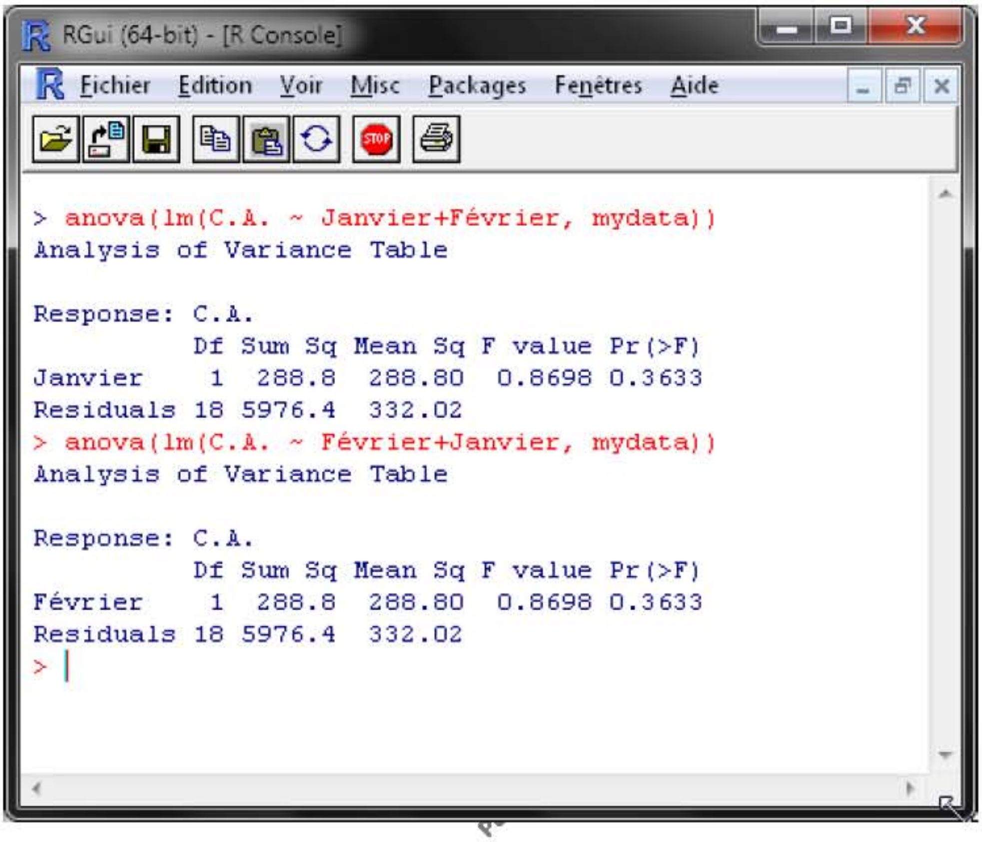
ou en passant donc par la régression linéaire:



Et maintenant comparons avec ce que nous avons vu dans le cours théorique:



Et ensuite nous procédons à une régression de tableau avec une analyse d'ANOVA pour voir que nous retombons bien sur nos pattes et avec les mêmes propriétés que celles discutées dans le cours théorique:



Exemple 252.: Régression quantile linéaire (globale)

R 3.0.2

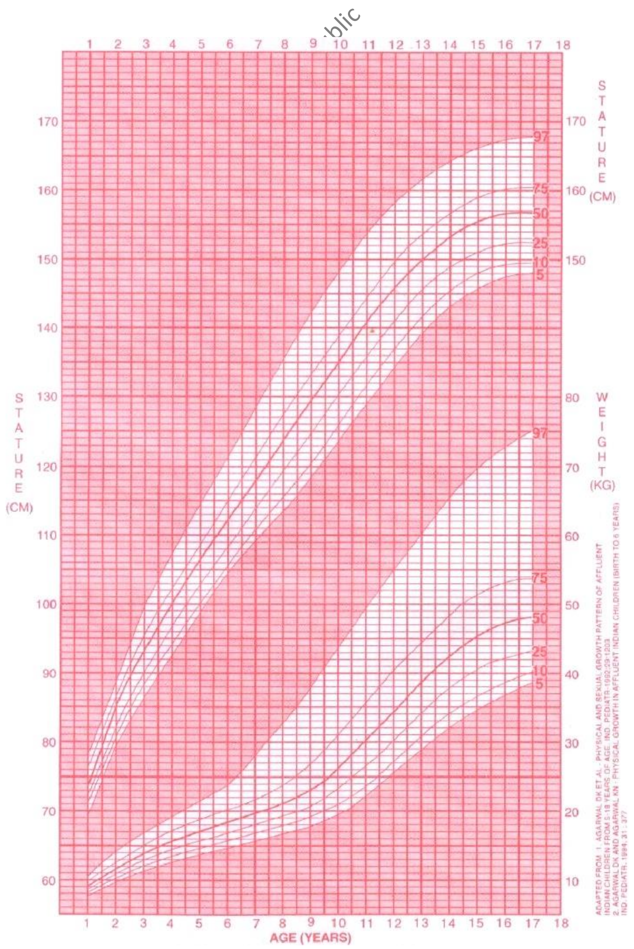
Donc comme vu dans le cours de méthodes numériques, la régression quantile (uni ou multivariée) non censurée qui fait partie des techniques de régressions non paramétriques (avec la régression boostrap) et qui de mon point de vu est plus une technique numérique qu'une technique purement statistique sur le fond...

Nous allons ici comparer le résultat donné par le package **quantreg** par rapport aux trois situations calculées avec Microsoft Excel dans le cours théorique:

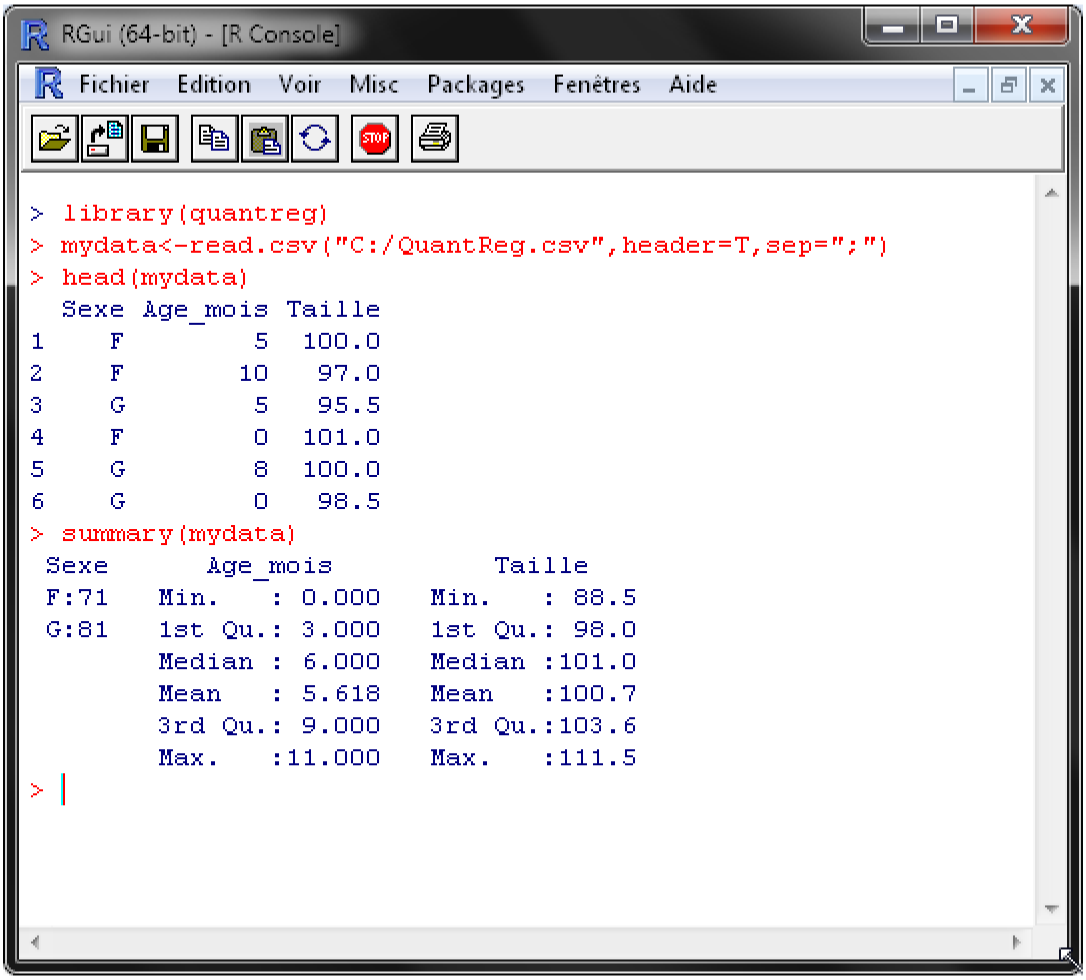
- La régression linéaire paramétrique ordinaire
- La régression linéaire OLS des quantiles
- La régression linéaire de la déviation absolue des quantiles par optimisation (LAD)

Remarque: Malheureusement le package **nlrq** de R qui faisait de la régression quantile non linéaire ne semble plus être maintenu.... Le lecteur intéressé pourra aussi avec le package **quantreg** faire une régression quantile par bootstrapping.

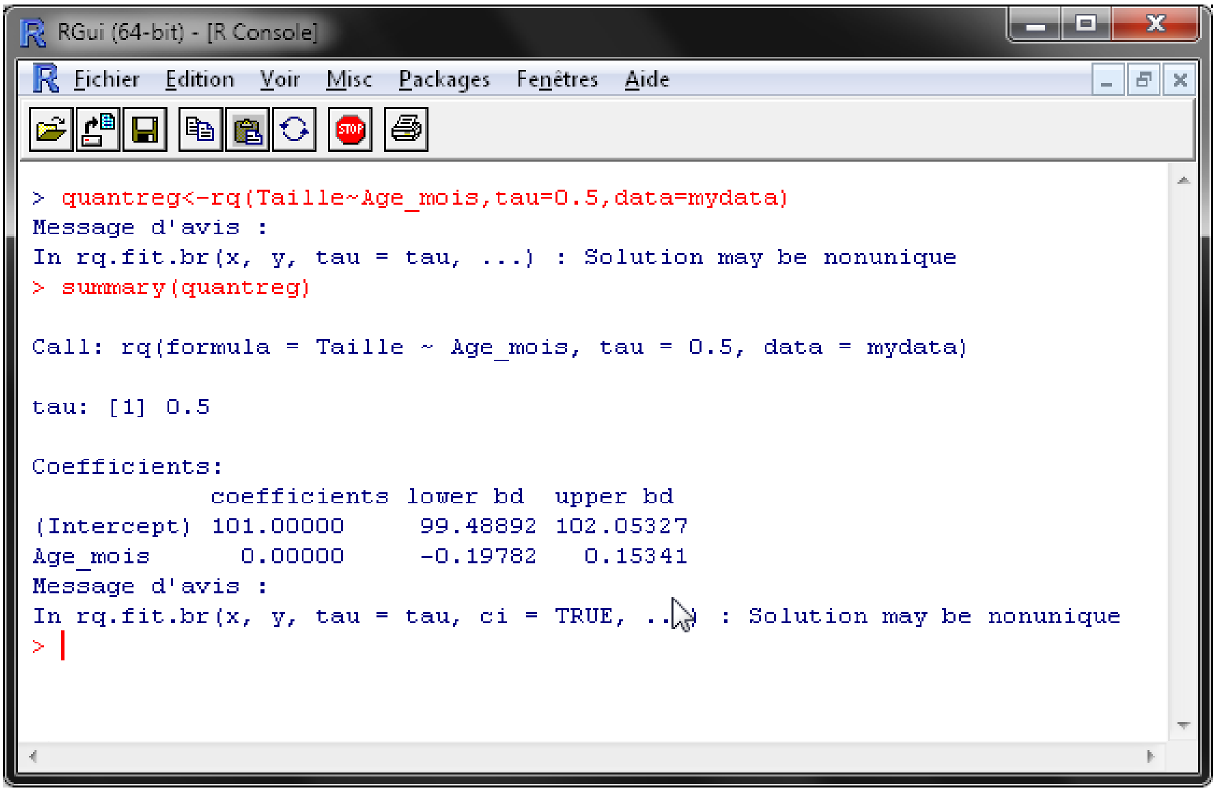
Rappelons que cette technique est très utile en finance et dans le domaine biomédical. Cependant on lui préfère dans les cas non linéaires l'interpolation par spline. Voici par exemple un cas typique de l'interpolation quantile par spline que nous avons presque tous connus en étant petit quand nous allions chez le docteur pour mesurer notre taille en fonction de notre âge:



Bon ce rappel étant fait, nous partons du même jeu de données fictifs que dans le cours théorique:



Maintenant, nous faisons une régression linéaire quantile de la médiane avec la commande **rq( )** du package **quantreg** et regardons ce que nous obtenons par rapport à ce que nous avions obtenu dans Microsoft Excel pour la médiane:



Donc c'est pas mal du tout par rapport à Microsoft Excel! Il en est de même pour d'autres quantiles!

Rappel: Il n'y a pas à ce jour de méthodes établies pour calculer correctement l'erreur standard des résidus ou un R^2 d'une régression quantile!! Mais si on veut vraiment par exemple comparer l'erreur standard entre les deux modèles:

```
> summary(lm(Taille~Age_mois,data=mydata))

Call:
lm(formula = Taille ~ Age_mois, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.3012  -2.7119   0.1652   2.7791  10.7211 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 100.62249    0.64621  155.71   <2e-16 ***
Age_mois      0.02234    0.09730    0.23    0.819    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.248 on 150 degrees of freedom
Multiple R-squared:  0.0003514, Adjusted R-squared:  -0.006313 
F-statistic: 0.05273 on 1 and 150 DF,  p-value: 0.8187

> sqrt(sum((lm(Taille~Age_mois,data=mydata)$residuals)^2)/150)
[1] 4.247892
> sqrt(sum((rq(Taille~Age_mois,tau=0.5,data=mydata)$residuals)^2)/150)
[1] 4.256203
Warning message:
In rq.fit.br(x, y, tau = tau, ...) : Solution may be nonunique
> 
```

Comme quoi… ce n'est pas toujours le non-paramétrique qui est le meilleur…

Maintenant, nous faisons plot de la régression linéaire OLS paramétrique avec des régressions linéaires quantiles pour différents quantiles:

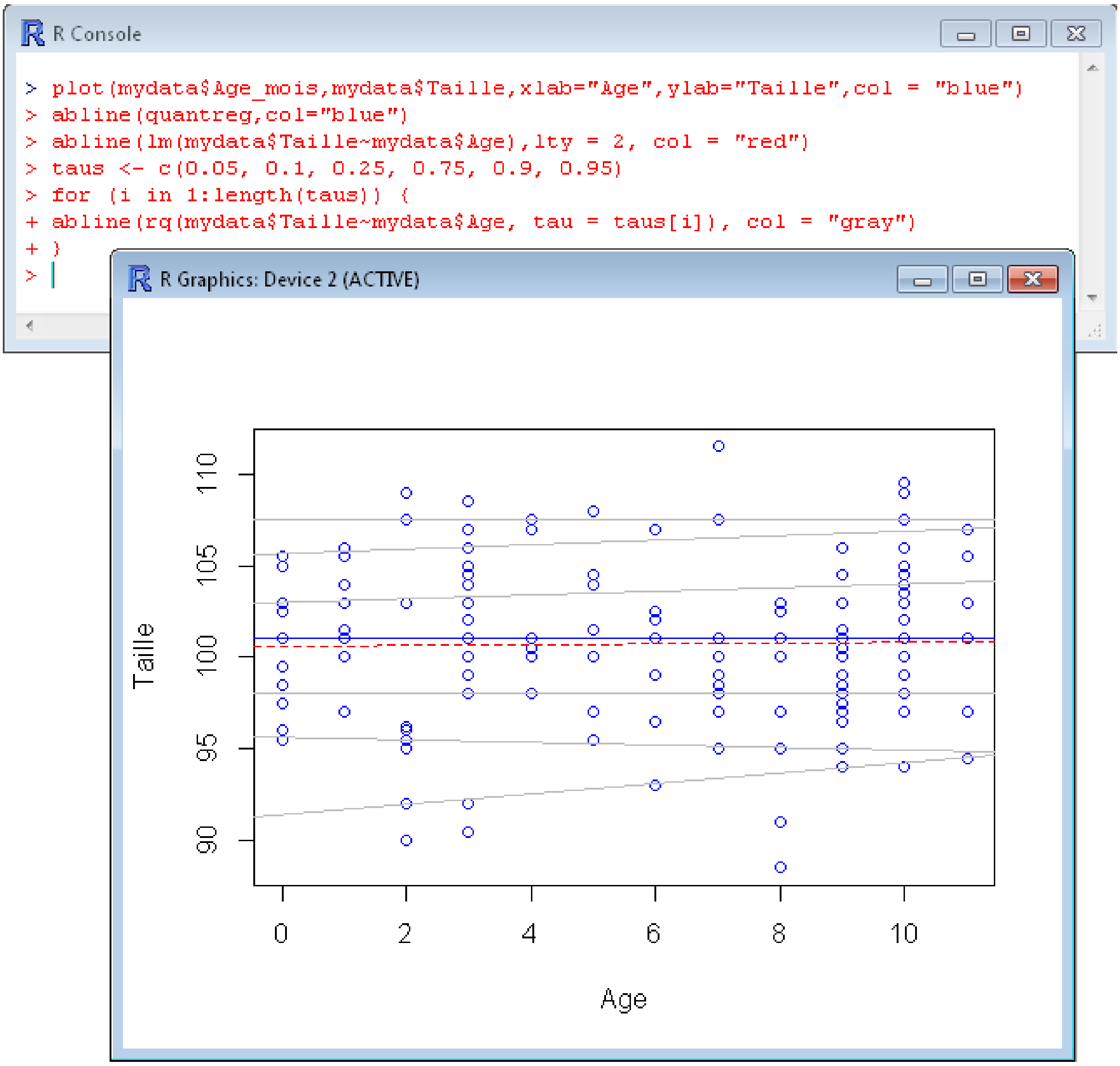
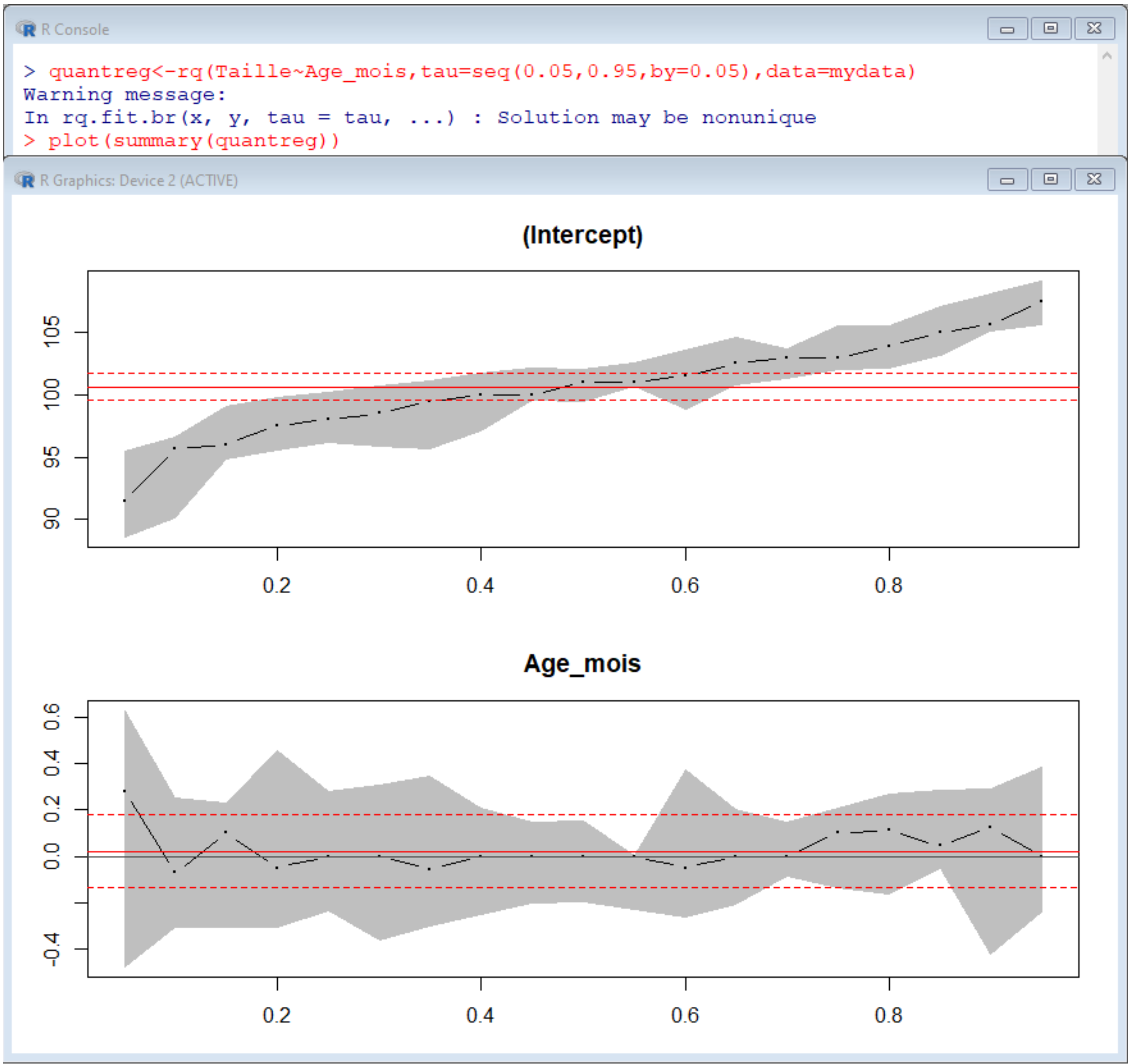


Figure 243 Graph d'une régression quantile

Comme nous pouvons le constater et l'avions fait observer dans le cours de méthodes numériques, des régressions quantiles peuvent donc bien se croiser!

Sinon on peut faire quelque chose de plus brut:



Divers quantiles sont représentés par l'axe X. La ligne centrale rouge indique les estimations des coefficients OLS et les lignes rouges en pointillés sont les intervalles de confiance autour de ces coefficients OLS pour divers quantiles. La ligne pointillée noire représente les estimations de régression quantile et la zone grise est l'intervalle de confiance pour divers quantiles. Nous pouvons voir que pour la seule variable, les deux régressions estimées coïncident pour la plupart des quantiles. Cependant pour l'ordonnée à l'origine ce n'est pas le cas!

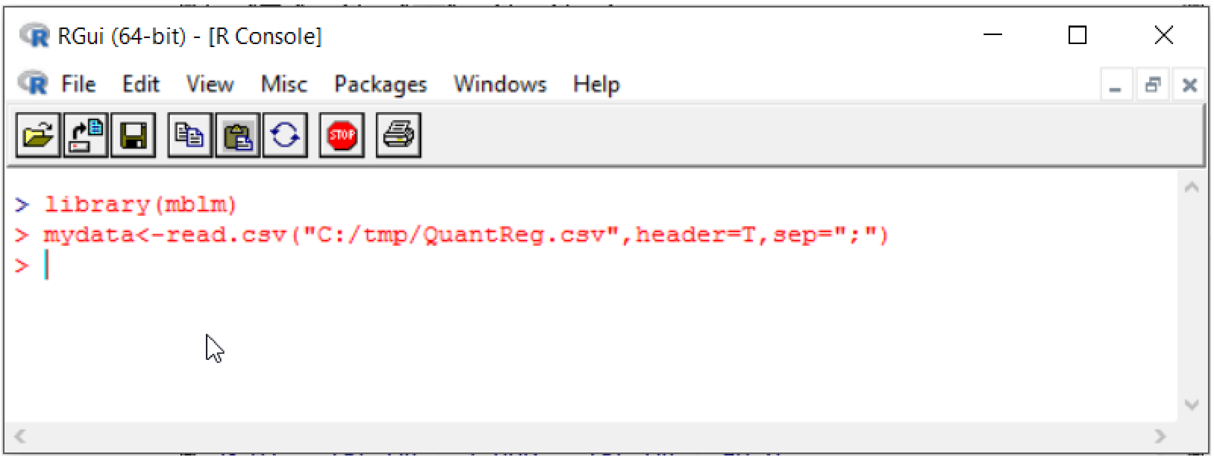
Par conséquent, notre utilisation de la régression quantile n'est pas justifiable pour de tels quantiles. En d'autres termes, nous voulons que les lignes rouge et grise se chevauchent le moins possible pour justifier notre utilisation de la régression quantile.

Exemple 253.: Estimateur de Theil-Sen (régression médiane)

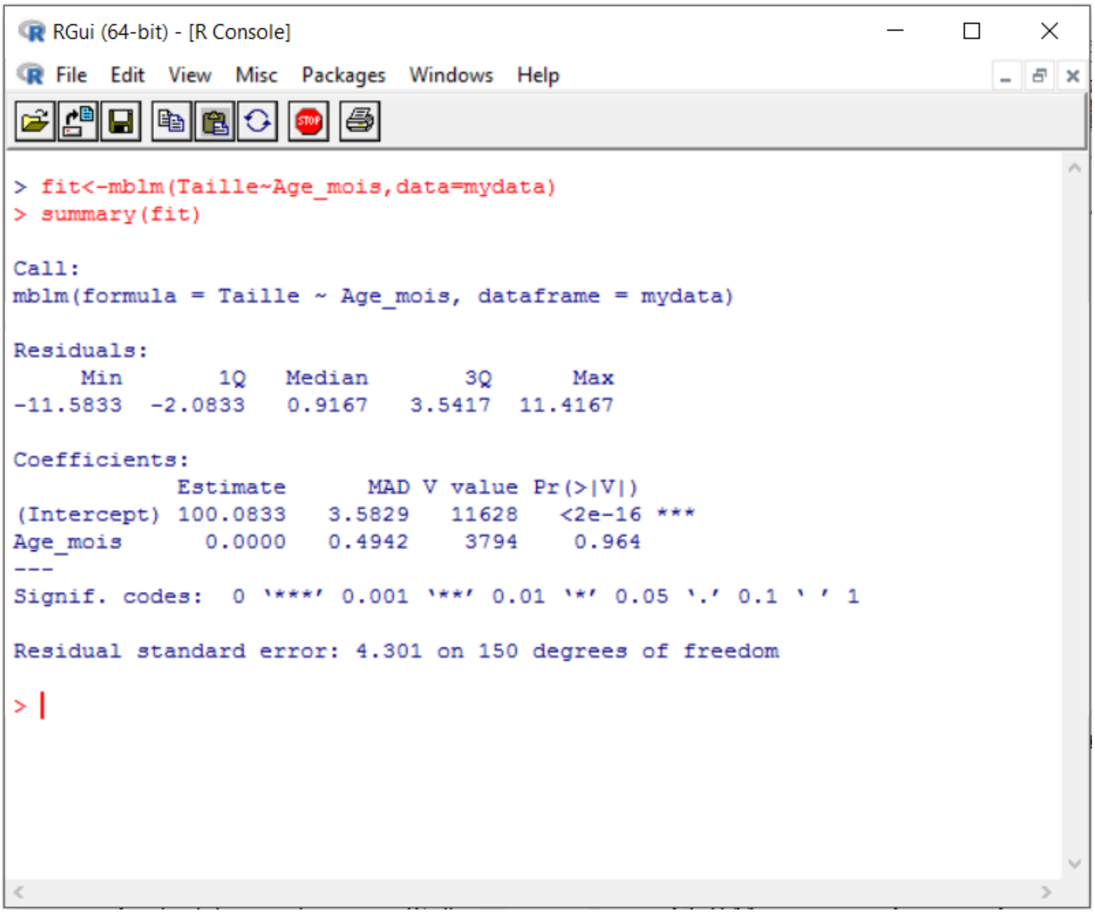
R 3.3.2

Comme nous l'avons vu dans le cours théorique, cette technique empirique est très simple à comprendre et à mettre en place. Nous allons voir ici qu'elle est disponible dans R bien évidemment et surtout la comparer à la régression quantile vue juste précédemment.

Donc d'abord nous chargeons le jeu de données:



Et ensuite nous utilisons la fonction **mblm( )** du package **mblm**:



Nous pouvons faire un plot tout simple:

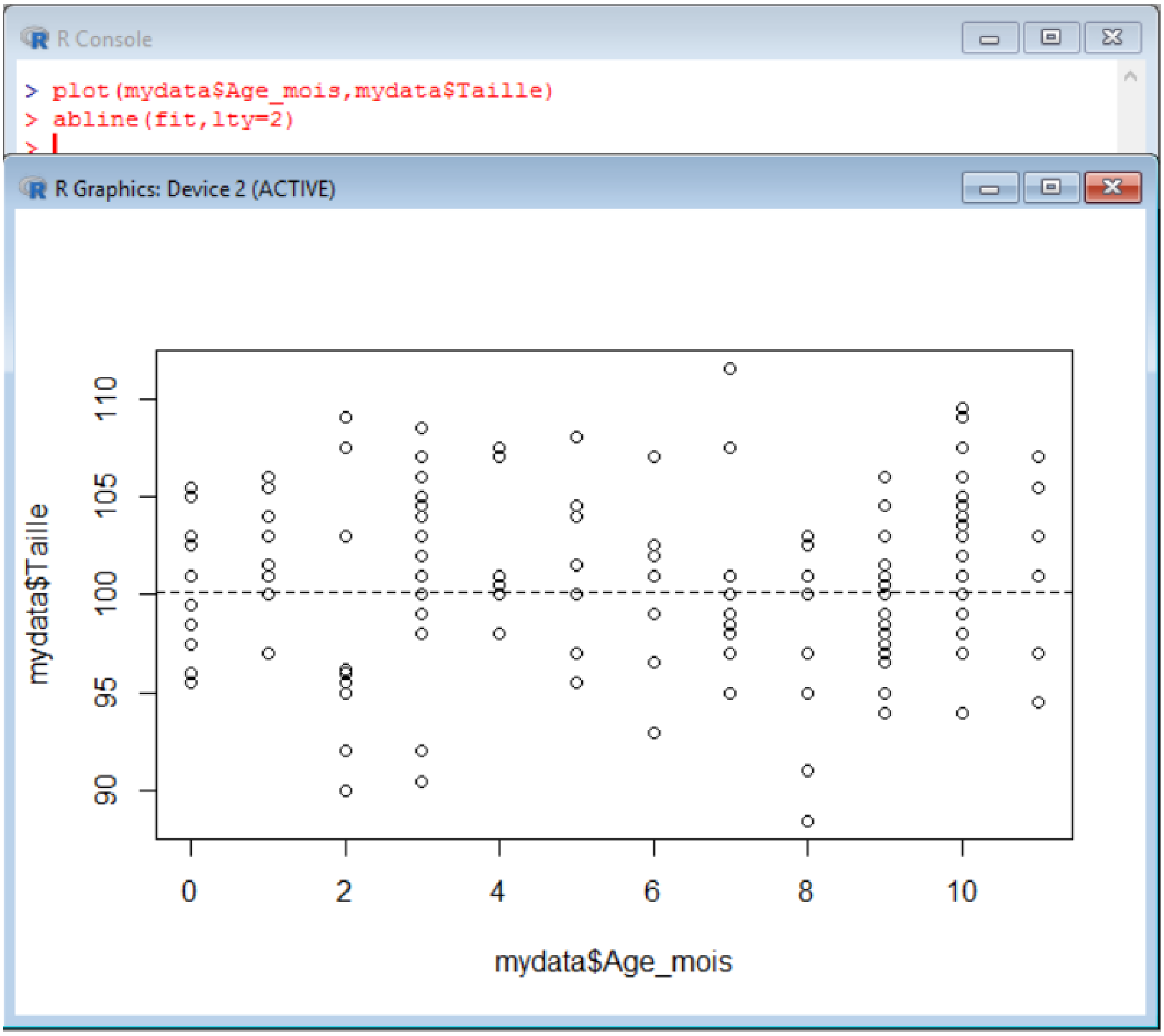
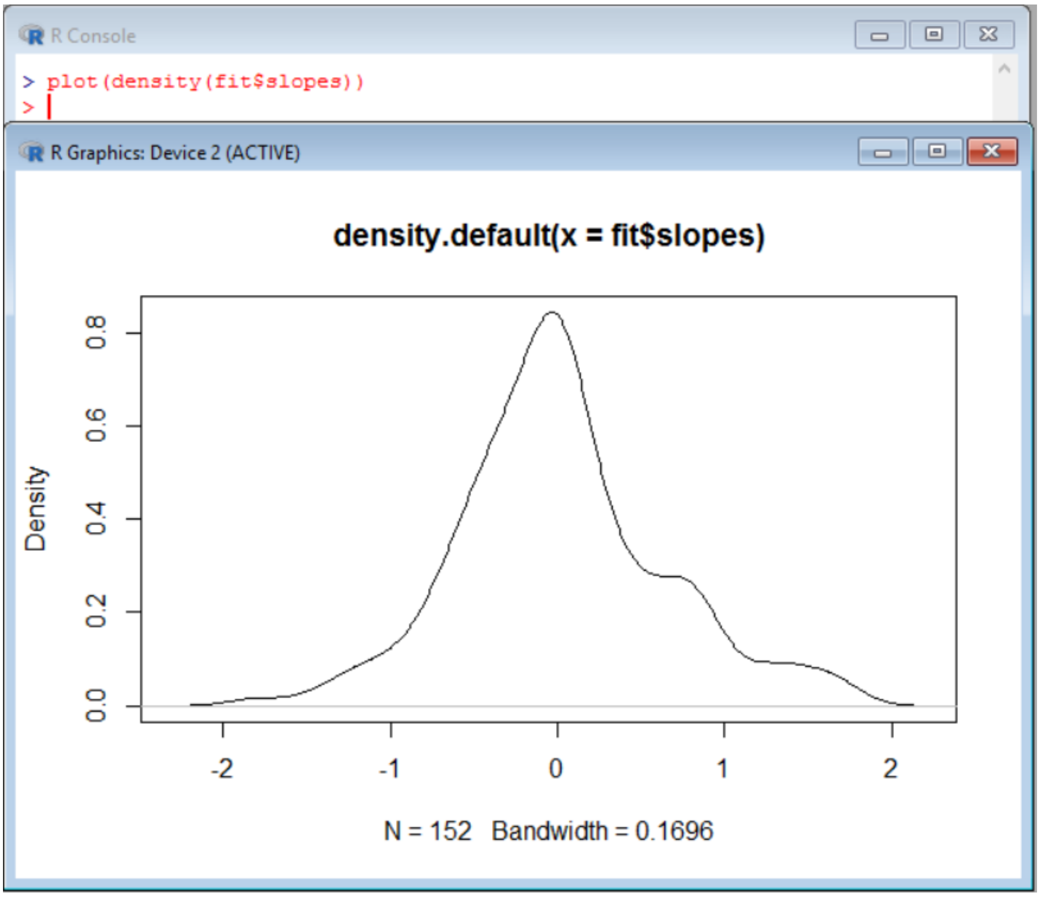
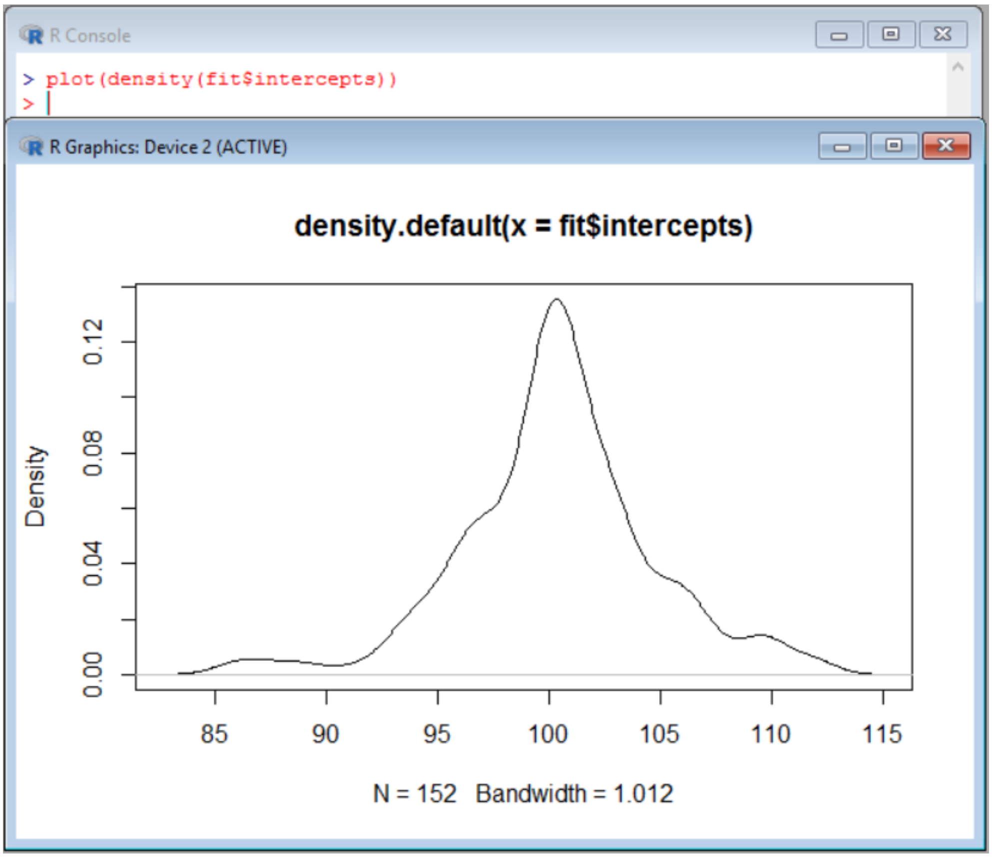


Figure 244 Graph d'une régression médiane

Ce qui est beaucoup plus intéressant par contre, c'est que nous pouvons faire un plot de la densité des pentes:



et de la densité des ordonnées à l'origine:



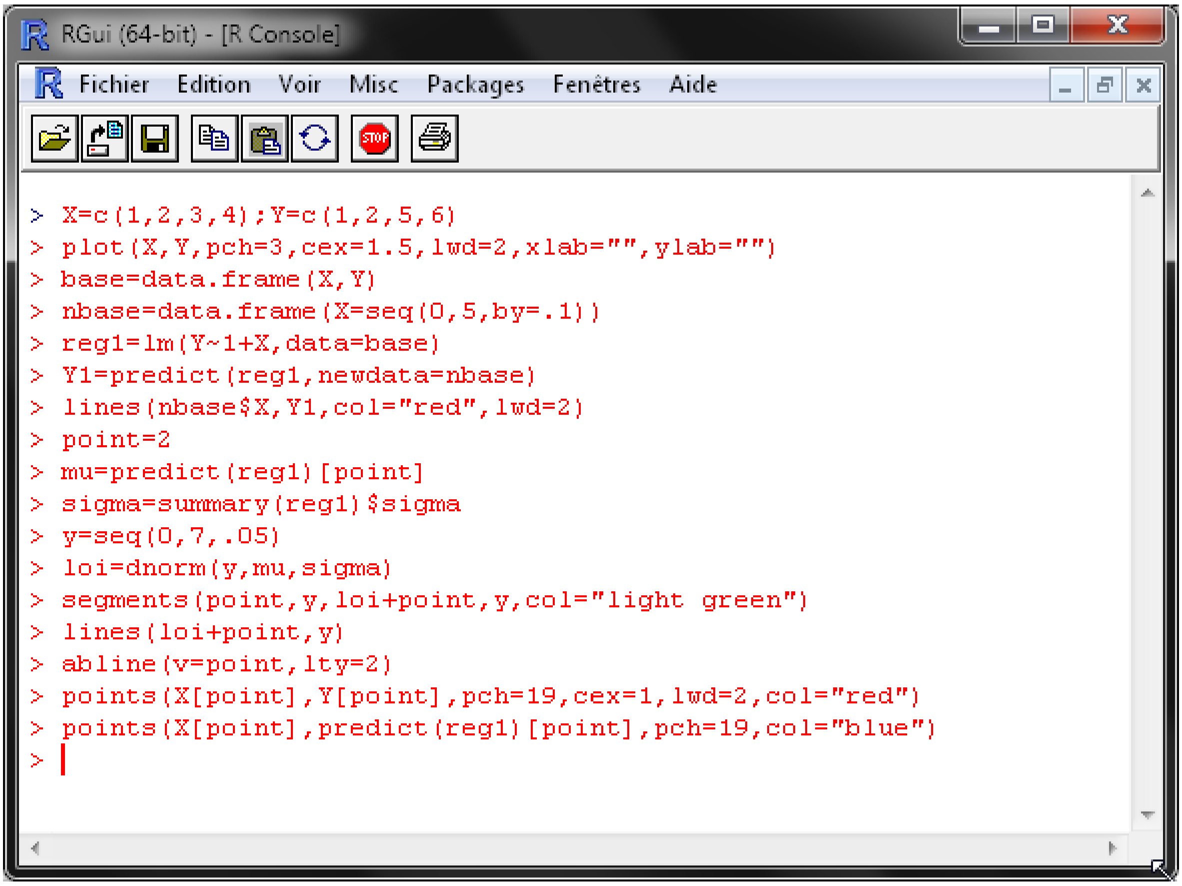
Exemple 254.: Régression linéaire univariée par moindres carrés ordinaires (modèle Gaussien) avec plot 2D/3D du modèle sous-jacent

R 3.0.2

Dans la littérature (la bonne de qualité!), nous voyons souvent les régressions linéaires univariées représentées avec la loi Normale à la verticale. Comme c'est une fonctionnalité souvent demandée voici typiquement le code avec l'aimable autorisation de Arthur CHARPENTIER pris de son billet de blog du mois d'octobre 2011:

http://freakonometrics.hypotheses.org/2280

Donc avec des données bidons cela donne d'abord au niveau des commandes.



Ce qui donne visuellement:

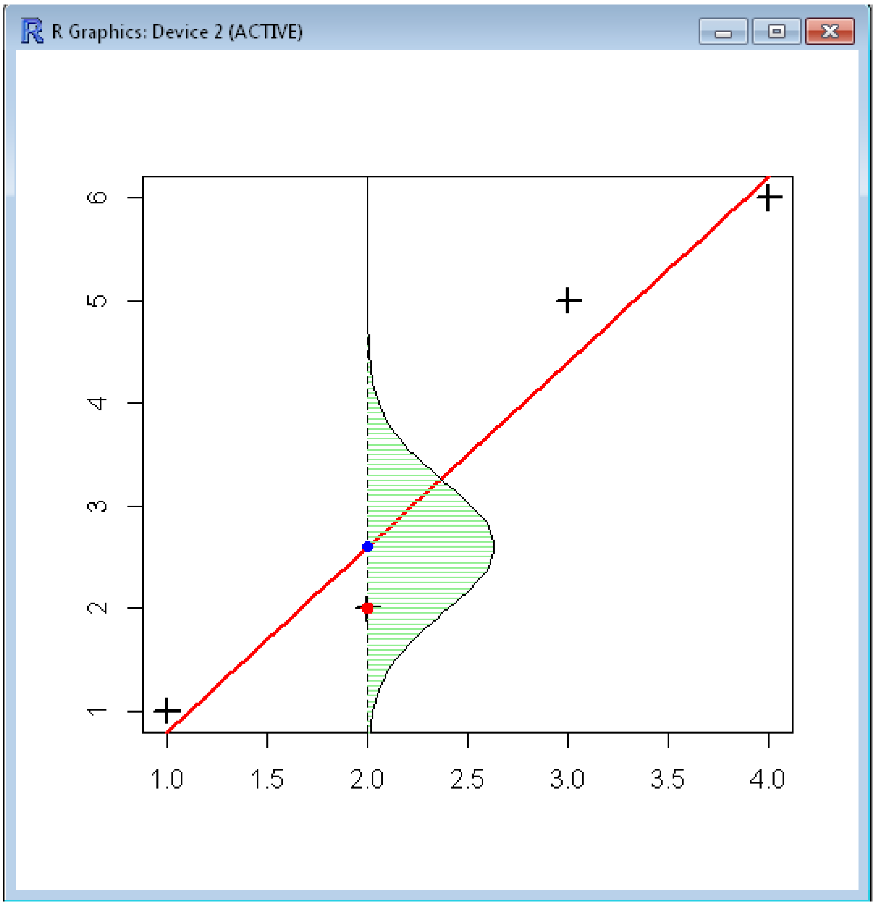


Figure 245 Graph de régression avec représentation de la Gaussienne sur un point

De même en perspective 3D et provenant toujours du même blog:

```
attach(cars)
n=2 #choix empirique
X=cars$speed
Y=cars$dist
df=data.frame(X,Y)
vX=seq(min(X)-2,max(X)+2,length=n)
vY=seq(min(Y)-15,max(Y)+15,length=n)
#on crée la box 3D à vide (remarquez l'astuce!)
mat=persp(vX,vY,matrix(0,n,n),zlim=c(0,.1),theta=-30,ticktype="detailed")
#mat va servir de matrice de projection pour la suite

#du classique:
regmdl=glm(Y~X,data=df,family=gaussian(link="identity"))
#on prépare x pour dessiner les lignes
x=seq(min(X),max(X),length=500)

#pas évident à deviner... mais logique une fois qu'on la vu!
regLine=trans3d(x,predict(regmdl,newdata=data.frame(X=x),type="response"),
rep(0,length(x)),mat)
sdgig=sqrt(summary(regmdl)$dispersion) #scalaire (valeur unique)
y1=qnorm(.95,predict(regmdl,newdata=data.frame(X=x),type="response"),sdgig)
y2=qnorm(.05,predict(regmdl,newdata=data.frame(X=x),type="response"),sdgig)

#maintenant on dessine tout cela!
#en faisant attention à l'ordre pour ne pas écraser ce
#qui a été dessiné auparavant
C=trans3d(c(x,rev(x)),c(y1,rev(y2)),rep(0,2*length(x)),mat)
polygon(C,border=NA,col="yellow")

lines(regLine,lwd=2)
```

```
upLine=trans3d(x,y1,rep(0,length(x)),mat)
lines(upLine,lty=2)

lowLine=trans3d(x,y2,rep(0,length(x)),mat)
lines(lowLine,lty=2)

C=trans3d(X,Y,rep(0,length(X)),mat)
points(C,pch=19,col="red")

#maintenant on dessine les gaussiennes en perspective
n=8
vX=seq(min(X),max(X),length=n)
mgig=predict(regmdl,newdata=data.frame(X=vX))
sdgig=sqrt(summary(regmdl)$dispersion)

#partie pas évidente à deviner!
for(j in n:1){
       stp=251
       x=rep(vX[j],stp)
       y=seq(min(min(Y)-
15,qnorm(.05,predict(regmdl,newdata=data.frame(X=vX[j]),type="response"),
sdgig)),max(Y)+15,length=stp)
       z0=rep(0,stp)
       z=dnorm(y, mgig[j], sdgig)
       C=trans3d(c(x,x),c(y,rev(y)),c(z,z0),mat)
       polygon(C,border=NA,col="light blue",density=40)
       C=trans3d(x,y,z0,mat)
       lines(C,lty=2)
       C=trans3d(x,y,z,mat)
       lines(C,col="blue")
}
```

Ce qui donne le non moins fameux visuel:

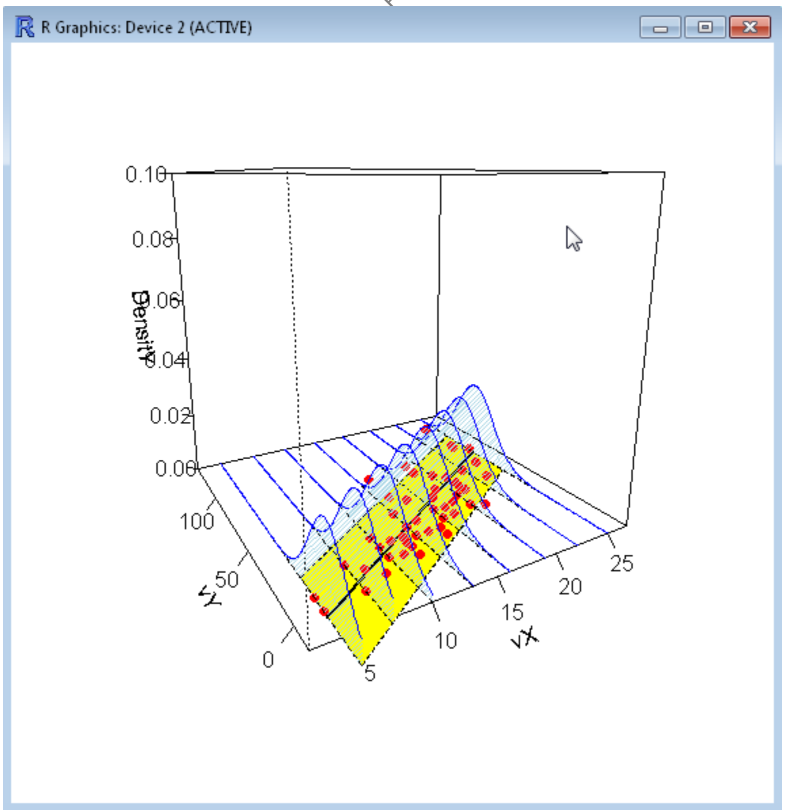


Figure 246 Graph de régression avec représentation 3D de multiples Gaussiennes

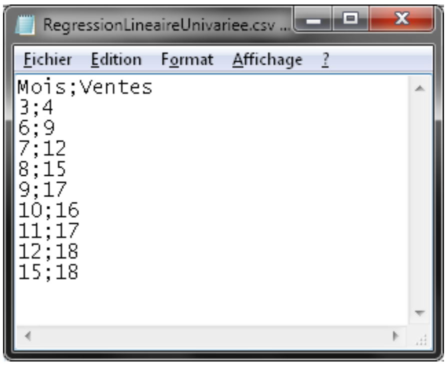
Et nous pouvons évidemment faire de même avec d'autres distribution (pensez aux modèles de régressions linéaires généralisées Binomiaux, ou Poissons).

Exemple 255.: Régression non linéaire univariée par moindres carrés ordinaires (modèle Gaussien)

R 3.0.2

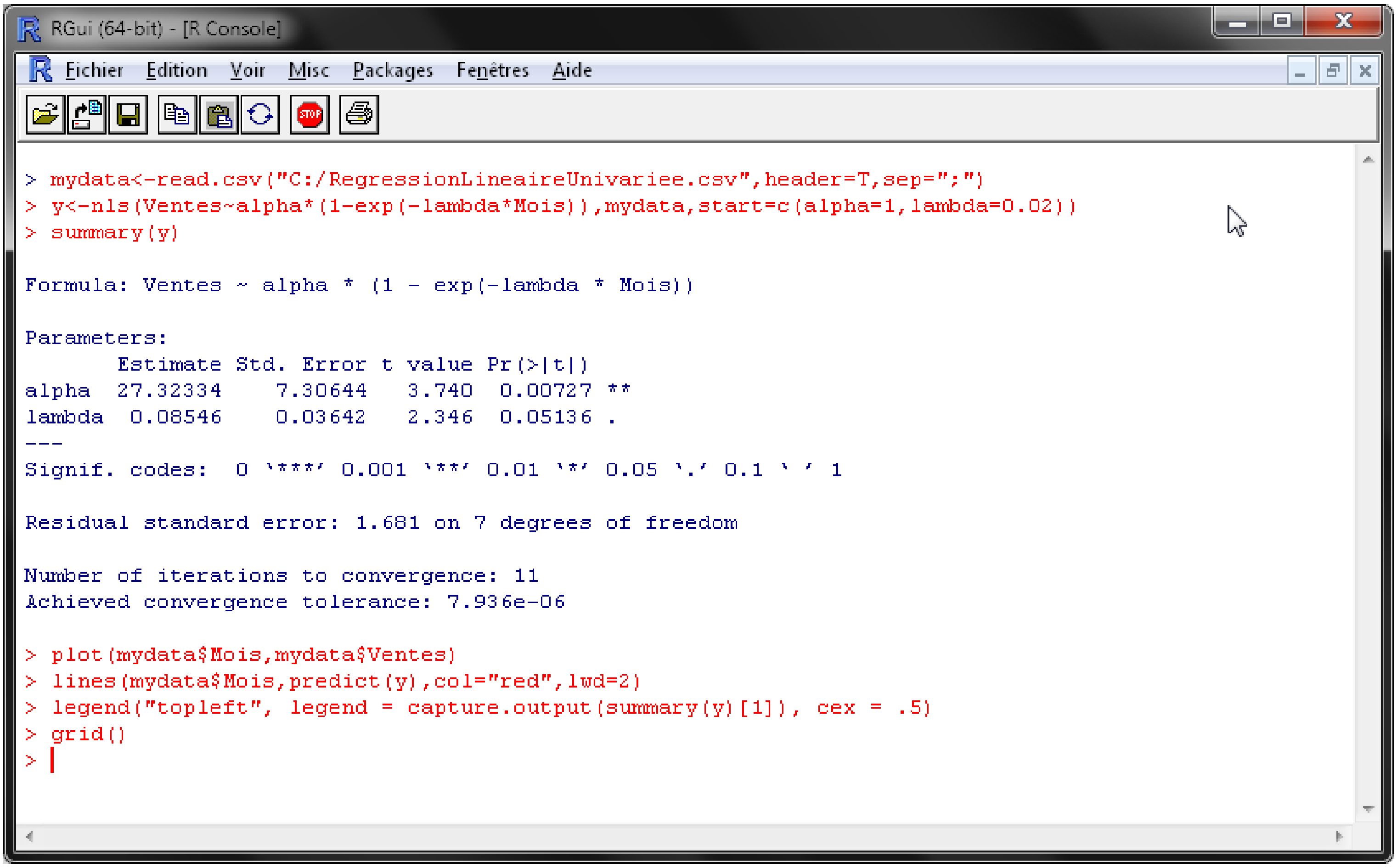
Nous pouvons faire comme nous l'avons vu dans le cours théorie une régression uni ou multivariée non linéaire par moindres carrées ordinaires.

En utilisant toujours les mêmes données:



```
Mois;Ventes
3;4
6;9
7;12
8;15
9;17
10;16
11;17
12;18
15;18
```

et en supposant un modèle exponentiel amorti, nous avons avec la commande **nls( )** :



```
> mydata<-read.csv("C:/RegressionLineaireUnivariee.csv",header=T,sep=";")
> y<-nls(Ventes~alpha*(1-exp(-lambda*Mois)),mydata,start=c(alpha=1,lambda=0.02))
> summary(y)

Formula: Ventes ~ alpha * (1 - exp(-lambda * Mois))

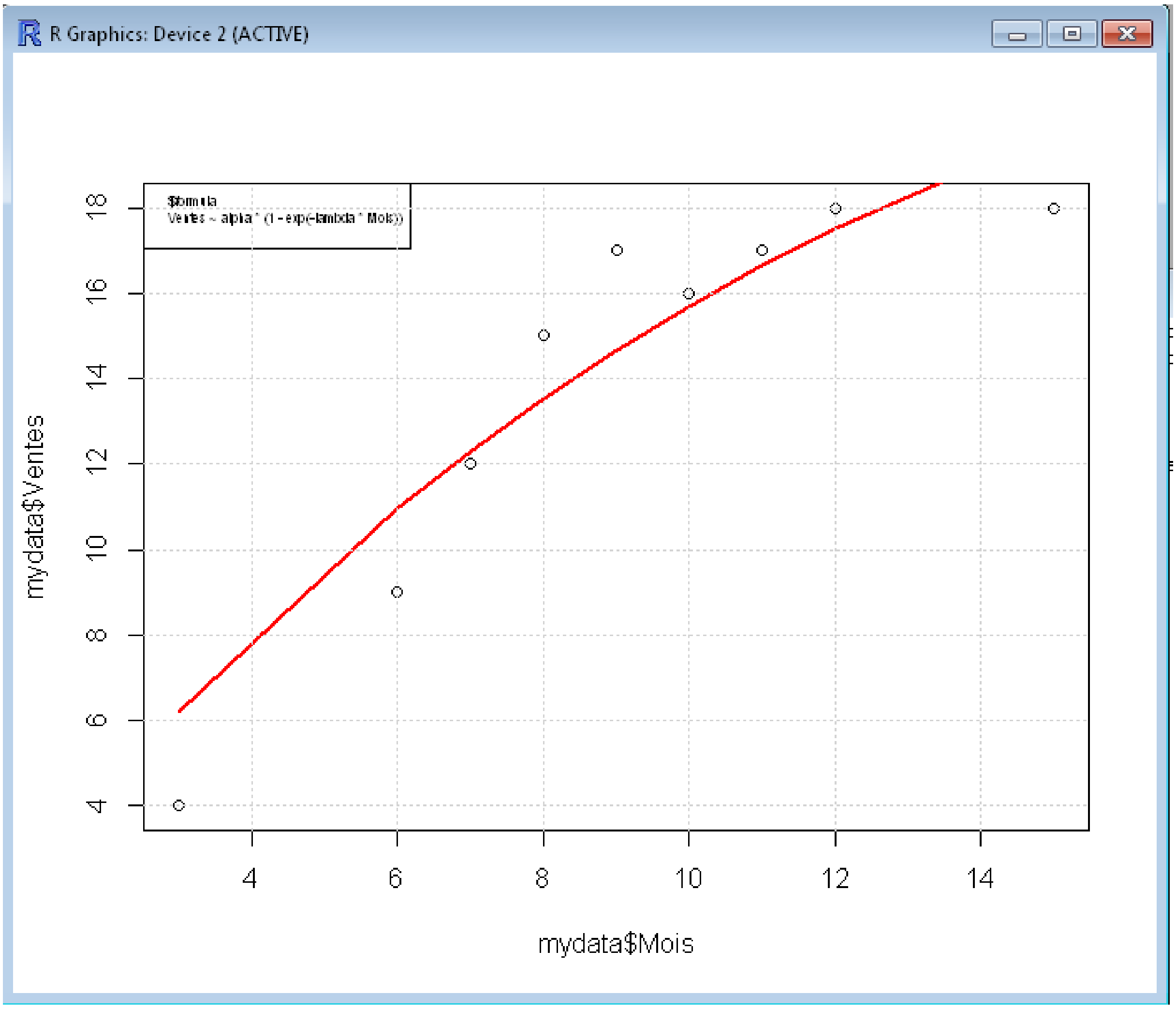
Parameters:
       Estimate Std. Error t value Pr(>|t|)
alpha  27.32334    7.30644   3.740  0.00727 **
lambda  0.08546    0.03642   2.346  0.05136 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.681 on 7 degrees of freedom

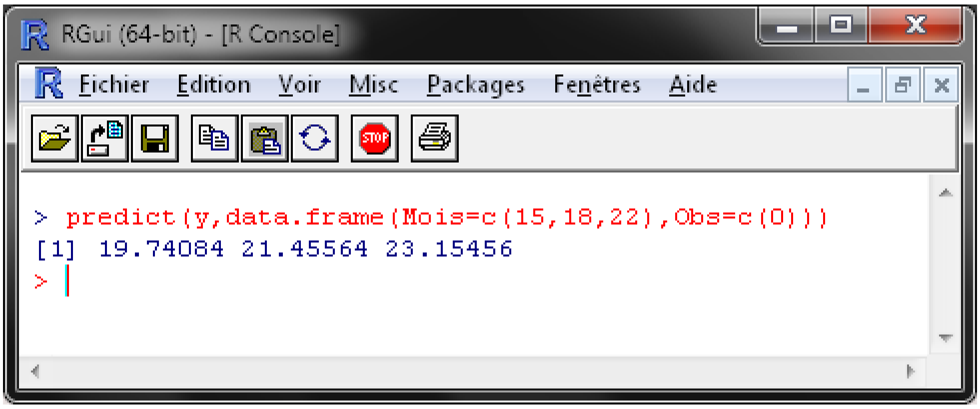
Number of iterations to convergence: 11
Achieved convergence tolerance: 7.936e-06

> plot(mydata$Mois,mydata$Ventes)
> lines(mydata$Mois,predict(y),col="red",lwd=2)
> legend("topleft", legend = capture.output(summary(y)[1]), cex = .5)
> grid()
>
```

Ce qui donne:



Ensuite, si nous voulons prédire de nouvelles valeurs:



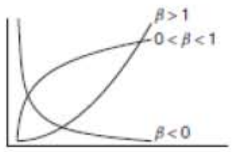
Tiré de l'ouvrage *Biostatistcal Analysis and Design using R* de Murrey Logan voici un excellent petit résumé des cas classiques des fonctions utilisées avec **nls( )** :

Function	**Preview**

Concave/convex functions

Power $(y = \alpha x^{\beta})$

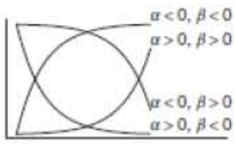
Used to describe a large range of physical and biological trends including allometric scaling relationships (e.g. Kleiber's law) and inverse square laws (e.g. Newtonian gravity). α defines the scale of the y-axis and β defines the magnitude and polatity of the rate of change and thus the degree of curvature




```
> nls(DV~a*IV^b, dataset, start=list(a=1,
    b=0.1))
```

Exponential $(y = \alpha e^{\beta x})$

Models non-asymptotic growth and decay. α defines the scale of the y-axis and increasing magnitude of β increases the curvature of the curve.



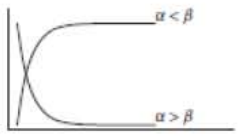

```
> nls(DV~a*exp(b*IV), dataset, start=list(a=1,
    b=0.1))
```

Aymptotic functions

Asymptotic exponential $(y = \alpha + (\beta - \alpha)e^{-e^{\gamma}x})$

Used to describe general asymptotic relationships.
Equivelent to the more simple $y = a - be^{-cx}$ when $a = \alpha$, $b = \beta - \alpha$ and $c = e^{\gamma}$

α - y value of horizontal asymptote. β - value of y when $x = 0$.
γ - natural log of rate of curvature

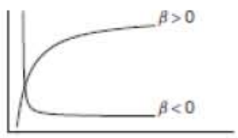



```
> nls(DV~a+b*exp(c*x),dataset,start=list(a=1,
    b=-1,c=-1))
> nls(DV~SSasymp(IV,a,b,c), dataset)
```

Michaelis-Menten $(y = \frac{\alpha x}{\beta + x})$

Used to relate rates of enzymatic reactions to substrate concentrations

$\alpha - y$ value of horizontal asymptote. β (*Mechaelis parameter*) - value of x at which half the asymptotic response is obtained.

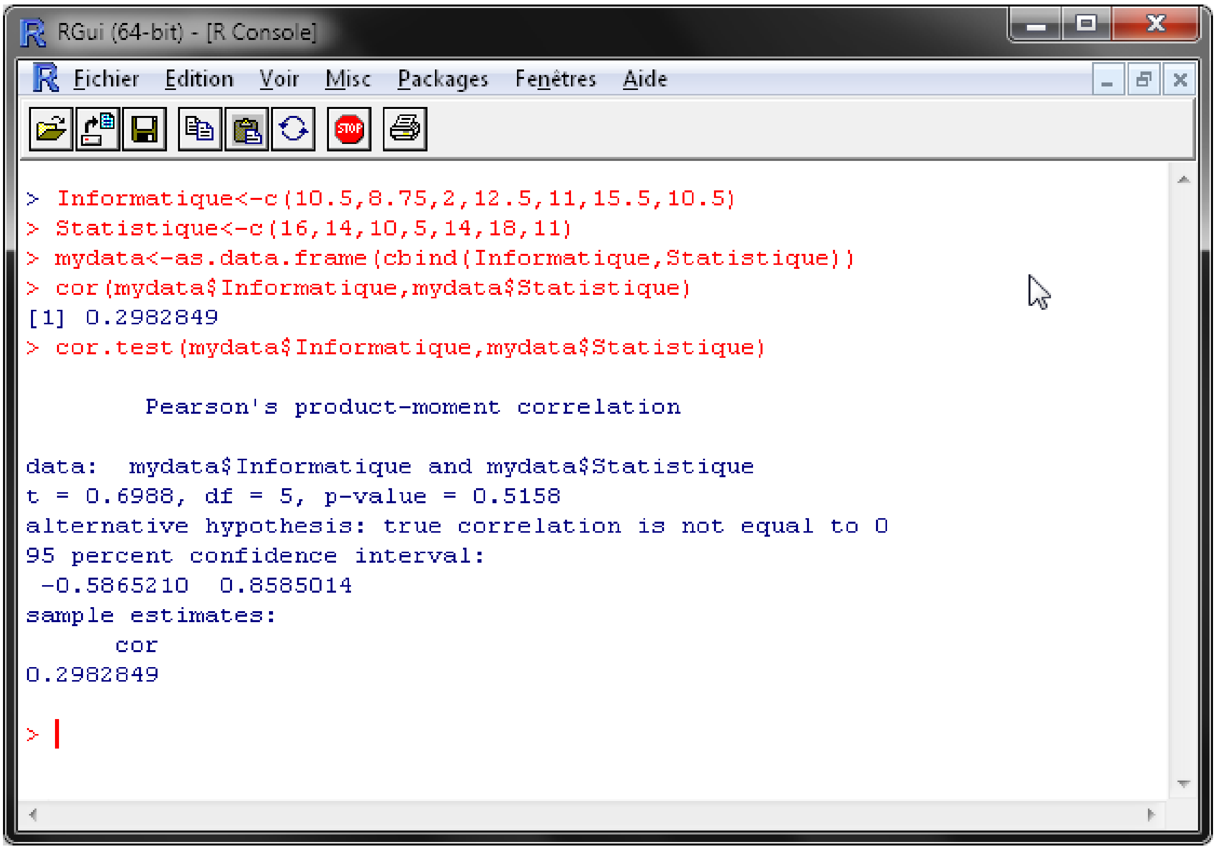



```
> nls(DV~(a*IV)/(b*IV), dataset,
    start=list(a=1, b=1))
> nls(DV~SSmicmen(IV,a,b), dataset)
```

Exemple 256.: Coefficient de corrélation de Pearson et test associé

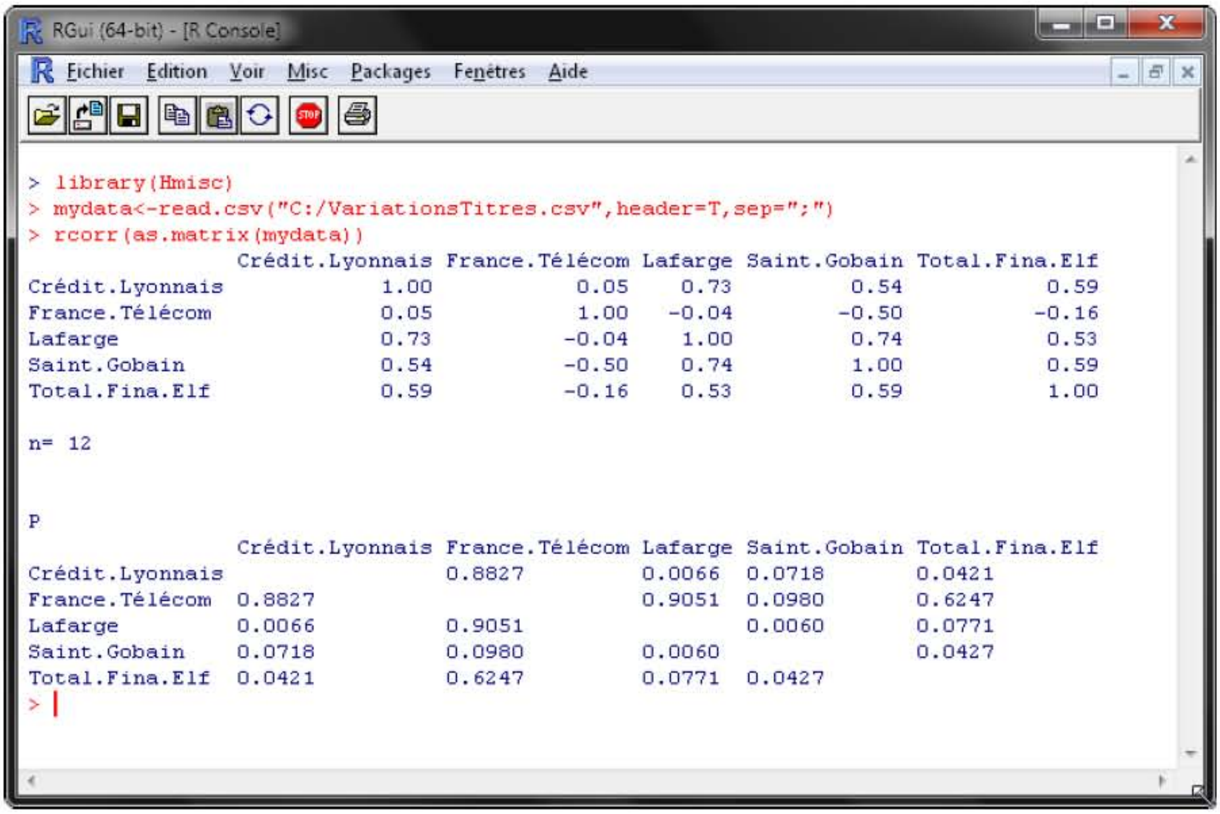
R 3.0.2

Là encore, nous allons vérifier que nous retrouvons les mêmes résultats que les calculs faits à la main après la démonstration mathématique dans le cours théorique du test et que dans Minitab et ce avec les mêmes données en utilisant la commande **cor.test( )** :



Nous retrouvons donc les mêmes valeurs à la différence que R donne des informations plus pertinentes et plus précises que Minitab (du moins par défaut).

Nous pouvons à l'aide de la commande **rcorr( )** du package **Hmisc** faire de multiples affichages de *p*-values du test de Pearson d'un coup. Voyons cela:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(Hmisc)
> mydata<-read.csv("C:/VariationsTitres.csv",header=T,sep=";")
> rcorr(as.matrix(mydata))
Crédit.Lyonnais France.Télécom Lafarge Saint.Gobain Total.Fina.Elf
Crédit.Lyonnais 1.00 0.05 0.73 0.54 0.59
France.Télécom 0.05 1.00 -0.04 -0.50 -0.16
Lafarge 0.73 -0.04 1.00 0.74 0.53
Saint.Gobain 0.54 -0.50 0.74 1.00 0.59
Total.Fina.Elf 0.59 -0.16 0.53 0.59 1.00
n= 12
P
Crédit.Lyonnais France.Télécom Lafarge Saint.Gobain Total.Fina.Elf
Crédit.Lyonnais 0.8827 0.0066 0.0718 0.0421
France.Télécom 0.8827 0.9051 0.0980 0.6247
Lafarge 0.0066 0.9051 0.0060 0.0771
Saint.Gobain 0.0718 0.0980 0.0060 0.0427
Total.Fina.Elf 0.0421 0.6247 0.0771 0.0427
>

Exemple 257.: Coefficient de corrélation bisérial ponctuel

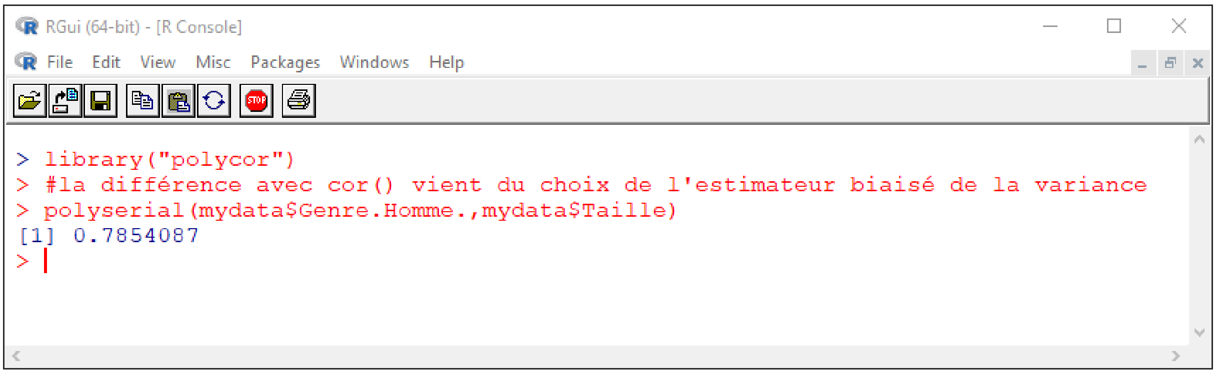
R 3.6.3

Rien d'extraordinaire ici puisque que comme nous l'avons démontré dans le cours théorique il s'agit d'un cas particulier du coefficient de corrélation de Pearson (avec la subtilité que la corrélation bisériale utilise l'estimateur biaisé de la variance).

Voyons donc un exemple en considérant le jeu de données suivant:

```
> (mydata<-read.csv("c:/tmp/Correlation_Biseriale_Ponctuelle.csv",sep=";",header=T))
   Genre.Homme. Taille
1             1   1.65
2             1   1.74
3             1   1.76
4             1   1.71
5             1   1.81
6             1   1.71
7             1   1.78
8             1   1.69
9             1   1.77
10            1   1.65
11            1   1.61
12            1   1.77
13            1   1.73
14            1   1.69
15            1   1.75
16            1   1.87
17            1   1.74
18            1   1.69
19            1   1.78
20            1   1.73
21            0   1.70
22            0   1.54
23            0   1.63
24            0   1.63
25            0   1.44
26            0   1.68
27            0   1.64
28            0   1.63
29            0   1.54
30            0   1.54
31            0   1.62
32            0   1.58
33            0   1.56
34            0   1.50
35            0   1.62
> #d'abord faisons le pearson classique
> cor(mydata$Genre.Homme.,mydata$Taille,method="pearson")
[1] 0.7458879
>
```

Voyons maintenant la fonction **polyserial( )** du package **polycor**:



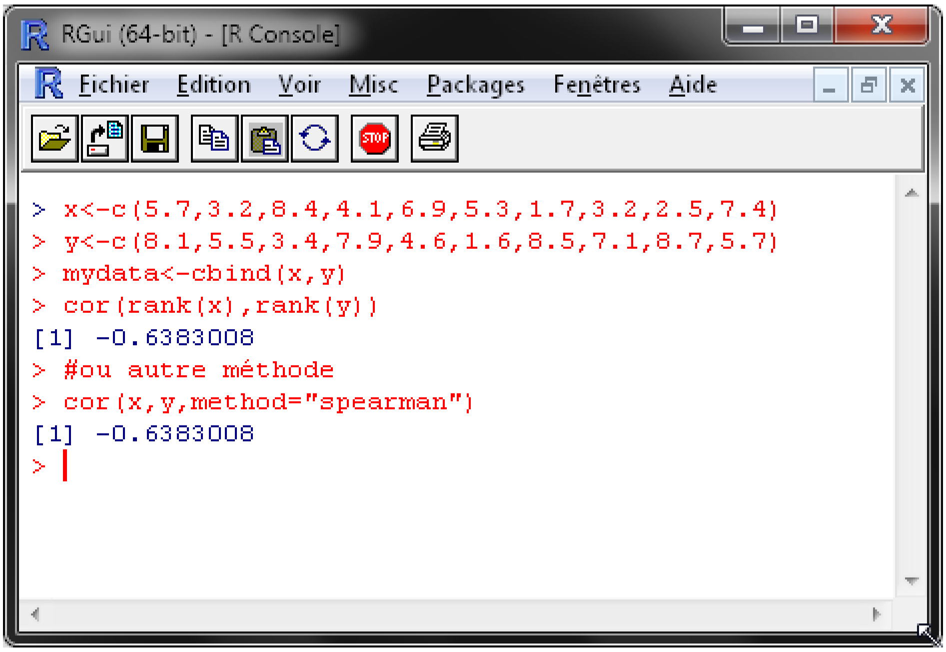
Ce qui est parfaitement conforme aux calculs faites avec Microsoft Excel.

Exemple 258.: Coefficient de corrélation de Spearman (Spearman rho)

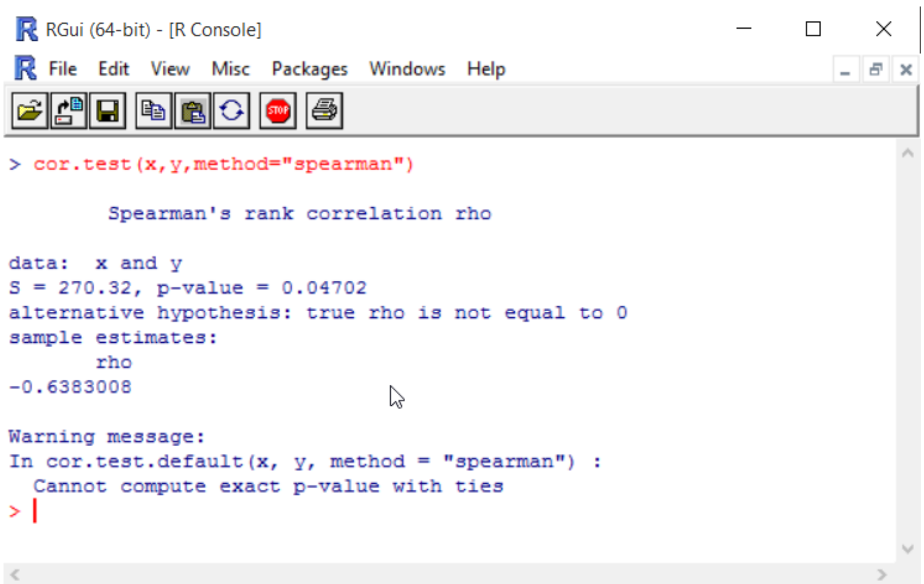
R 3.0.2

Nous allons partir ici des mêmes données que celles utilisées dans le cours théorique pour encore une fois vérifier que nous retombons sur la même chose ou pas et le comparer au résultat renvoyé par Tanagra.

D'abord en utilisant les commandes **cor( )** et **rank( )** et ensuite en spécifiant la méthode



et même si nous n'avons pas démontré dans le cours théorique comment faire le test d'hypothèse classique, voici la commande si nécessaire:



Exemple 259.: Régression linéaire multiple (avec ou sans interactions)

R 3.0.2

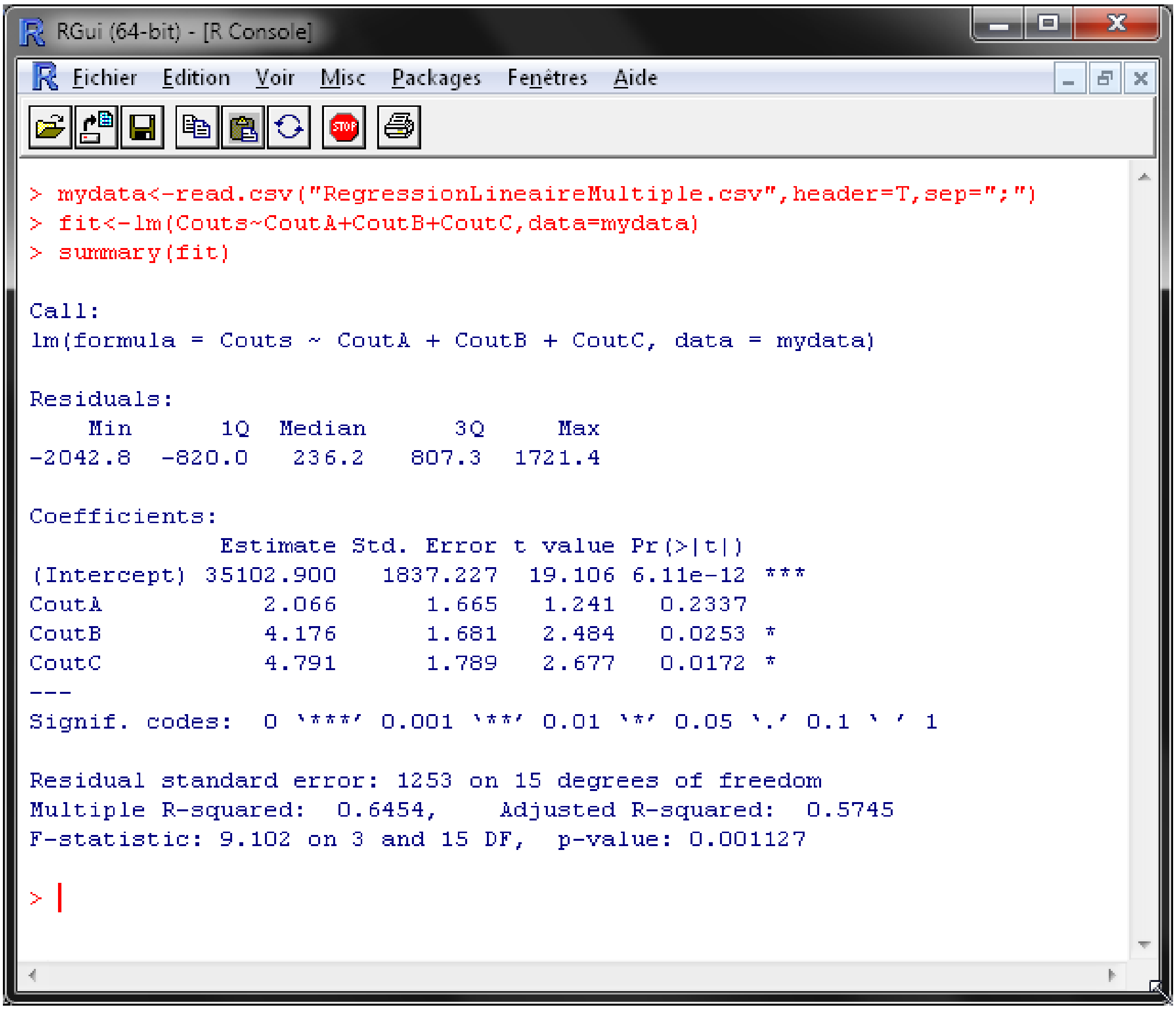
Maintenant faisons la même régression linéaire multiple que dans le cours théorique et Minitab avec la commande **lm( )** en utilisant les données ci-dessous:

RegressionLineaireMultiple.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

Allons-y d'abord sans interactions:



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> mydata<-read.csv("RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> summary(fit)

Call:
lm(formula = Couts ~ CoutA + CoutB + CoutC, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-2042.8  -820.0   236.2   807.3  1721.4

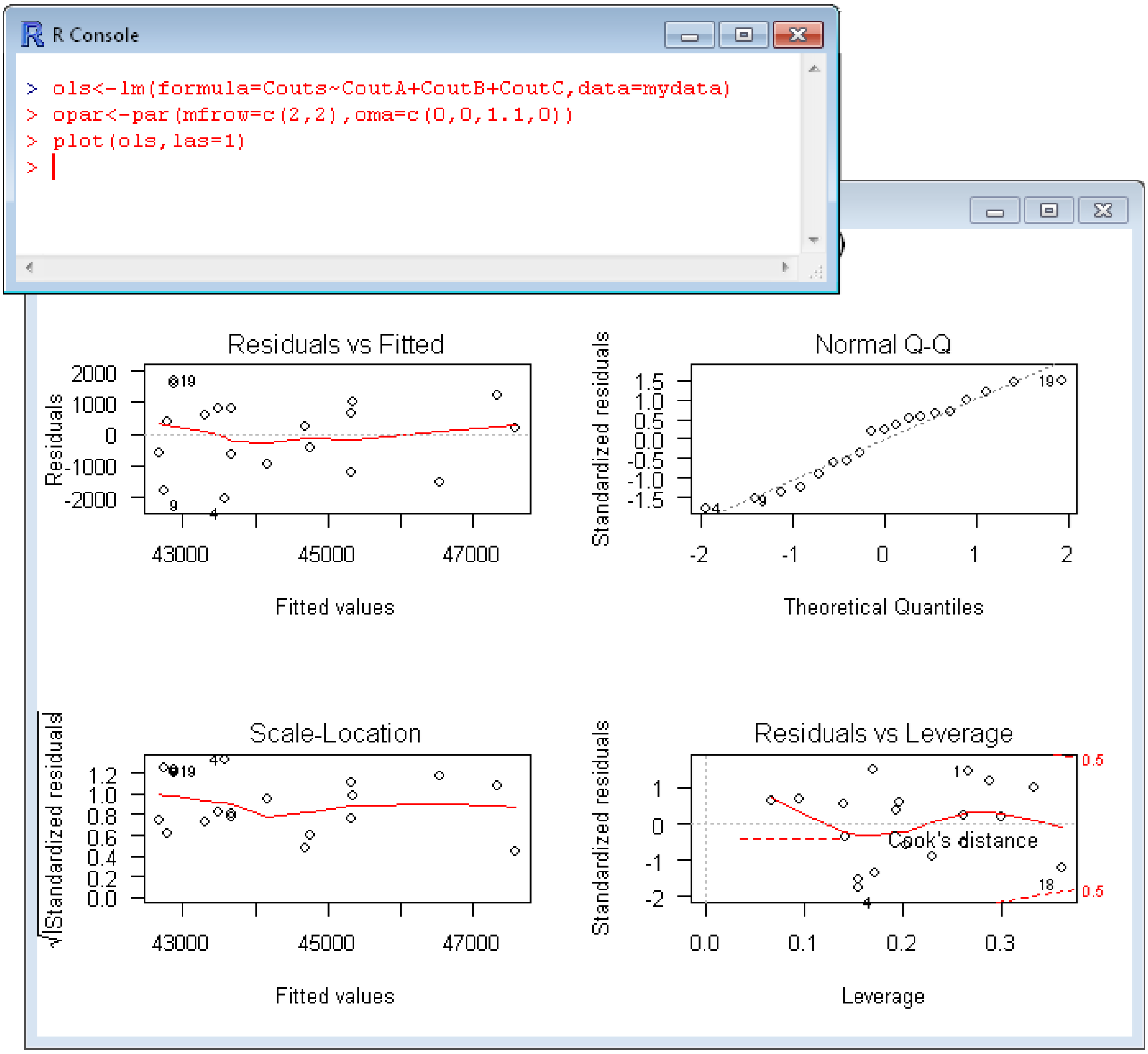
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 35102.900   1837.227  19.106 6.11e-12 ***
CoutA           2.066      1.665   1.241   0.2337
CoutB           4.176      1.681   2.484   0.0253 *
CoutC           4.791      1.789   2.677   0.0172 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1253 on 15 degrees of freedom
Multiple R-squared:  0.6454,    Adjusted R-squared:  0.5745
F-statistic: 9.102 on 3 and 15 DF,  p-value: 0.001127

>
```

Nous retrouvons donc les mêmes données qu'avec Minitab et MS Excel.

Agrémentons un peu ceci avec les analyses habituelles qualitatives:



et le package **car** en utilisant la commande **scatterplotMatrix( )** qui montre la distribution de chaque variable, l'interpolation linéaire avec chaque autre variable et aussi une régression LOESS (voir plus loin):

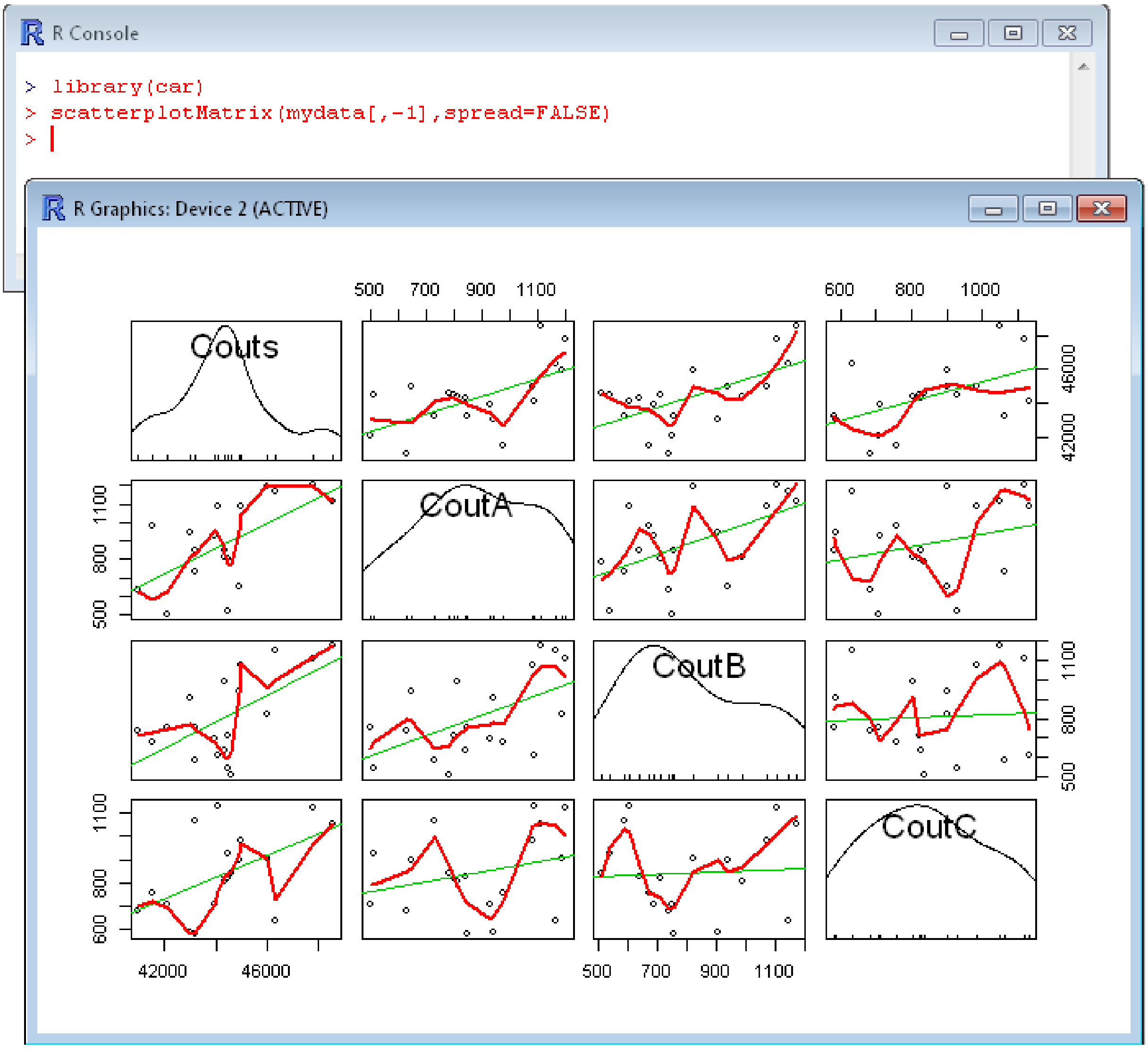
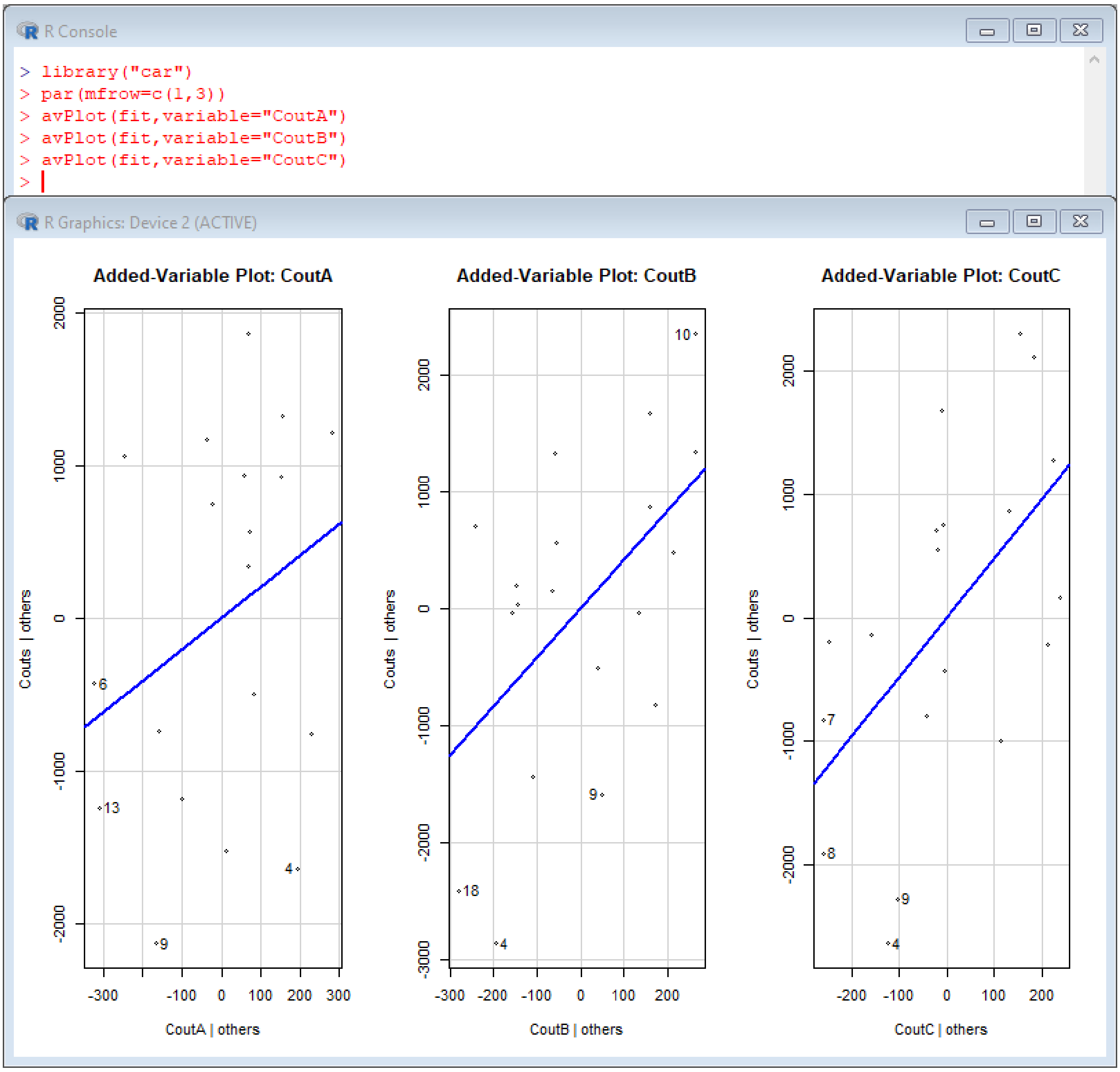


Figure 247 Graph de lattice pour régression LOESS

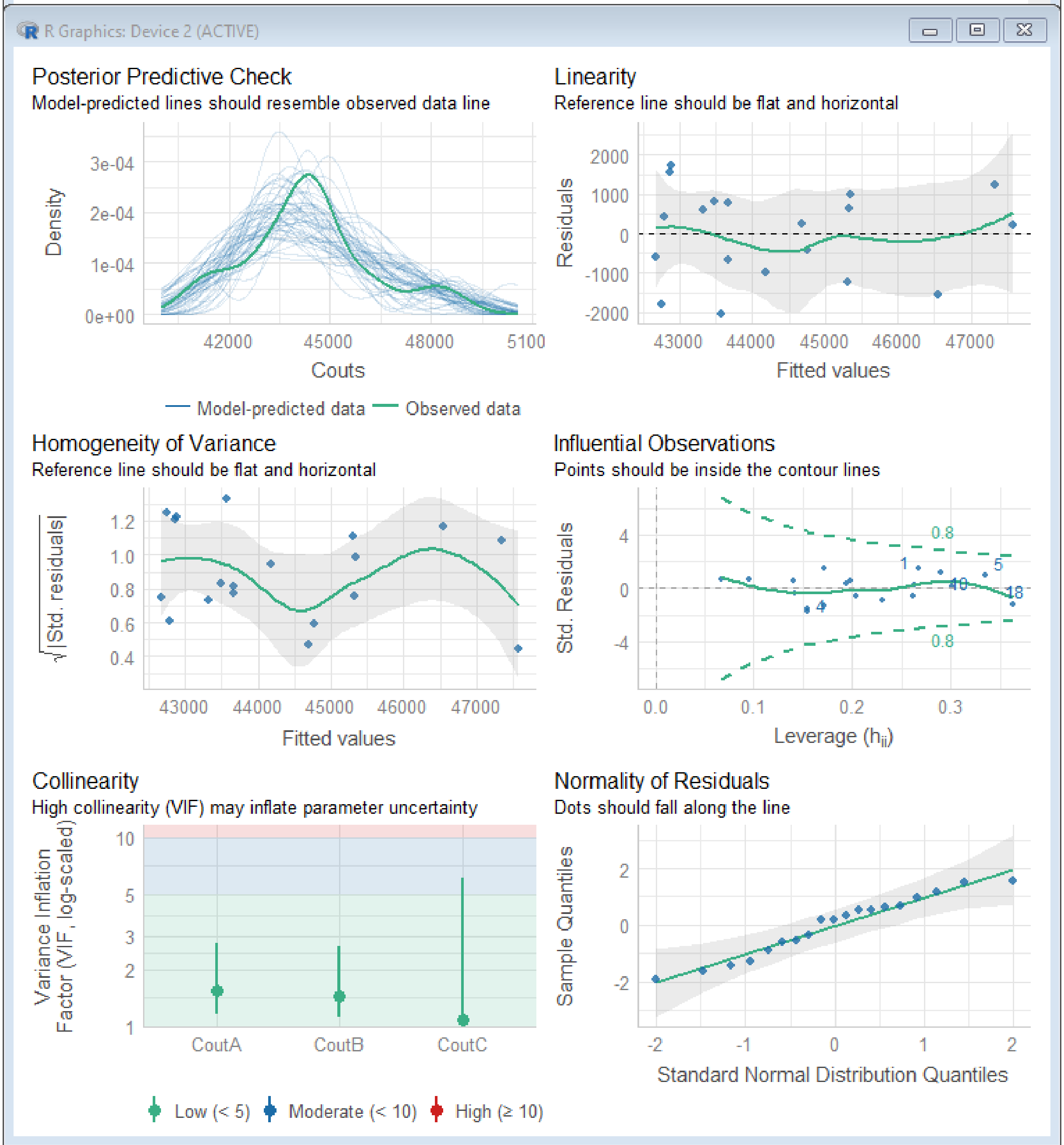
Toujours avec le package car, nous pouvons visualiser la contribution marginale pour chaque variable explicative:



La véracité de ce diagramme est facilement vérifiable avec Microsoft Excel (mais dans un temps beaucoup plus long...).

Cependant l'esthétique du graphique de diagnostique précédent est pas des plus élégantes. Nous pouvons faire mieux avec le package **performance** et et sa fonction **check_model( )** comme illustré ci-dessous:

```
> library("performance")
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",sep=";",head=T)
> fit<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> check_model(fit)
>
```



Nous pouvons aussi améliorer le tableau de sortie des coefficient avec la fonction **summ( )** du package **jtools**:

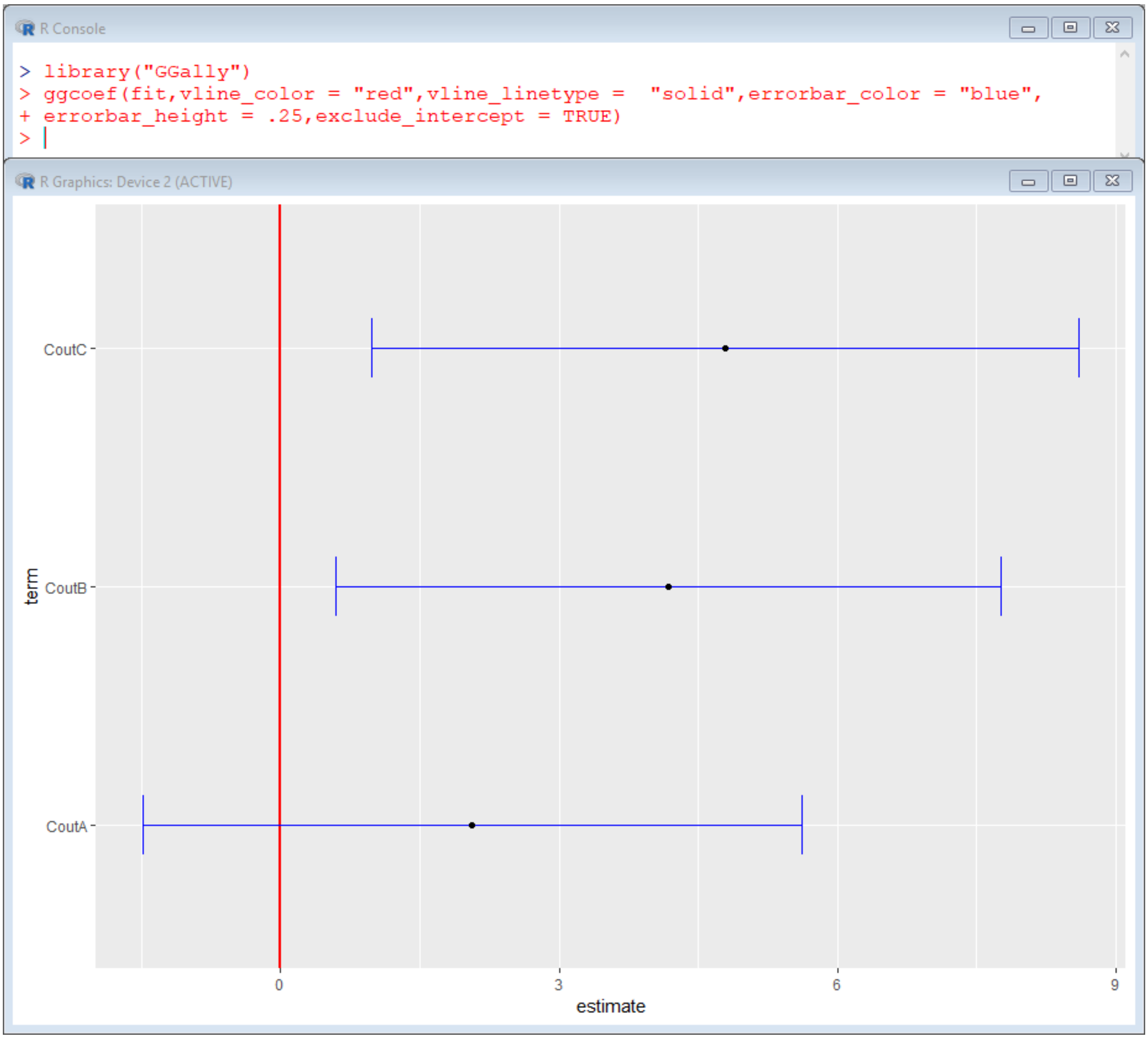
```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> library("jtools")
> summ(model,confint=TRUE,digits=3,vifs=TRUE)
MODEL INFO:
Observations: 19
Dependent Variable: Couts
Type: OLS linear regression

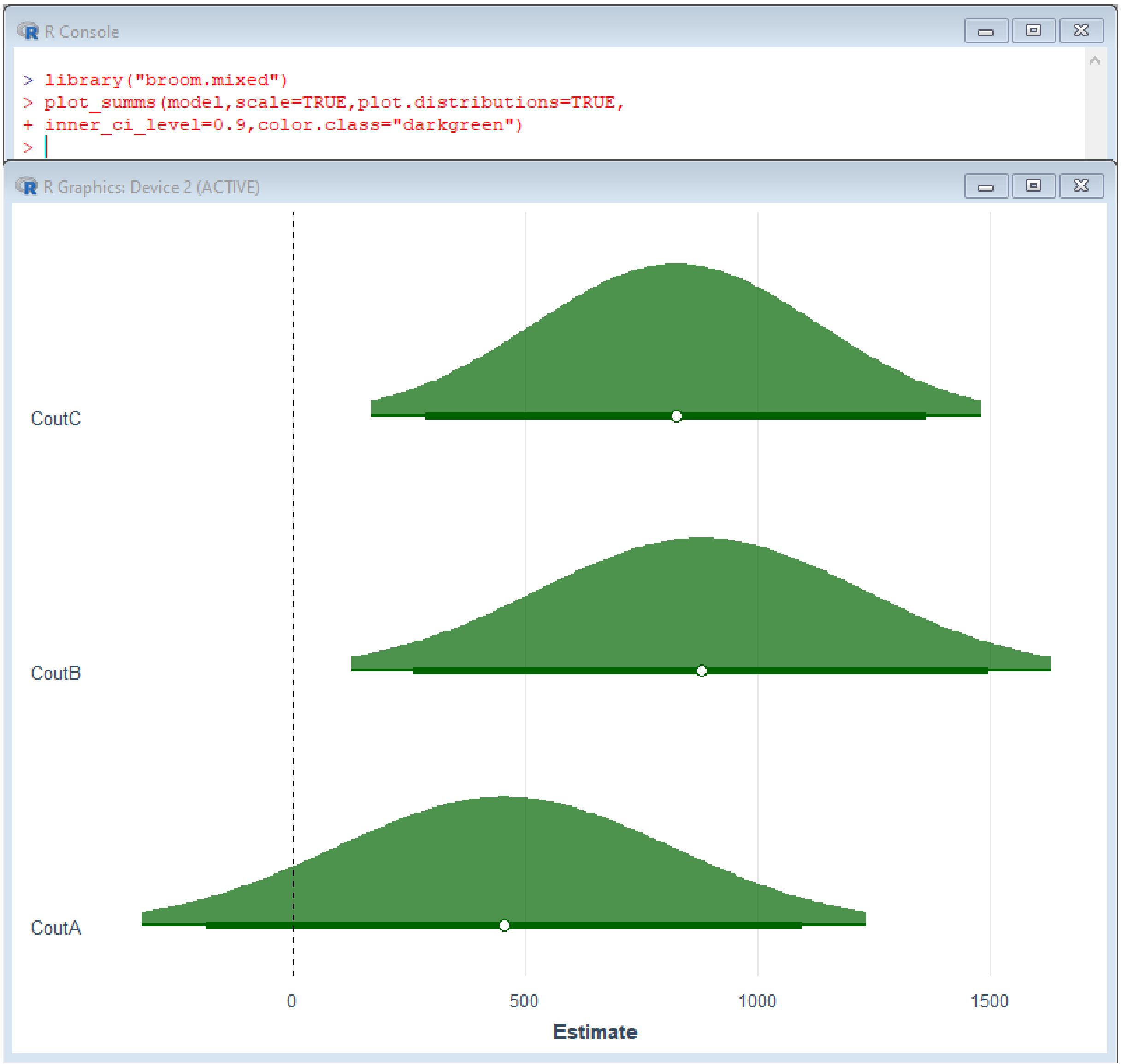
MODEL FIT:
F(3,15) = 9.102, p = 0.001
R² = 0.645
Adj. R² = 0.575

Standard errors: OLS
------------------------------------------------------------------------------
                          Est.        2.5%       97.5%    t val.       p     VIF
----------------- ----------- ----------- ----------- --------- ------- -------
(Intercept)          35102.900   31186.944   39018.857    19.106   0.000
CoutA                    2.066      -1.483       5.615     1.241   0.234   1.542
CoutB                    4.176       0.593       7.760     2.484   0.025   1.436
CoutC                    4.791       0.977       8.604     2.677   0.017   1.090
------------------------------------------------------------------------------
> 
```

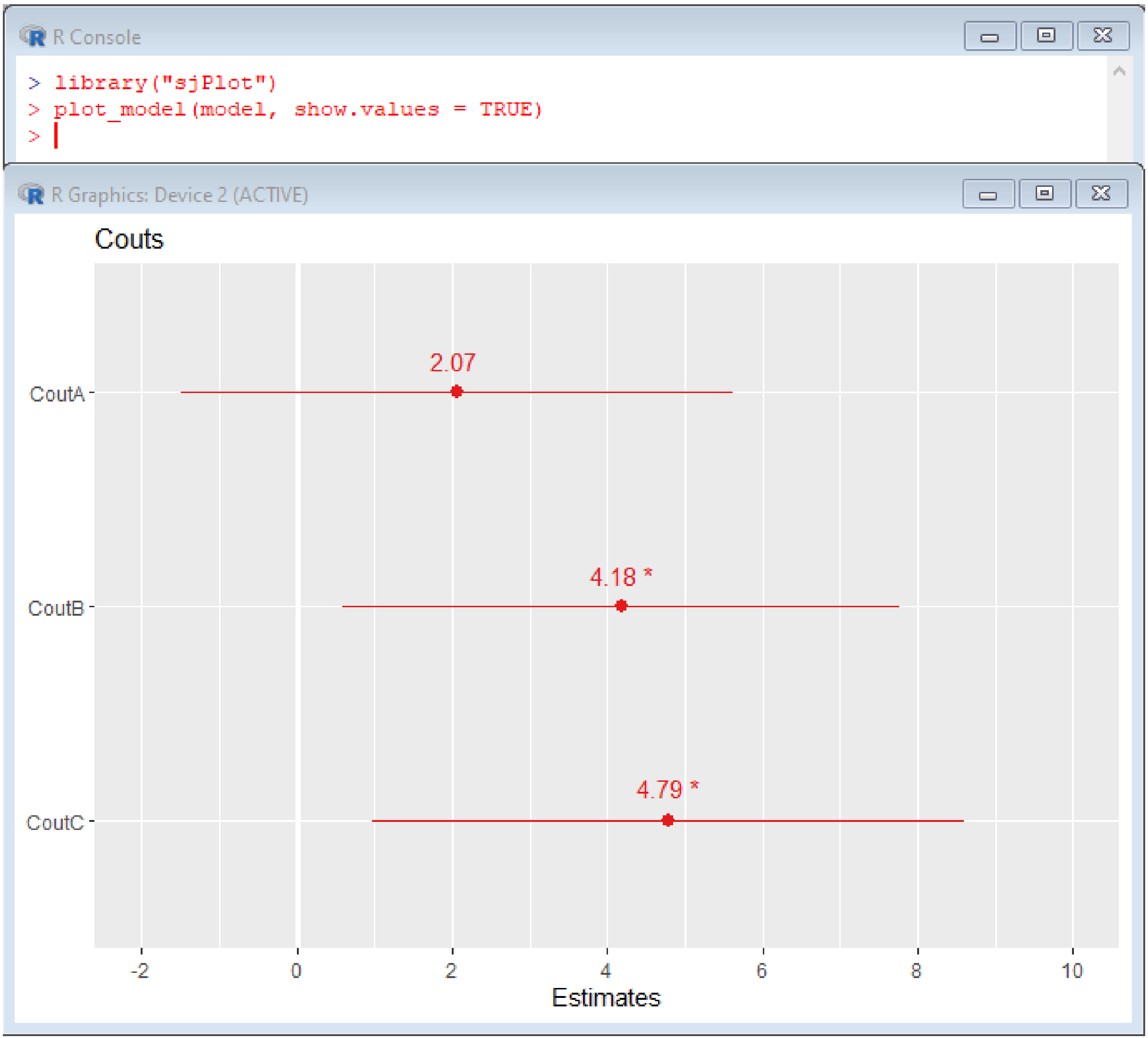
Nous pouvons aussi faire un diagnostique des intervalles de confiance des coefficients à l'aide du package **GGally** et sa fonction **ggcoef( )**:



Ou avec la fonction **plot_summs( )** du package **broom.mixed**:



Ou encore avec la fonction **plot_model( )** du package **sjPlot**:



Si nous voulons mettre en place des interactions, la syntaxe sera:

```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide

> mydata<-read.csv("RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutB+CoutC+CoutA:CoutB,data=mydata)
> summary(fit)

Call:
lm(formula = Couts ~ CoutA + CoutB + CoutC + CoutA:CoutB, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-1819.9  -551.6   290.3   841.6  1323.2

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.551e+04  5.909e+03   7.701 2.13e-06 ***
CoutA       -8.260e+00  5.824e+00  -1.418   0.1780
CoutB       -8.734e+00  7.192e+00  -1.214   0.2447
CoutC        4.079e+00  1.707e+00   2.390   0.0315 *
CoutA:CoutB  1.322e-02  7.188e-03   1.839   0.0872 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1164 on 14 degrees of freedom
Multiple R-squared:  0.7144,    Adjusted R-squared:  0.6328
F-statistic: 8.757 on 4 and 14 DF,  p-value: 0.0009292

>
```

Nous pouvons évidemment aussi très facilement faire des prévisions à nouveau avec la commande **predict( )** :

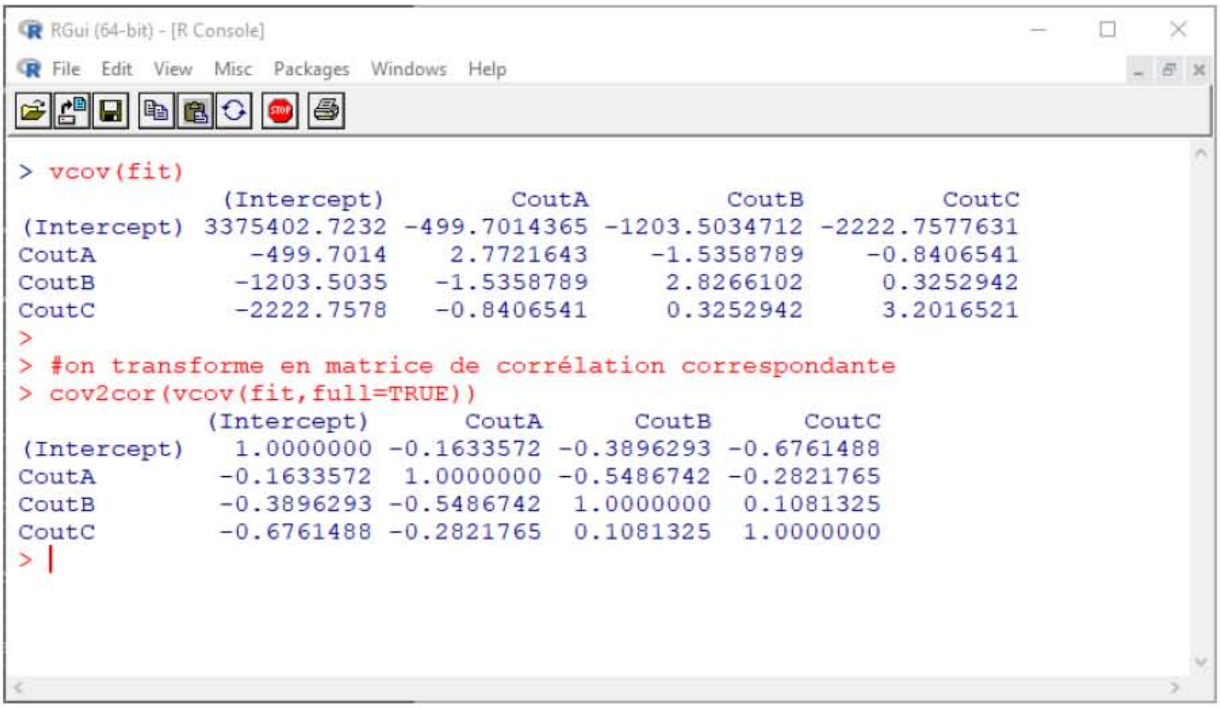
```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide

> mydata<-read.csv("C:/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> predict(fit,data.frame(CoutA=750,CoutB=800,CoutC=900),interval="prediction")
       fit      lwr      upr
1 44305.03 41508.93 47101.12
>
```

Matrice des variances-covariances du modèle

Nous avons vu dans le cours théorique l'importance de la matrice des variances-covariances dans l'analyse post-hoc de la régression multivariée pour calculer des nombreux autres statistiques y relatives (typiquement celles qui vont suivre).

Voyons comment obtenir cette dernière à l'aide de la fonction **vcov( )**:

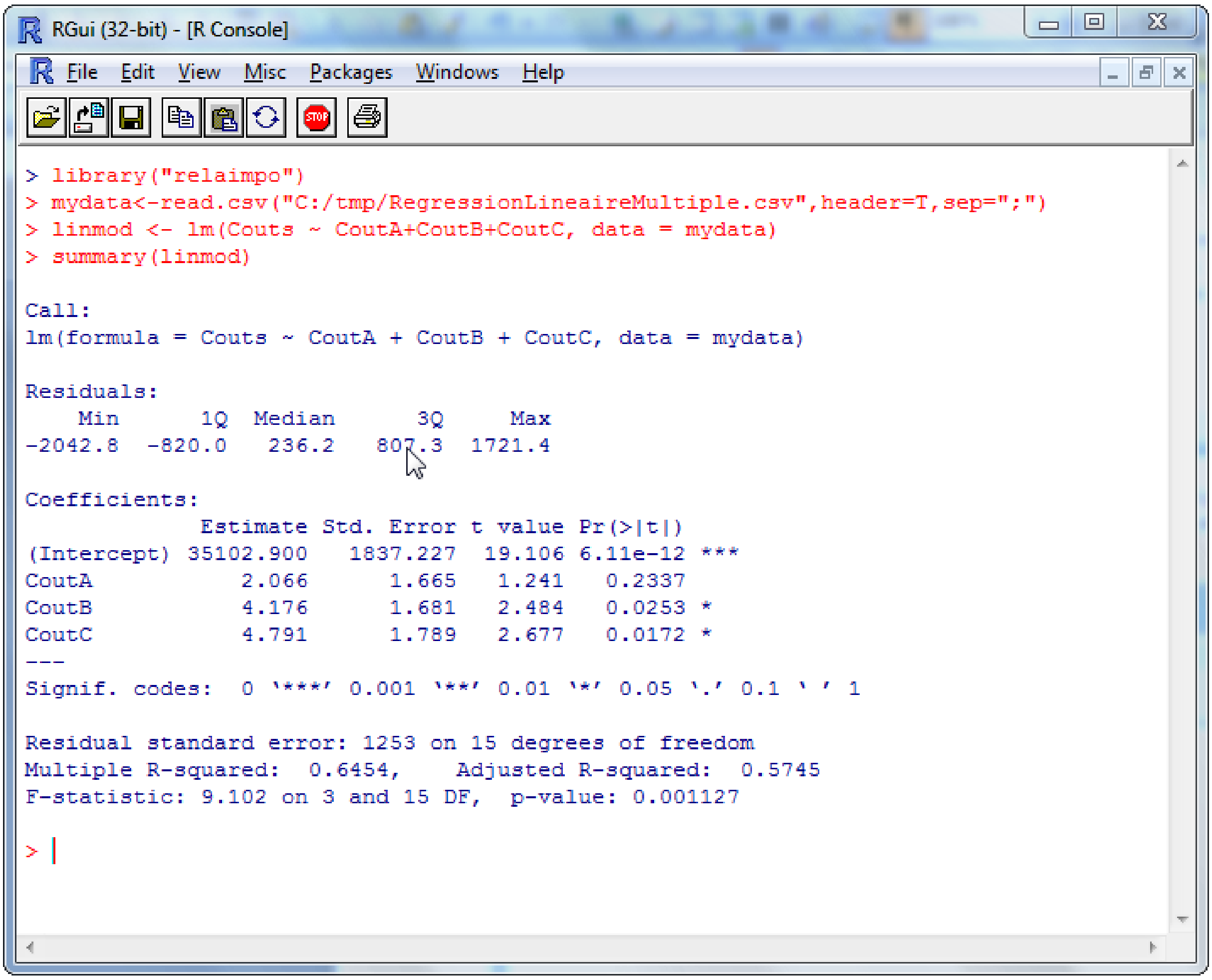


```
> vcov(fit)
              (Intercept)        CoutA         CoutB         CoutC
(Intercept) 3375402.7232 -499.7014365 -1203.5034712 -2222.7577631
CoutA          -499.7014    2.7721643    -1.5358789    -0.8406541
CoutB         -1203.5035   -1.5358789     2.8266102     0.3252942
CoutC         -2222.7578   -0.8406541     0.3252942     3.2016521
>
> #on transforme en matrice de corrélation correspondante
> cov2cor(vcov(fit,full=TRUE))
            (Intercept)      CoutA      CoutB      CoutC
(Intercept)   1.0000000 -0.1633572 -0.3896293 -0.6761488
CoutA        -0.1633572  1.0000000 -0.5486742 -0.2821765
CoutB        -0.3896293 -0.5486742  1.0000000  0.1081325
CoutC        -0.6761488 -0.2821765  0.1081325  1.0000000
> 
```

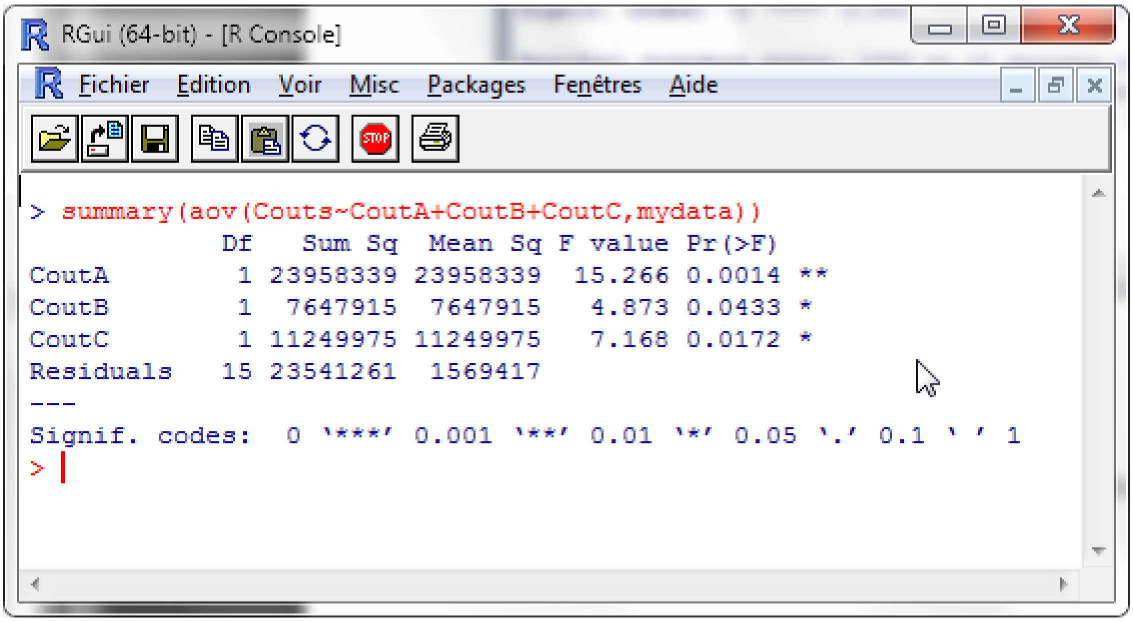
Importance relative des variables explicatives

Comme nous le savons à chaque régression il est possible d'associer une ANOVA et à chaque variable nous pouvons associer la fraction du carré des erreurs qu'elle prend en compte par rapport au carré des erreurs totales! Avec un simple ratio il est alors possible de définir une importance relative de chaque coefficient. Cependant il existe de nombreuses autres techniques empiriques pour définir une importance relative. Voyons un exemple avec le package **realimpo**!

D'abord le standard:

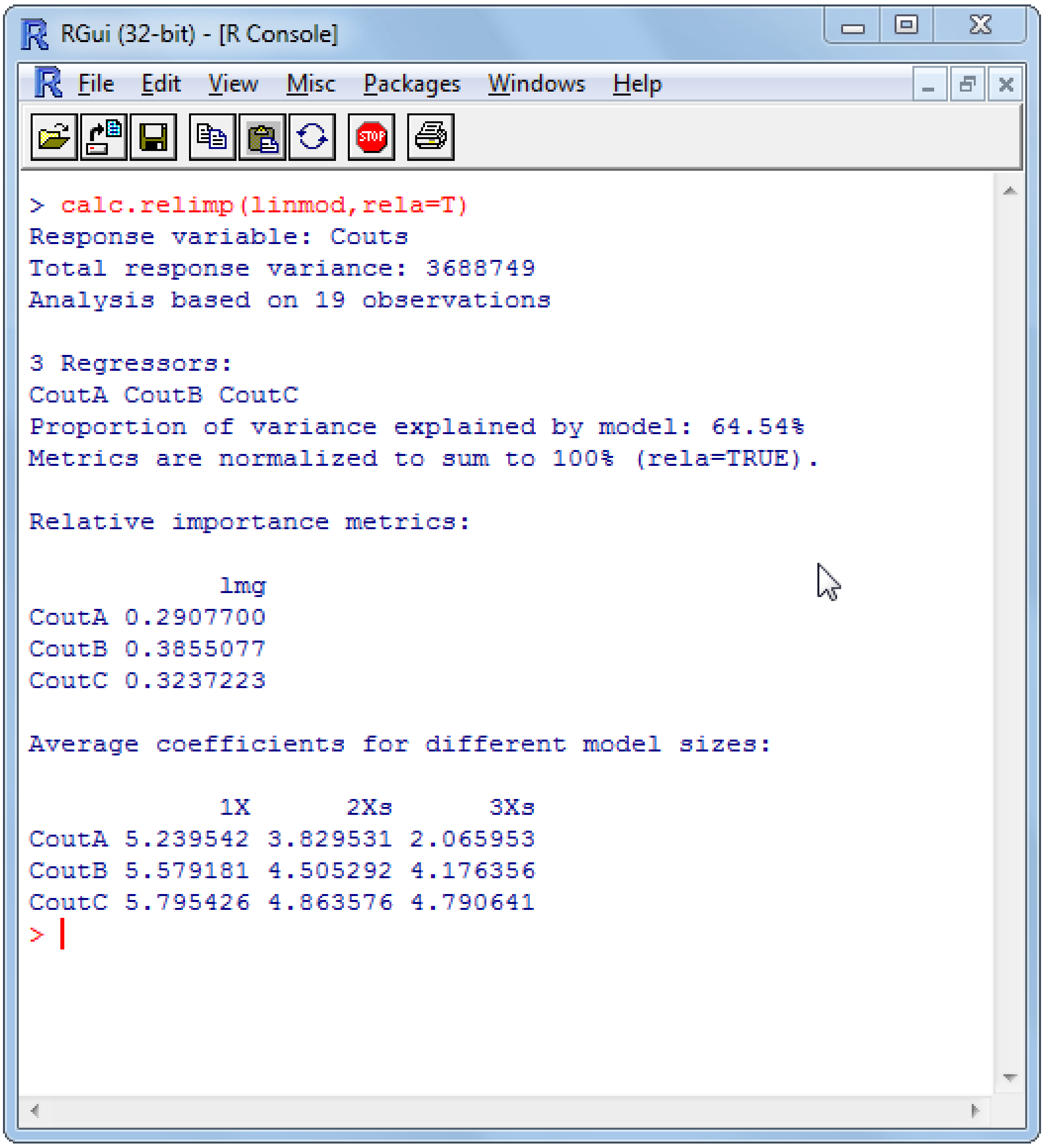


et l'ANOVA correspondante:



Donc selon l'ANOVA, le *CoutA* représente 36% du total du *Sum Sq* et *CoutB* 11.5%, *CoutC* 16.9% (somme de 64.54% pour les trois variables).

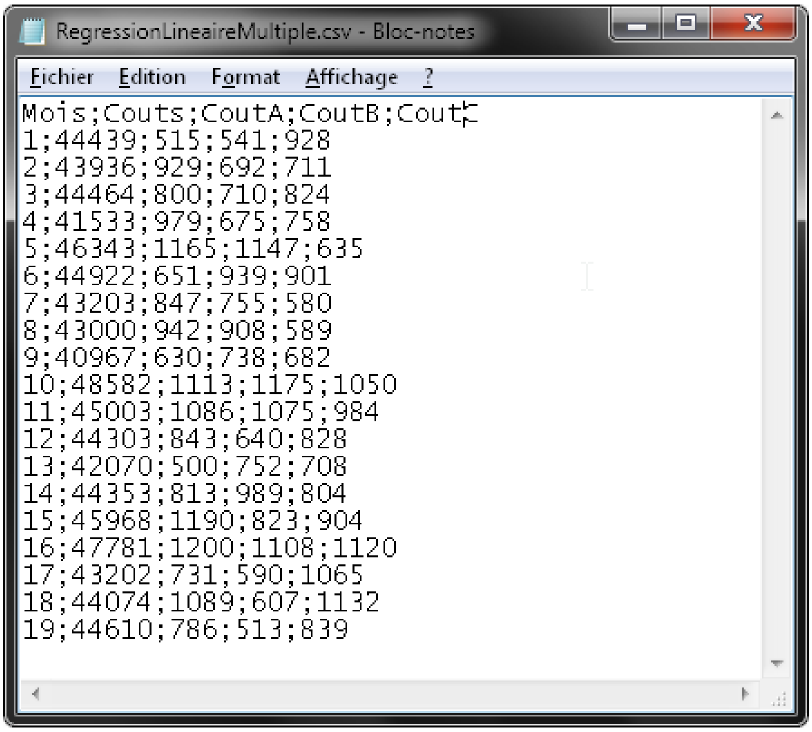
Si nous utilisons la fonction **calc.relimp( )** du package spécialisé **relimpo** (relative importance):



Nous retrouvons le 64.54% par contre pour les coûts les importances relatives sont en importance inverse. Cela est dû au modèle empirique utilisé (car il en existe plus d'une dizaine…).

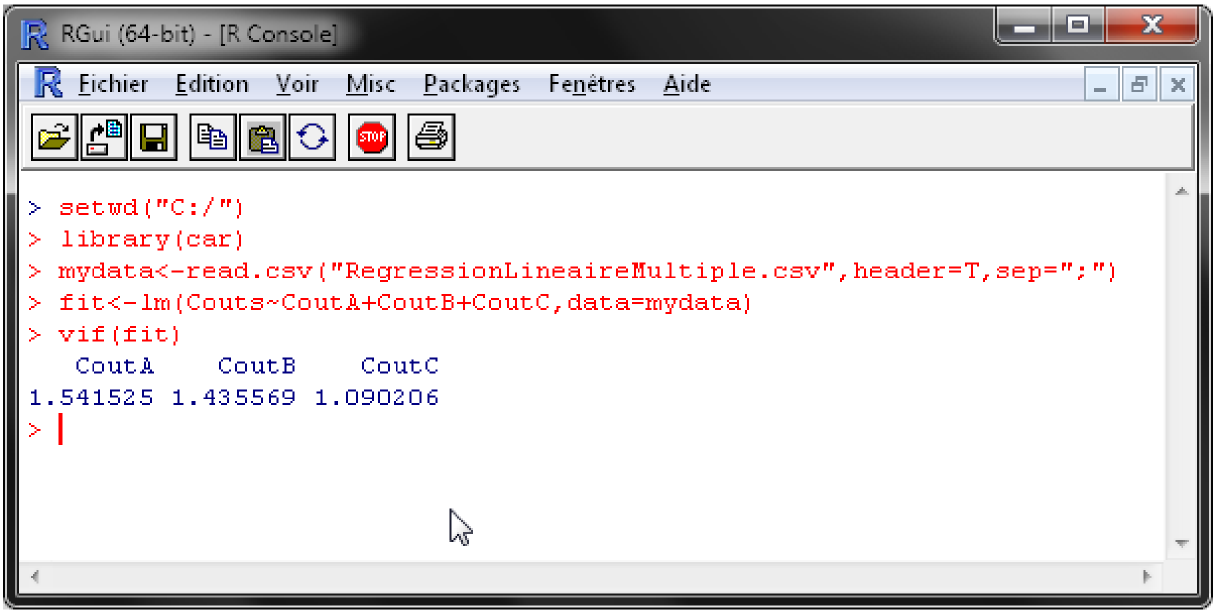
Facteur d'Inflation de la Variance (VIF)

Encore une fois nous allons contrôler les calculs faits à la main dans le cours théorique et obtenus aussi avec Microsoft Excel en utilisant toujours les mêmes données:




```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

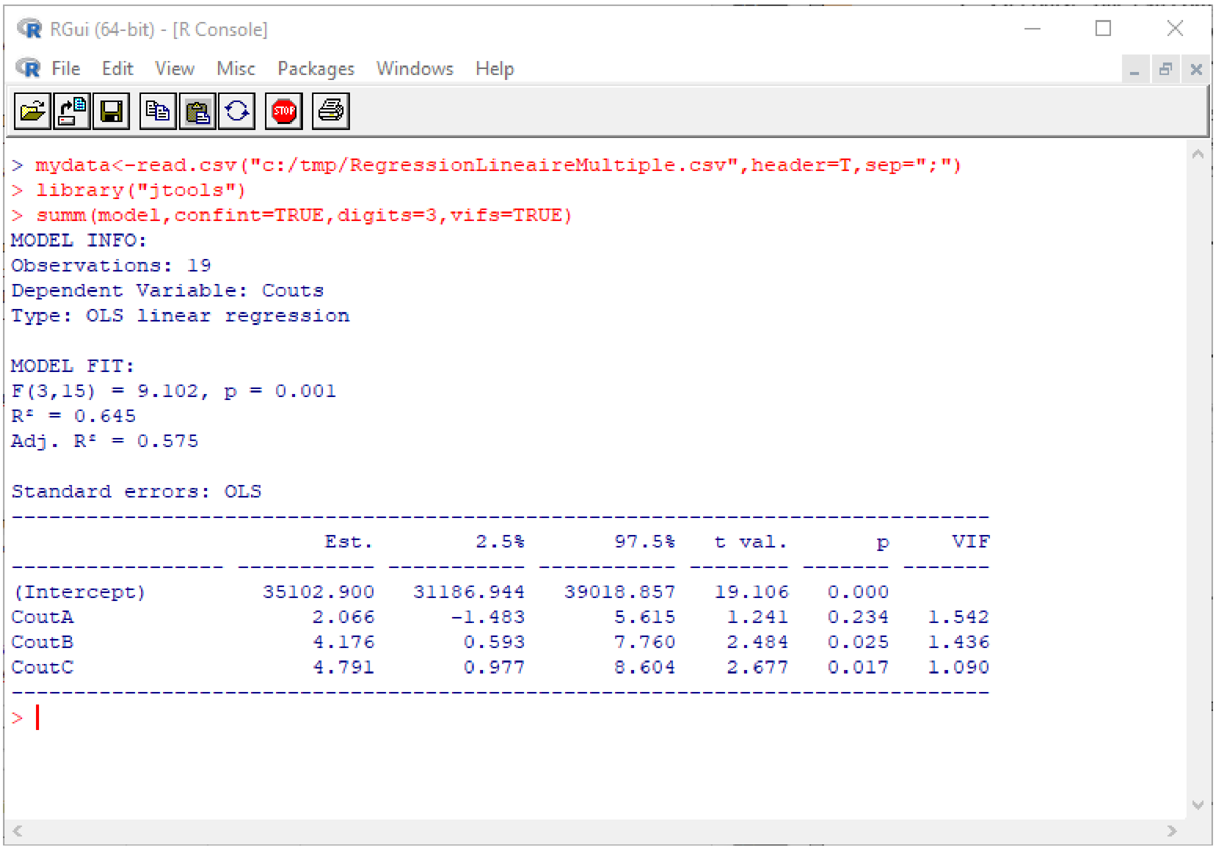
Nous utilisons pour cela la commande **vif( )** du package **car**:



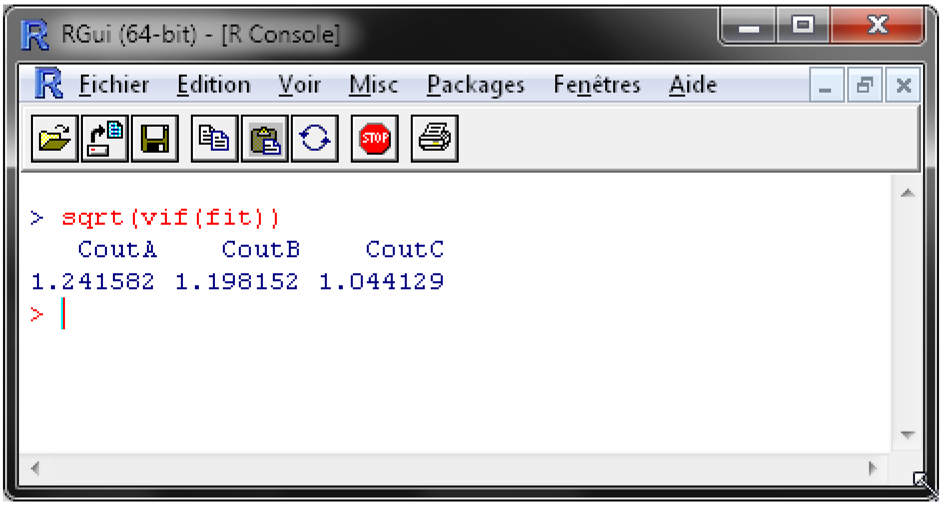

```
> setwd("C:/")
> library(car)
> mydata<-read.csv("RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> vif(fit)
   CoutA    CoutB    CoutC
1.541525 1.435569 1.090206
>
```

Nous retrouvons effectivement les valeurs calculées manuellement en cours et donc les mêmes conclusions.

Notez que nou savions obtenus les mêmes résultats plus haut avec le package jtools:

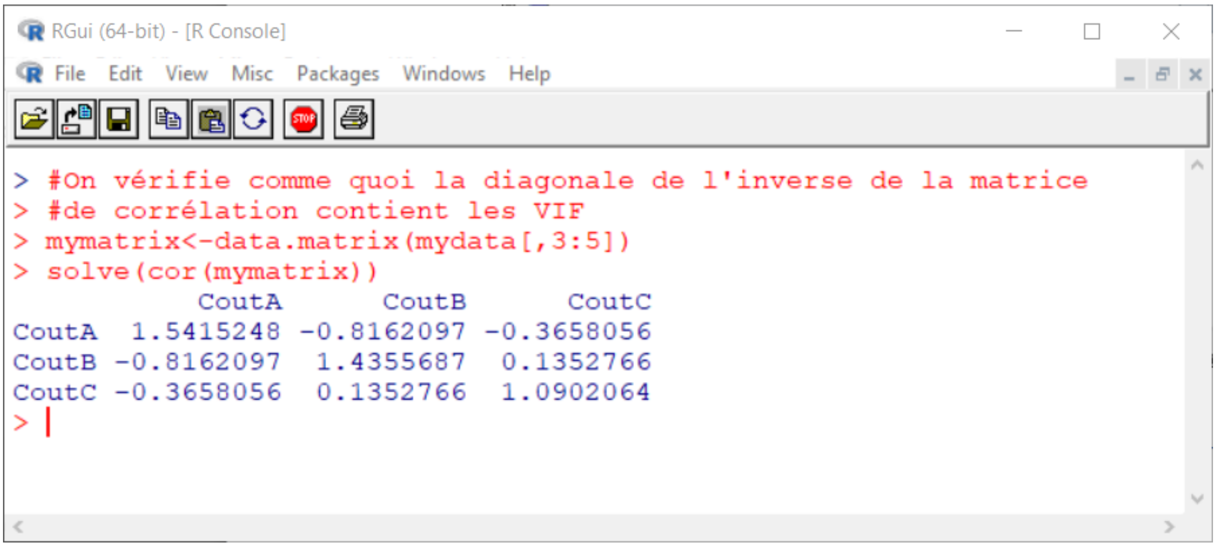


Nous pouvons faire rapidement le calcul que Minitab ne fait pas et qui consiste dans la règle empirique que la racine carrée du VIF soit plus grande que 2 alors il y a multicolinéarité:



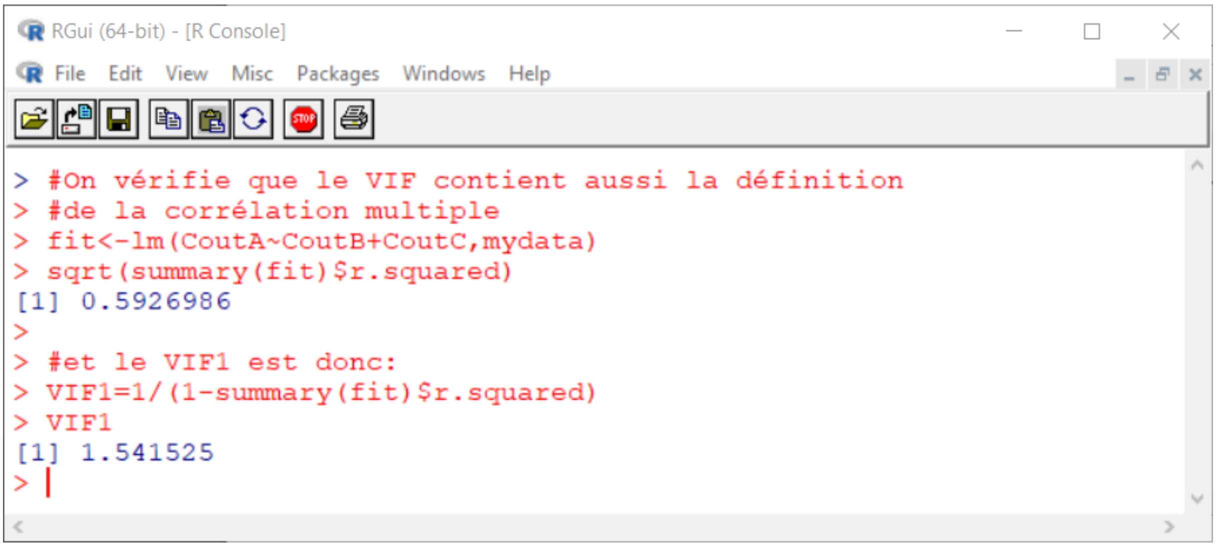
Maintenant vérifions que nous pouvons retrouver (par exemple!) la première valeur de VIF précédent de différentes façon selon ce que nous avons démontré dans le cours théorique.

D'abord avec le cas le plus simple qui consiste à retrouver les VIF dans l'inverse de la matrice de corrélation:

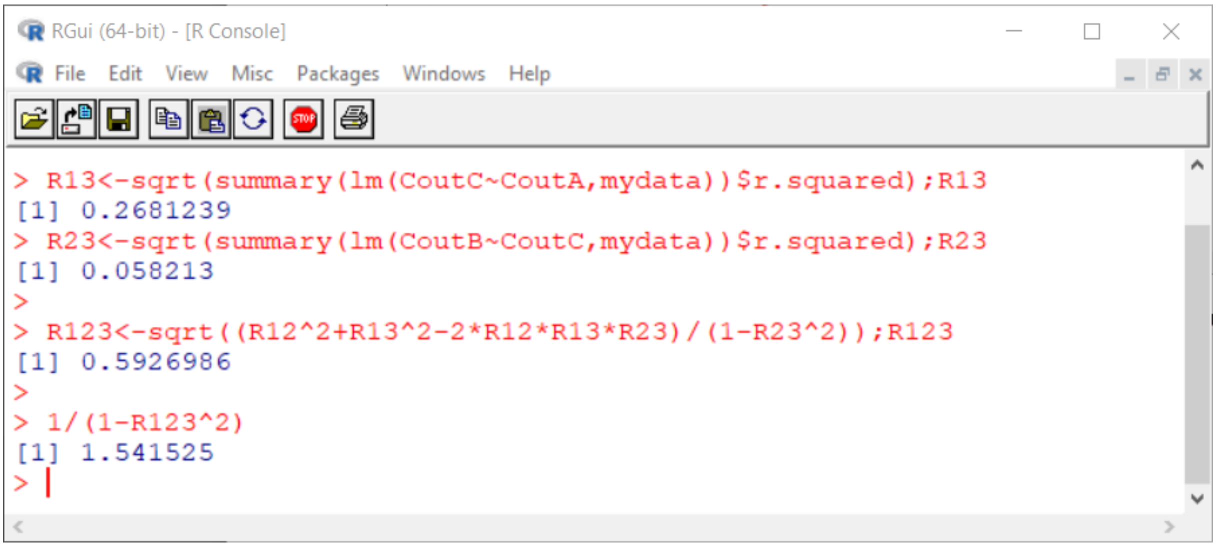


Ce qui est bien vérifié!

Maintenant, souvenons-nous dans le cours théorique que nous avons vu que ce qui définit le coefficient de corrélation multiple est dans la définition du VIF lui-même et particulièrement dans la corrélation des variables entre elles:



Et nous retombons bien sur le résultat attendu! Nous pouvons aller encore plus loin en calculant le coefficient de corrélation multiple de façon explicite et à nouveau le VIF:



Bref voilà une application numérique qui "démontre" bien les différentes définitions et concepts vus dans le cours théorique.

Erreur d'ajustement (lack-of-fit)

Dans le cours théorique nous avons démontré une astuce mathématique qui permet dans un certain sens de mettre en évidence le manque d'ajustement d'un modèle linéaire (avec ou sans interactions) qui consistait à décomposer l'erreur des résidus en erreur pure et erreur d'ajustement. Même si cette décomposition s'applique aussi à la régression univariée, il nous a semblé plus judicieux de présenter ce sujet ici dans la section concernant la régression multivariée (rappelons que la seule condition est d'avoir de multiples mesures pour une abscisse donnée!).

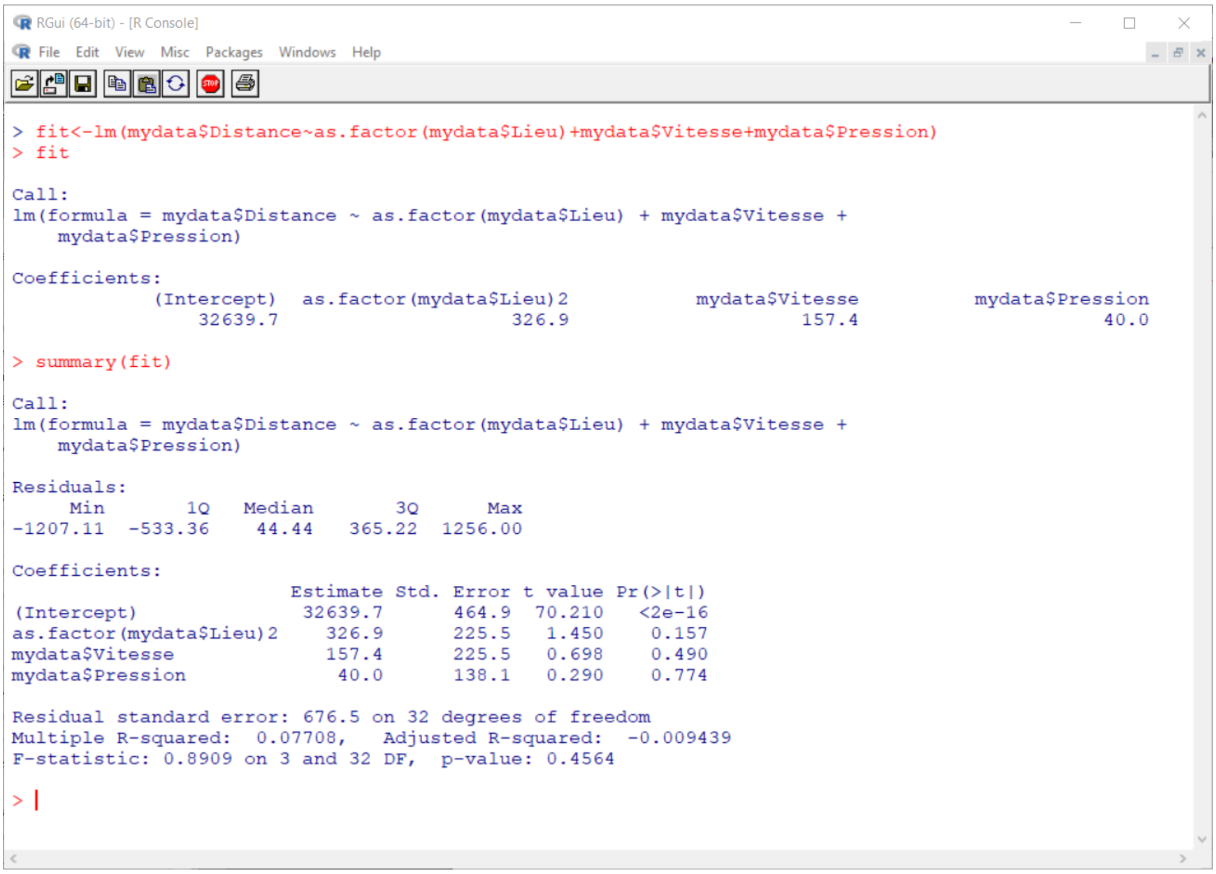
Le but ici va être de voir si nous sommes dans un premier temps capable de reproduire les calculs faits dans Microsoft Excel (à la main) et si nous retombons sur les mêmes résultats que Minitab.

Pour cela, nous partons du jeu de données suivant qui pour rappel est un plan d'expérience d'une série de 12 mesures répétées 3 fois (qui nous est bien connu dans le cours théorique!):

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> mydata<-read.csv("c:/tmp/LackOfFit.csv",header=T,sep=";")
> mydata
   Mesure Lieu Vitesse Pression Distance
1       1    1       1        1    32700
2       2    1       1        2    33430
3       3    1       1        3    31710
4       4    1       2        1    32680
5       5    1       2        2    34070
6       6    1       2        3    33220
7       7    2       1        1    33180
8       8    2       1        2    34430
9       9    2       1        3    33570
10     10    2       2        1    33270
11     11    2       2        2    33440
12     12    2       2        3    32840
13     13    1       1        1    32750
14     14    1       1        2    33360
15     15    1       1        3    32100
16     16    1       2        1    32270
17     17    1       2        2    33100
18     18    1       2        3    33700
19     19    2       1        1    32160
20     20    2       1        2    34280
21     21    2       1        3    33300
22     22    2       2        1    33080
23     23    2       2        2    33570
24     24    2       2        3    33210
25     25    1       1        1    32960
26     26    1       1        2    32910
27     27    1       1        3    32220
28     28    1       2        1    33130
29     29    1       2        2    33610
30     30    1       2        3    33285
31     31    2       1        1    32640
32     32    2       1        2    34460
33     33    2       1        3    32570
34     34    2       2        1    32415
35     35    2       2        2    34204
36     36    2       2        3    32470
```

Faisons d'abord la régression linéaire classique sans interactions comme effectuée dans le cours théorique et l'ANOVA de la régression classique avec les fonctions **lm( )** et **summary( )** :



```
> fit<-lm(mydata$Distance~as.factor(mydata$Lieu)+mydata$Vitesse+mydata$Pression)
> fit

Call:
lm(formula = mydata$Distance ~ as.factor(mydata$Lieu) + mydata$Vitesse +
    mydata$Pression)

Coefficients:
           (Intercept)  as.factor(mydata$Lieu)2           mydata$Vitesse          mydata$Pression
               32639.7                    326.9                    157.4                     40.0

> summary(fit)

Call:
lm(formula = mydata$Distance ~ as.factor(mydata$Lieu) + mydata$Vitesse +
    mydata$Pression)

Residuals:
     Min       1Q   Median       3Q      Max
-1207.11  -533.36    44.44   365.22  1256.00

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)              32639.7      464.9  70.210   <2e-16
as.factor(mydata$Lieu)2    326.9      225.5   1.450    0.157
mydata$Vitesse             157.4      225.5   0.698    0.490
mydata$Pression             40.0      138.1   0.290    0.774

Residual standard error: 676.5 on 32 degrees of freedom
Multiple R-squared:  0.07708,   Adjusted R-squared:  -0.009439
F-statistic: 0.8909 on 3 and 32 DF,  p-value: 0.4564

>
```

Nous retrouvons donc les mêmes résultats qu'avec Microsoft Excel et Minitab. Enfin, faisons l'analyse du manque d'ajustement qui nous intéresse avec la fonction **pureErrorAnova( )** du package **alr3**:

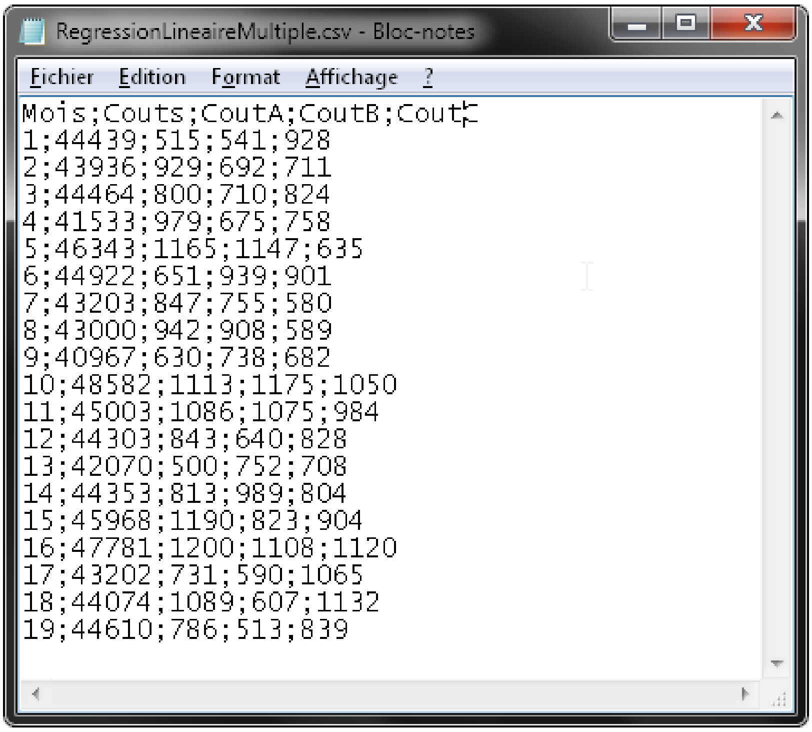
```
> library("alr3")
> pureErrorAnova(fit)
Analysis of Variance Table

Response: mydata$Distance
                        Df   Sum Sq Mean Sq F value   Pr(>F)
as.factor(mydata$Lieu)   1   961707  961707  6.7848  0.01554
mydata$Vitesse           1   223099  223099  1.5740  0.22171
mydata$Pression          1    38400   38400  0.2709  0.60749
Residuals               32 14645181  457662
 Lack of fit             8 11243331 1405416  9.9152 5.45e-06
 Pure Error             24  3401851  141744
>
```

Nous retrouvons bien tous les résultats attendus avec les mêmes conclusions!

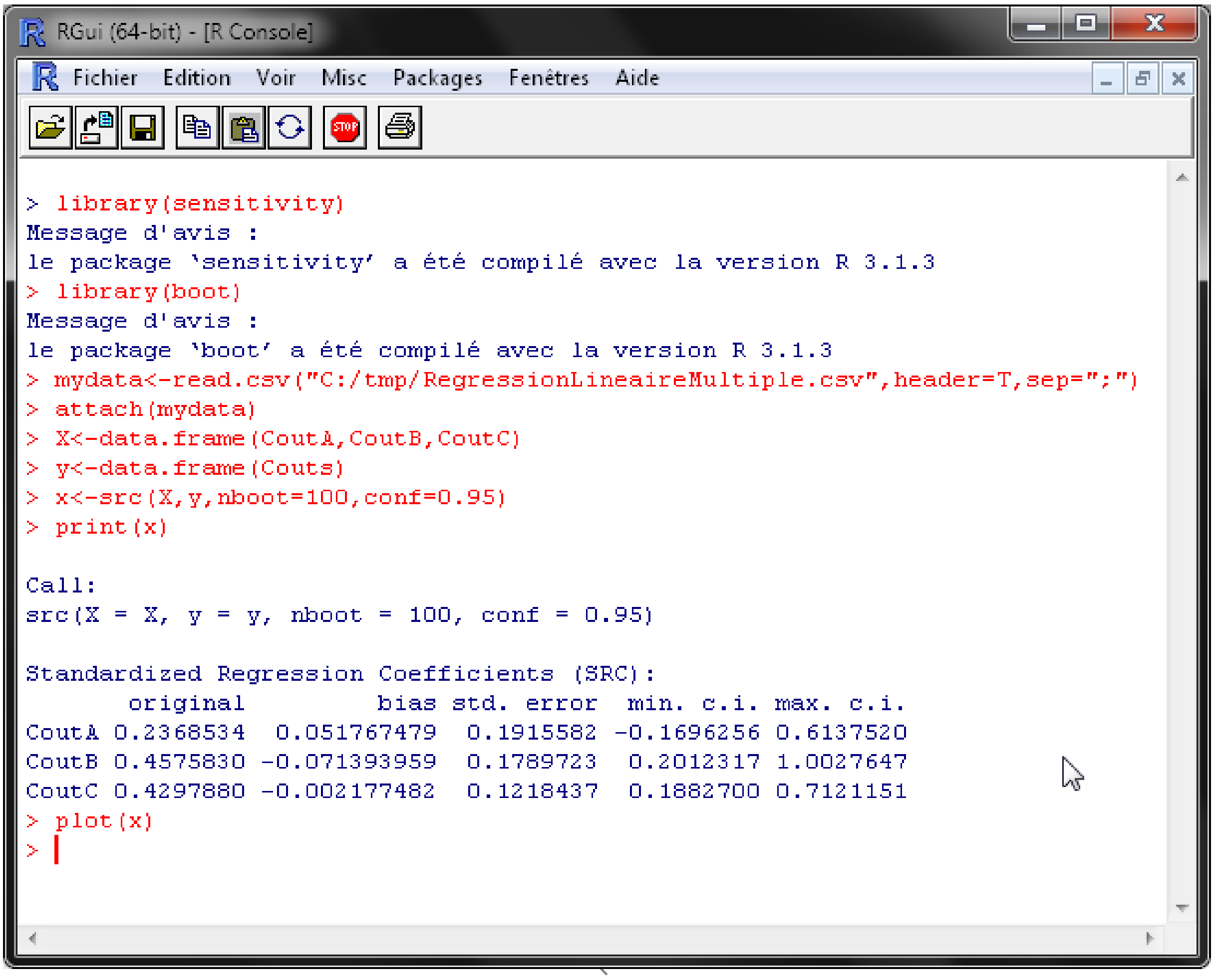
Analyse de la sensibilité par coefficients de régression standardisés

Encore une fois nous allons contrôler les calculs faits à la main dans le cours théorique et obtenus aussi avec Microsoft Excel en utilisant toujours les mêmes données:




```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

Pour cela nous allons utiliser la fonction **src( )** du package **sensitivity** qui nous donnera les coefficients de régression standardisés ainsi que le package **boost** pour faire du boosting dans le but d'obtenir les intervalles de confiance:



Et le graphique associé:

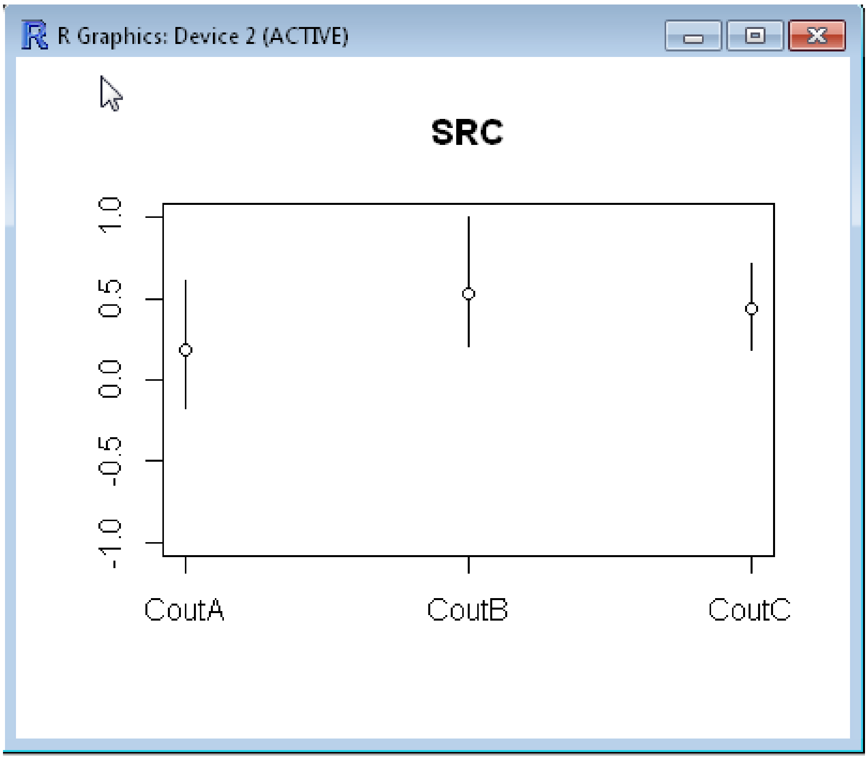


Figure 248 Graphique de diagnostique d'analyse de la sensibilité

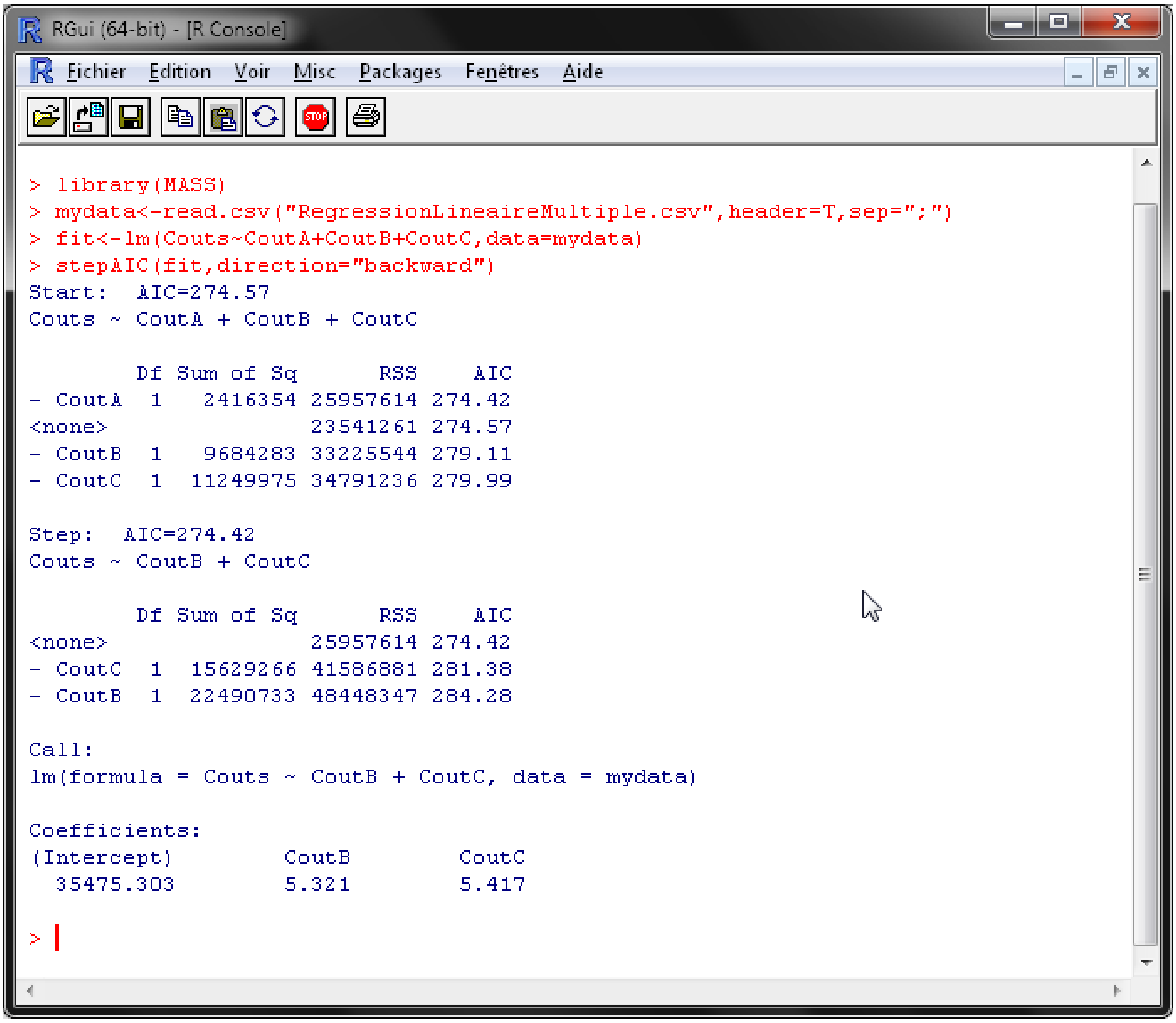
"Régression" pas à pas ascendante ou descendante

Ici nous allons encore une fois vérifier que nous retrouvons les mêmes résultats que dans le cours Minitab et R et ainsi qu'avec les calculs faits à la main suite à la présentation de cette technique empirique dans le cours théorique.

Remarque: Rappelons que la "régression pas à pas" n'est pas une régression mais juste une méthode d'analyse!

Première méthode (descendante package MASS)

Nous utilisons pour cela la commande **stepAIC( )** du package **MASS**:



```
> library(MASS)
> mydata<-read.csv("RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> stepAIC(fit,direction="backward")
Start:  AIC=274.57
Couts ~ CoutA + CoutB + CoutC

        Df Sum of Sq      RSS    AIC
- CoutA  1   2416354 25957614 274.42
<none>               23541261 274.57
- CoutB  1   9684283 33225544 279.11
- CoutC  1  11249975 34791236 279.99

Step:  AIC=274.42
Couts ~ CoutB + CoutC

        Df Sum of Sq      RSS    AIC
<none>               25957614 274.42
- CoutC  1  15629266 41586881 281.38
- CoutB  1  22490733 48448347 284.28

Call:
lm(formula = Couts ~ CoutB + CoutC, data = mydata)

Coefficients:
(Intercept)        CoutB        CoutC
  35475.303        5.321        5.417

>
```

Nous voyons que nous avons moins d'options que dans Minitab ce qui est bien dommage car les options proposées par ce dernier sont très pertinentes. Cependant dans R, nous avons le AIC (Akaike Information Criterion) et que nous n'avons pas à ce jour dans Minitab ce qui est bien dommage.

Nous voyons cependant que le meilleur modèle résultant est le même car R s'arrête dès que le modèle se dégrade (le but étant de minimiser l'AIC d'où le nom de la commande...).

Petit interlude: l'AIC (1973) est une mesure empirique d'une pseudo-distance entropique entre la distribution de probabilités vraie et celle estimée et appelé divergence de Kullback-Leibler:

$$D_S(p,q)=\sum p(x)\log\frac{p(x)}{q(x)}=\sum p(x)\log p(x)-\sum p(x)\log q(x)$$

et nous remarquons de suite que si $p(x)=q(x)$ la distance entropique est alors nulle. Akaike s'est concentré sur le deuxième terme uniquement car il contient la distribution supposée:

$$\sum p(x)\log q(x)$$

Ce dernier est par construction toujours négatif puisque nous prenons le log d'une valeur toujours inférieure à l'unité! Ensuite après quelques manipulations mathématiques discutables (probabilités conditionnelles, approximations de Taylor, convergence en loi, etc.) nous arrivons à:

$$AIC(k)=-2L\left(\hat{\vec{\theta}}\right)+2k$$

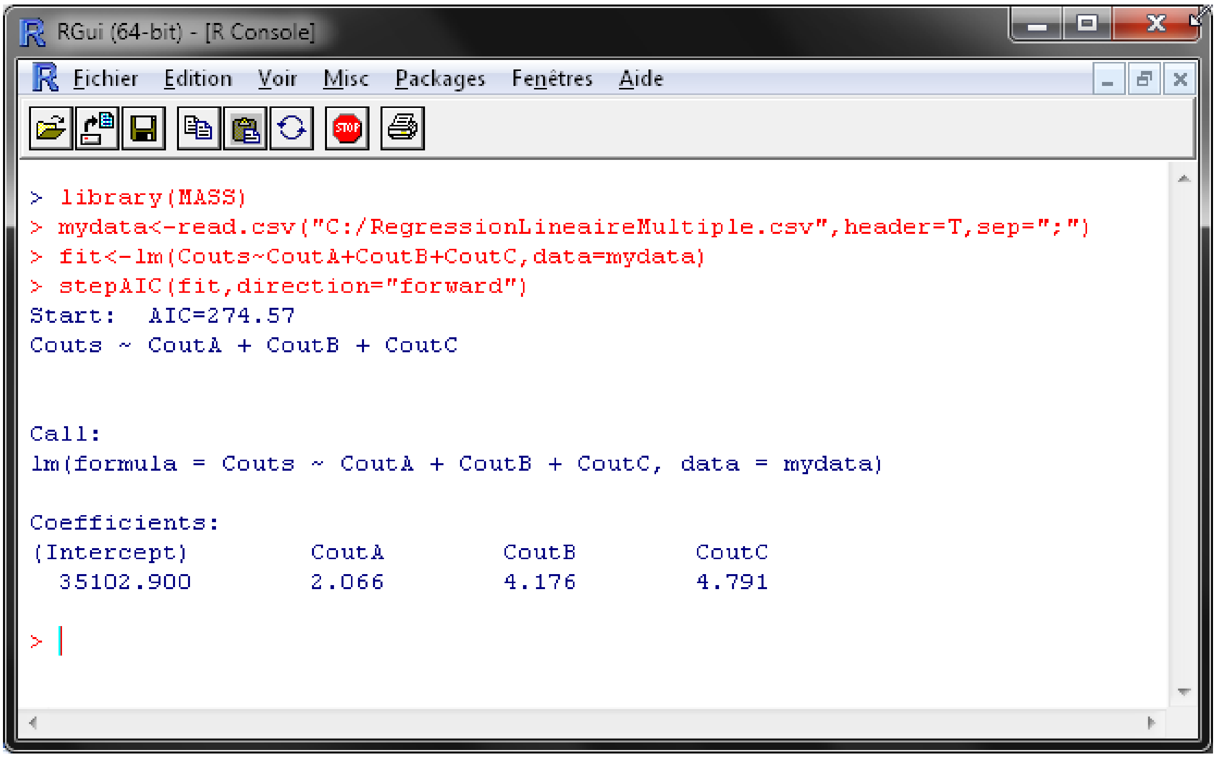
qui ne marche donc que pour des gros échantillons vu les hypothèses des développements sous-jacents.

Dans tous les cas à sélection de modèles qui utilisent le même nombre de mesures k, il faut donc privilégier le modèle qui a la plus petite valeur (donc si c'est négatif, plus c'est négatif mieux c'est). Rappelons que le maximum de vraisemblance L décrit par construction d'autant mieux un modèle que ce dernier est grand! Dans le cas de modèles qui n'utilisent pas le même nombre de points de mesure k la sélection est à ma connaissance plus délicate à cause de la présence du $+2k$…

Depuis il y a eu d'autres modèles comme le TIC (Takuchi's Information Criterion) développé en 1976 et le BIC (Baysian Information Criterion).

Première méthode (ascendante package MASS)

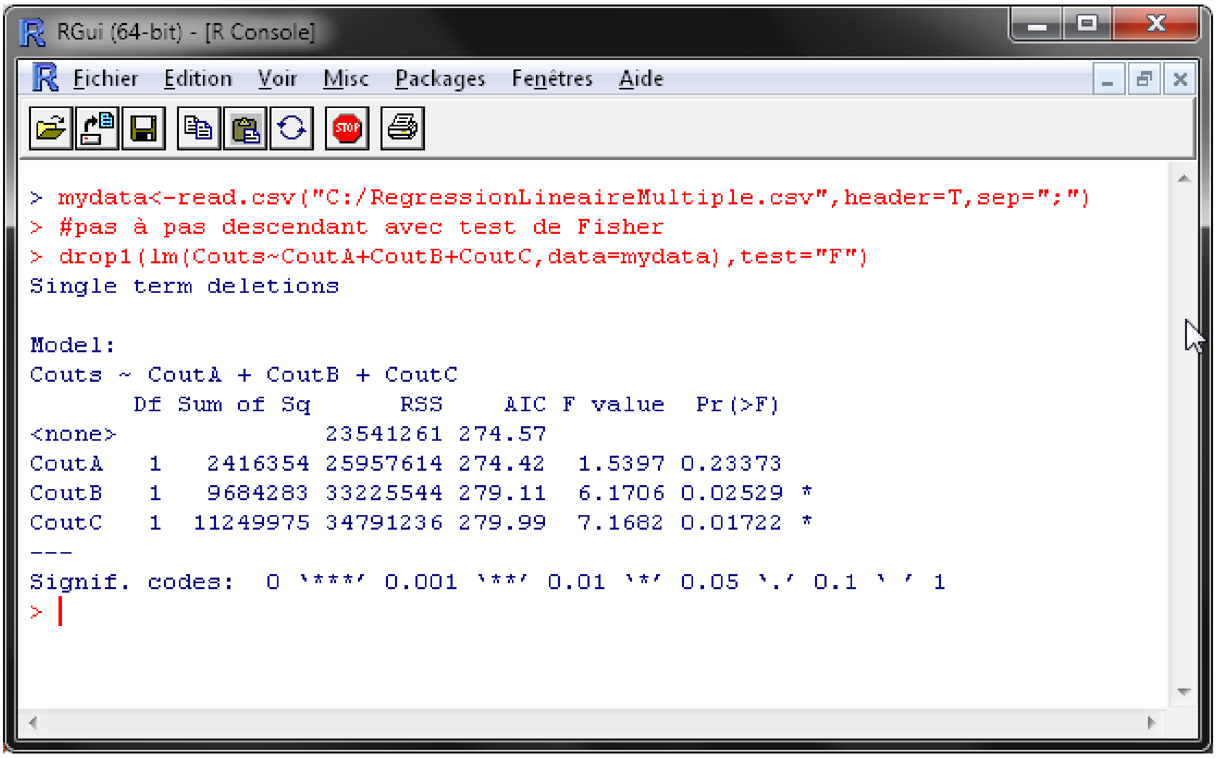
Même commande et même principe qu'avant:



Nous retrouvons donc bien le même résultat que dans Minitab et qu'à la main (ou que dans MS Excel). Le seul point qui est un peu dommage c'est que R s'arrête à la première étape. Cela aurait été bien d'en montrer plusieurs comme le fait Minitab.

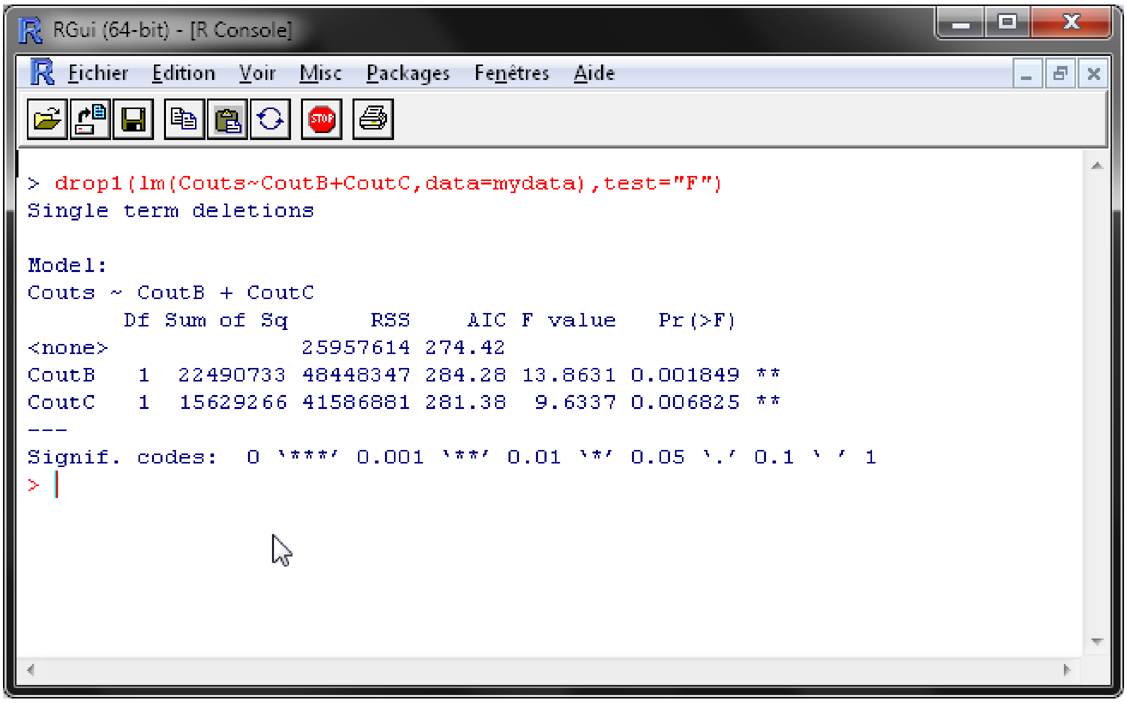
Deuxième méthode (descendante sans package)

Nous allons utiliser la commande **drop1( )** native de R:



Donc ici à l'aide des p-value on voit que c'est *CoutC* qui est le plus significatif, ensuite Cout*B* et ainsi de suite. Nous voyons aussi en face l'AIC ce qui est toujours utile.

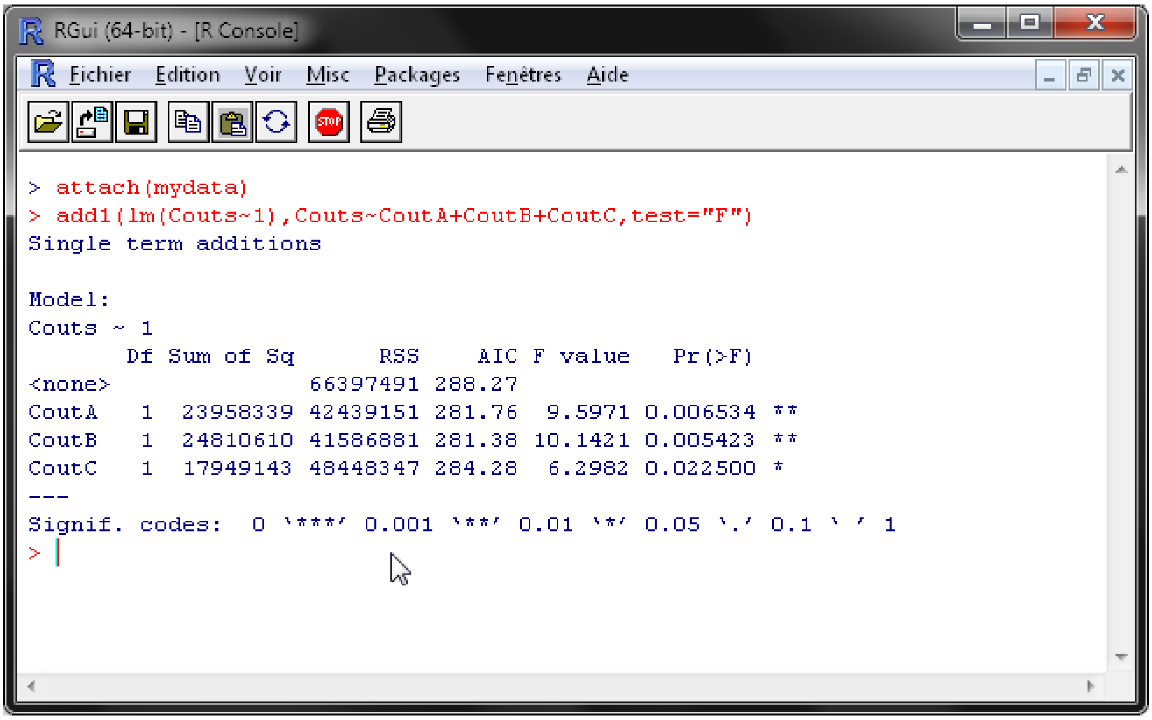
On peut ensuite s'amuser à enlever la variable la moins significative:



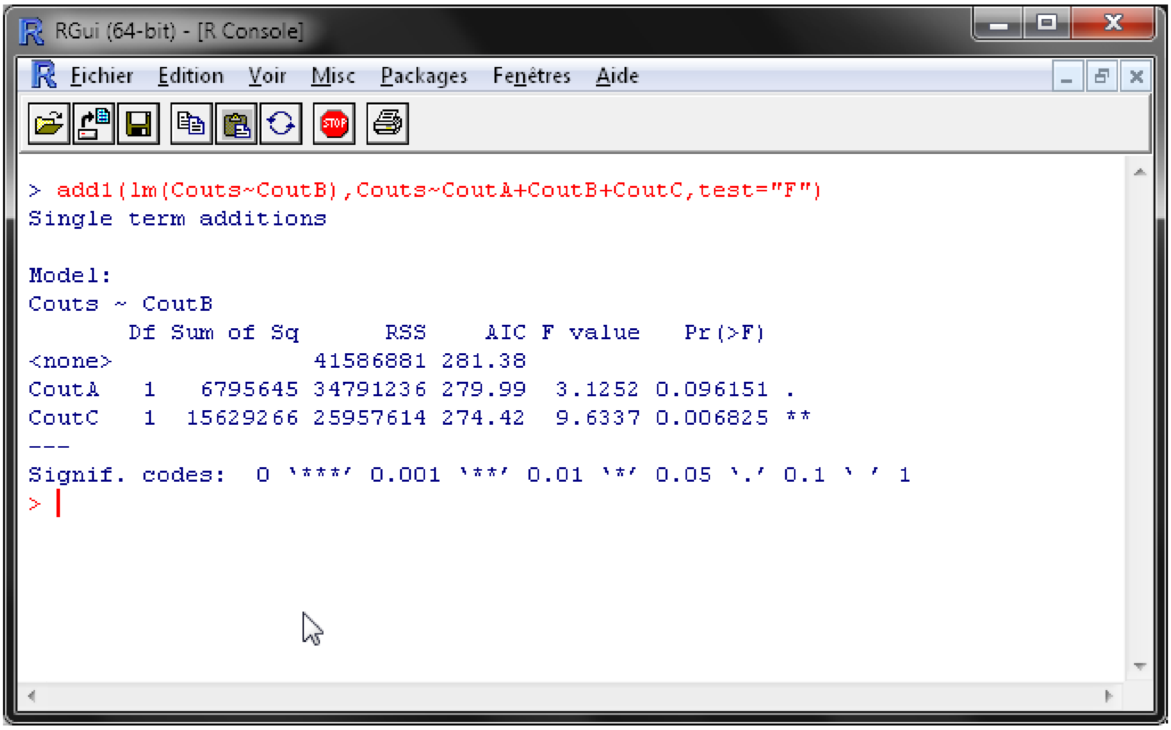
et ainsi de suite...

Deuxième méthode (ascendante sans package)

Nous allons utiliser la commande **add1( )** native de R:



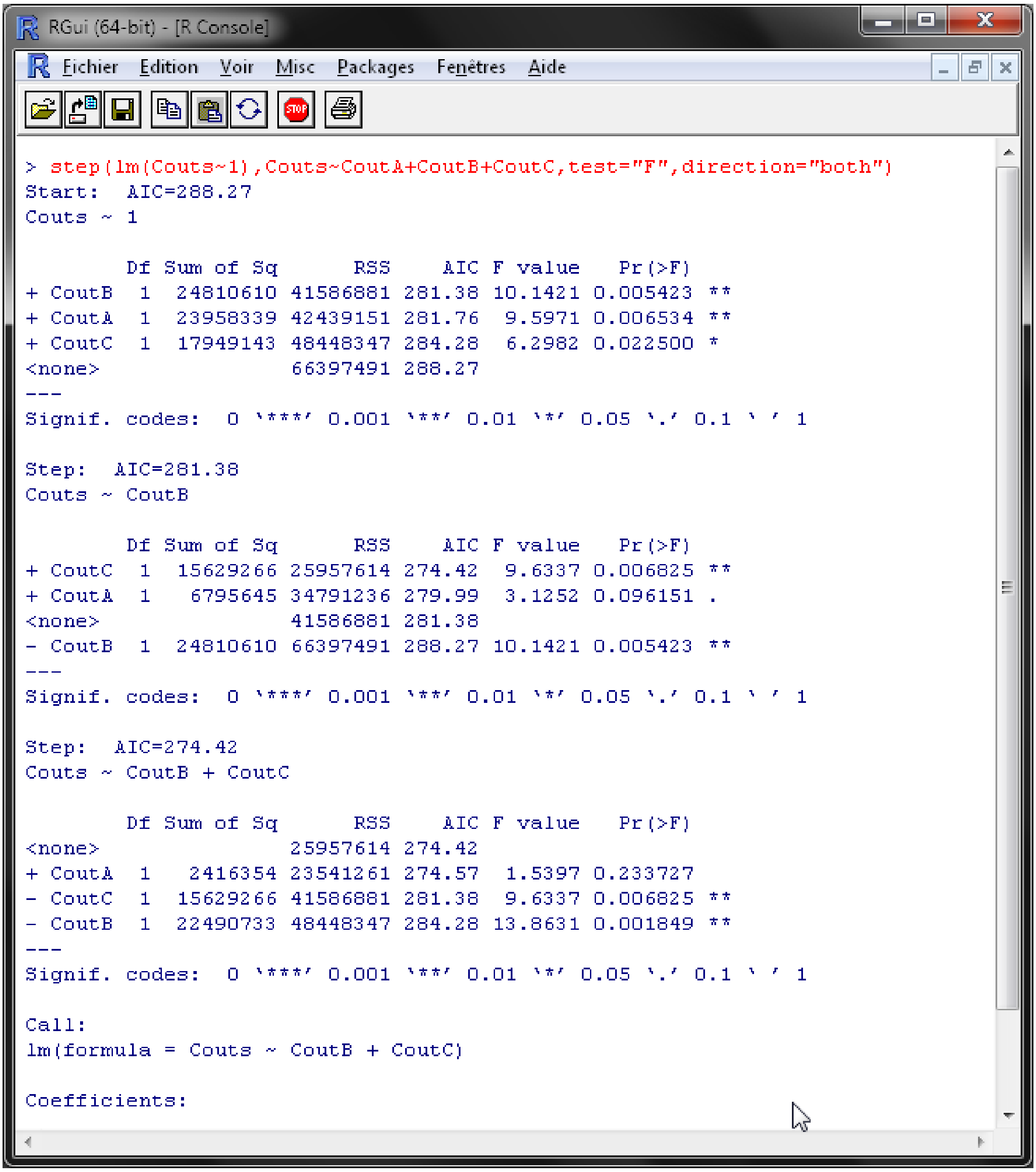
Là nous voyons que c'est *CoutB* qui est le plus significatif. C'est donc toujours conforme à Minitab et aux calculs faits à la main. Nous pouvons donc ensuite, nous amuser à ajouter cette variable au modèle:



et ainsi de suite...

Troisième méthode (ascendante sans package)

Nous allons utiliser la commande **step( )** native de R qui permet d'obtenir automatiquement les séquences de la deuxième méthode vue précédemment:



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> step(lm(Couts~1),Couts~CoutA+CoutB+CoutC,test="F",direction="both")
Start:  AIC=288.27
Couts ~ 1

        Df Sum of Sq      RSS    AIC F value   Pr(>F)
+ CoutB  1  24810610 41586881 281.38 10.1421 0.005423 **
+ CoutA  1  23958339 42439151 281.76  9.5971 0.006534 **
+ CoutC  1  17949143 48448347 284.28  6.2982 0.022500 *
<none>                66397491 288.27
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step:  AIC=281.38
Couts ~ CoutB

        Df Sum of Sq      RSS    AIC F value   Pr(>F)
+ CoutC  1  15629266 25957614 274.42  9.6337 0.006825 **
+ CoutA  1   6795645 34791236 279.99  3.1252 0.096151 .
<none>                41586881 281.38
- CoutB  1  24810610 66397491 288.27 10.1421 0.005423 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step:  AIC=274.42
Couts ~ CoutB + CoutC

        Df Sum of Sq      RSS    AIC F value   Pr(>F)
<none>                25957614 274.42
+ CoutA  1   2416354 23541261 274.57  1.5397 0.233727
- CoutC  1  15629266 41586881 281.38  9.6337 0.006825 **
- CoutB  1  22490733 48448347 284.28 13.8631 0.001849 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Call:
lm(formula = Couts ~ CoutB + CoutC)

Coefficients:
```

Quatrième méthode (meilleur sous ensemble sur R^2 ou C_p de Mallows avec package leaps)

Nous allons ici utiliser la commande **leaps( )** du package **leaps**. Cela nous donne en nous basant sur le coefficient de corrélation ajusté comme critère à maximiser:

```
> library(leaps)
> mydata<-read.csv("C:/RegressionLineaireMultiple.csv",header=T,sep=";")
> #on va forcer l'ordonnée à l'origine comme étant nulle par la même occasion
> fit<-model.matrix(lm(Couts~-1+CoutA+CoutB+CoutC,data=mydata))
> leaps(fit,mydata$Couts,method="adjr2")
$which
      1     2     3
1 FALSE  TRUE FALSE
1  TRUE FALSE FALSE
1 FALSE FALSE  TRUE
2 FALSE  TRUE  TRUE
2  TRUE FALSE  TRUE
2  TRUE  TRUE FALSE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 2 2 3 3 3 4

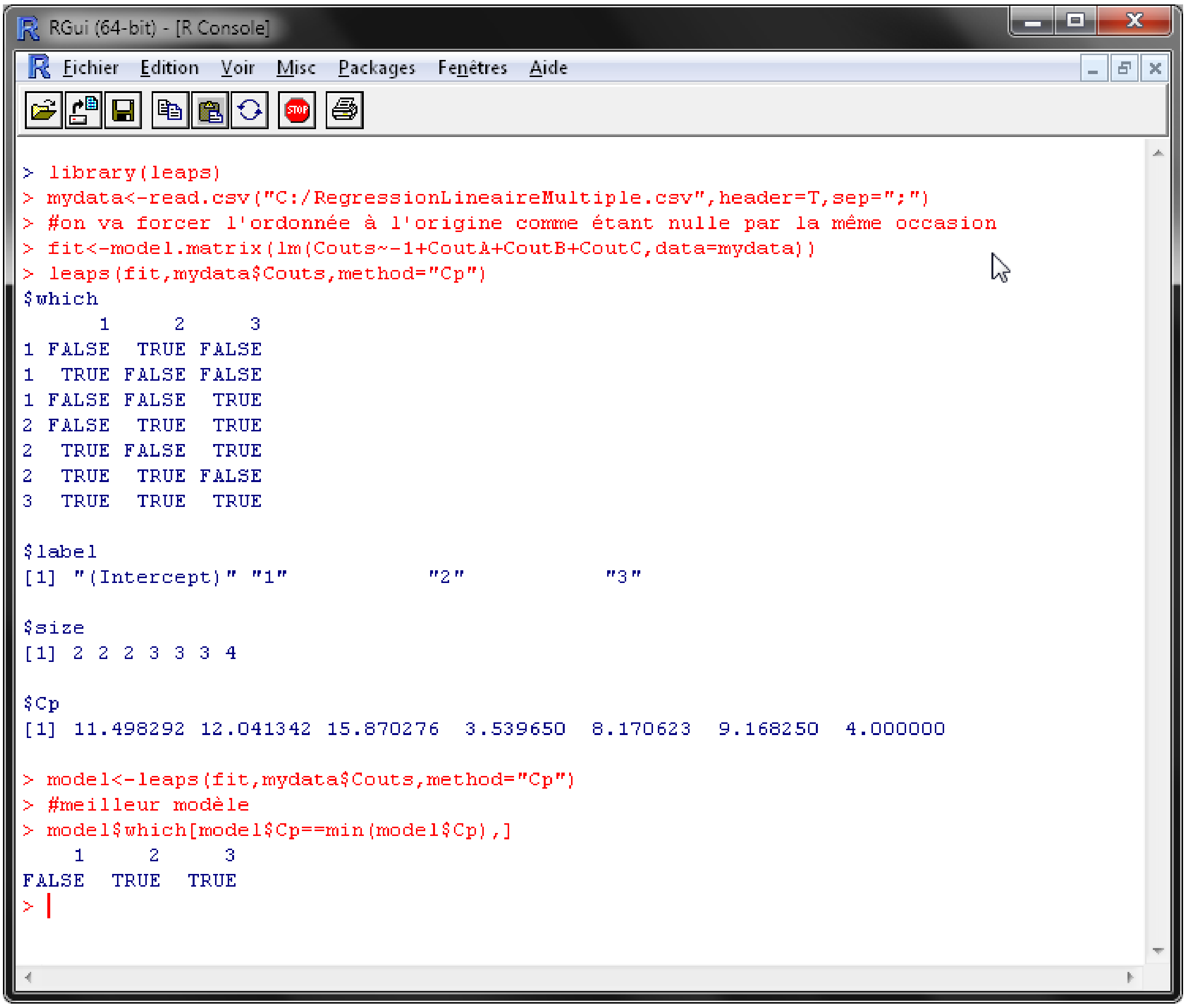
$adjr2
[1] 0.3368248 0.3232339 0.2274068 0.5601895 0.4370459 0.4105178 0.5745394

> model<-leaps(fit,mydata$Couts,method="adjr2")
> #meilleur modèle
> model$which[model$adjr2==max(model$adjr2),]
   1    2    3
TRUE TRUE TRUE
>
```

Donc le meilleur modèle au sens du coefficient de corrélation ajusté est celui qui contient les trois variables explicatives.

Bien que la conclusion soit la même que dans Minitab, le lecteur pour test que si nous ne forçons pas l'ordonnée à l'origine comme étant nulle, la commande **leaps( )** n'arrive pas à s'exécuter (contrairement à Minitab où nous pouvons forcer ou non l'ordonnée à l'origine).

Sinon en nous basant sur le critère de Mallows:



Comme quoi... comme on dit toujours: il n'y a pas de bons modèles. Il n'y a que des modèles meilleurs que les autres!

Quatrième méthode (meilleur sous ensemble sur R^2, C_p de Mallows, AIC et BIC avec package biology)

En utilisant le package **biology** et sa fonction **Model.selection( )** mais ne fonctionnant que dans les versions antérieures à R 2.9.0 et pour lequel il faut passer par l'installation du fichier zip biology.zip:



Nous obtenons:




```
> mydata<-read.csv("C:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~-1+CoutA+CoutB+CoutC,data=mydata)
> Model.selection(fit)


Model selection
Response: Couts
                     r.squared  Adj.r.sq P        Cp      AIC     AICc  deltaAIC  wAIC      BIC Select
1. CoutA             0.3608320 0.3232339 1 -13.86159 281.7637 284.6209 7.3405203 0.012 283.6526
2. CoutB             0.3736679 0.3368248 1 -13.88445 281.3783 284.2354 6.9550755 0.014 283.2671      *
3. CoutC             0.2703286 0.2274068 1 -13.70039 284.2798 287.1370 9.8566315 0.003 286.1687
4. CoutA+CoutB       0.4760158 0.4105178 2 -12.06674 279.9883 282.8454 5.5651101 0.029 282.8216
5. CoutA+CoutC       0.4995964 0.4370459 2 -12.10874 279.1134 281.9706 4.6902255 0.045 281.9467
6. CoutB+CoutC       0.6090573 0.5601895 2 -12.30370 274.4232 277.2803 0.0000000 0.465 277.2565      *
7. CoutA+CoutB+CoutC 0.6454495 0.5745394 3 -10.36852 274.5667 277.4238 0.1435037 0.432 278.3445


Model averaging
Response: Couts
      Estimate Unconditional_SE  Lower95CI Upper95CI
CoutA 1.246843         1.091428 -0.8923553  3.386041
CoutB 4.463543         1.602168  1.3232940  7.603792
CoutC 4.799863         1.736051  1.3972026  8.202523
```

Analyse perturbationnelle

Bon c'est un outil qui fait un peu doublon avec les intervalles de toléreance et de prédiction de la régression. Mais certains préfèrent cette approche. Alors voyons de quoi il s'agit et pour cela faisons les préparatifs habituels:

```
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv", header=T,sep=";")
> (model<-lm(Couts~-1+CoutA+CoutB+CoutC,data=mydata))

Call:
lm(formula = Couts ~ -1 + CoutA + CoutB + CoutC, data = mydata)

Coefficients:
 CoutA   CoutB   CoutC
 7.263  16.692  27.906

> summary(model)

Call:
lm(formula = Couts ~ -1 + CoutA + CoutB + CoutC, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max
-10684  -1658   2002   5406   8263

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
CoutA    7.263      8.006   0.907 0.377770
CoutB   16.692      7.546   2.212 0.041865
CoutC   27.906      6.425   4.343 0.000503

Residual standard error: 6106 on 16 degrees of freedom
Multiple R-squared:  0.9841,    Adjusted R-squared:  0.9811
F-statistic: 329.5 on 3 and 16 DF,  p-value: 1.373e-14

> 
```

Ensuite, nous utilisons la fonction **perturb( )** du package **perturb**:

```
> #perturbation de 1 sigma sur les trois variables
> library("perturb")
> attach(mydata) #absolument indispensable!!!
> p<-perturb(model,pvars=c("CoutA","CoutB","CoutC"),prange=c(1,1,1),niter=200)
> summary(p)
$formula
[1] "Couts ~ -1 + CoutA + CoutB + CoutC"

$pvars
[1] "CoutA" "CoutB" "CoutC"

$prange
[1] 1 1 1

$ptrans2
character(0)

$formula2
[1] "Couts ~ -1 + CoutA.1 + CoutB.1 + CoutC.1"

$distribution
[1] "normal"

$summ
           mean       s.d.      min      max
CoutA  7.256873 0.06061076  7.10110  7.40755
CoutB 16.698688 0.06064994 16.55945 16.88242
CoutC 27.905810 0.04410600 27.79697 28.03634

$dec.places
[1] 3

$full
[1] FALSE

$dots
[1] ""

attr(,"class")
[1] "summary.perturb"
>
```

De ces résultats nous pouvons ensuite faire de nombreuses choses diverses et variées. Le lecteur pourra consulter la documentation du package pour appliquer la fonction **perturb( )** sur des régressions comportant des variables catégorielles.

Recherche automatique de la meilleure régression polynomiale

Avec le package **glmulti** et la fonction du même nom nous pouvons dans la version **x86** de R demander à chercher le meilleur modèle quadratique avec interactions ou non. Pour cela, nous écrivons:



Exemple 260.: Régression linéaire uni ou multivariée non-négative (NNLS)

R 3.6.1

Dans la pratique, il arrive que nous souhaitons une régression linéaires par la méthode des moindres carrés dont aucun coefficient soit négatif (pour des raisons de cohérence avec la réalité).

Pour montrer comment faire un NNLS (non-negative Least-Square) dans R, nous allons reprendre le jeu de donnnées précédent et pour comparaison, nous allons refaire une simple régression linéaire forcée à l'origine:

```
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> mydata
   Mois Couts CoutA CoutB CoutC
1     1 44439   515   541   928
2     2 43936   929   692   711
3     3 44464   800   710   824
4     4 41533   979   675   758
5     5 46343  1165  1147   635
6     6 44922   651   939   901
7     7 43203   847   755   580
8     8 43000   942   908   589
9     9 40967   630   738   682
10   10 48582  1113  1175  1050
11   11 45003  1086  1075   984
12   12 44303   843   640   828
13   13 42070   500   752   708
14   14 44353   813   989   804
15   15 45968  1190   823   904
16   16 47781  1200  1108  1120
17   17 43202   731   590  1065
18   18 44074  1089   607  1132
19   19 44610   786   513   839
> fit<-lm(Couts~-1+CoutA+CoutB+CoutC,data=mydata)
> summary(fit)

Call:
lm(formula = Couts ~ -1 + CoutA + CoutB + CoutC, data = mydata)

Residuals:
   Min      1Q Median      3Q     Max
-10684  -1658   2002   5406    8263

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
CoutA    7.263      8.006   0.907 0.377770
CoutB   16.692      7.546   2.212 0.041865
CoutC   27.906      6.425   4.343 0.000503

Residual standard error: 6106 on 16 degrees of freedom
Multiple R-squared:  0.9841,    Adjusted R-squared:  0.9811
F-statistic: 329.5 on 3 and 16 DF,  p-value: 1.373e-14

> |
```

Maintenant voyons comment à l'aide de la fonction **nnls( )** du package **nnls** comment faire une telle régression:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("nnls")
> (x<-as.matrix(mydata[,3:5]))
      CoutA CoutB CoutC
 [1,]   515   541   928
 [2,]   929   692   711
 [3,]   800   710   824
 [4,]   979   675   758
 [5,]  1165  1147   635
 [6,]   651   939   901
 [7,]   847   755   580
 [8,]   942   908   589
 [9,]   630   738   682
[10,]  1113  1175  1050
[11,]  1086  1075   984
[12,]   843   640   828
[13,]   500   752   708
[14,]   813   989   804
[15,]  1190   823   904
[16,]  1200  1108  1120
[17,]   731   590  1065
[18,]  1089   607  1132
[19,]   786   513   839
> (y<-mydata[,2])
 [1] 44439 43936 44464 41533 46343 44922 43203 43000 40967 48582 45003 44303 42070 44353
[15] 45968 47781 43202 44074 44610
>
> (fitNNLS <- nnls(x, y))
Nonnegative least squares model
x estimates: 7.262658 16.69233 27.90647
residual sum-of-squares: 596467703
reason terminated: The solution has been computed sucessfully.
> fitNNLS$x
[1]  7.262658 16.692332 27.906474
>
```

Et heureusement nous voyons qu'au moins dans le cas présent que nous retombons sur la valeur de la régression classique (c'est le test minimum à faire pour voir que le résultat ait au moins un minimum de cohérence!).

Exemple 261.: Régression linéaire uni ou multivariée contrainte

R 3.6.1

Bon ici le but est de reproduire le calcul que nous avions fait avec MATLAB (par ailleurs la fonction R correspondante a exactement le même nom!). Cependant rappelons que souvent l'utilisation de la régression contrainte est dangereuse car elle peut faire croire qu'il y a qu'une seule solution au problème contraint alors qu'en réalité il y en a une infinité.

D'abord, reprenons la même capture d'écran que la documentation de MATLAB (cela aide aussi à comprendre ce qui va suivre):

$$\min_{x} \frac{1}{2} \|C \cdot x - d\|_2^2 \text{ such that } \begin{cases} A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{cases}$$

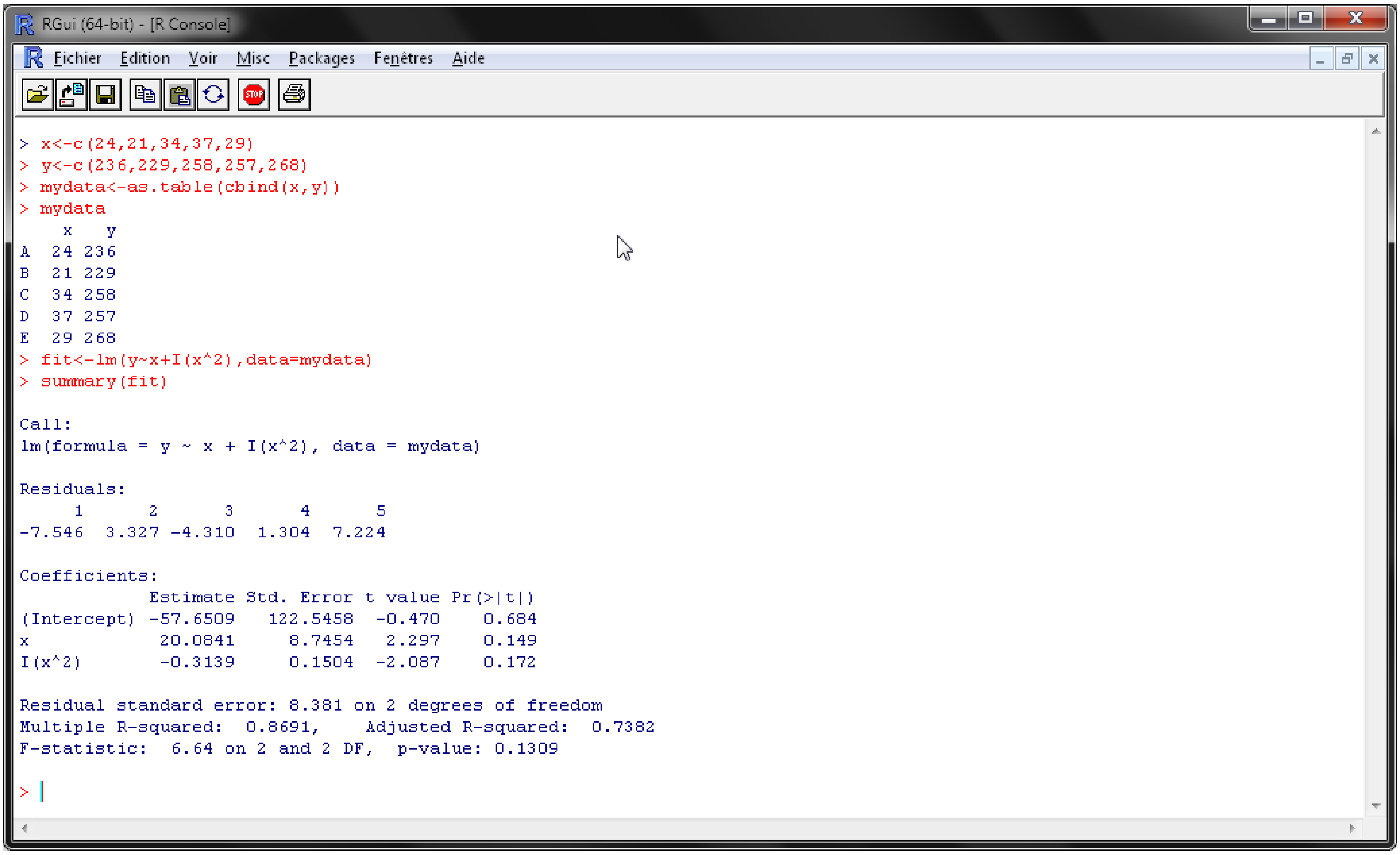
Et maintenant allons-y toujours en nous basant sur le même jeu de données et la fonction **lsqlincon( )** du package **pracma**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("pracma")
> (mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";"))
   Mois Couts CoutA CoutB CoutC
1     1 44439   515   541   928
2     2 43936   929   692   711
3     3 44464   800   710   824
4     4 41533   979   675   758
5     5 46343  1165  1147   635
6     6 44922   651   939   901
7     7 43203   847   755   580
8     8 43000   942   908   589
9     9 40967   630   738   682
10   10 48582  1113  1175  1050
11   11 45003  1086  1075   984
12   12 44303   843   640   828
13   13 42070   500   752   708
14   14 44353   813   989   804
15   15 45968  1190   823   904
16   16 47781  1200  1108  1120
17   17 43202   731   590  1065
18   18 44074  1089   607  1132
19   19 44610   786   513   839
> (x<-as.matrix(mydata[,c(3:5)]))
      CoutA CoutB CoutC
 [1,]   515   541   928
 [2,]   929   692   711
 [3,]   800   710   824
 [4,]   979   675   758
 [5,]  1165  1147   635
 [6,]   651   939   901
 [7,]   847   755   580
 [8,]   942   908   589
 [9,]   630   738   682
[10,]  1113  1175  1050
[11,]  1086  1075   984
[12,]   843   640   828
[13,]   500   752   708
[14,]   813   989   804
[15,]  1190   823   904
[16,]  1200  1108  1120
[17,]   731   590  1065
[18,]  1089   607  1132
[19,]   786   513   839
> (y<-mydata[,2])
 [1] 44439 43936 44464 41533 46343 44922 43203 43000 40967 48582 45003 44303 42070 44353
[15] 45968 47781 43202 44074 44610
> lsqlincon(x, y,Aeq = c(1, 1, 1), beq = 1,lb=c(0.3,0.2,0.4),ub=c(0.5,0.3,0.5))
[1] 0.4 0.2 0.4
>
```

Exemple 262.: Régression polynomiale

R 3.0.2

Nous poursuivons en faisant la même régression polynomiale que dans le cours théorique et Minitab avec la commande **lm( )** :




```
> x<-c(24,21,34,37,29)
> y<-c(236,229,258,257,268)
> mydata<-as.table(cbind(x,y))
> mydata
   x   y
A  24 236
B  21 229
C  34 258
D  37 257
E  29 268
> fit<-lm(y~x+I(x^2),data=mydata)
> summary(fit)

Call:
lm(formula = y ~ x + I(x^2), data = mydata)

Residuals:
     1      2      3      4      5
-7.546  3.327 -4.310  1.304  7.224

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.6509   122.5458  -0.470    0.684
x            20.0841     8.7454   2.297    0.149
I(x^2)       -0.3139     0.1504  -2.087    0.172

Residual standard error: 8.381 on 2 degrees of freedom
Multiple R-squared:  0.8691,	Adjusted R-squared:  0.7382
F-statistic:  6.64 on 2 and 2 DF,  p-value: 0.1309

>
```

Nous retrouvons là encore une fois les mêmes résultats qu'avec Minitab et le cours théorique. La différence étant qu'avec Minitab nous avons automatiquement une série de graphiques récapitulatifs du modèle.

Nous pouvons cependant compléter avec le package **ggplot2** comme nous l'avons fait avant:

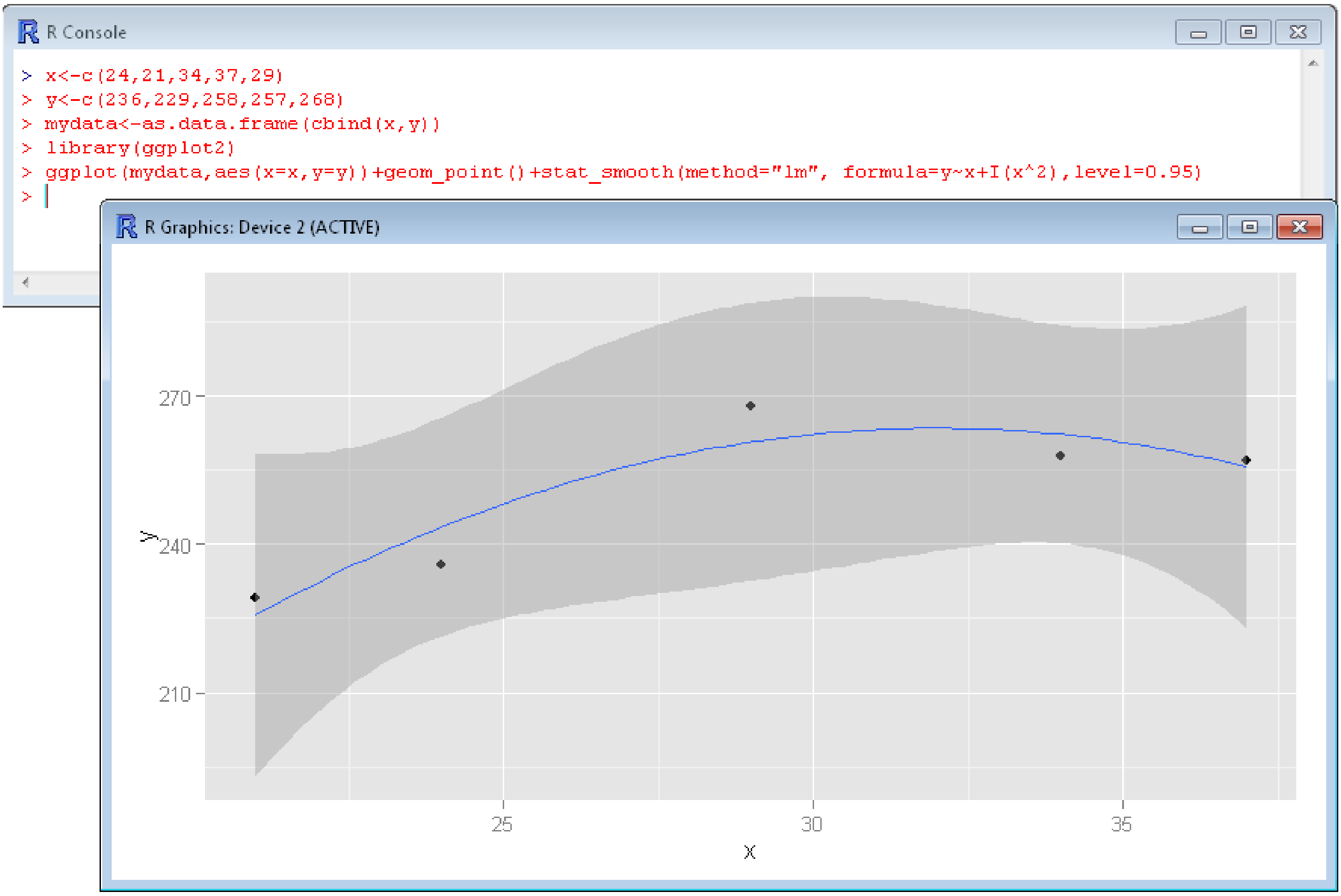
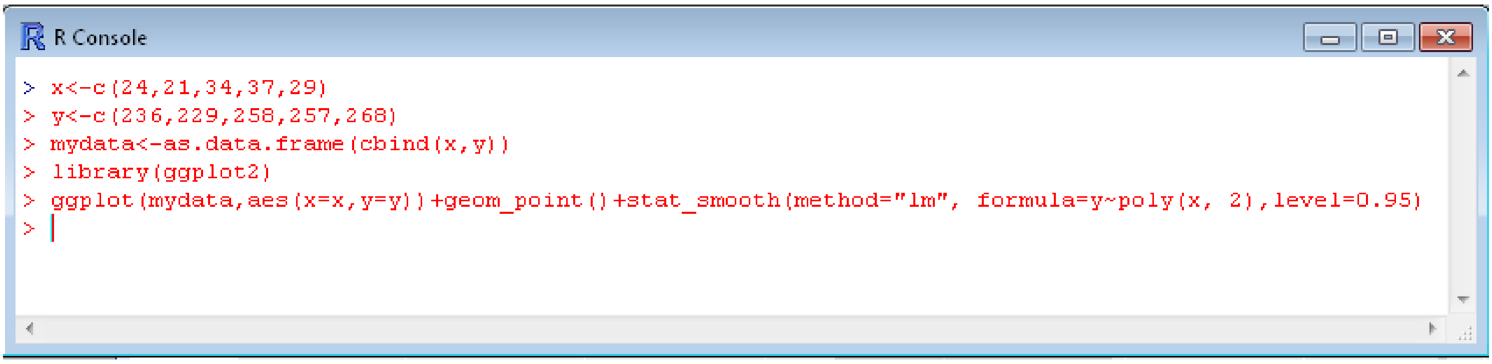
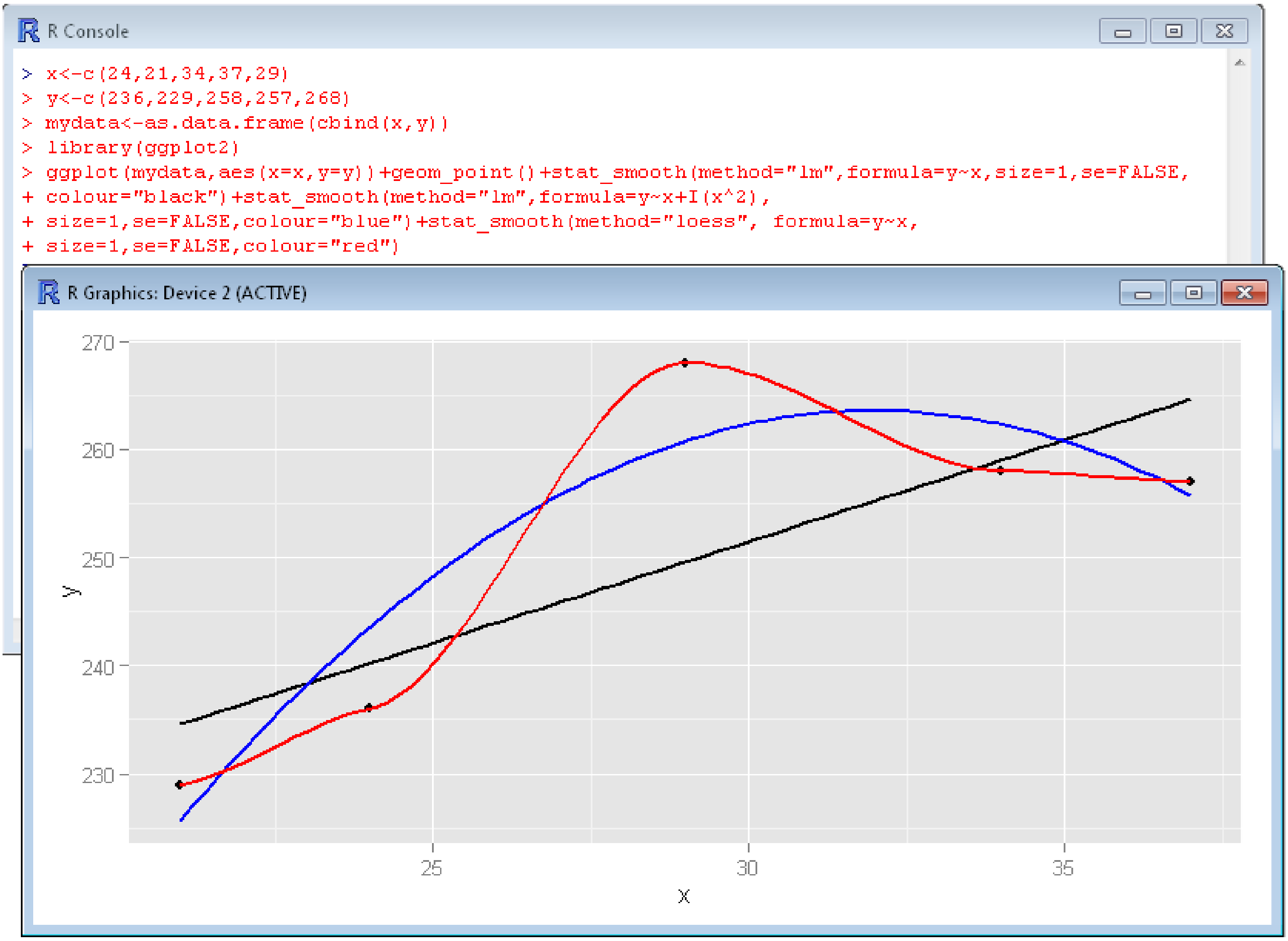


Figure 249 Graph régression polynomiale avec intervalle de confiance (ggplot)

Ou avec une syntaxe alternative équivalente:



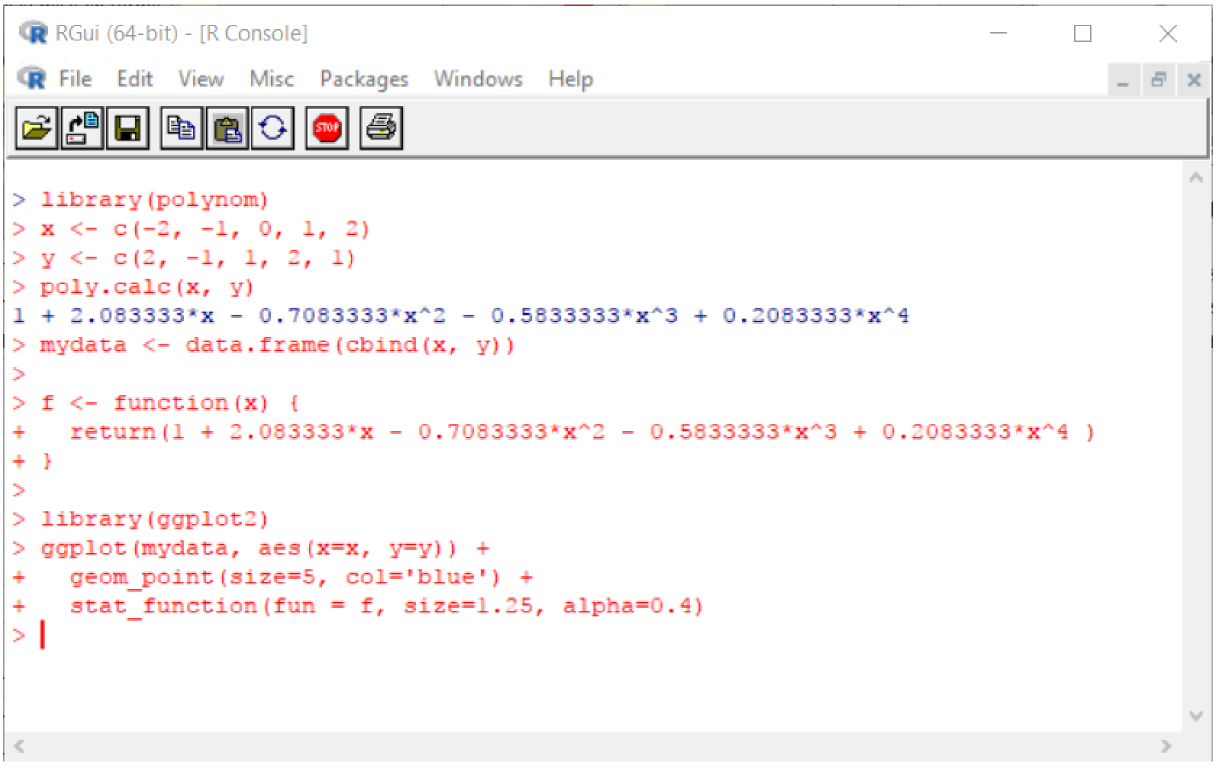
Nous pouvons aussi comparer plusieurs modèles en même temps:



Nous pouvons revenir sur un cas plus compliqué comme nous l'avons fait avec Minitab pour vérifier les possibilités de R. Nous allons donc utiliser les mêmes données que dans le cours Minitab:

Interpolation de Lagrange

Nous allons voir que nous pouvons effectuer des interpolations de Lagrange comme celles étudiées dans le cours théorique. Le script n'est de loin pas trivial. Il utilise les packages **polynom** et **ggplot2** et particulièrement la fonction **poly.calc( )** du premier package:



Ce qui donne visuellement:

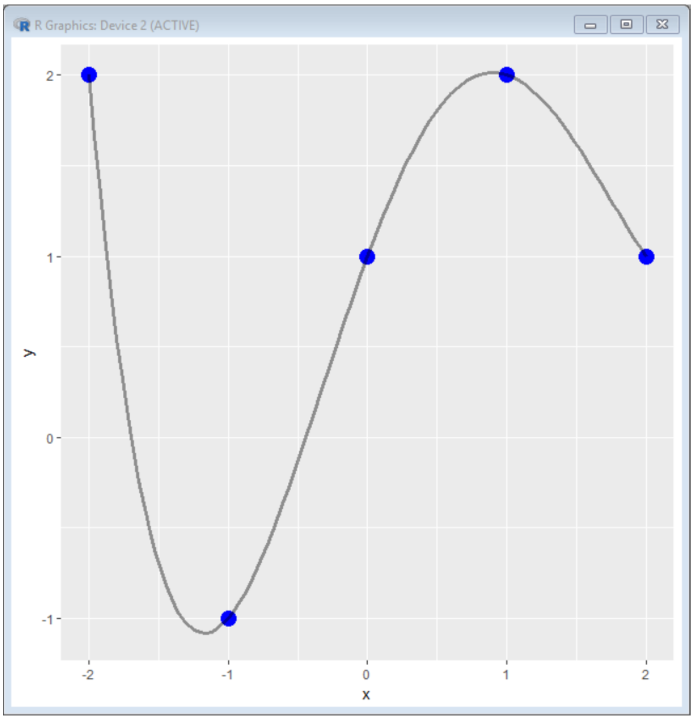


Figure 250 Graph interpolation de Lagrange (ggplot)

Exemple 263.: Régression pondérée

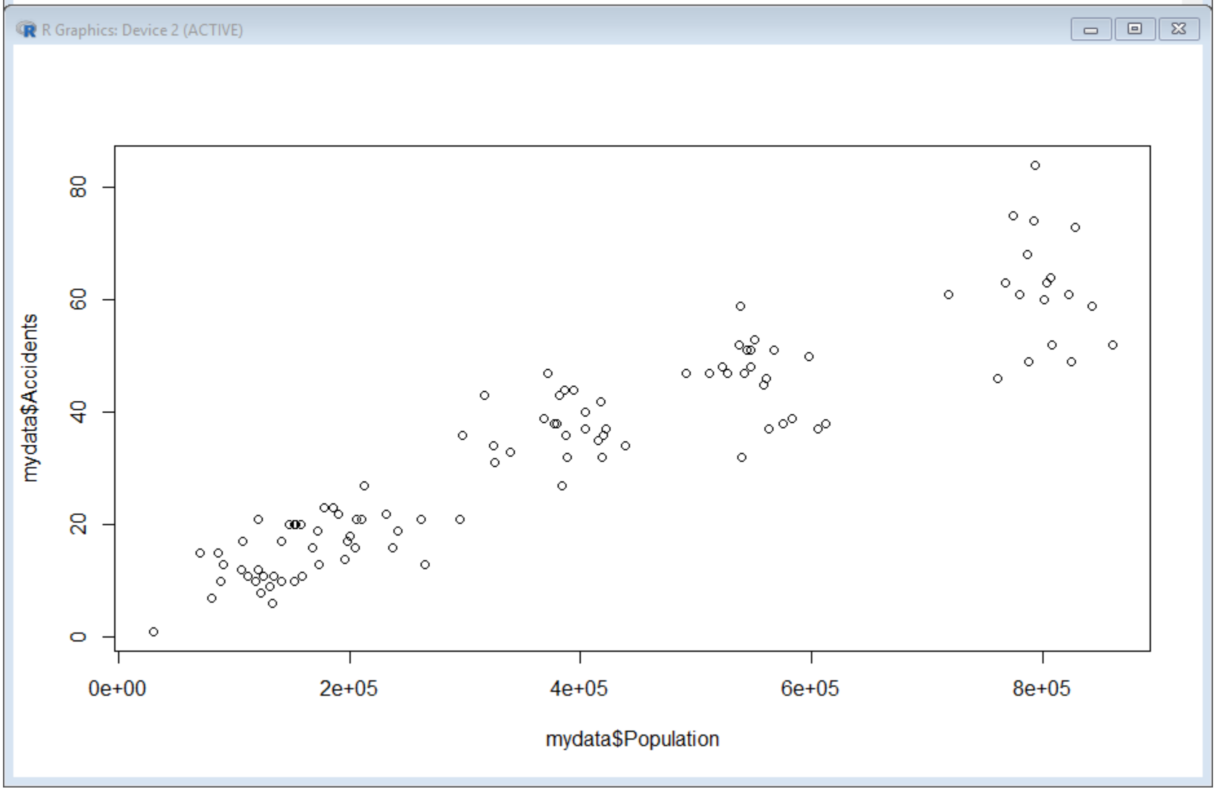
R 3.5.1

Nous avons aussi étudié mathématiquement les principes sous-jacents de la régression pondérée et surtout de sa difficulté à choisir les poids dans certaines situations.

Voyons cependant si en reprenant l'exemple de Minitab si nous retombons sur les mêmes résultats.

D'abord importons les données et voyons-en un graph:

```
> mydata <- read.csv("c:/tmp/RegressionPonderee.csv", header=T, sep=",")
> head(mydata)
  Accidents AccidentRate Population
1        20  1.35498e-04   147604.0
2        38  6.21013e-05   611903.2
3        17  1.20308e-04   141304.5
4        16  7.81061e-05   204849.5
5        20  1.31211e-04   152426.5
6        37  8.78015e-05   421405.0
> plot(x=mydata$Population,y=mydata$Accidents)
> 
```



Voyons déjà la à quoi ressemble le modèle naïf de base avec la variance des résidus globales (étant donné que dans le modèle naïf les résidus sont homoscédastiques alors il n'y a effectivement qu'une variance pour rappel):

```
> model.1 <- lm(Accidents~ Population,mydata)
> summary(model.1)

Call:
lm(formula = Accidents ~ Population, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-16.0837  -4.5023   0.3836   4.4076  21.1181

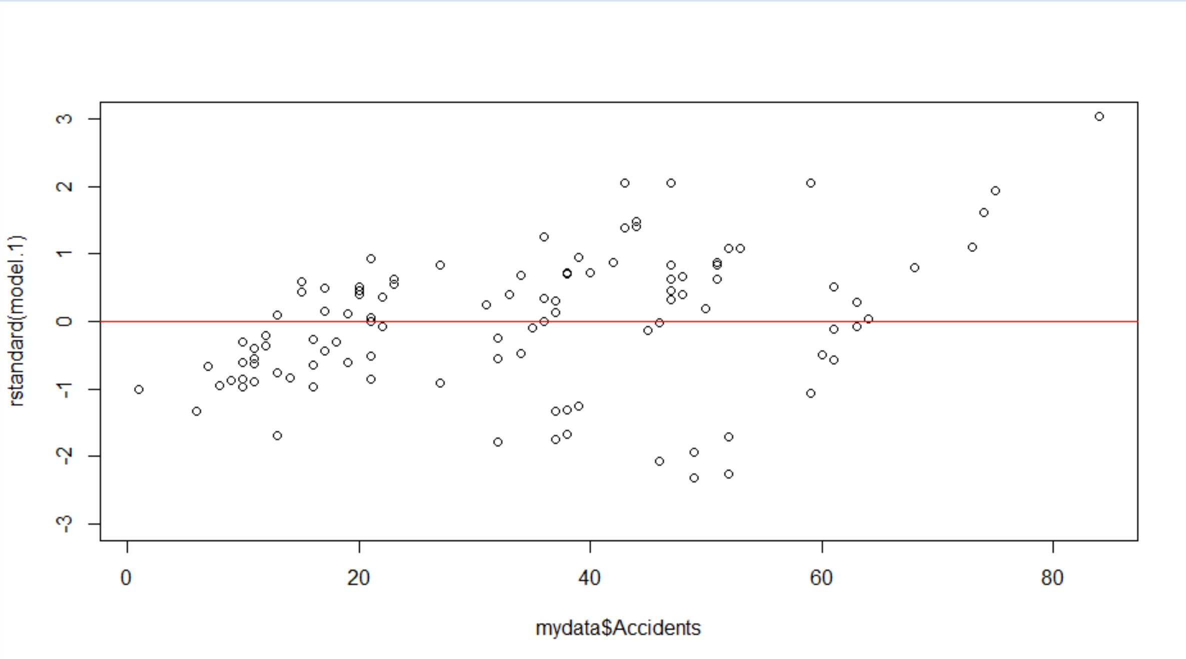
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.822e+00  1.313e+00   4.434 2.28e-05
Population  7.186e-05  2.858e-06  25.147  < 2e-16

Residual standard error: 7.085 on 105 degrees of freedom
Multiple R-squared:  0.8576,    Adjusted R-squared:  0.8562
F-statistic: 632.4 on 1 and 105 DF,  p-value: < 2.2e-16

> #variance des résidus
> sigma(model.1)
[1] 7.084646
>
```

Maintenant voyons le graph des résidus en fonction de l'abscisse:

```
> plot(mydata$Accidents, rstandard(model.1),ylim=c(-3,3))
> abline(h=0, col="red")
>
```



Nous voyons clairement qu'il y a un effet de dispersion net (effet "fan") qui a l'air d'être proportionnel à l'abscisse. Utilisons donc l'inverse de cette dernière comme poids:

```
> wts <- 1/mydata$Population
> head(wts,40)
 [1] 6.774885e-06 1.634246e-06 7.076917e-06 4.881633e-06 6.560540e-06 2.373014e-06
 [7] 4.998895e-06 2.623418e-06 1.784775e-06 1.243911e-06 2.394608e-06 1.652055e-06
[13] 2.390540e-06 7.536439e-06 1.269378e-06 8.142500e-06 1.212627e-06 5.055221e-06
[19] 1.839789e-06 1.282155e-06 5.399769e-06 1.774682e-06 1.165165e-05 4.741859e-06
[25] 6.514328e-06 2.582458e-06 5.254899e-06 2.473752e-06 1.913300e-06 1.270154e-06
[31] 2.591861e-06 1.713715e-06 2.575765e-06 9.383604e-06 1.238147e-06 6.557786e-06
[37] 1.162444e-06 4.223076e-06 1.815588e-06 5.095425e-06
> model.2 <- lm(Accidents~ Population, weights=wts,data=mydata)
> summary(model.2)

Call:
lm(formula = Accidents ~ Population, data = mydata, weights = wts)

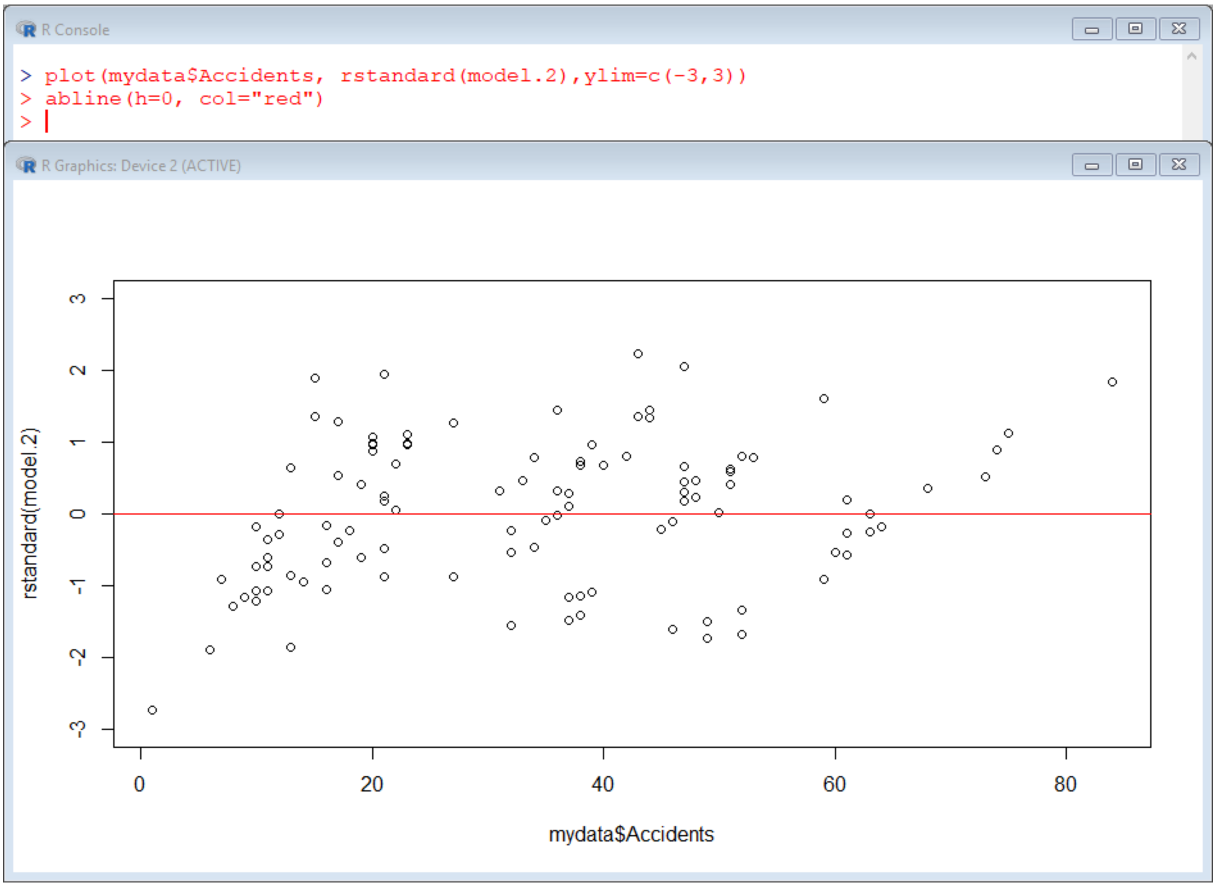
Weighted Residuals:
       Min         1Q     Median         3Q        Max 
-0.0295870 -0.0085203  0.0006693  0.0087642  0.0262133 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.815e+00  8.672e-01    4.40 2.61e-05
Population  7.698e-05  2.864e-06   26.88  < 2e-16

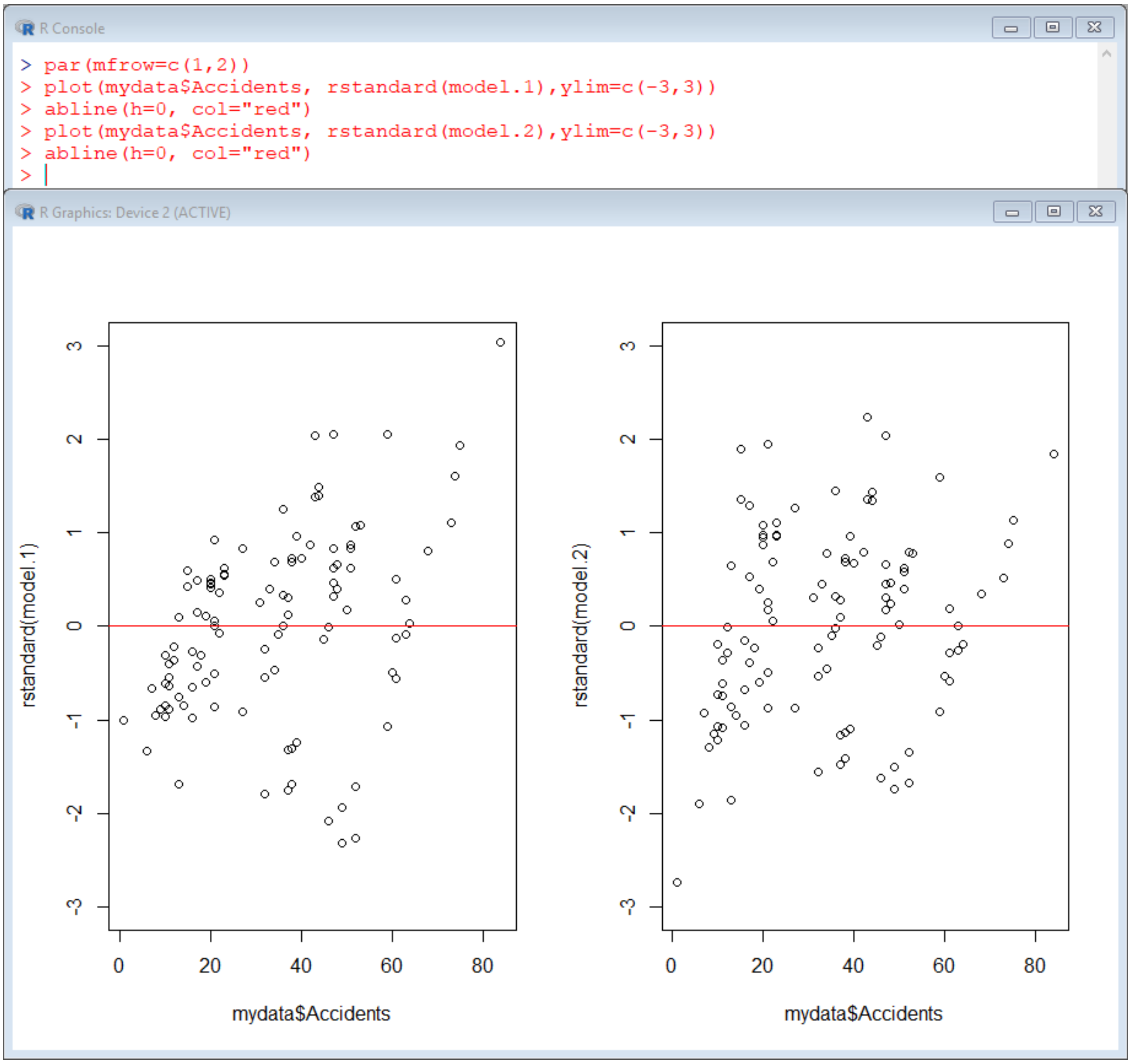
Residual standard error: 0.01178 on 105 degrees of freedom
Multiple R-squared:  0.8731,	Adjusted R-squared:  0.8719 
F-statistic: 722.7 on 1 and 105 DF,  p-value: < 2.2e-16

> 
```

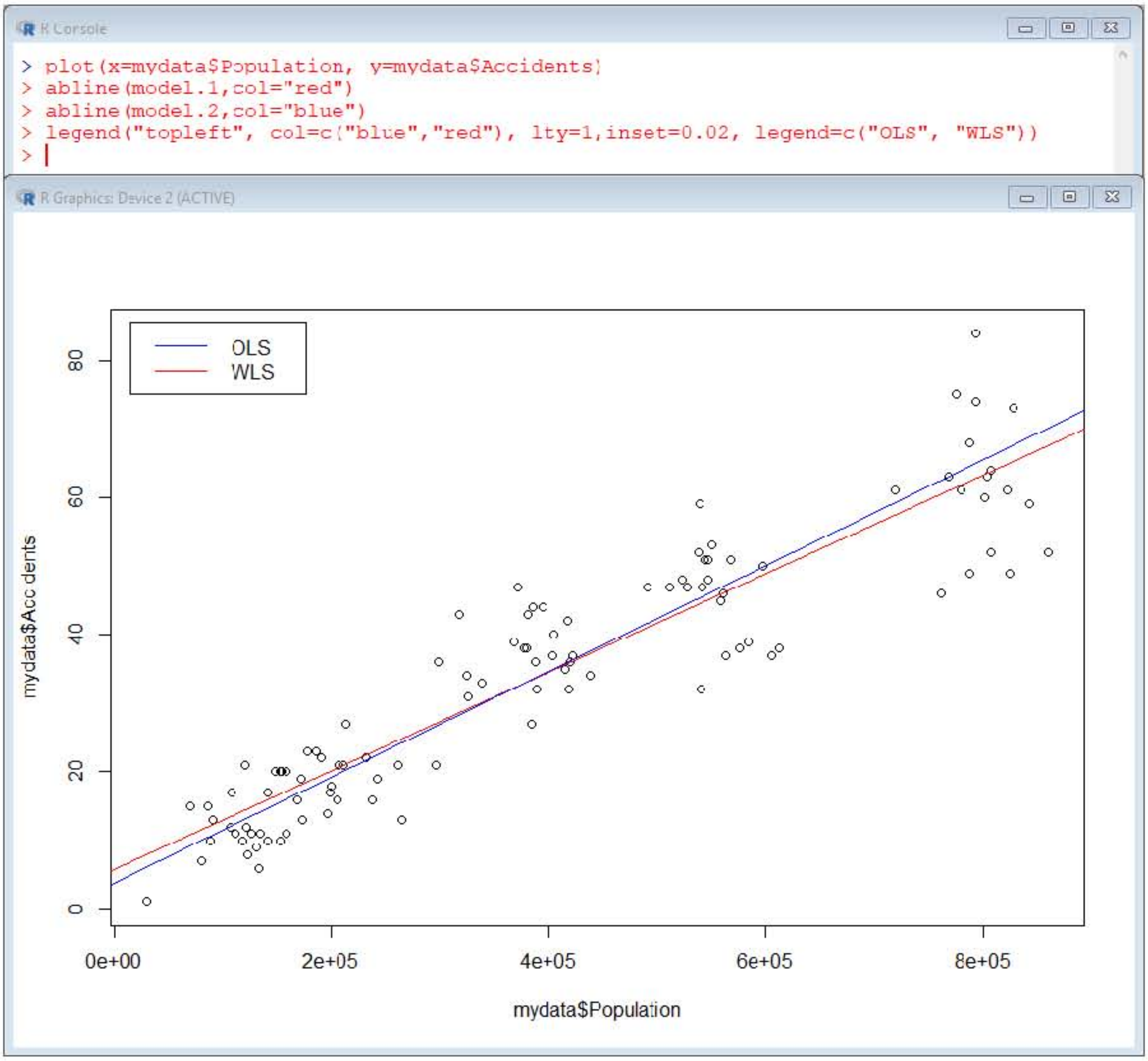
Nous voyons bien évidemment que les paramètres du modèle changent alors. Voyons maintenant le graph des résidus:



Comparons les deux graphs de résidus côte à côte:



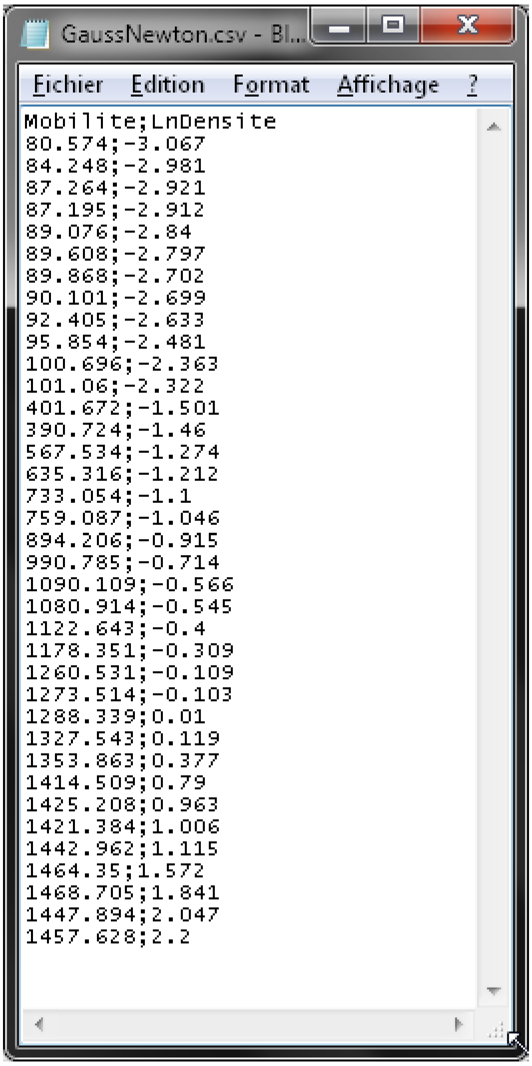
Et comparons également le résultat visuel des deux modèles:


R Console
> plot(x=mydata$Population, y=mydata$Accidents)
> abline(model.1,col="red")
> abline(model.2,col="blue")
> legend("topleft", col=c("blue","red"), lty=1,inset=0.02, legend=c("OLS", "WLS"))
>
R Graphics: Device 2 (ACTIVE)
OLS
WLS
mydata$Accidents
0
20
40
60
80
0e+00
2e+05
4e+05
6e+05
8e+05
mydata$Population

Exemple 264.: Régression non linéaire générale

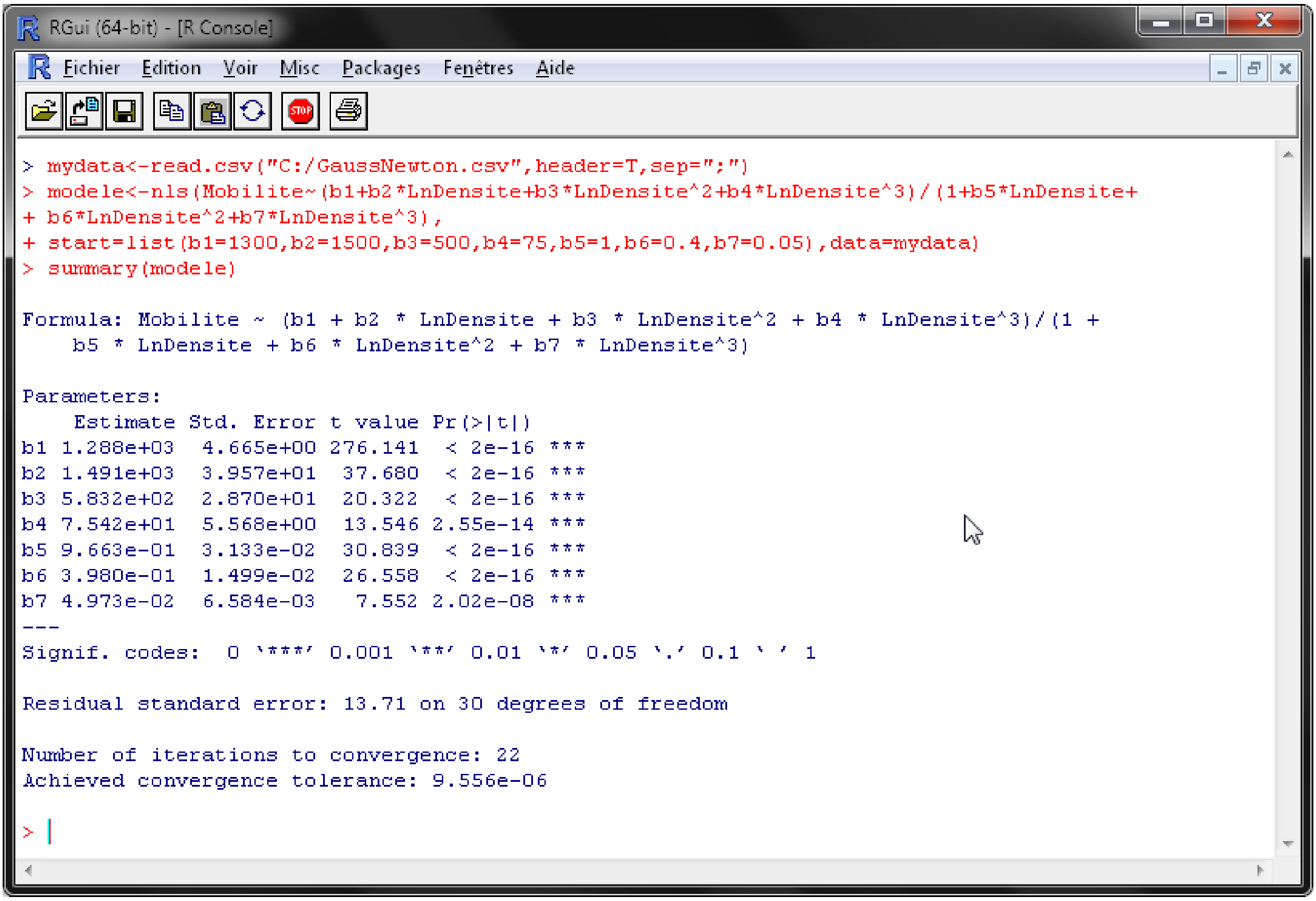
R 3.0.2

Le but ici va être de reprendre l'exemple fait dans le cours Microsoft Excel utilisant la méthode de Gauss-Newton et aussi la comparer avec ce que nous avions obtenu avec Minitab 16. Nous partons donc des données suivantes:




```
Mobilite;LnDensite
80.574;-3.067
84.248;-2.981
87.264;-2.921
87.195;-2.912
89.076;-2.84
89.608;-2.797
89.868;-2.702
90.101;-2.699
92.405;-2.633
95.854;-2.481
100.696;-2.363
101.06;-2.322
401.672;-1.501
390.724;-1.46
567.534;-1.274
635.316;-1.212
733.054;-1.1
759.087;-1.046
894.206;-0.915
990.785;-0.714
1090.109;-0.566
1080.914;-0.545
1122.643;-0.4
1178.351;-0.309
1260.531;-0.109
1273.514;-0.103
1288.339;0.01
1327.543;0.119
1353.863;0.377
1414.509;0.79
1425.208;0.963
1421.384;1.006
1442.962;1.115
1464.35;1.572
1468.705;1.841
1447.894;2.047
1457.628;2.2
```

Ensuite, nous utilisons la commande native pour y répliquer le modèle théorique supposé:




```
> mydata<-read.csv("C:/GaussNewton.csv",header=T,sep=";")
> modele<-nls(Mobilite~(b1+b2*LnDensite+b3*LnDensite^2+b4*LnDensite^3)/(1+b5*LnDensite+
+ b6*LnDensite^2+b7*LnDensite^3),
+ start=list(b1=1300,b2=1500,b3=500,b4=75,b5=1,b6=0.4,b7=0.05),data=mydata)
> summary(modele)

Formula: Mobilite ~ (b1 + b2 * LnDensite + b3 * LnDensite^2 + b4 * LnDensite^3)/(1 +
    b5 * LnDensite + b6 * LnDensite^2 + b7 * LnDensite^3)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
b1 1.288e+03  4.665e+00 276.141  < 2e-16 ***
b2 1.491e+03  3.957e+01  37.680  < 2e-16 ***
b3 5.832e+02  2.870e+01  20.322  < 2e-16 ***
b4 7.542e+01  5.568e+00  13.546 2.55e-14 ***
b5 9.663e-01  3.133e-02  30.839  < 2e-16 ***
b6 3.980e-01  1.499e-02  26.558  < 2e-16 ***
b7 4.973e-02  6.584e-03   7.552 2.02e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.71 on 30 degrees of freedom

Number of iterations to convergence: 22
Achieved convergence tolerance: 9.556e-06

> 
```

Nous remarquons donc que le nombre d'itérations est inférieur à celui de Minitab. Par contre nous avons moins d'information concernant l'erreur du modèle que ce que donne Minitab ou Microsoft Excel de façon automatique. Avec un plus pour Minitab qui donne l'intervalle de confiance au lieu de la valeur critique de la distribution de Student!

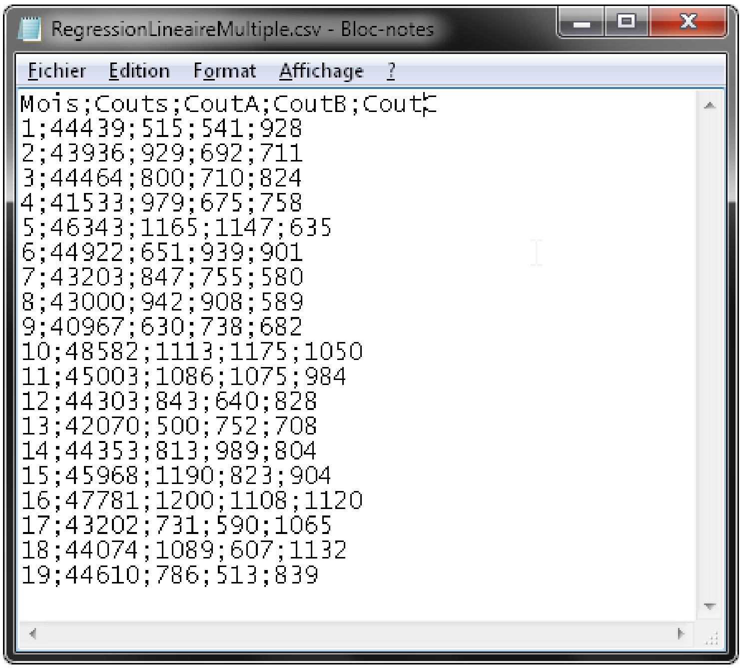
Sinon nous retrouvons exactement les mêmes valeurs de coefficients qu'avec Minitab et la même erreur résiduelle.

Exemple 265.: Régression linéaire/polynomiale 3D

R 3.0.2

Bon cet exercice n'est que pour les paillettes car avoir un modèle à seulement deux variables explicatives est super limitatif (c'est toujours le problème des graphes car en dehors de la 2D et de la 3D... c'est fini!).

Nous travaillons à nouveau avec le fichier contenant les Coûts:

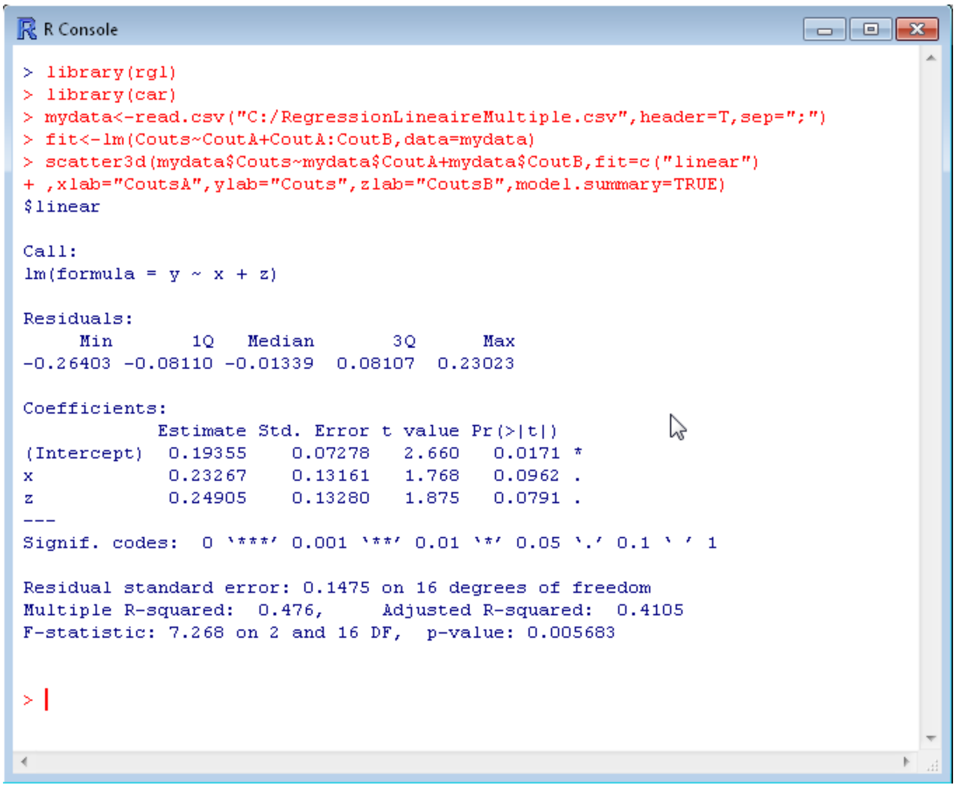

RegressionLineaireMultiple.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

Et maintenant nous jouons avec la commande **scatter3d( )** et **lm( )** des package **car** et **rgl**.

D'abord nous tentons de voir ce qu'un modèle purement linéaire avec deux variables explicatives donne:



```
> library(rgl)
> library(car)
> mydata<-read.csv("C:/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutA:CoutB,data=mydata)
> scatter3d(mydata$Couts~mydata$CoutA+mydata$CoutB,fit=c("linear")
+ ,xlab="CoutsA",ylab="Couts",zlab="CoutsB",model.summary=TRUE)
$linear

Call:
lm(formula = y ~ x + z)

Residuals:
     Min       1Q   Median       3Q      Max
-0.26403 -0.08110 -0.01339  0.08107  0.23023

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.19355    0.07278   2.660   0.0171 *
x            0.23267    0.13161   1.768   0.0962 .
z            0.24905    0.13280   1.875   0.0791 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1475 on 16 degrees of freedom
Multiple R-squared:  0.476,     Adjusted R-squared:  0.4105
F-statistic: 7.268 on 2 and 16 DF,  p-value: 0.005683

> 
```

Soit graphiquement:

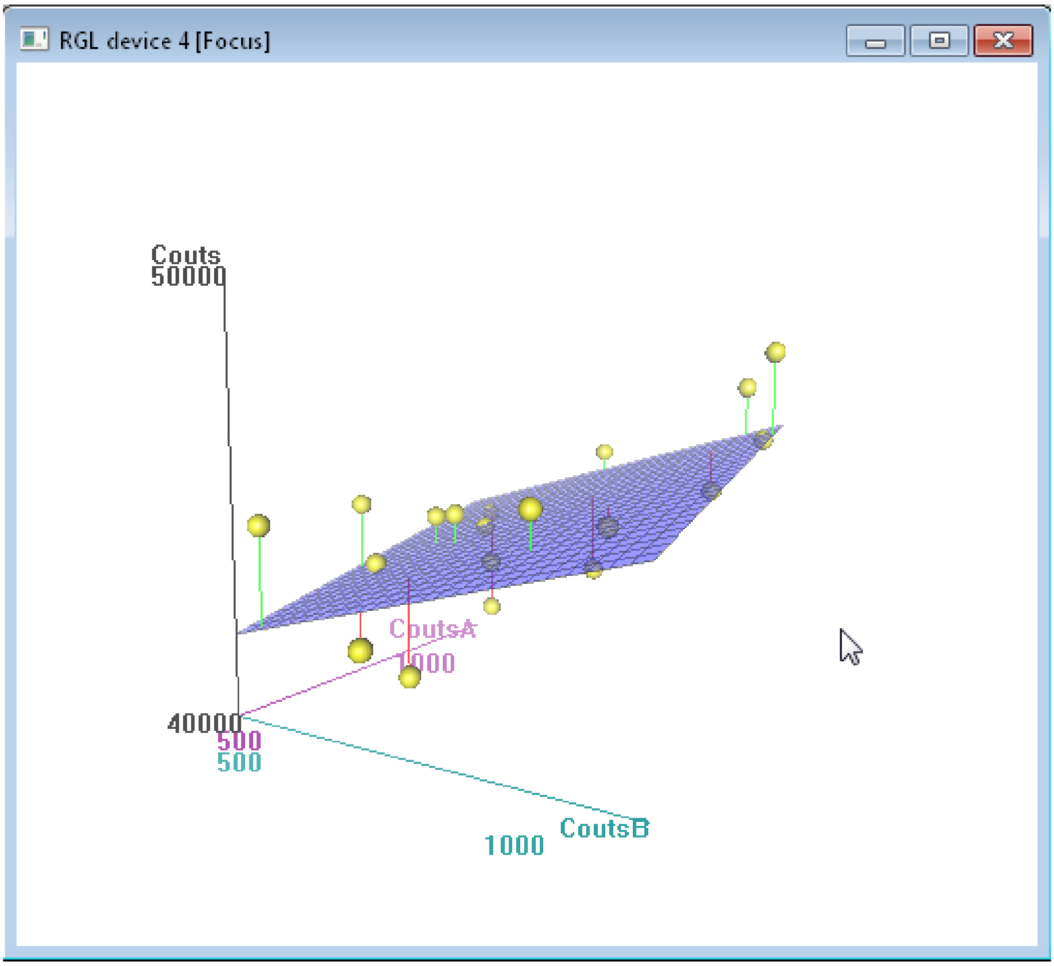


Figure 251 Graphique plan de régression linéaires 3D avec erreurs verticales et points

Nous pouvons faire mieux et optionnellement sans utiliser le package **car** et en affichant aussi l'ellipsoïde de contrôle:

```
> library(rgl)
> mydata<-read.csv("C:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> x<-mydata$CoutA;y<-mydata$CoutB;z<-mydata$CoutC
>
> #on afficher les sphères
> plot3d(x, y, z, col="blue", box = FALSE, type ="s", radius = 20)
> #modèle linéaire
> fit <- lm(z ~ x + y)
> coefs <- coef(fit)
> a <- coefs["x"]; b <- coefs["y"]; c <- -1
> d <- coefs["(Intercept)"]
> #on affiche le plan
> rgl.planes(a, b, c, d, alpha=0.5, color = "red")
> title3d(color="black","","","CoutA","CoutB","CoutC")
>
> ellipse <- ellipse3d(cov(cbind(x,y,z)),
+ centre=c(mean(x), mean(y), mean(z)), level = 0.95)
> wire3d(ellipse, col = "#D95F02",  lit = FALSE)
>
```

Ce qui donne:

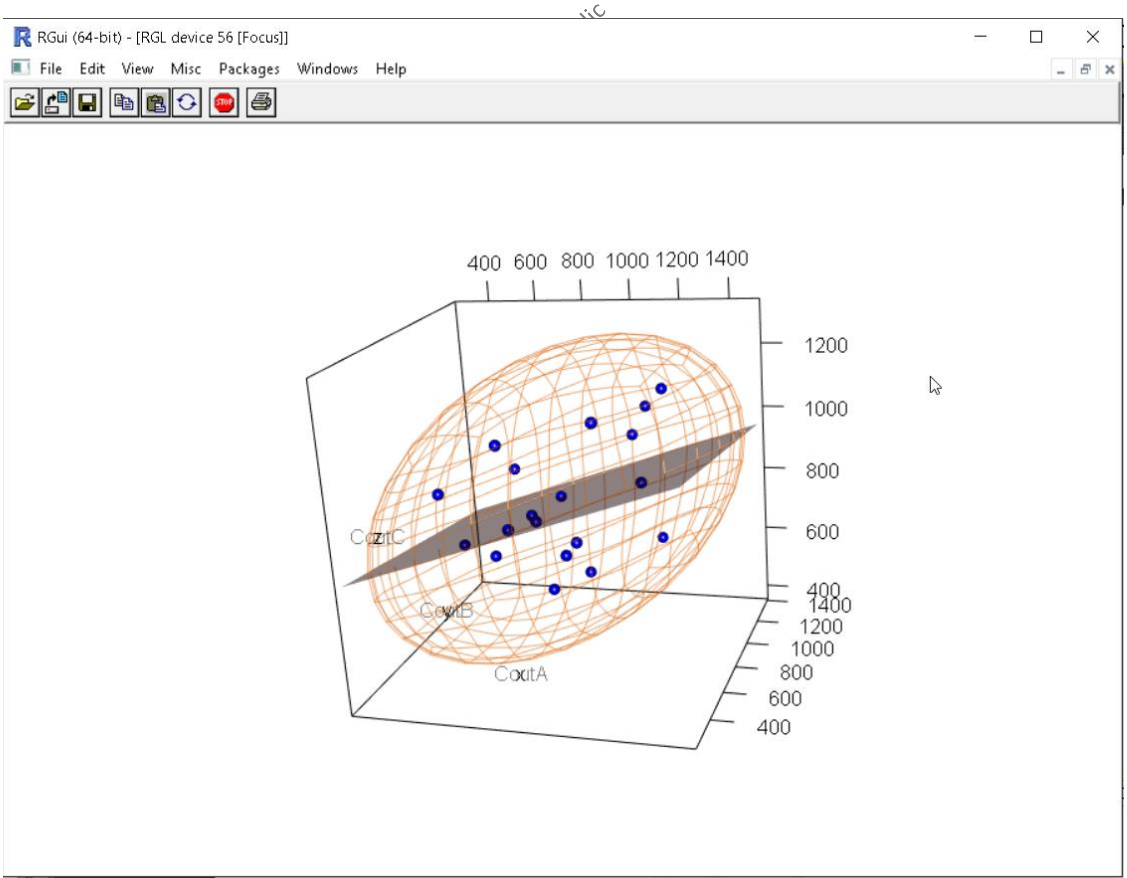


Figure 252 Graph de plan de régression 3D avec ellipse de confiance

Avec un modèle quadratique seul avec interactions:

```
> library(rgl)
> library(car)
> mydata<-read.csv("C:/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit<-lm(Couts~CoutA+CoutA:CoutB,data=mydata)
> scatter3d(mydata$Couts~mydata$CoutA+mydata$CoutB,fit=c("quadratic")
+ ,xlab="CoutsA",ylab="Couts",zlab="CoutsB",model.summary=TRUE)
$quadratic

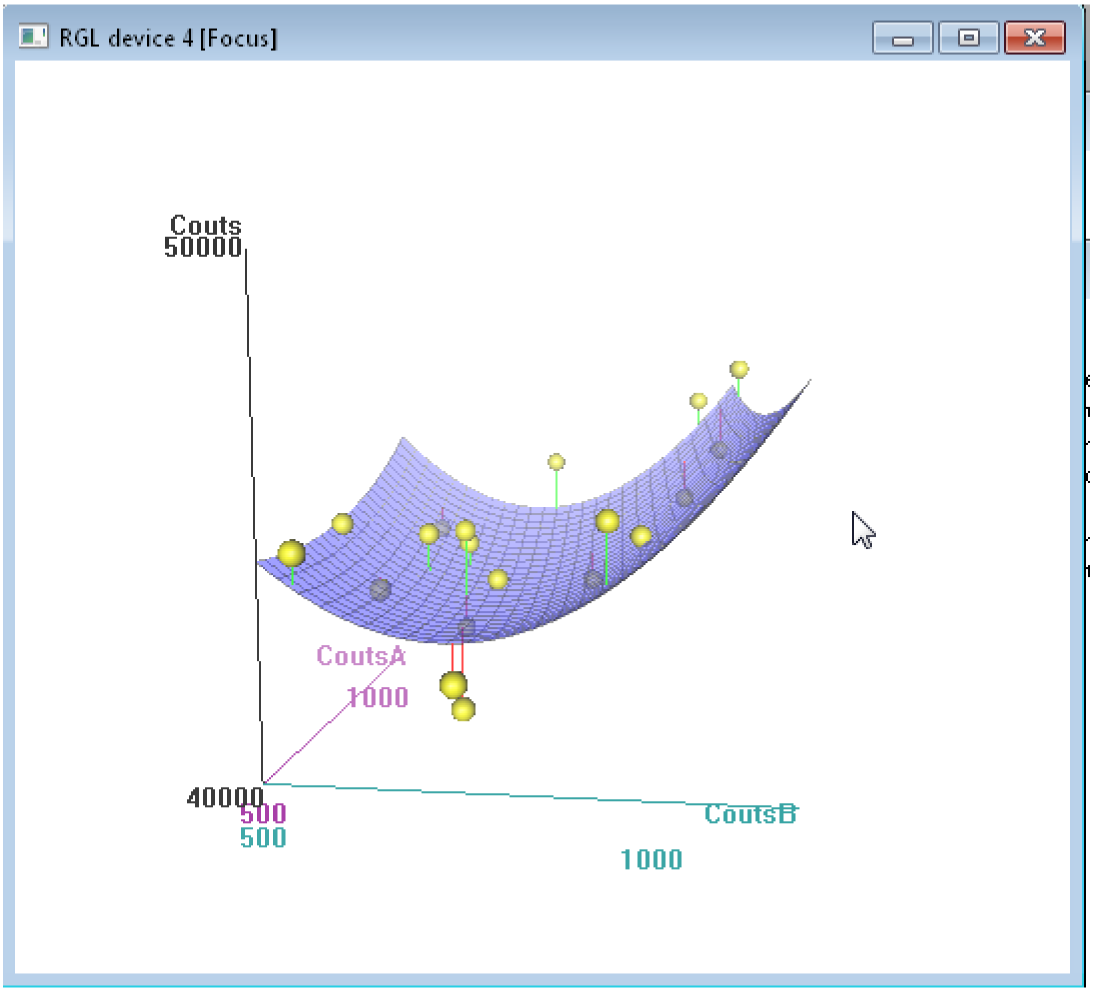
Call:
lm(formula = y ~ (x + z)^2 + I(x^2) + I(z^2))

Residuals:
     Min       1Q   Median       3Q      Max 
-0.21216 -0.07394  0.01990  0.07327  0.15238 

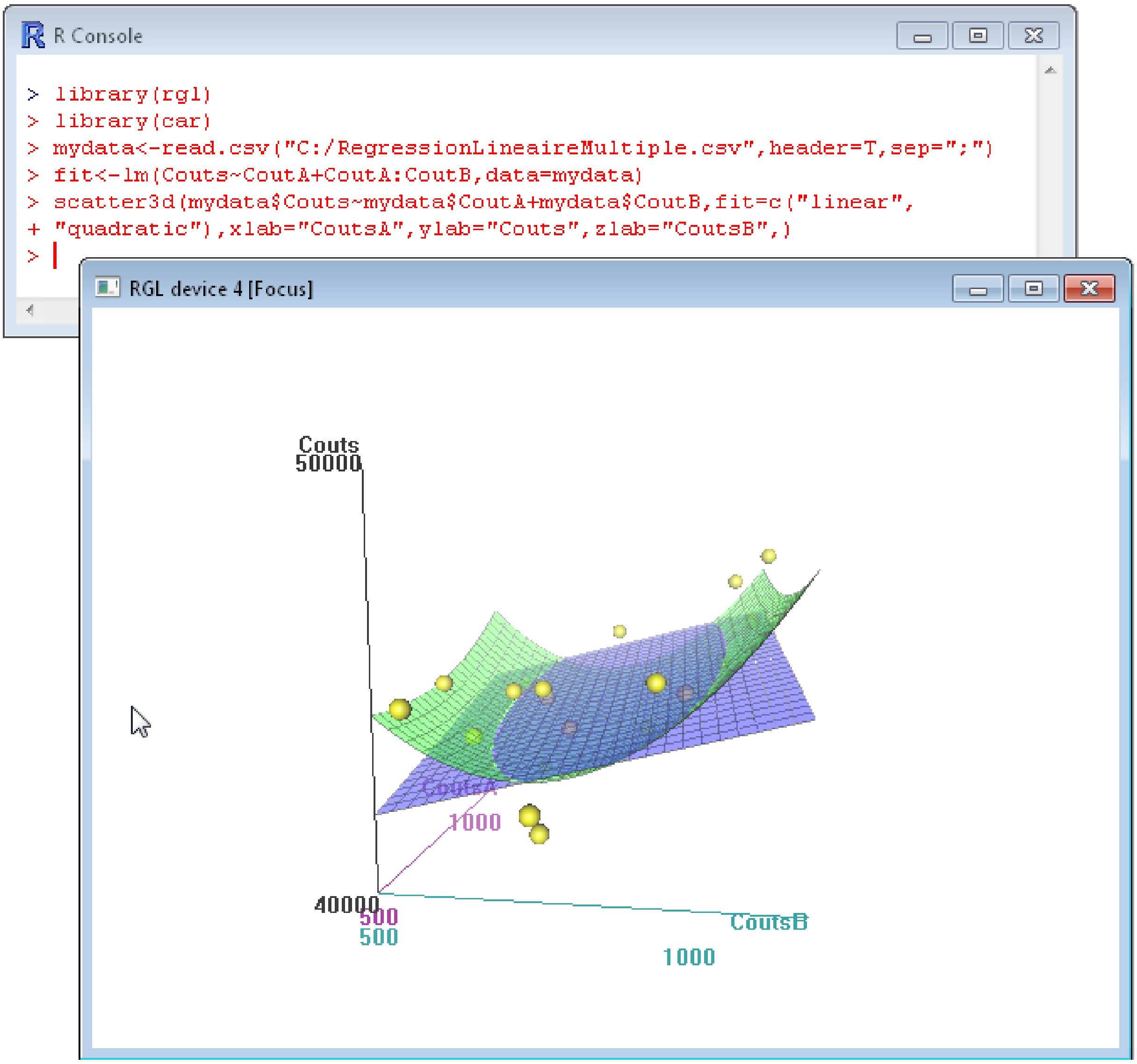
Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  0.42985    0.11149   3.856  0.00199 **
x           -0.05245    0.35190  -0.149  0.88381   
z           -0.81985    0.39810  -2.059  0.06007 . 
I(x^2)       0.29629    0.39695   0.746  0.46870   
I(z^2)       1.15046    0.58644   1.962  0.07156 . 
x:z         -0.18049    0.63479  -0.284  0.78064   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1257 on 13 degrees of freedom
Multiple R-squared:  0.6906,    Adjusted R-squared:  0.5717 
F-statistic: 5.805 on 5 and 13 DF,  p-value: 0.004952
```

Soit graphiquement:

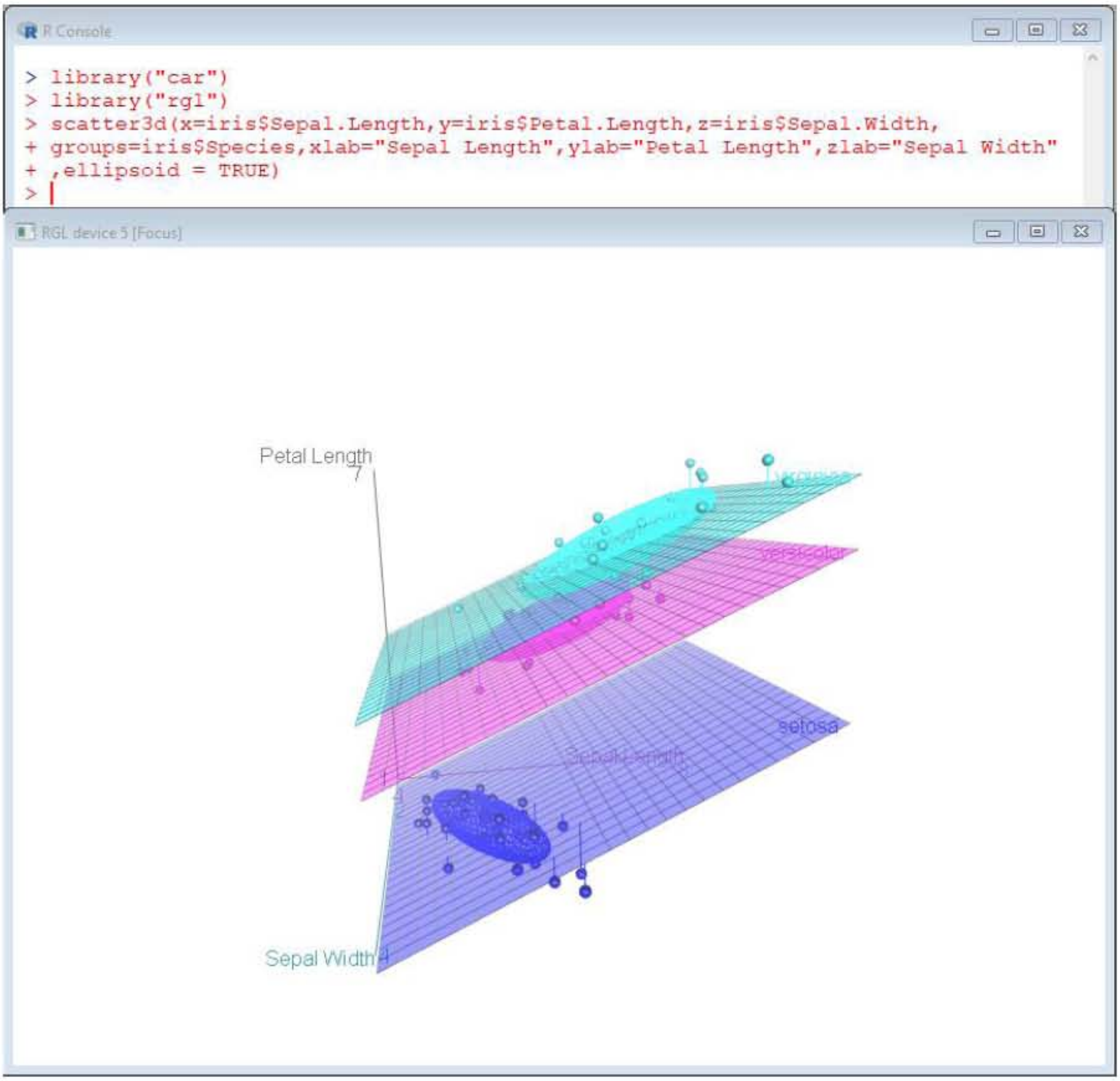


Et en mélangeant les deux (par contre à ce moment là on ne peut plus avoir les projections des points sur le modèle):



Bref c'est joli mais très limite puisque à trois dimensions et de toute façon c'est aussi peu précis (un bon tableau avec des chiffres en statistiques ça reste toujours la base!).

Enfin pour ceux qui veulent comparer plusieurs modèles de régression (comme si on avait un modèle à trois variables explicatives dont l'une est maintenue fixe):


R Console
> library("car")
> library("rgl")
> scatter3d(x=iris$Sepal.Length,y=iris$Petal.Length,z=iris$Sepal.Width,
+ groups=iris$Species,xlab="Sepal Length",ylab="Petal Length",zlab="Sepal Width"
+ ,ellipsoid = TRUE)
RGL device 5 [Focus]
Petal Length
versicolor
setosa
Sepal Width

Exemple 266.: Régression régularisée ridge (L_2)

R 3.5.1

Comme nous l'avons mentionné dans le cours théorique:

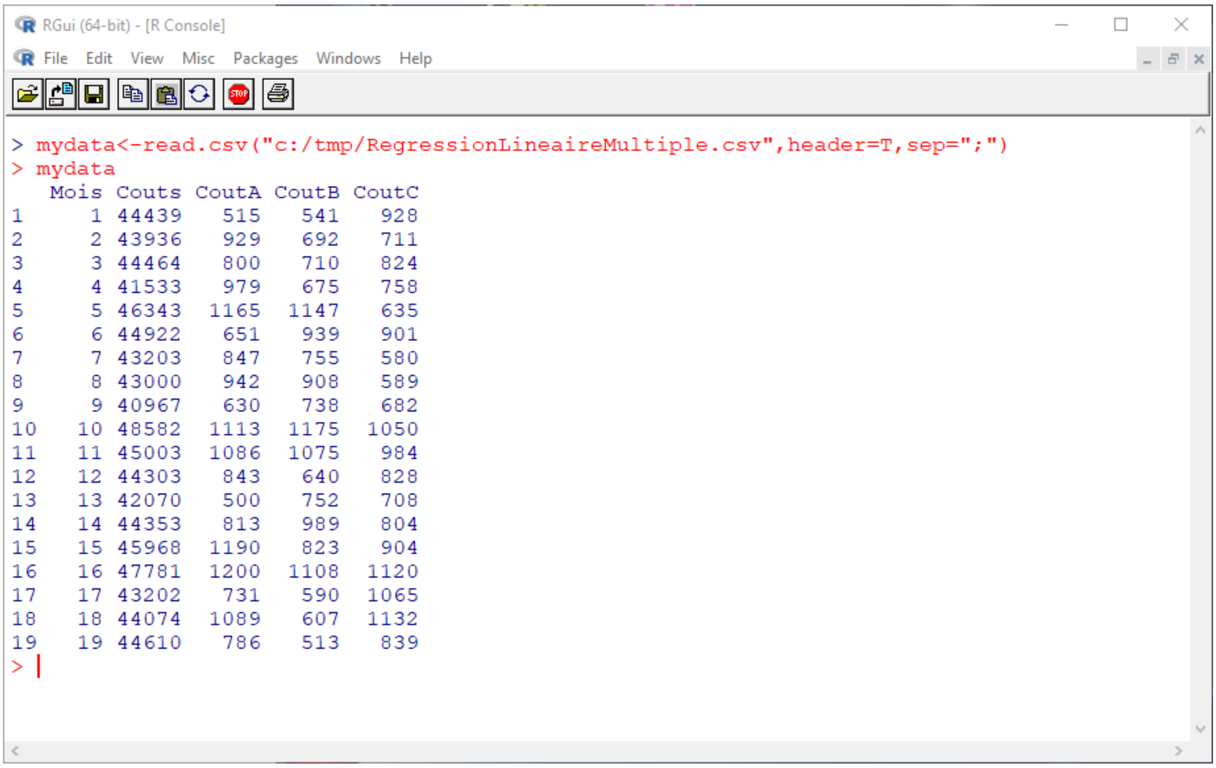
1. Il s'agit plus d'une technique d'ingénierie statistique que de statistiques pures (raisons pour lesquelles on va voir des différences importantes selon les packages dans la valeur du lambda optimal car entre les différentes formulations de la fonction de coût et les méthodes paramétriques… on a pas fini de se mélanger les pinceaux en absence de normes ISO)

2. La régression régularisée ridge (ainsi que LASSO ou Elastic Net) ont plus leur place dans la section de machine learning (d'ailleurs nous reviendrons sur ces 3 types de régression dans le chapitre y relatif) que dans cette section sur la régression puisqu'aujourd'hui nous n'avons pas de relation analytique connu pour faire de l'inférence statistique sur les coefficients (autres que les bootstrapping bien évidemment!)

Pour information, à ce jour et à ma connaissance, il n'existe pas (malheureusement) un seul et unique package qui fait en seulement quelques lignes (4-5 lignes) ce que nous allons voir ci-dessous.

Avec le package MASS

Donc commençons avec le package **MASS** (les exemples ci-dessous peuvent probablement être améliorés)!



```
> summary(lm(Couts~CoutA+CoutB+CoutC,data=mydata))

Call:
lm(formula = Couts ~ CoutA + CoutB + CoutC, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-2042.8  -820.0   236.2   807.3  1721.4

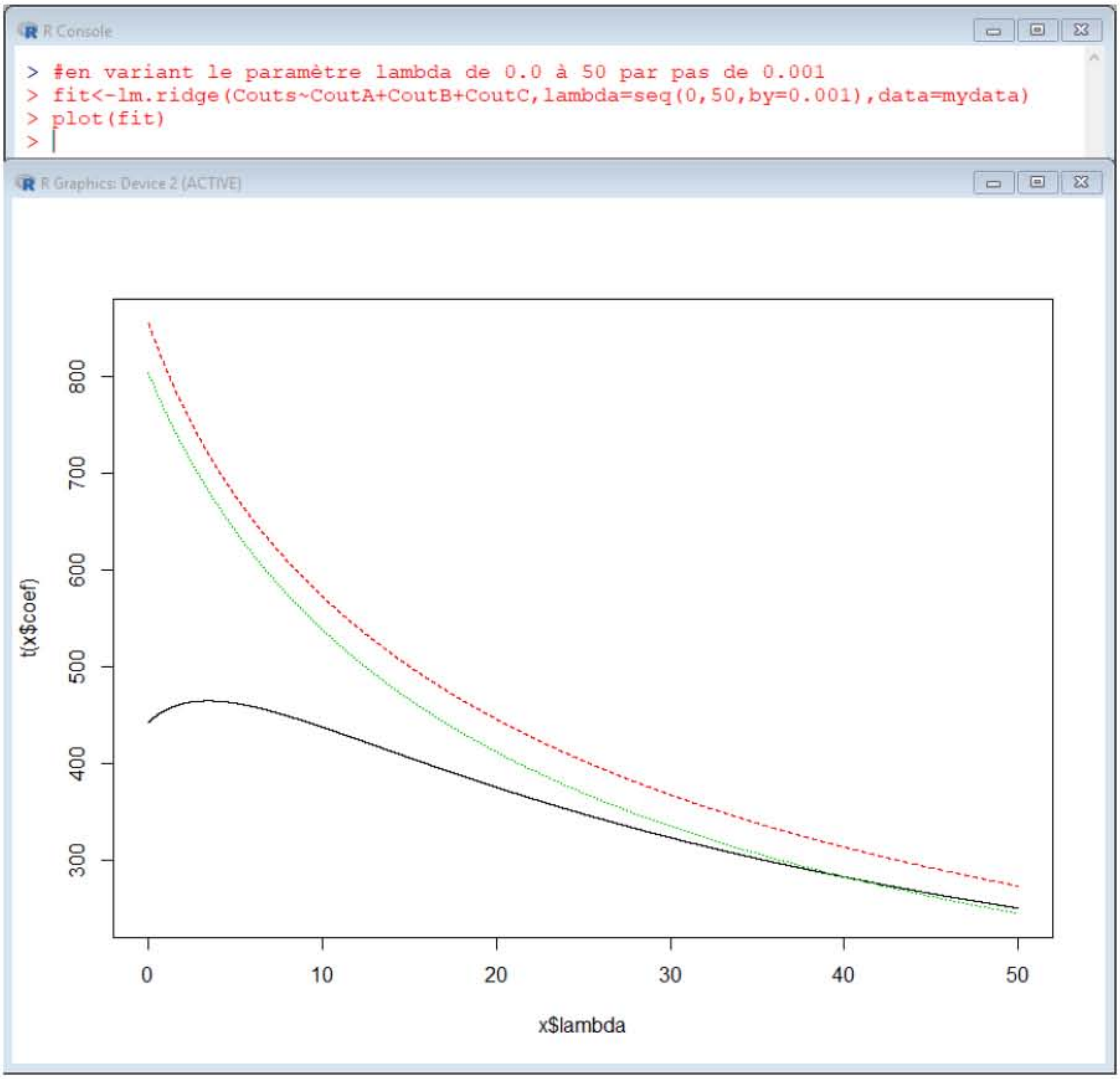
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 35102.900   1837.227  19.106 6.11e-12
CoutA           2.066      1.665   1.241   0.2337
CoutB           4.176      1.681   2.484   0.0253
CoutC           4.791      1.789   2.677   0.0172

Residual standard error: 1253 on 15 degrees of freedom
Multiple R-squared:  0.6454,    Adjusted R-squared:  0.5745
F-statistic: 9.102 on 3 and 15 DF,  p-value: 0.001127

> library("MASS")
> #régression ridge avec lambda=0
> lm.ridge(Couts~CoutA+CoutB+CoutC,data=mydata)
                    CoutA        CoutB        CoutC
35102.900449     2.065953     4.176356     4.790641
> |
```

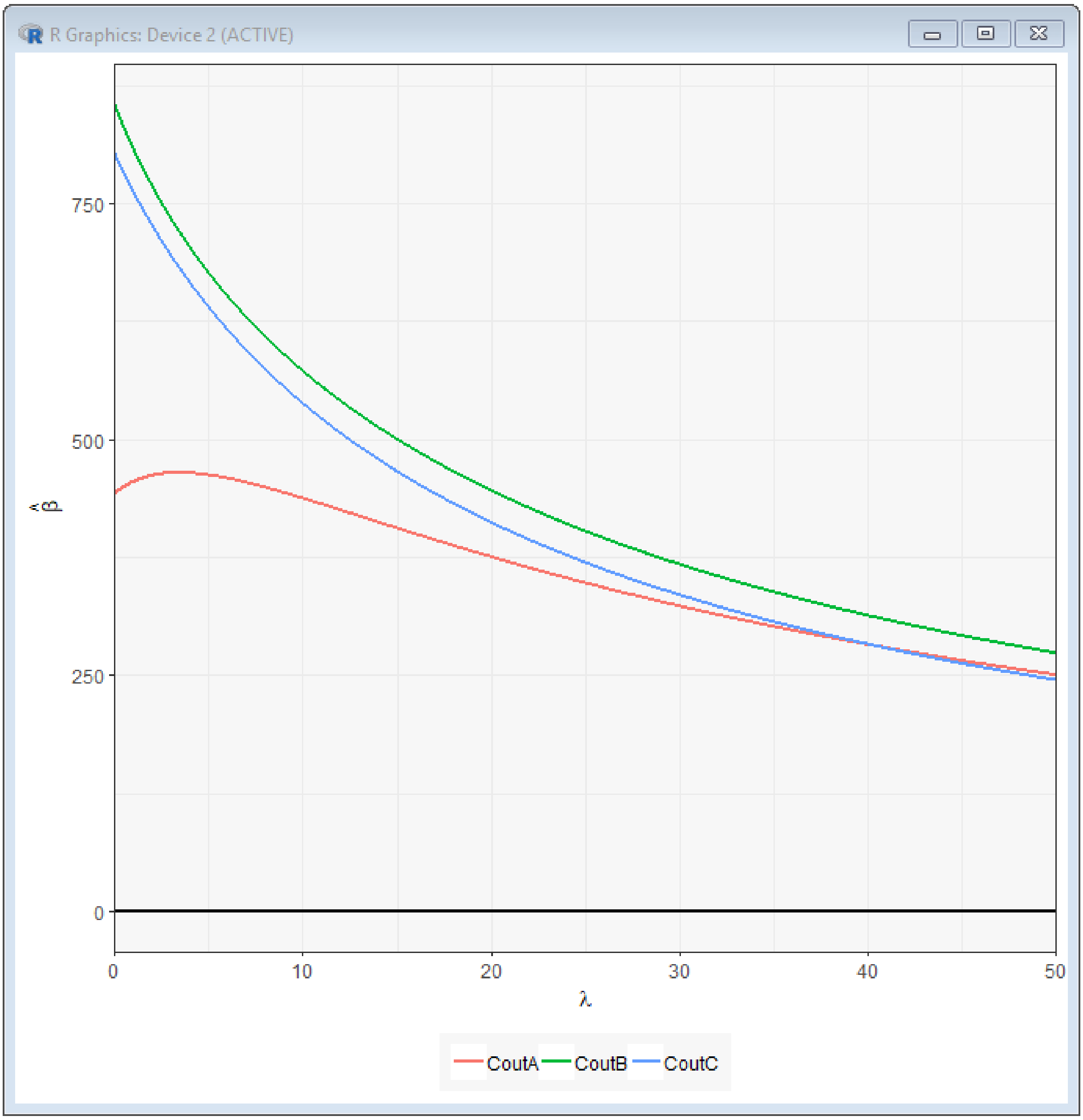
```
> fit<-lm.ridge(Couts~CoutA+CoutB+CoutC,lambda=0,data=mydata)
> fit$coef
   CoutA    CoutB    CoutC
442.7705 855.3994 803.4398
> coef(fit)[1] #ordonnée à l'origine

35102.9
> fit$GCV
       0
91958.05
> |
```


R Console
> #en variant le paramètre lambda de 0.0 à 50 par pas de 0.001
> fit<-lm.ridge(Couts~CoutA+CoutB+CoutC,lambda=seq(0,50,by=0.001),data=mydata)
> plot(fit)
R Graphics: Device 2 (ACTIVE)
t(x$coef)
300
400
500
600
700
800
0
10
20
30
40
50
x$lambda

```
> #version un peu plus sexy du graphique
> library("dplyr")
> library("broom") #pour la fonction tidy()
> library("psych")
> library("ggplot2")
> reg_ridge_df<-fit %>% tidy()
> headTail(reg_ridge_df)
  lambda       GCV  term estimate  scale
1      0  91958.05 CoutA   442.77 214.32
2      0  91955.59 CoutA   442.79 214.32
3      0  91953.13 CoutA    442.8 214.32
4      0  91950.67 CoutA   442.82 214.32
5    ...       ...  <NA>      ...    ...
6     50 122794.37 CoutC   244.79 167.71
7     50 122794.97 CoutC   244.79 167.71
8     50 122795.57 CoutC   244.79 167.71
9     50 122796.16 CoutC   244.78 167.71
> ggplot(reg_ridge_df,aes(lambda,estimate,color=term))+
+ geom_line(size=1)+geom_hline(yintercept=0,size=1)+
+ scale_x_continuous(expand=c(0,0))+
+ labs(x=expression(lambda),y=expression(hat(beta)))+theme_bw()+
+ theme(panel.background=element_rect(fill="gray97"),
+ legend.background=element_rect(fill="gray97"),
+ legend.position="bottom",legend.title=element_blank())
> 
```

Ce qui donne:

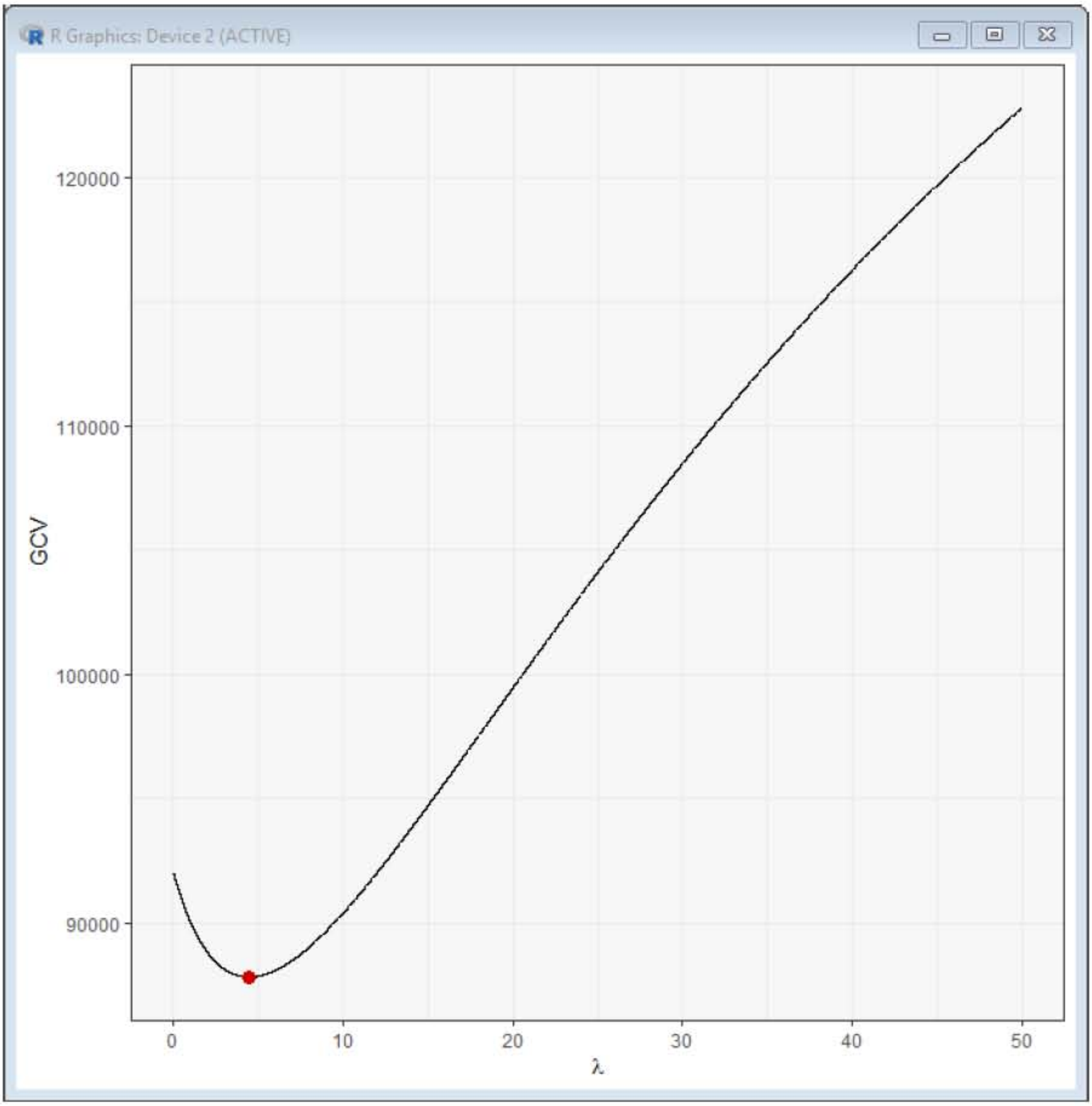


Pour extraire la valeur optimale de lambda selon la méthode GCV (et donc toujours sur la base du package MASS), on écrira:

```
> MASS::select(fit)
modified HKB estimator is 0.9975516
modified L-W estimator is 0.6957899
smallest value of GCV  at 4.442
> lambdas<-fit %>% glance()
> lambdas$lambdaGCV
[1] 4.442
>
```

```
> ggplot(reg_ridge_df,aes(lambda,GCV))+geom_line()+
+ geom_point(x=lambdas$lambdaGCV,y=min(reg_ridge_df$GCV),color="red3",size=3)+
+ labs(x=expression(lambda),y=expression(GCV))+theme_bw()+
+ theme(panel.background=element_rect(fill="gray97"))
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
GCV
120000
110000
100000
90000
0
10
20
30
40
50
λ

Avec le package ridge

```
> library("ridge")
> # lambda optimal automatiquement calculé par une méthode semi-paramétrique
> fit<-linearRidge(Couts~ CoutA+CoutB+CoutC, data = mydata,scaling = "scale")
Warning message:
In linearRidge(Couts ~ CoutA + CoutB + CoutC, data = mydata, scaling = "scale") :
  lambda is chosen automatically so scaling is set to "corrForm"
> summary(fit)

Call:
linearRidge(formula = Couts ~ CoutA + CoutB + CoutC, data = mydata,
    scaling = "corrForm")


Coefficients:
             Estimate Scaled estimate Std. Error (scaled) t value (scaled)
(Intercept) 35796.125              NA                  NA               NA
CoutA           2.158        2015.768            1234.859            1.632
CoutB           3.726        3326.788            1208.857            2.752
CoutC           4.305        3146.940            1119.922            2.810
            Pr(>|t|)
(Intercept)       NA
CoutA        0.10260
CoutB        0.00592
CoutC        0.00495

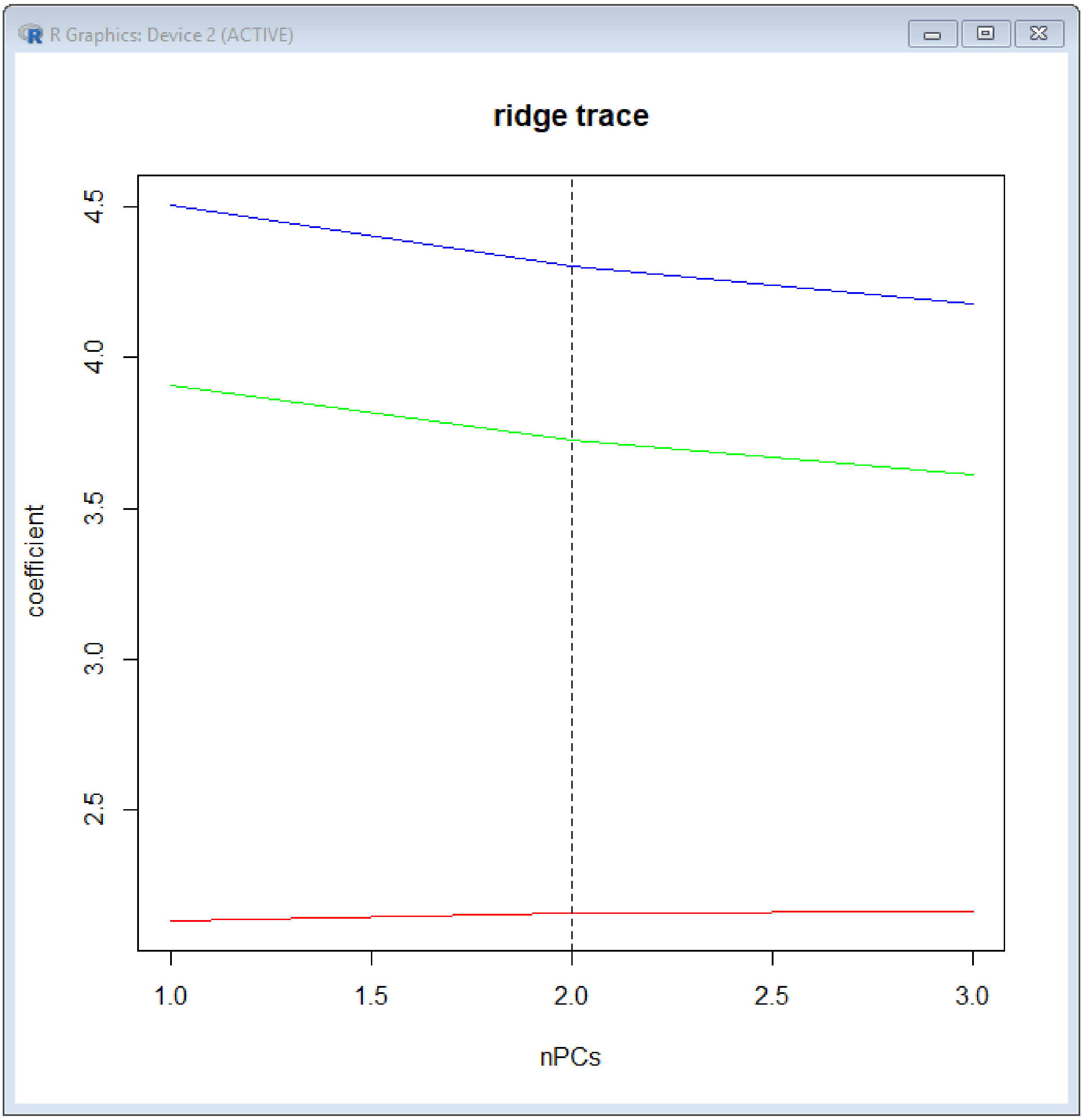
Ridge parameter: 0.1129874, chosen automatically, computed using 2 PCs

Degrees of freedom: model 2.616 , variance 2.292 , residual 2.939

> fit$lambda
[1] 0.06192115 0.11298745 0.14766389
> plot(fit)
> 
```

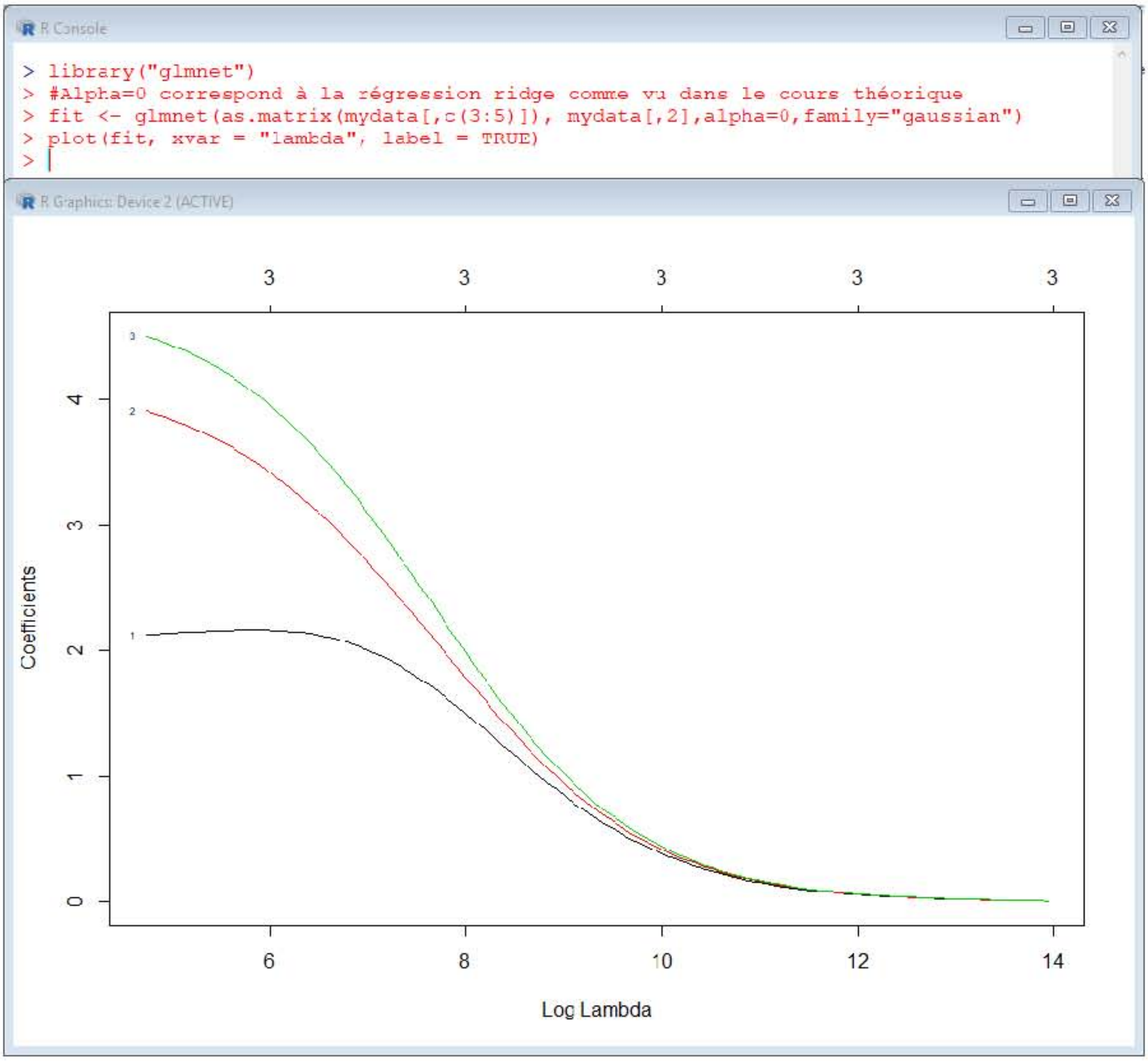
On voit donc ici une différent significative. Mais c'est dû à deux choses:

1. Les coefficients ne sont pas standardisés
2. Le package utilise la méthode proposée par Cule et al (2012)

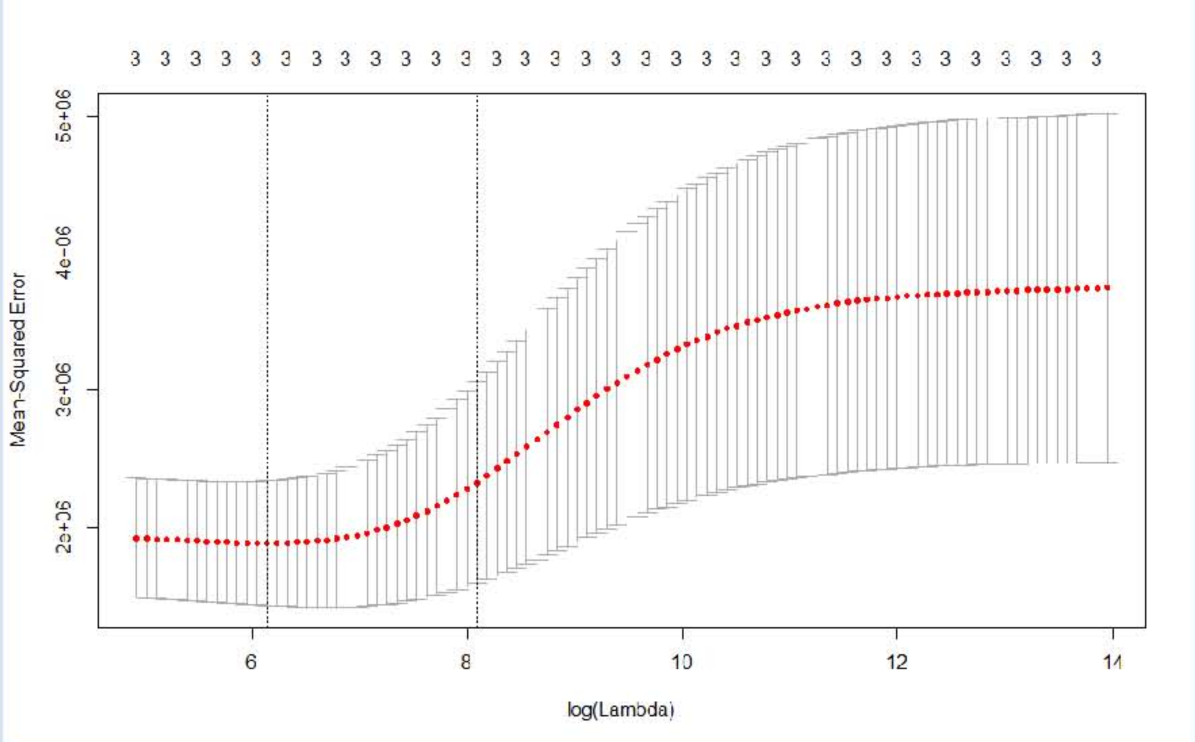

R Graphics: Device 2 (ACTIVE)
ridge trace
coefficient
4.5
4.0
3.5
3.0
2.5
1.0
1.5
2.0
2.5
3.0
nPCs

Avec le package glmnet

Là encore on va voir des différences entre le lambda. Bien que glmnet utilise à ce jour aussi le GCV comme MASS, la différence est significative.



```
> #attention sans set.seed le résultat sera différent à chaque exécution!
> cv.glmmod <- cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=0,family="gaussian")
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> plot(cv.glmmod)
> (best.lambda <- cv.glmmod$lambda.min)
[1] 461.3203
> log(best.lambda)
[1] 6.134093
>
```




```
> #coefficients correspondants aux lambda optimal
> coef(cv.glmmod,"lambda.min")[,1]
 (Intercept)        CoutA        CoutB        CoutC
36476.855113     2.160472     3.342054     3.864020
>
```

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #toujours avec les packages dlplyr et broom
> #on peut formater la sorte afin de voir l'évolution
> #de l'algorithme
> fit<-fit %>% tidy()
> psych::headTail(fit)
          term step estimate    lambda dev.ratio
1  (Intercept)    1 44355.42 1142725.5         0
2        CoutA    1        0 1142725.5         0
3        CoutB    1        0 1142725.5         0
4        CoutC    1        0 1142725.5         0
5         <NA>  ...      ...       ...       ...
6  (Intercept)  100 35494.02    114.27      0.64
7        CoutA  100     2.13    114.27      0.64
8        CoutB  100     3.91    114.27      0.64
9        CoutC  100     4.51    114.27      0.64
> 
```

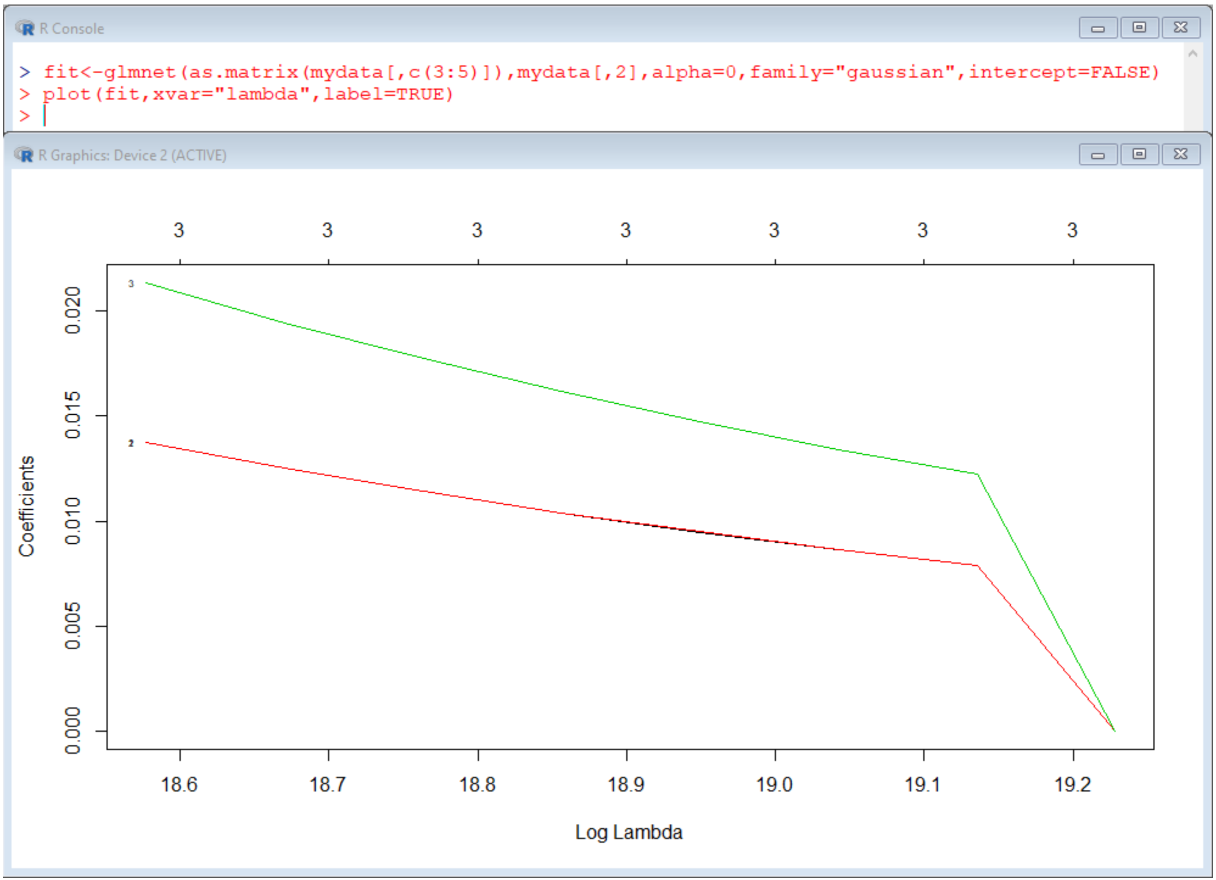
On peut aussi faire des prévisions:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("glmnet")
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit_ridge<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=0,family="gaussian")
> cv.glmmod.ridge <- cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=0,family="gaussian")
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> predict(fit_ridge, newx = as.matrix(mydata[,c(3:5)]), s = cv.glmmod.ridge$lambda.min)
             1
 [1,] 42972.41
 [2,] 43529.00
 [3,] 43754.88
 [4,] 43764.12
 [5,] 45287.08
 [6,] 44512.89
 [7,] 43050.87
 [8,] 43811.67
 [9,] 42924.14
[10,] 46900.12
[11,] 46242.63
[12,] 43625.85
[13,] 42792.46
[14,] 44652.33
[15,] 45297.28
[16,] 47135.82
[17,] 44144.68
[18,] 45240.58
[19,] 43114.21
> 
```

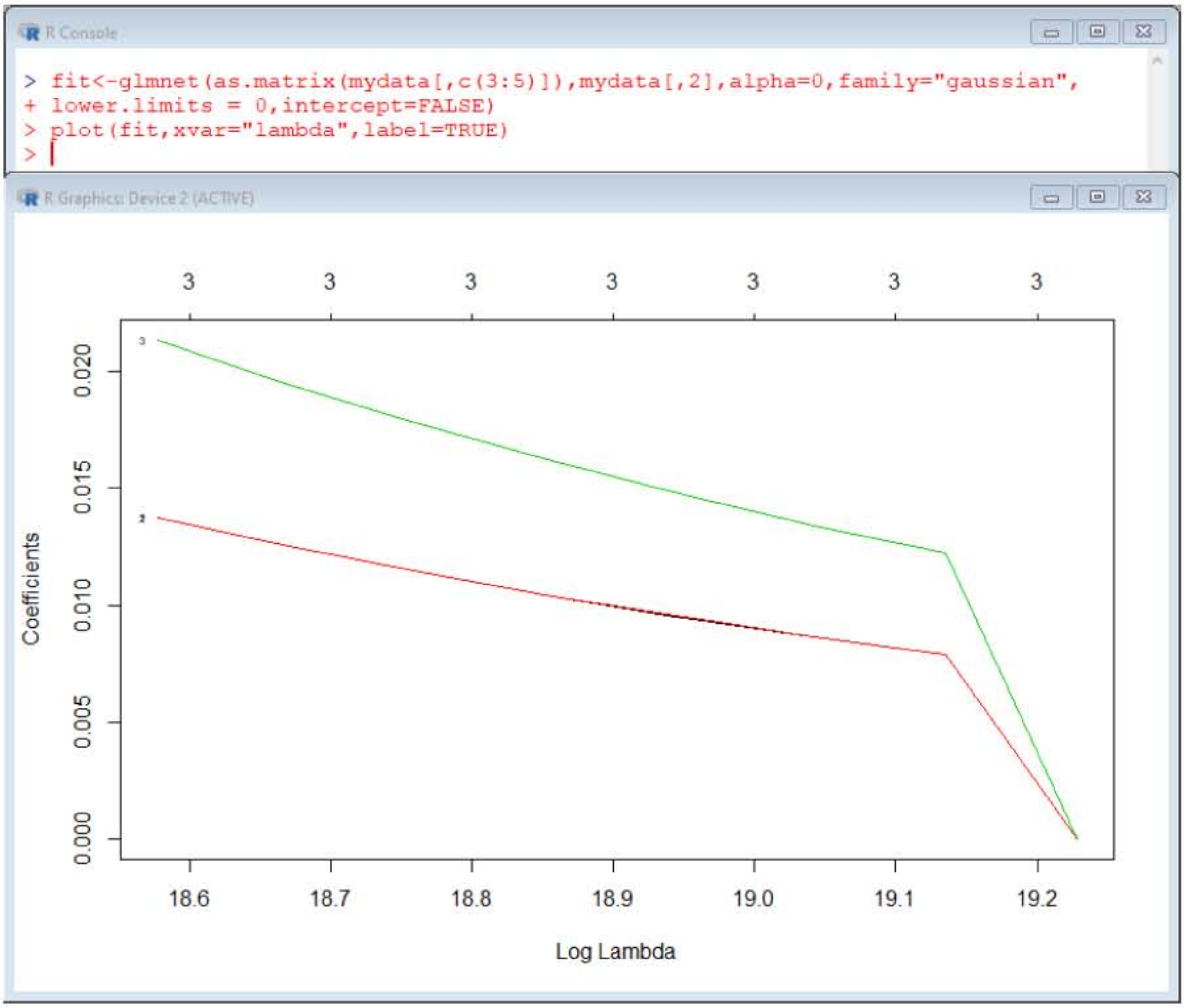
Forçage à l'origine

On peut forcer la même analyse que précédemment à passer par l'origine en écrivant:



Coefficients non-négatifs

On peut forcer aussi en plus les coefficients à être non-négatifs (au cas où…!) avec:

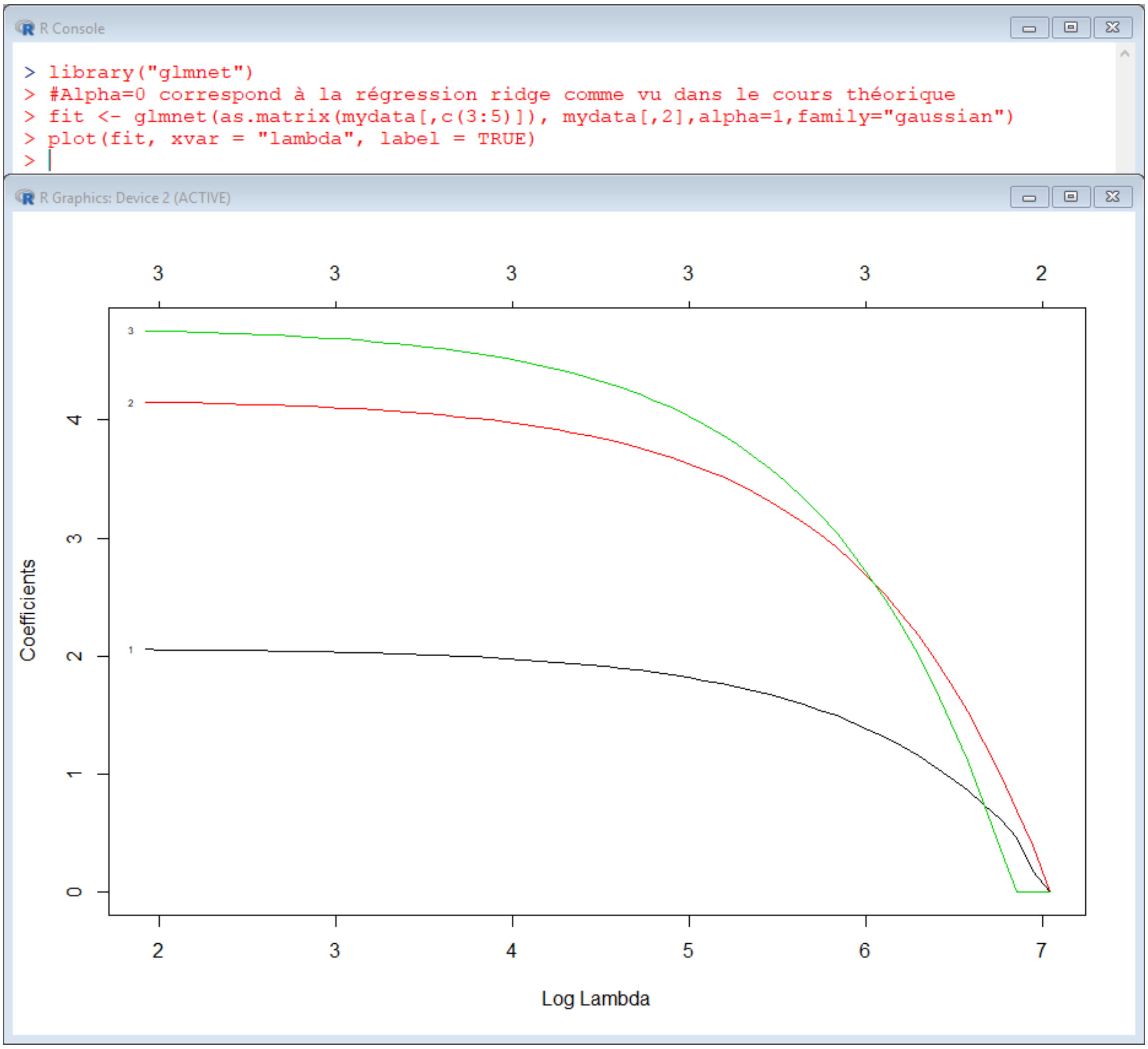

R Console
> fit<-glmnet(as.matrix(mydata[,c(3:5)]),mydata[,2],alpha=0,family="gaussian",
+ lower.limits = 0,intercept=FALSE)
> plot(fit,xvar="lambda",label=TRUE)
>
R Graphics: Device 2 (ACTIVE)
3 3 3 3 3 3 3
0.020
0.015
0.010
0.005
0.000
Coefficients
18.6 18.7 18.8 18.9 19.0 19.1 19.2
Log Lambda

Exemple 267.: Régression LASSO (L_1)

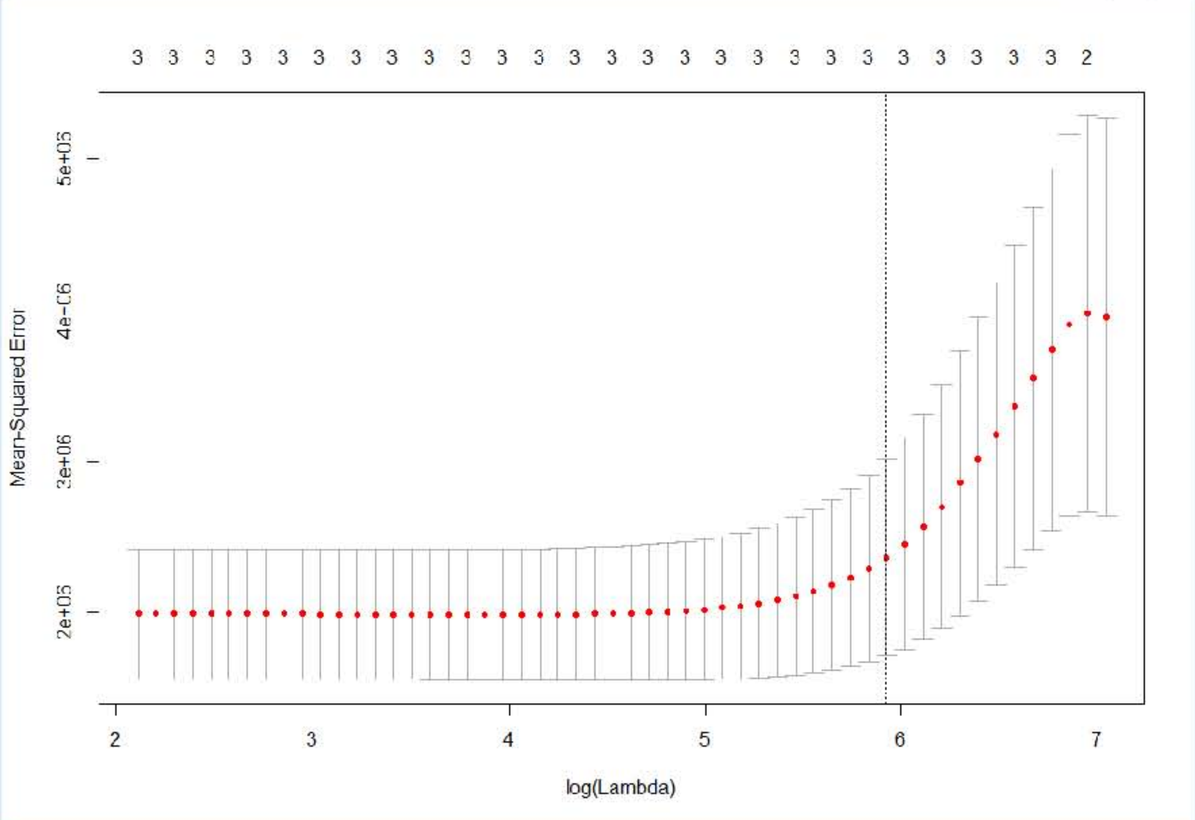
R 3.5.1

Nous allons ici nous concentrer que sur le package **glmnet** puisque nous nous sommes bien rendus compte de ses avantages.

Il suffit reprendre l'exemple précédent mais avec alpha qui vaut 1:

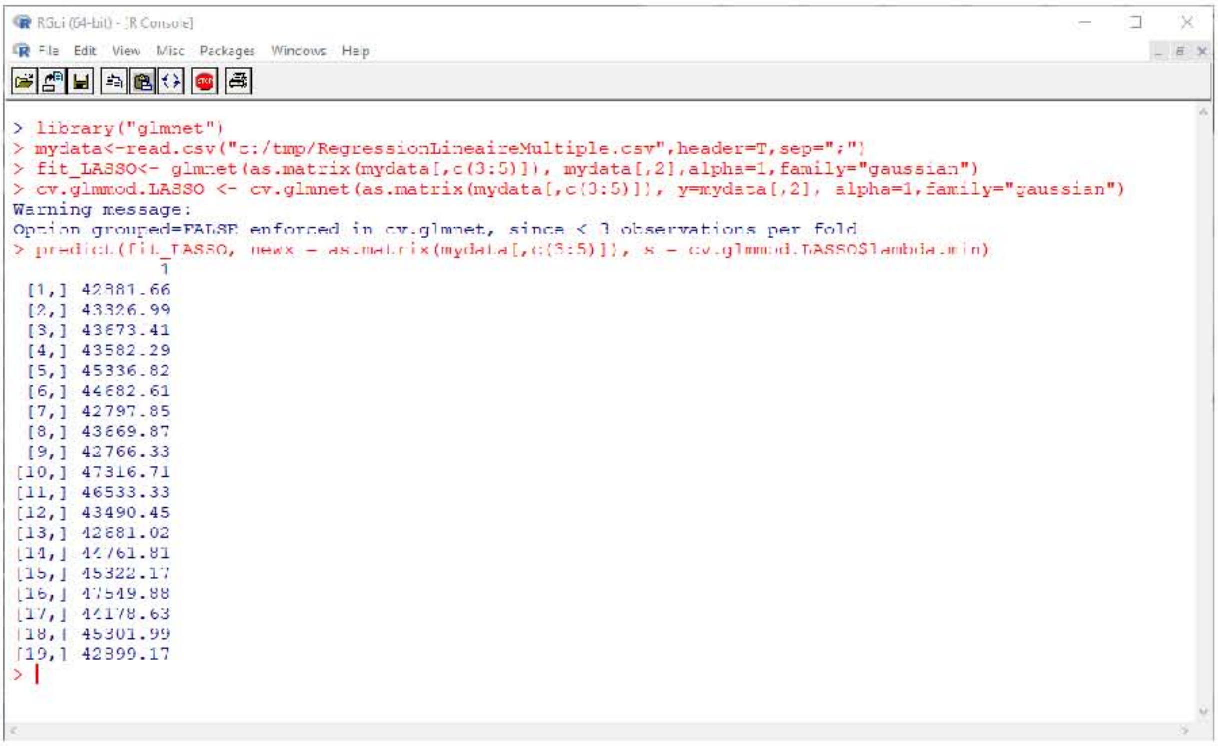


```
> #attention sans set.seed le résultat sera différent à chaque exécution!
> cv.glmmod <- cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=1,family="gaussian")
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> plot(cv.glmmod)
> (best.lambda <- cv.glmmod$lambda.min)
[1] 48.32364
> log(best.lambda)
[1] 3.878024
>
```




```
> coef(cv.glmmod,"lambda.min")[,1]
 (Intercept)        CoutA        CoutB        CoutC 
35527.858022     1.985055     3.998204     4.542858 
>
```

Sinon on peut comme toujours faire des prévisions:

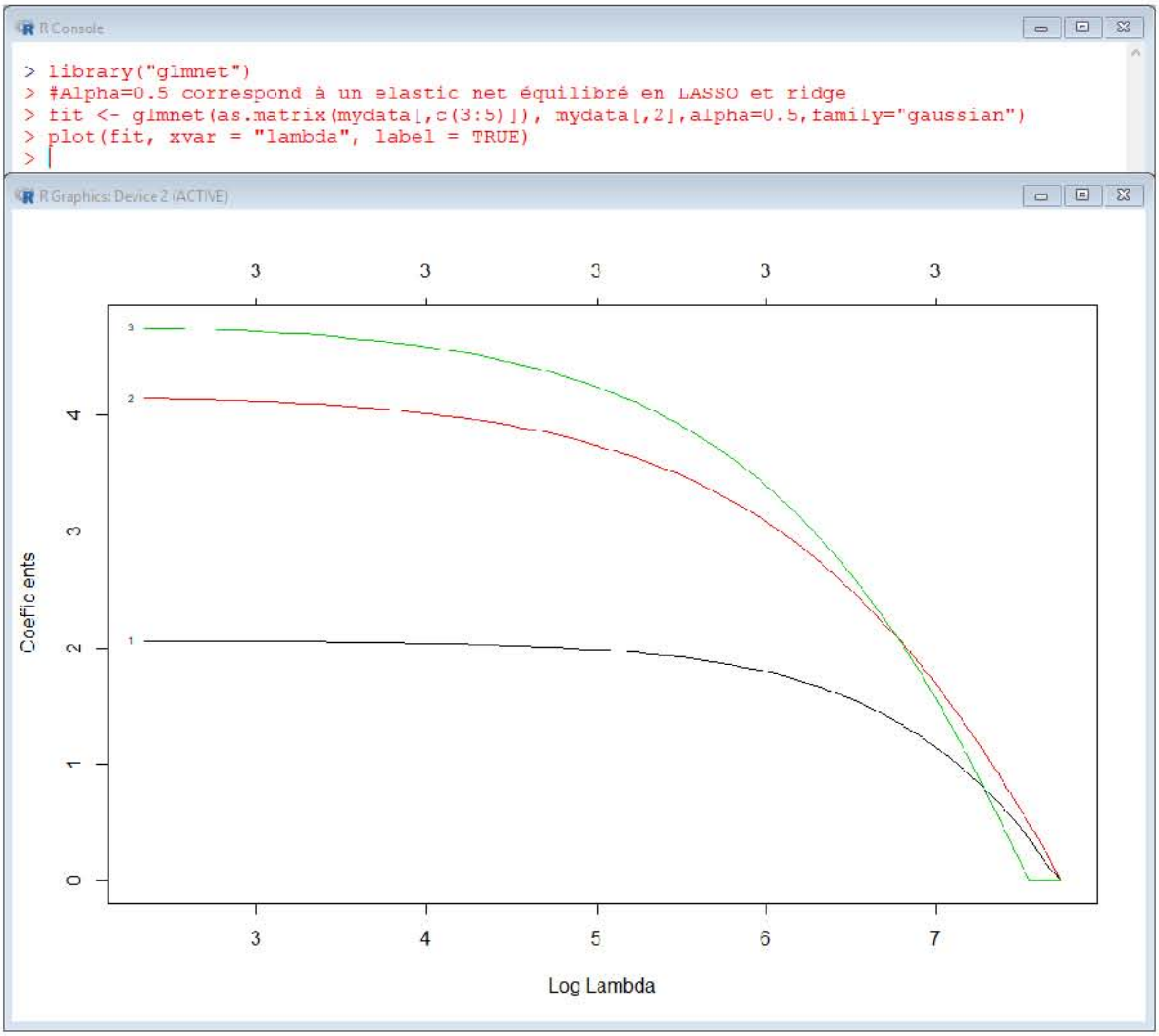


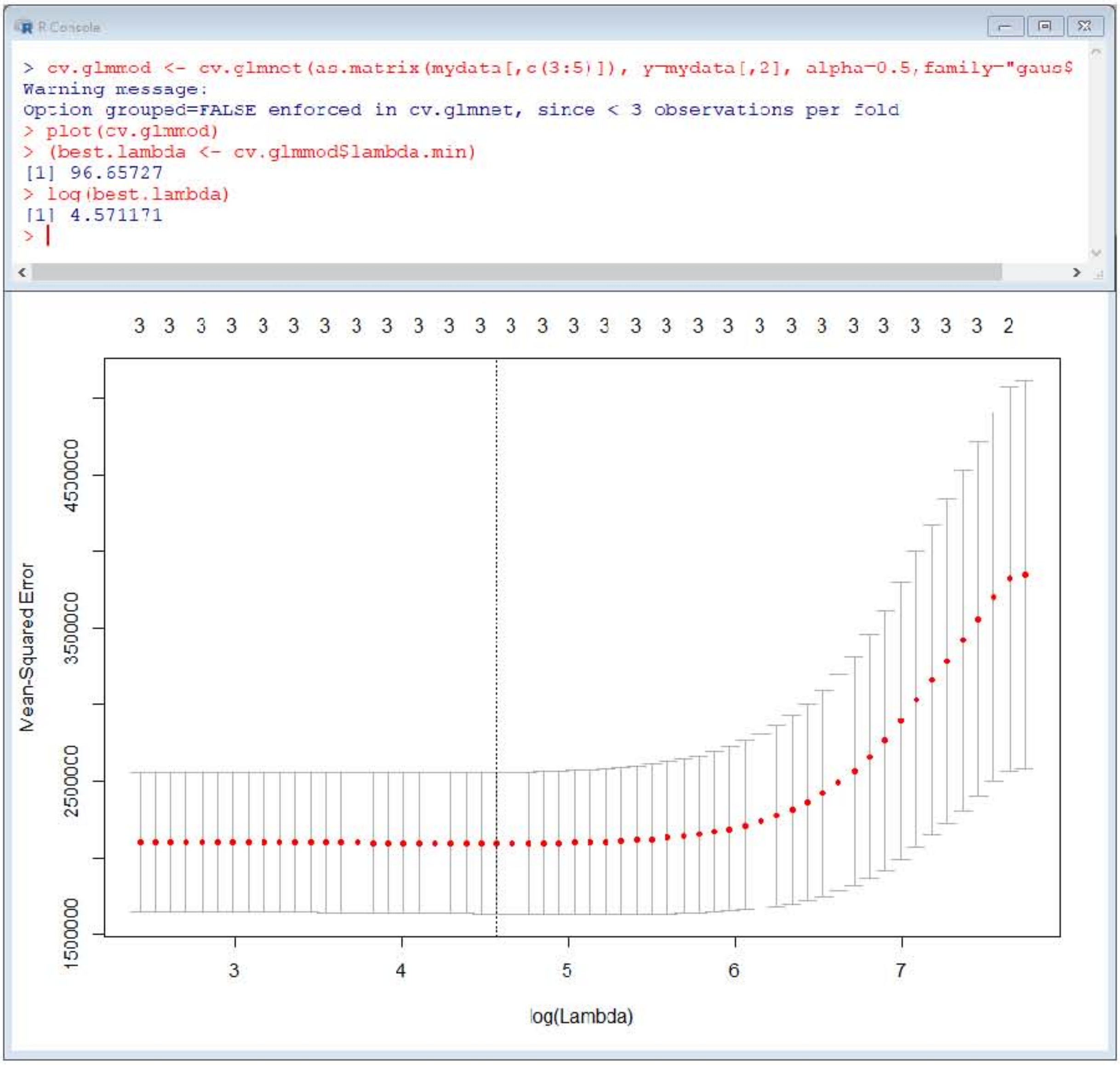
```
> library("glmnet")
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit_LASSO<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=1,family="gaussian")
> cv.glmmod.LASSO <- cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=1,family="gaussian")
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> predict(fit_LASSO, newx = as.matrix(mydata[,c(3:5)]), s = cv.glmmod.LASSO$lambda.min)
             1
 [1,] 42881.66
 [2,] 43326.99
 [3,] 43673.41
 [4,] 43582.29
 [5,] 45336.82
 [6,] 44682.61
 [7,] 42797.85
 [8,] 43669.87
 [9,] 42766.33
[10,] 47316.71
[11,] 46533.33
[12,] 43490.45
[13,] 42681.02
[14,] 44761.81
[15,] 45322.17
[16,] 47549.88
[17,] 44178.63
[18,] 45301.99
[19,] 42399.17
>
```

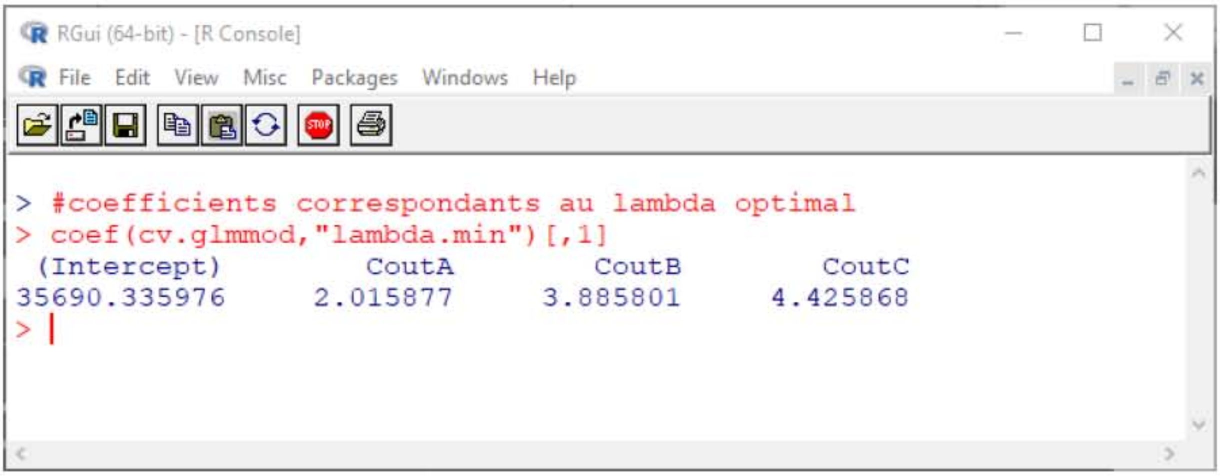
Exemple 268.: Régression Elastic Net (L_1+L_2)

R 3.5.1

Pour l'elastic net on va à nouveau se focaliser sur le package **glmnet** mais avec un alpha de 0.5.

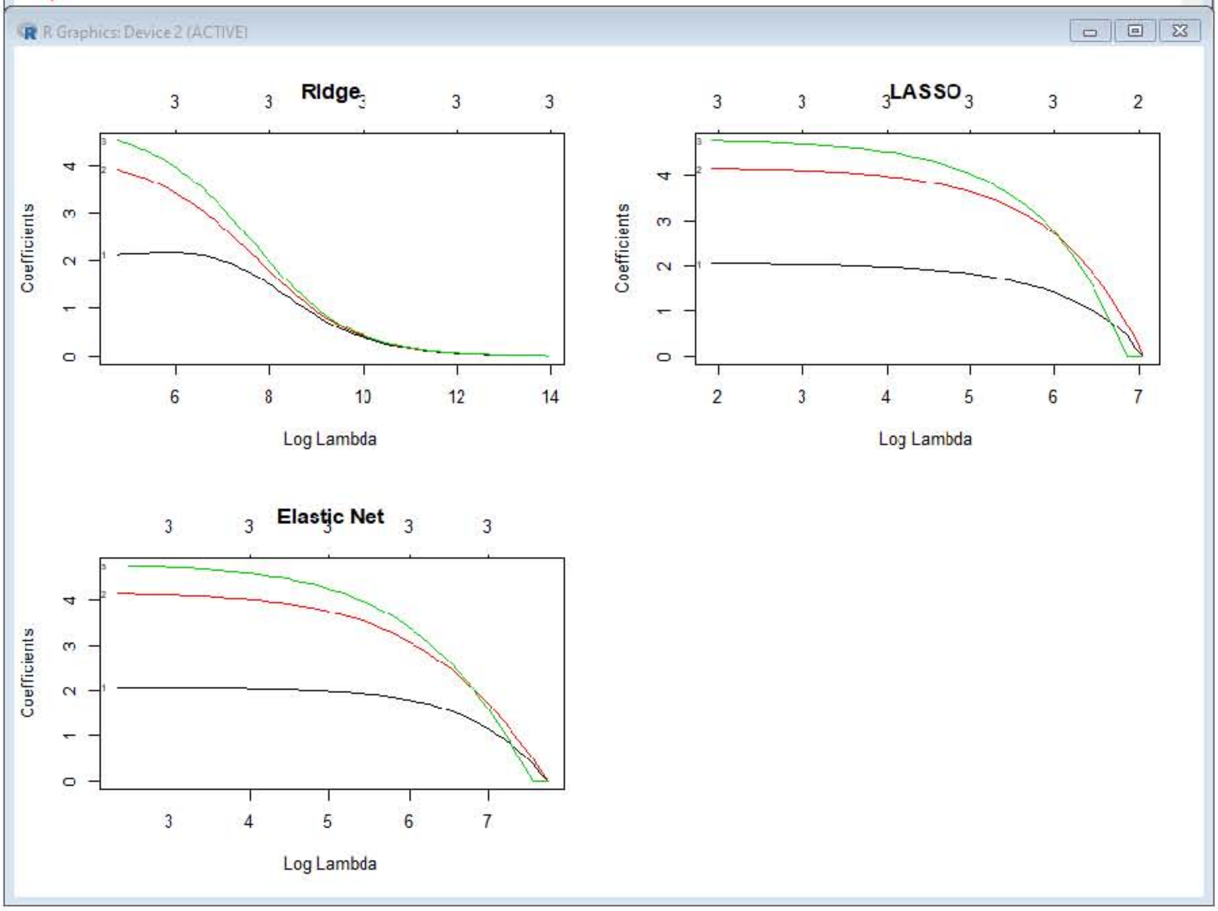






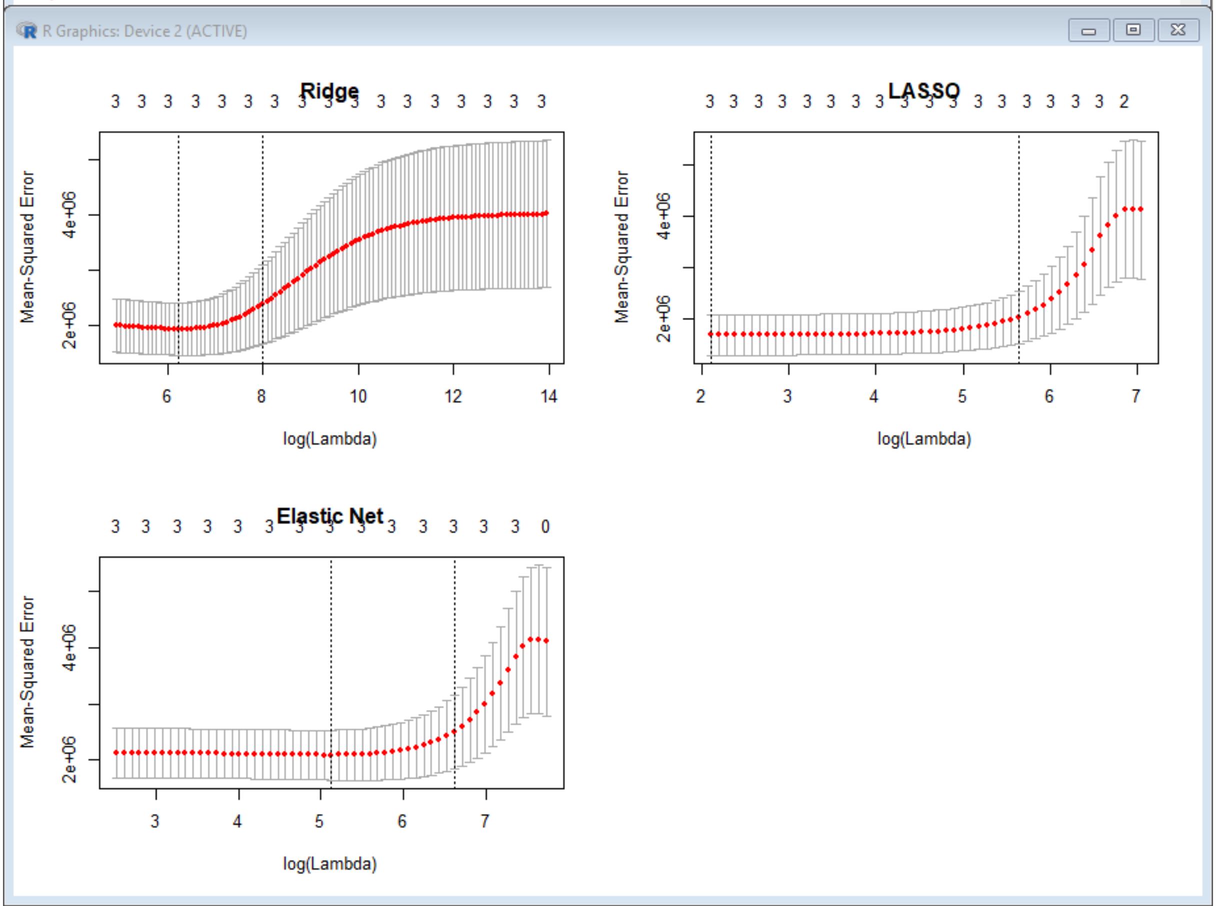
Finalement on peut "s'amuser" à comparer tous les modèles:

```
> library("glmnet")
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> fit_OLS<-lm(CoutS~CoutA+CoutB+CoutC,data=mydata)
> fit_ridge<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=0,family="gaussian")
> fit_LASSO<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=1,family="gaussian")
> fit_EN<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=0.5,family="gaussian")
> par(mfrow=c(2,2))
> plot(fit_ridge, xvar = "lambda", label = TRUE, main="Ridge")
> plot(fit_LASSO, xvar = "lambda", label = TRUE, main="LASSO")
> plot(fit_EN, xvar = "lambda", label = TRUE, main="Elastic Net")
>
```



Ainsi que:

```
> par(mfrow=c(2,2))
> plot(cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=0,family="gaussian"),$
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> plot(cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=1,family="gaussian"),$
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> plot(cv.glmnet(as.matrix(mydata[,c(3:5)]), y=mydata[,2], alpha=0.5,family="gaussian"$
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
>
```

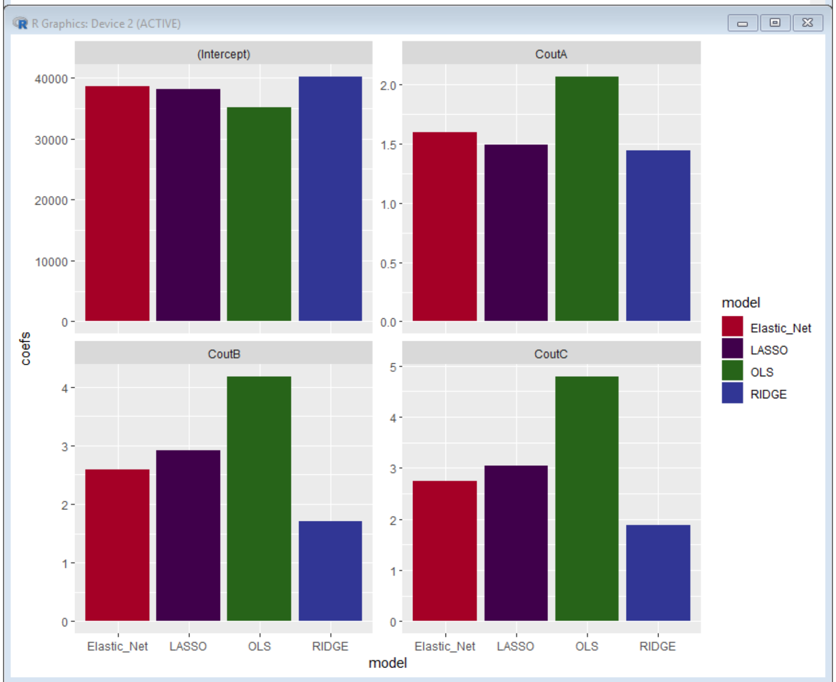


On peut comparer aussi tous les coefficients graphiquement. Pour cela on va d'abord traiter les données (y'a peut-être plus efficace mais bon…):

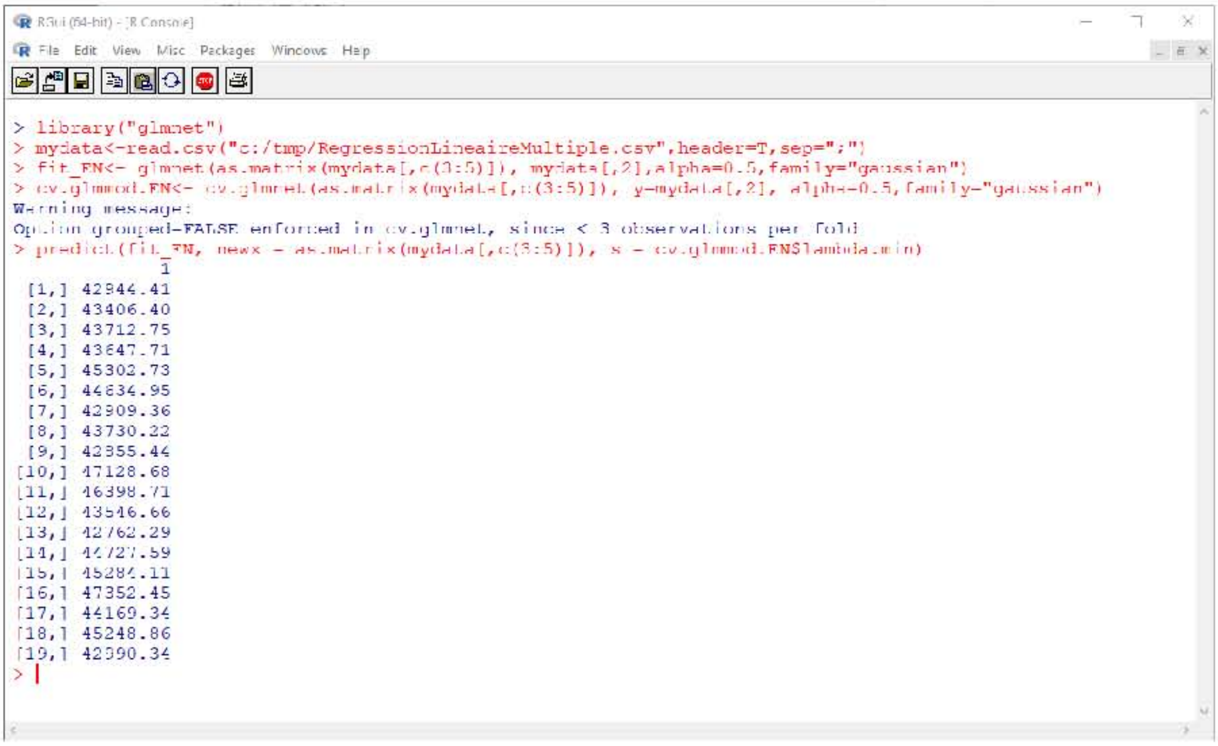
```
> library("glmnet")
> library("tidyr")
>
> mydata<-read.csv("RegressionLineaireMultiple.csv",header=T,sep=";")
> fit_OLS<-lm(Couts~CoutA+CoutB+CoutC,data=mydata)
> fit_ridge<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=0,family="gaussian")
> fit_LASSO<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=1,family="gaussian")
> fit_EN<- glmnet(as.matrix(mydata[,c(3:5)]), mydata[,2],alpha=0.5,family="gaussian")
>
> df_coefs = cbind(coef(fit_OLS),
+             as.data.frame.matrix(cbind(
+              coef(fit_ridge, s = cv.glmnet(as.matrix(mydata[, c(3:5)]), y = mydata[, 2], alpha = 0, family = "gaussian")$lambda.1se),
+              coef(fit_LASSO, s = cv.glmnet(as.matrix(mydata[, c(3:5)]), y = mydata[, 2], alpha = 1, family = "gaussian")$lambda.1se),
+              coef(fit_EN, s = cv.glmnet(as.matrix(mydata[, c(3:5)]), y = mydata[, 2], alpha = 0.5, family = "gaussian")$lambda.1se))))
Warning messages:
1: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
2: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
3: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> colnames(df_coefs) <- c('OLS', 'RIDGE', 'LASSO', 'Elastic_Net')
> df_coefs
                  OLS        RIDGE        LASSO  Elastic_Net
(Intercept) 35102.900449 40103.873951 38101.053534 38518.069800
CoutA           2.065953     1.444314     1.494429     1.601419
CoutB           4.176356     1.703065     2.919954     2.590357
CoutC           4.790641     1.889658     3.042825     2.752742
> df_coefs['variable'] <- rownames(df_coefs)
> df_coefs
                  OLS        RIDGE        LASSO  Elastic_Net    variable
(Intercept) 35102.900449 40103.873951 38101.053534 38518.069800 (Intercept)
CoutA           2.065953     1.444314     1.494429     1.601419       CoutA
CoutB           4.176356     1.703065     2.919954     2.590357       CoutB
CoutC           4.790641     1.889658     3.042825     2.752742       CoutC
> df_coefs <- gather(df_coefs, model, coefs, OLS:Elastic_Net)
> df_coefs
      variable       model        coefs
1  (Intercept)         OLS 35102.900449
2        CoutA         OLS     2.065953
3        CoutB         OLS     4.176356
4        CoutC         OLS     4.790641
5  (Intercept)       RIDGE 40103.873951
6        CoutA       RIDGE     1.444314
7        CoutB       RIDGE     1.703065
8        CoutC       RIDGE     1.889658
9  (Intercept)       LASSO 38101.053534
10       CoutA       LASSO     1.494429
11       CoutB       LASSO     2.919954
12       CoutC       LASSO     3.042825
13 (Intercept) Elastic_Net 38518.069800
14       CoutA Elastic_Net     1.601419
15       CoutB Elastic_Net     2.590357
16       CoutC Elastic_Net     2.752742
```

… et on peut passer alors à la partie graphique:

```
> library("ggplot2")
> colfill <- c('#A50026','#40004B','#276419','#313695')
> ggplot(df_coefs, aes(x = model, y = coefs, fill = model)) +
+     geom_bar(stat = 'identity')  +
+     facet_wrap(~variable, ncol = 2, scales = "free_y") +
+     scale_fill_manual(values = colfill)
>
```



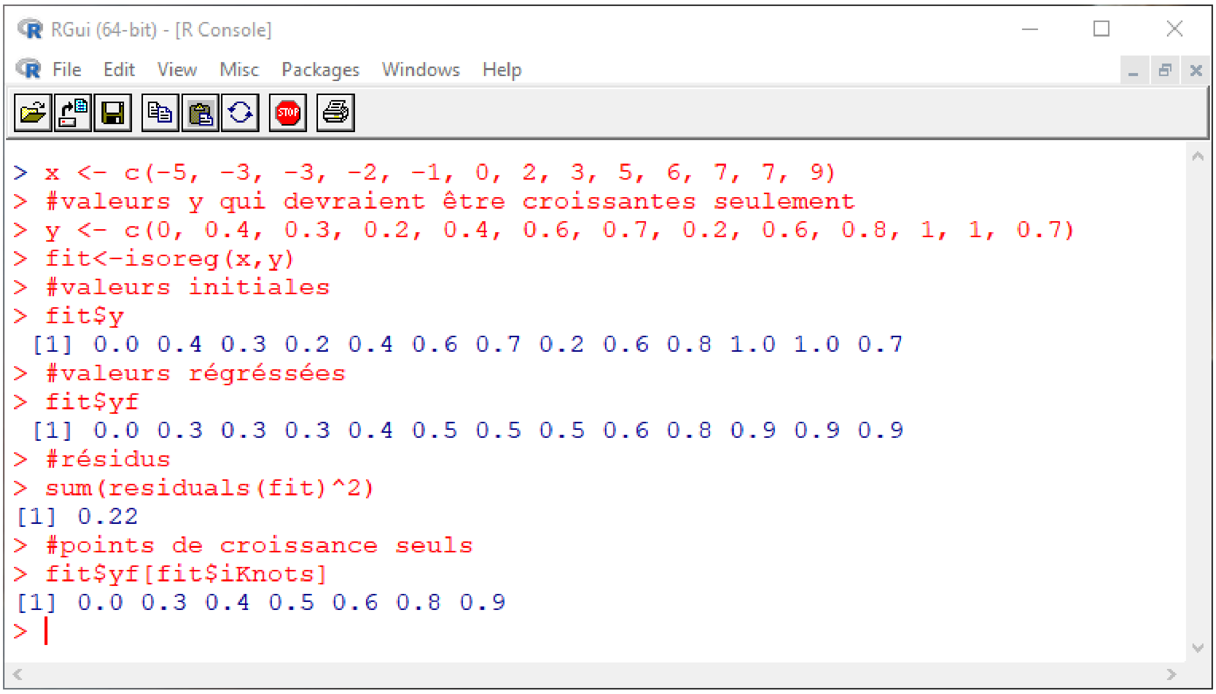
À nouveau on peut faire des prévisions:



Exemple 269.: Régression Isotonique

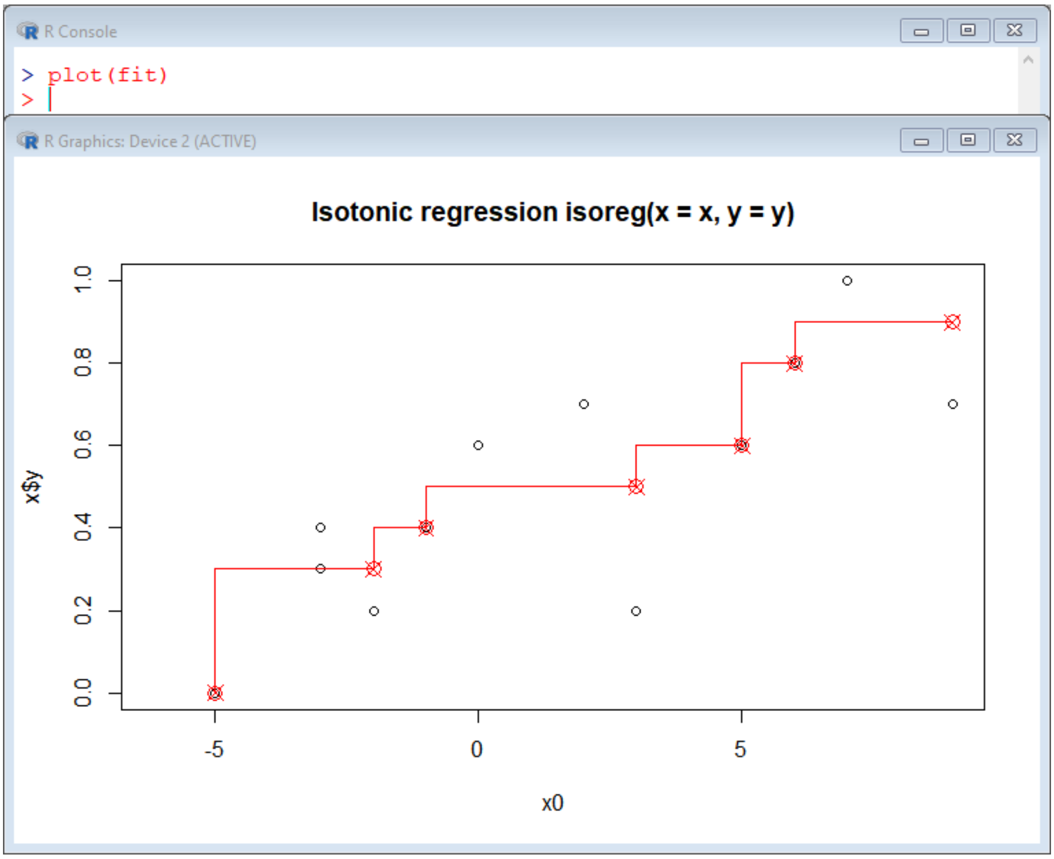
R 3.6.1

Reproduisons ici l'exemple détaillé fait à la main dans le cours théorique:

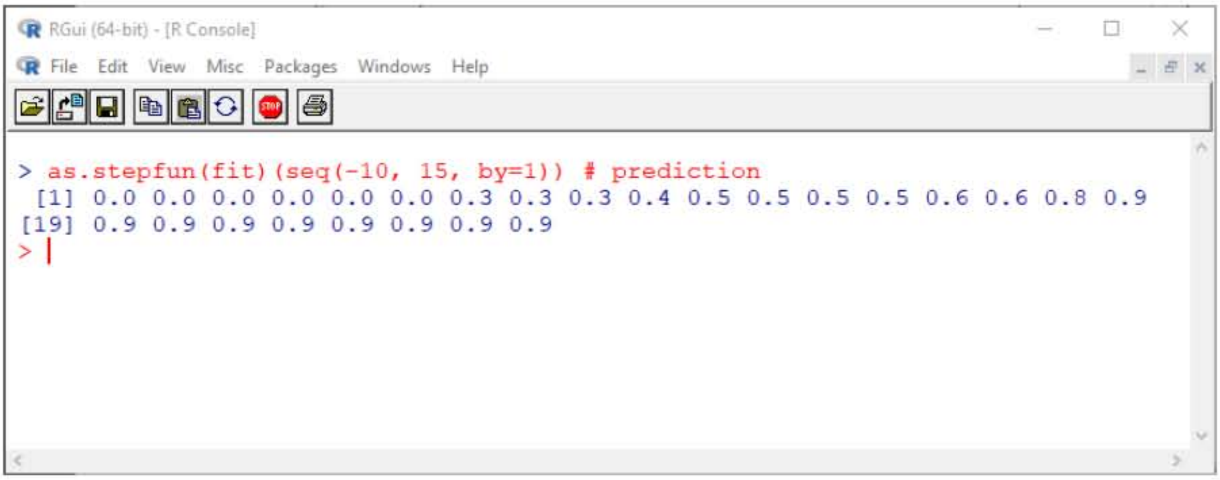


```
> x <- c(-5, -3, -3, -2, -1, 0, 2, 3, 5, 6, 7, 7, 9)
> #valeurs y qui devraient être croissantes seulement
> y <- c(0, 0.4, 0.3, 0.2, 0.4, 0.6, 0.7, 0.2, 0.6, 0.8, 1, 1, 0.7)
> fit<-isoreg(x,y)
> #valeurs initiales
> fit$y
 [1] 0.0 0.4 0.3 0.2 0.4 0.6 0.7 0.2 0.6 0.8 1.0 1.0 0.7
> #valeurs régréssées
> fit$yf
 [1] 0.0 0.3 0.3 0.3 0.4 0.5 0.5 0.5 0.6 0.8 0.9 0.9 0.9
> #résidus
> sum(residuals(fit)^2)
[1] 0.22
> #points de croissance seuls
> fit$yf[fit$iKnots]
[1] 0.0 0.3 0.4 0.5 0.6 0.8 0.9
>
```

Nous retrouvons bien les même résultats. Voyons si nous obtenons aussi le même graphique:



Voyons si nous pouvons faire de la prévision et de l'interpolation:



Exemple 270.: Support Vector Regression

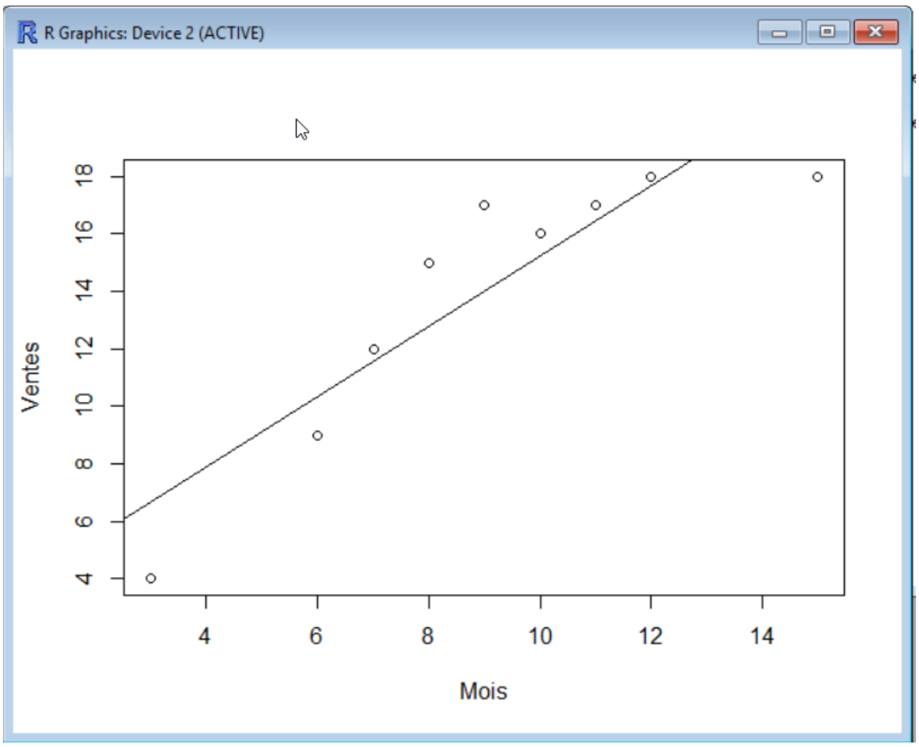
R 3.2.1 x86

Il paraît assez évident que SVM peut aussi être utilisé pour la régression linéaire ou non linéaire et performe bien souvent bien mieux que les techniques paramétriques classiques. Voyons comment avec un exemple particulier univarié.

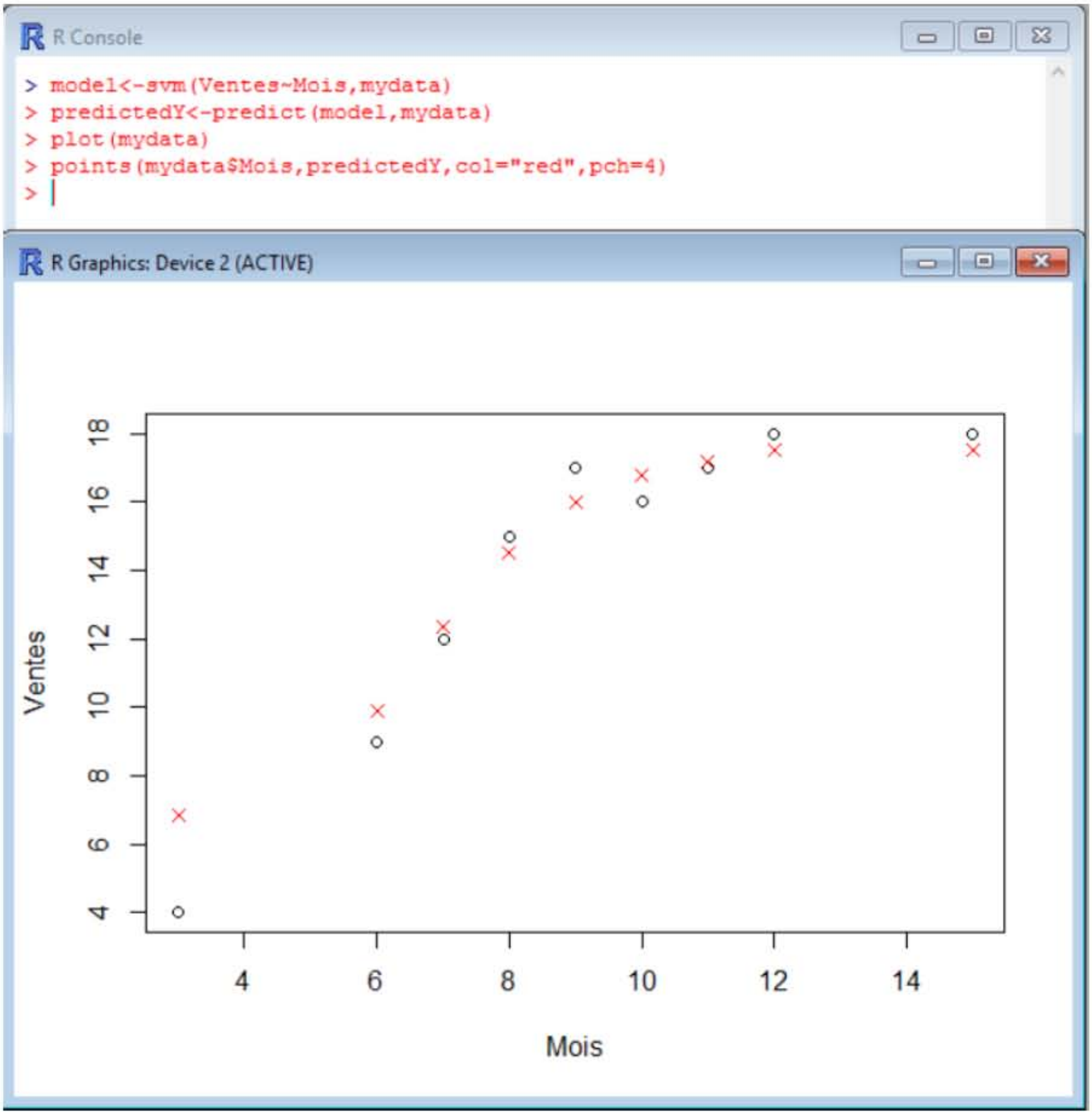
Commençons avec le cas classique:

```
> mydata<-read.csv("c:/tmp/RegressionUnivarieeSVM.csv",sep=";",header=T)
> mydata
  Mois Ventes
1    3      4
2    6      9
3    7     12
4    8     15
5    9     17
6   10     16
7   11     17
8   12     18
9   15     18
> plot(mydata)
> model <- lm(Ventes ~ Mois, mydata)
> abline(model)
>
```

Ce qui donne:



Et en utilisant le SVM, le résultat est bien évidemment meilleur:

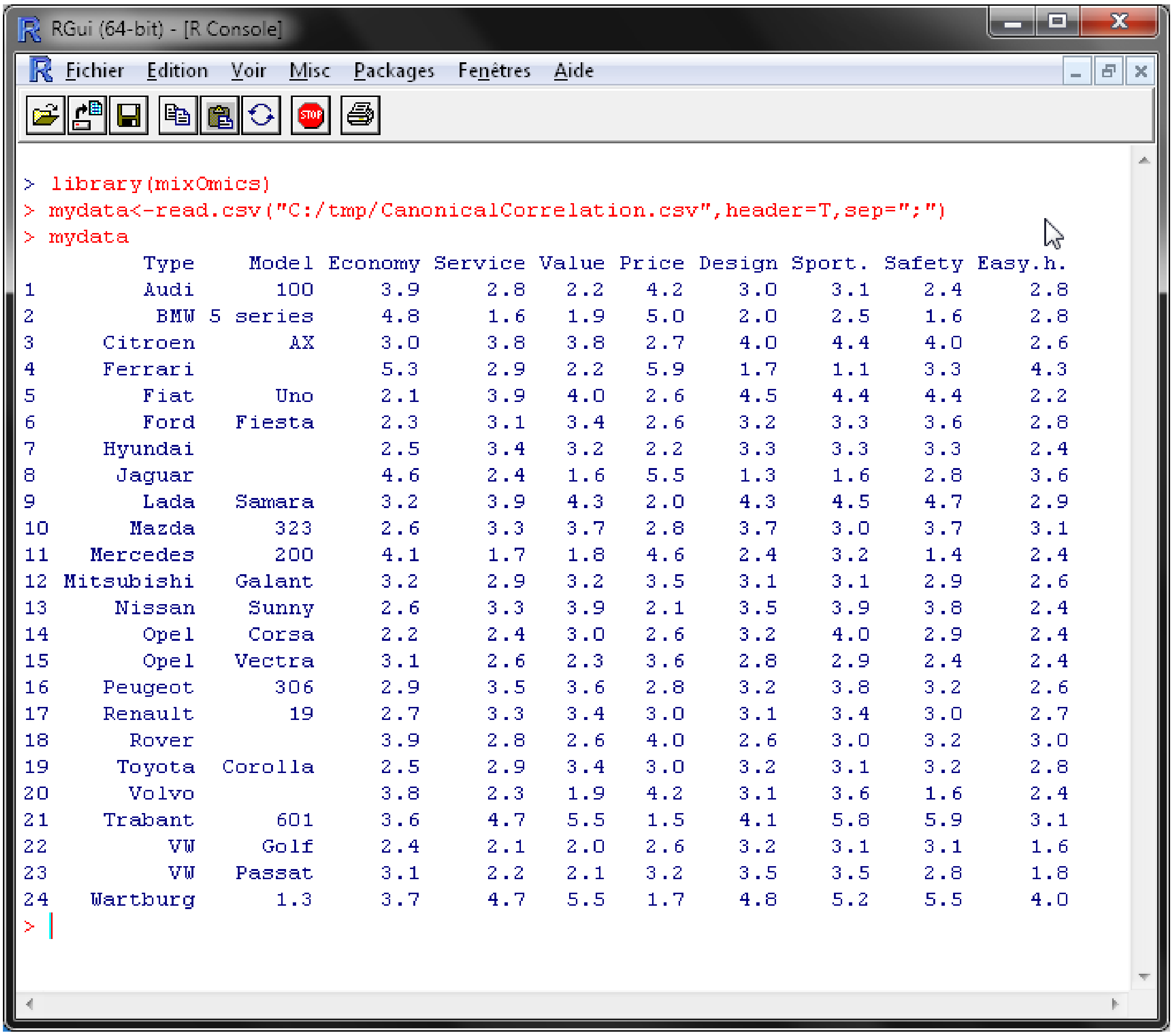


Exemple 271.: Analyse de la corrélation canonique

R 3.1.1 et R 4.1

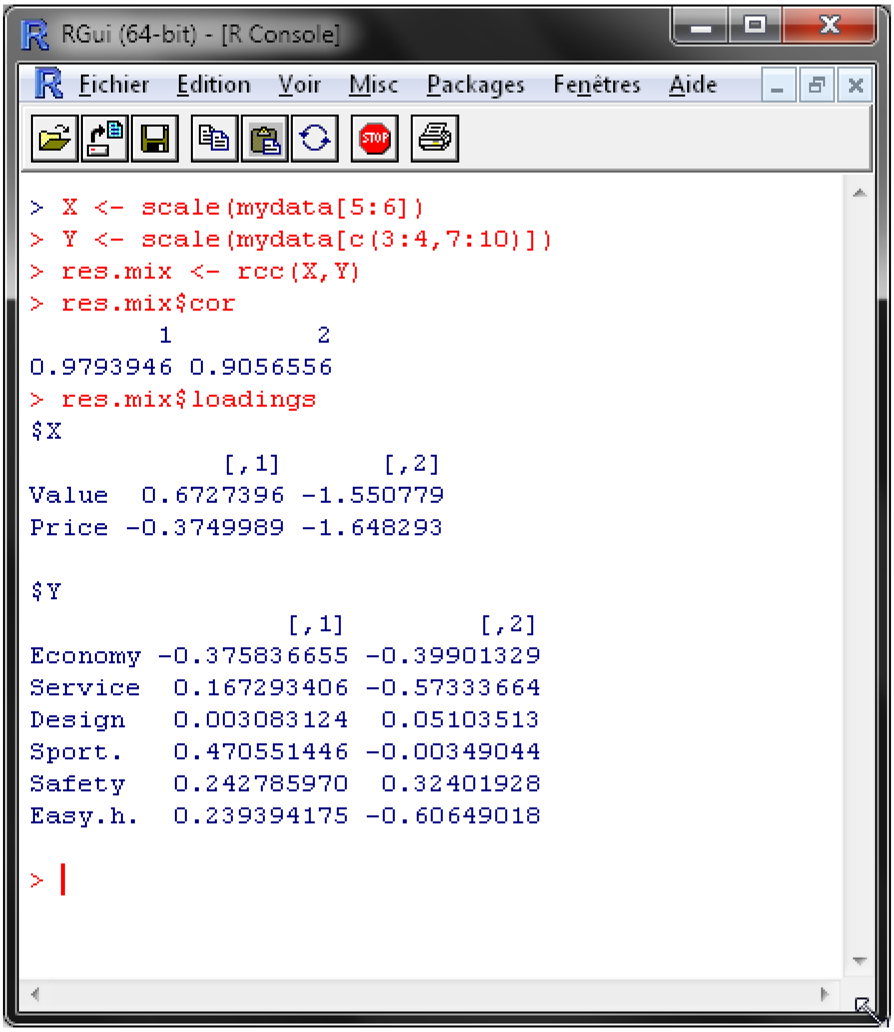
Commençons par les captures que nous avions fait à l'époque de R 3.1.1.

Nous utilisons le jeu de données du cours théorique:



```
> library(mixOmics)
> mydata<-read.csv("C:/tmp/CanonicalCorrelation.csv",header=T,sep=";")
> mydata
         Type     Model Economy Service Value Price Design Sport. Safety Easy.h.
1        Audi       100     3.9     2.8   2.2   4.2    3.0    3.1    2.4     2.8
2          BMW 5 series     4.8     1.6   1.9   5.0    2.0    2.5    1.6     2.8
3     Citroen        AX     3.0     3.8   3.8   2.7    4.0    4.4    4.0     2.6
4     Ferrari               5.3     2.9   2.2   5.9    1.7    1.1    3.3     4.3
5        Fiat       Uno     2.1     3.9   4.0   2.6    4.5    4.4    4.4     2.2
6        Ford    Fiesta     2.3     3.1   3.4   2.6    3.2    3.3    3.6     2.8
7     Hyundai               2.5     3.4   3.2   2.2    3.3    3.3    3.3     2.4
8      Jaguar               4.6     2.4   1.6   5.5    1.3    1.6    2.8     3.6
9        Lada    Samara     3.2     3.9   4.3   2.0    4.3    4.5    4.7     2.9
10      Mazda       323     2.6     3.3   3.7   2.8    3.7    3.0    3.7     3.1
11   Mercedes       200     4.1     1.7   1.8   4.6    2.4    3.2    1.4     2.4
12 Mitsubishi    Galant     3.2     2.9   3.2   3.5    3.1    3.1    2.9     2.6
13     Nissan     Sunny     2.6     3.3   3.9   2.1    3.5    3.9    3.8     2.4
14       Opel     Corsa     2.2     2.4   3.0   2.6    3.2    4.0    2.9     2.4
15       Opel    Vectra     3.1     2.6   2.3   3.6    2.8    2.9    2.4     2.4
16    Peugeot       306     2.9     3.5   3.6   2.8    3.2    3.8    3.2     2.6
17    Renault        19     2.7     3.3   3.4   3.0    3.1    3.4    3.0     2.7
18      Rover               3.9     2.8   2.6   4.0    2.6    3.0    3.2     3.0
19     Toyota   Corolla     2.5     2.9   3.4   3.0    3.2    3.1    3.2     2.8
20      Volvo               3.8     2.3   1.9   4.2    3.1    3.6    1.6     2.4
21    Trabant       601     3.6     4.7   5.5   1.5    4.1    5.8    5.9     3.1
22         VW      Golf     2.4     2.1   2.0   2.6    3.2    3.1    3.1     1.6
23         VW    Passat     3.1     2.2   2.1   3.2    3.5    3.5    2.8     1.8
24   Wartburg       1.3     3.7     4.7   5.5   1.7    4.8    5.2    5.5     4.0
>
```

En utilisant la fonction **rcc( )** du package **mixOmics** nous obtenons:



Il s'agit bien des valeurs obtenues dans le cours théorique. Nous retrouvons les mêmes corrélations qu'avec MATLAB mais pas les mêmes coefficients (ceux-ci-dessus étant centrés réduits au contraire de MATLAB).

Maintenant passons à un exemple fait en formation avec R 4.1 pour comparer les résultats de SAS et de R. Comme le package **mixOmics** n'y est plus disponible au jour où nous écrivons ces lignes, nous ferons l'exemple avec le package **CCA** et sa fonction **cc( )**.

D'abord nous chargeons les données:

```
> library("CCA")
> mydata<-read.csv("c:/tmp/Sales.txt",header=T,sep=";")
> str(mydata)
'data.frame':   50 obs. of  7 variables:
 $ Sales.Growth         : num  93 88.8 95 101.3 102 ...
 $ Sales.Profitability : num  96 91.8 100.3 103.8 107.8 ...
 $ New.Account.Sales   : num  97.8 96.8 99 106.8 103 ...
 $ Creativity          : int  9 7 8 13 10 10 9 18 10 14 ...
 $ Mechanical.Reasoning: int  12 10 12 14 15 14 12 20 17 18 ...
 $ Abstract.Reasoning  : int  9 10 9 12 12 11 9 15 13 11 ...
 $ Mathematics         : int  20 15 26 29 32 21 25 51 31 39 ...
> head(mydata)
  Sales.Growth Sales.Profitability New.Account.Sales Creativity Mechanical.Reasoning
1         93.0                96.0              97.8          9                   12
2         88.8                91.8              96.8          7                   10
3         95.0               100.3              99.0          8                   12
4        101.3               103.8             106.8         13                   14
5        102.0               107.8             103.0         10                   15
6         95.8                97.5              99.3         10                   14
  Abstract.Reasoning Mathematics
1                  9          20
2                 10          15
3                  9          26
4                 12          29
5                 12          32
6                 11          21
> 
```

Nous faisons d'abord une première analyse descriptive des différentes matrices de corrélations:

```
> X<-mydata[1:3]
> Y<-mydata[4:7]
> matcor(X, Y)
$Xcor
                   Sales.Growth Sales.Profitability New.Account.Sales
Sales.Growth          1.0000000           0.9260758         0.8840023
Sales.Profitability   0.9260758           1.0000000         0.8425232
New.Account.Sales     0.8840023           0.8425232         1.0000000

$Ycor
                     Creativity Mechanical.Reasoning Abstract.Reasoning Mathematics
Creativity            1.0000000            0.5907360          0.1469074   0.4126395
Mechanical.Reasoning  0.5907360            1.0000000          0.3859502   0.5745533
Abstract.Reasoning    0.1469074            0.3859502          1.0000000   0.5663721
Mathematics           0.4126395            0.5745533          0.5663721   1.0000000

$XYcor
                     Sales.Growth Sales.Profitability New.Account.Sales Creativity
Sales.Growth            1.0000000           0.9260758         0.8840023  0.5720363
Sales.Profitability     0.9260758           1.0000000         0.8425232  0.5415080
New.Account.Sales       0.8840023           0.8425232         1.0000000  0.7003630
Creativity              0.5720363           0.5415080         0.7003630  1.0000000
Mechanical.Reasoning    0.7080738           0.7459097         0.6374712  0.5907360
Abstract.Reasoning      0.6744073           0.4653880         0.6410886  0.1469074
Mathematics             0.9273116           0.9442960         0.8525682  0.4126395
                     Mechanical.Reasoning Abstract.Reasoning Mathematics
Sales.Growth                    0.7080738          0.6744073   0.9273116
Sales.Profitability             0.7459097          0.4653880   0.9442960
New.Account.Sales               0.6374712          0.6410886   0.8525682
Creativity                      0.5907360          0.1469074   0.4126395
Mechanical.Reasoning            1.0000000          0.3859502   0.5745533
Abstract.Reasoning              0.3859502          1.0000000   0.5663721
Mathematics                     0.5745533          0.5663721   1.0000000

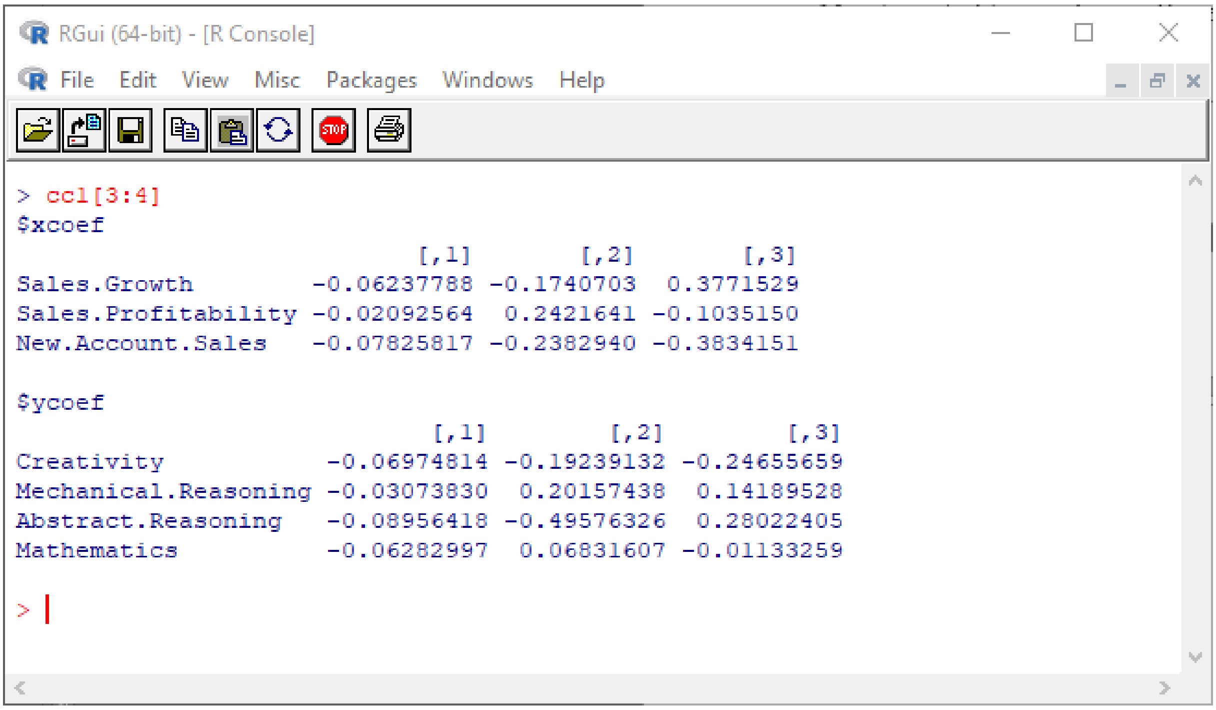
> 
```

Ensuite nous obtenir les corrélations canoniques et nous voyons que nous obtenons bien les mêmes que dans SAS:

```
> cc1$cor
[1] 0.9944827 0.8781065 0.3836057
> cc1$cor^2
[1] 0.9889958 0.7710711 0.1471533
> 
```

On voit que 98.9 % de la variation dans U_1 de s'explique par la variation de $V1$, et 77.11 % de la variation dans U_2 s'explique par V_2, mais seulement 14.72 % de la variation dans U_3 s'explique par V_3. Ces deux premières corrélations canoniques sont très élevées et suggèrent que seules les deux premières corrélations canoniques sont importantes.

Maintenant obtenons les coefficients des coefficients de la régression canonique pour noter que là aussi nous obtenons les mêmes qu'avec SAS:



Et donc nous avons par exemple:

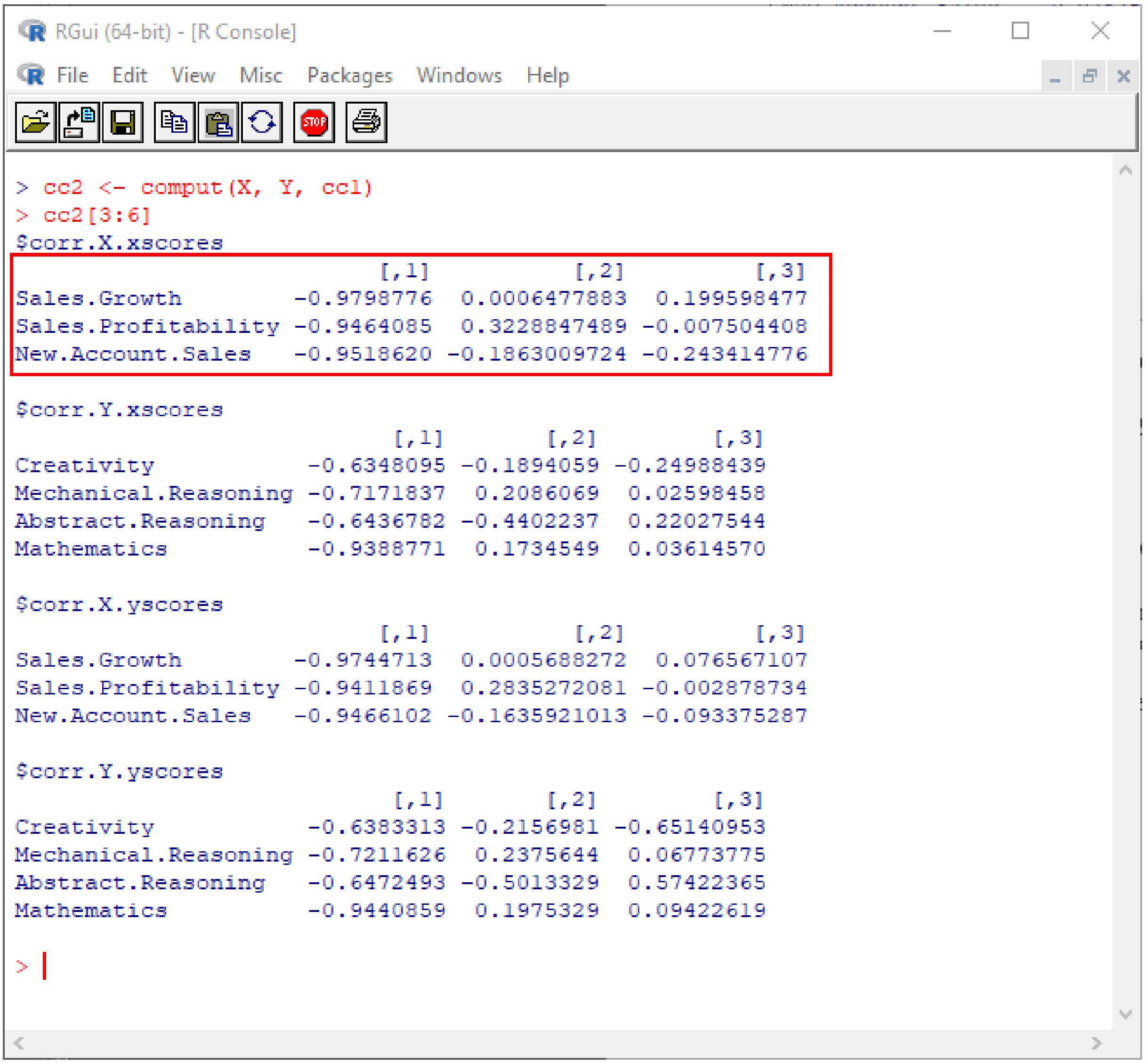
$$U_1 = -0.0624X_{growth} - 0.209X_{profit} - 0.0783X_{new}$$

et respectivement:

$$V_1 = -0.0697Y_{create} - 0.0307Y_{mech} - 0.0628X_{math}$$

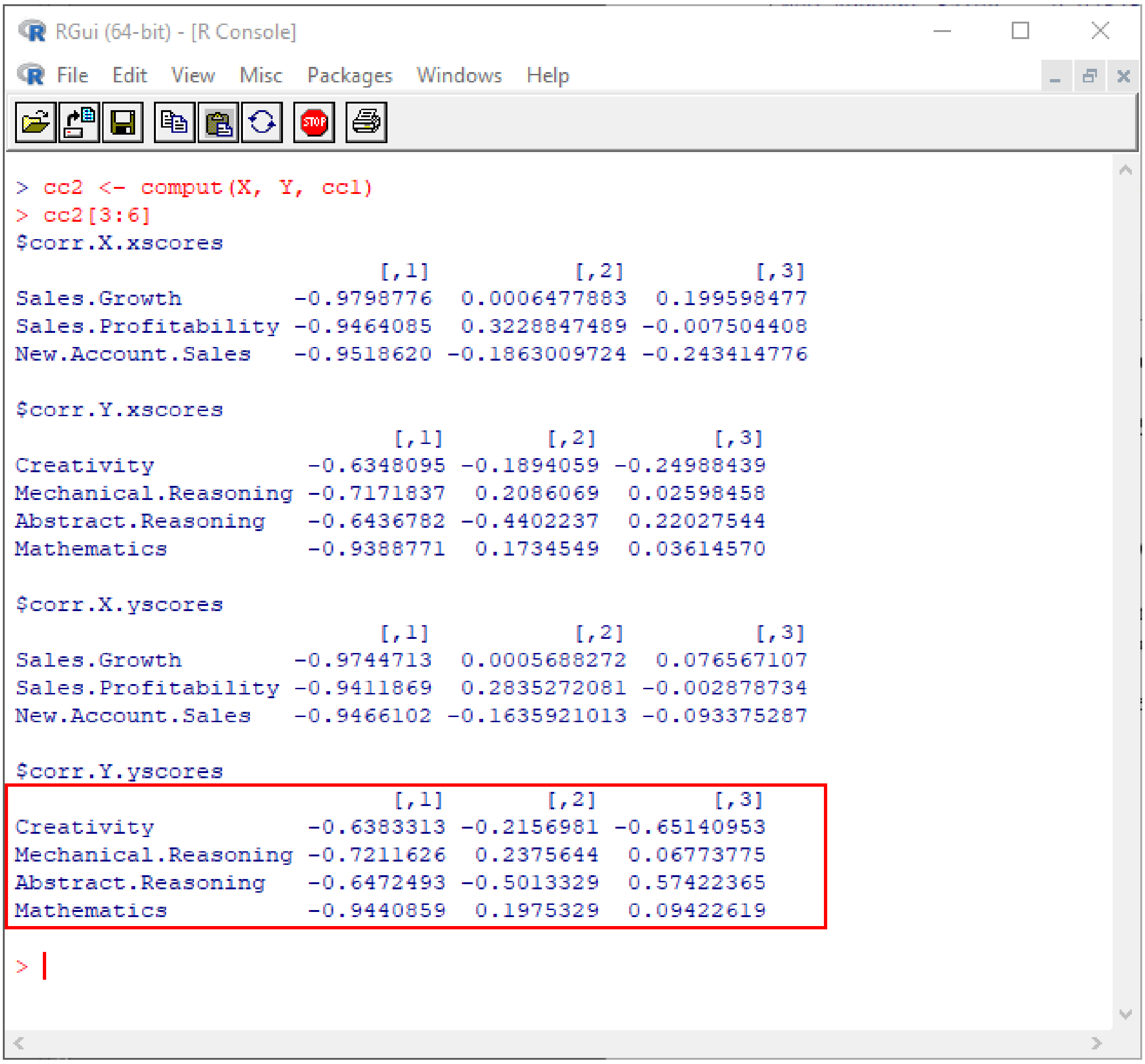
Pour interpréter chaque composante, nous devons calculer les corrélations entre chaque variable et la variable canonique correspondante.

Les corrélations entre les variables de vente et les variables canoniques pour la performance des ventes se trouvent ici :



En regardant la première variable canonique pour les ventes ci-dessus (première colonne de la matrice entourée en rouge), nous voyons que toutes les corrélations sont uniformément grandes. Par conséquent, nous pouvons considérer cette variable canonique comme une mesure globale de la performance commerciale. Pour la deuxième variable canonique pour les performances des ventes, aucune des corrélations n'est particulièrement importante et, par conséquent, cette variable canonique fournit peu d'informations sur les données. Encore une fois, nous avions décidé plus tôt de ne pas regarder les troisièmes paires de variables canoniques.

Les corrélations entre les scores des tests et les variables canoniques pour les scores des tests se trouvent également dans:

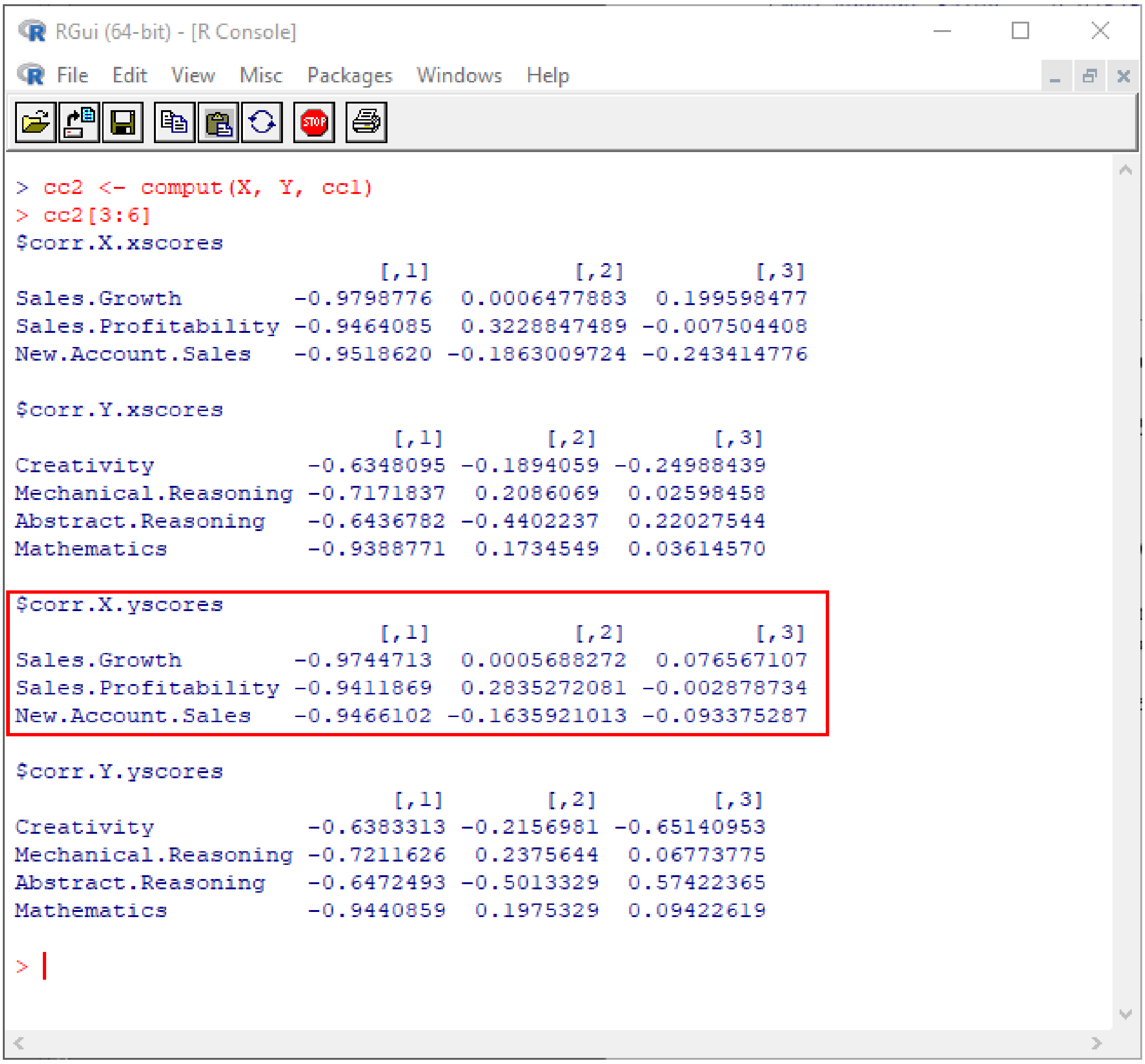


Étant donné que toutes les corrélations sont importantes pour la première variable canonique, cela peut également être considéré comme une mesure globale de la performance au test, cependant, il est plus fortement corrélé avec les résultats des tests de mathématiques. La plupart des corrélations avec la deuxième variable canonique sont faibles. Certains suggèrent que cette variable peut être négativement corrélée avec le raisonnement abstrait.

En mettant ces deux dernières conclusions ensemble, nous voyons que le meilleur prédicteur de la performance des ventes est les résultats des tests de mathématiques, car cet indicateur se démarque le plus.

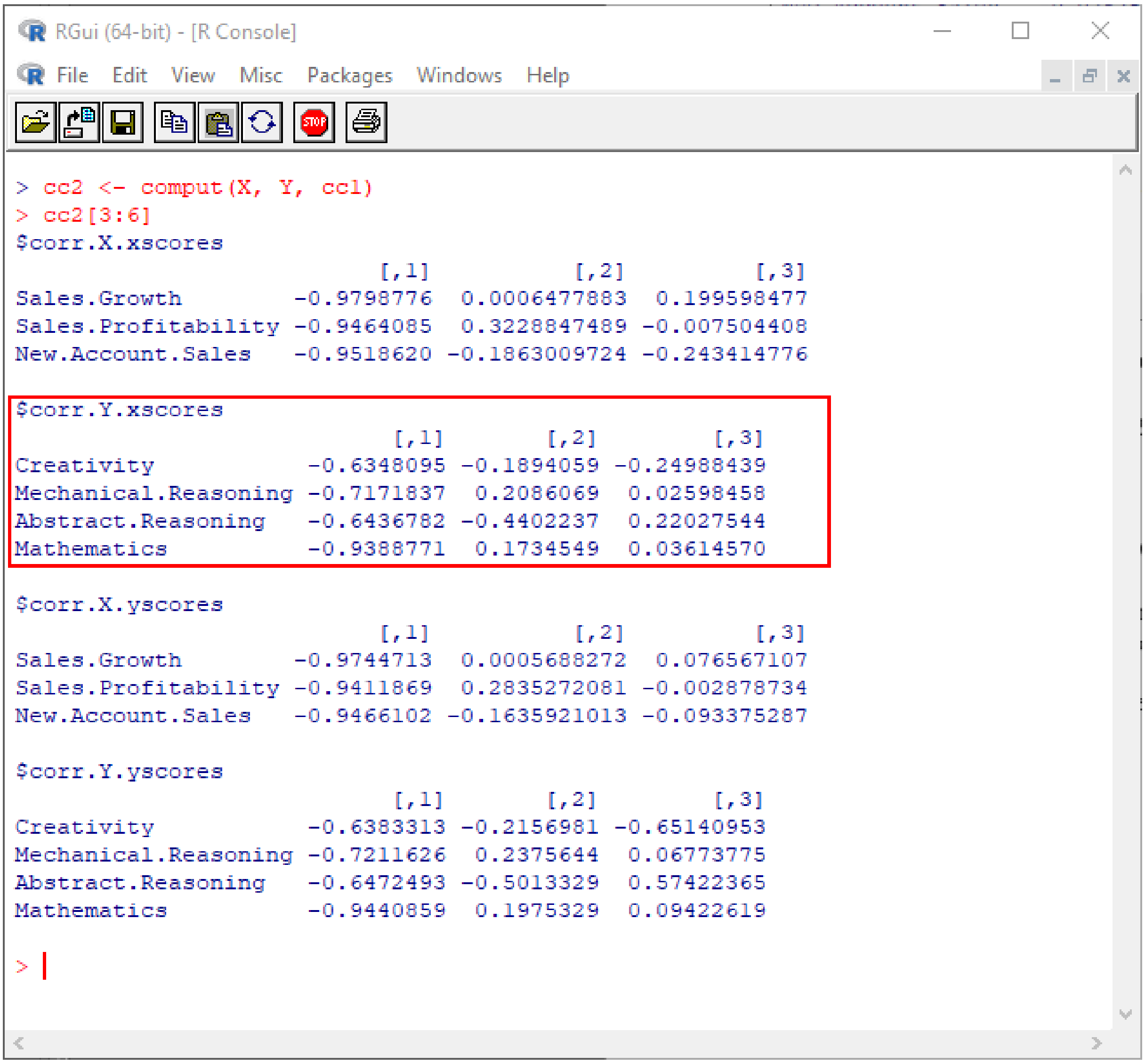
Ces résultats sont encore renforcés en examinant les corrélations entre chaque ensemble de variables et le groupe opposé de variables canoniques.

Les corrélations entre les variables de vente et la première variable canonique pour les résultats des tests:



Nous pouvons voir que ces trois corrélations sont fortes et montrent un modèle similaire à celui de la variable canonique pour les ventes. La raison en est évidente : la première corrélation canonique est très élevée.

Les corrélations entre les scores des tests et la première variable canonique pour les ventes:



```
> cc2 <- comput(X, Y, cc1)
> cc2[3:6]
$corr.X.xscores
                         [,1]          [,2]         [,3]
Sales.Growth       -0.9798776  0.0006477883  0.199598477
Sales.Profitability -0.9464085  0.3228847489 -0.007504408
New.Account.Sales   -0.9518620 -0.1863009724 -0.243414776

$corr.Y.xscores
                          [,1]       [,2]        [,3]
Creativity           -0.6348095 -0.1894059 -0.24988439
Mechanical.Reasoning -0.7171837  0.2086069  0.02598458
Abstract.Reasoning   -0.6436782 -0.4402237  0.22027544
Mathematics          -0.9388771  0.1734549  0.03614570

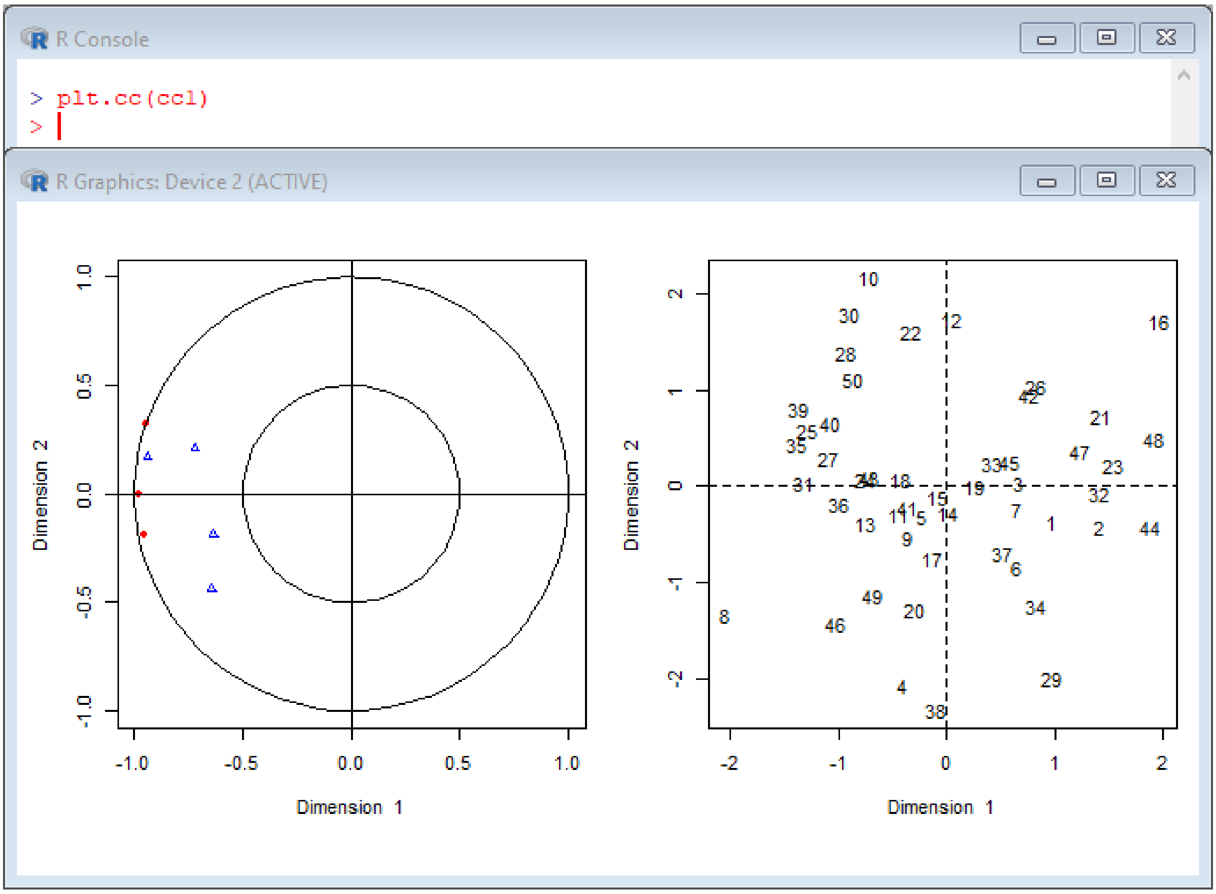
$corr.X.yscores
                         [,1]          [,2]         [,3]
Sales.Growth       -0.9744713  0.0005688272  0.076567107
Sales.Profitability -0.9411869  0.2835272081 -0.002878734
New.Account.Sales   -0.9466102 -0.1635921013 -0.093375287

$corr.Y.yscores
                          [,1]       [,2]        [,3]
Creativity           -0.6383313 -0.2156981 -0.65140953
Mechanical.Reasoning -0.7211626  0.2375644  0.06773775
Abstract.Reasoning   -0.6472493 -0.5013329  0.57422365
Mathematics          -0.9440859  0.1975329  0.09422619

>
```

Ces deux derniers résultats confirment que la performance des ventes est mieux prédite par les résultats des tests de mathématiques.

Nous pouvons aussi faire un graphique avec la fonction **plt.cc( )** du package CCA:



Enfin on peut faire comme dans SAS, obtenir le résultat du test d'hypothèse que les corrélations canoniques de la ligne actuelle et toutes celles qui suivent sont nulles:

$$
\begin{aligned}
X_1 &= \beta_{10} + \beta_{11}Y_1 + \beta_{12}Y_2 + \cdots + \beta_{1q}Y_q + \epsilon_1 \\
X_2 &= \beta_{20} + \beta_{21}Y_1 + \beta_{22}Y_2 + \cdots + \beta_{2q}Y_q + \epsilon_2 \\
&\vdots \\
X_p &= \beta_{p0} + \beta_{p1}Y_1 + \beta_{p2}Y_2 + \cdots + \beta_{pq}Y_q + \epsilon_p
\end{aligned}
$$

Ce qui nous donne en utilisant la fonction **p.asym( )** du package **CCP**:

```
> (rho <- cc1$cor)
[1] 0.9944827 0.8781065 0.3836057
> (n <- dim(X)[1])
[1] 50
> (p <- length(X))
[1] 3
> (q <- length(Y))
[1] 4
>
> library("CCP")
> ## Calculate p-values using the F-approximations of different test statisti$
> p.asym(rho, n, p, q, tstat = "Wilks")
Wilks' Lambda, using F-approximation (Rao's F):
              stat    approx df1      df2      p.value
1 to 3:  0.002148472 87.391525  12 114.0588 0.000000e+00
2 to 3:  0.195241267 18.526265   6  88.0000 8.248957e-14
3 to 3:  0.852846693  3.882233   2  45.0000 2.783536e-02
> p.asym(rho, n, p, q, tstat = "Hotelling")
 Hotelling-Lawley Trace, using F-approximation:
               stat     approx df1 df2    p.value
1 to 3:  93.4151751 324.358247  12 125 0.00000000
2 to 3:   3.5407119  25.768514   6 131 0.00000000
3 to 3:   0.1725437   3.939747   2 137 0.02169625
> p.asym(rho, n, p, q, tstat = "Pillai")
 Pillai-Bartlett Trace, using F-approximation:
              stat   approx df1 df2      p.value
1 to 3:  1.9072202 19.63454  12 135 0.000000e+00
2 to 3:  0.9182244 10.36532   6 141 1.680060e-09
3 to 3:  0.1471533  3.79122   2 147 2.480577e-02
> p.asym(rho, n, p, q, tstat = "Roy")
 Roy's Largest Root, using F-approximation:
              stat   approx df1 df2 p.value
1 to 1:  0.9889958 1011.088   4  45       0

 F statistic for Roy's Greatest Root is an upper bound.
>
```

Dans notre exemple, nous rejetons donc l'hypothèse nulle selon laquelle il n'y a pas de relation entre les deux ensembles de variables et concluons que les deux ensembles de variables sont dépendants. Notez également que l'hypothèse nulle ci-dessus équivaut également à tester l'hypothèse nulle selon laquelle toutes les p paires de variables canoniques ne sont pas corrélées

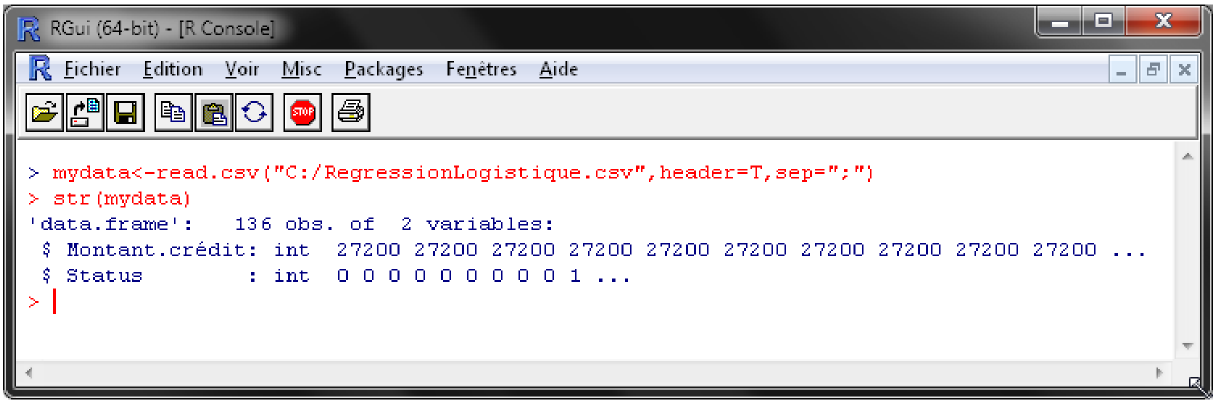
Exemple 272.: Régressions logistiques (logit et probit)
R 3.6.2

Régression binomiale (binaire/scoring) logit

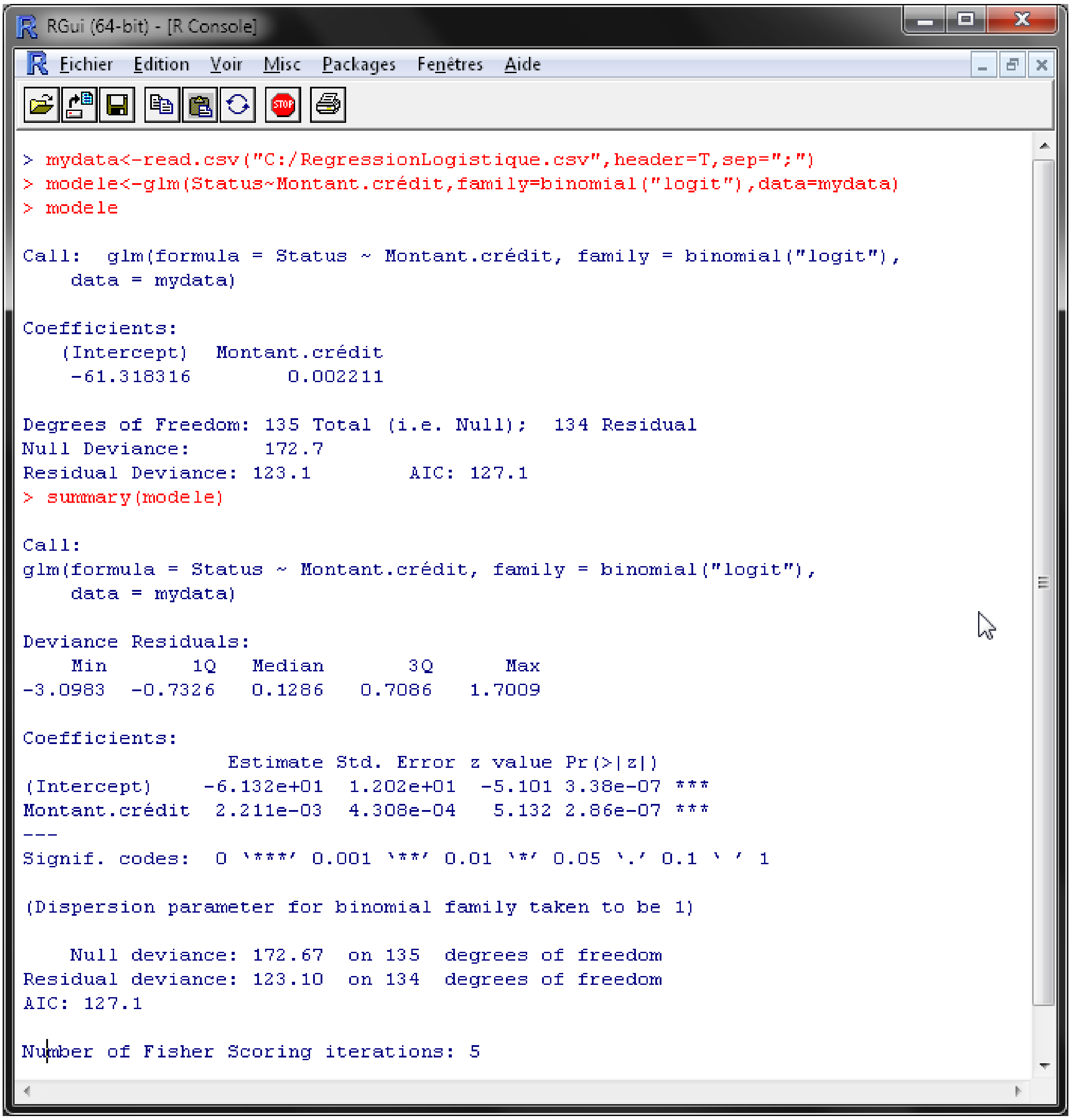
Comme nous l'avons vu dans le cours de statistique théorique, le problème d'une régression linéaire simple (bivariée) ou multiple à variable expliquée de type binaire c'est que le modèle théorique peut prendre toute valeur dans $\mathbb{R}$ ce qui est évidemment une aberration puisque dans le cas d'une variable expliquée de type 0/1, le modèle théorique ne doit jamais donner une valeur inférieure à 0 ou supérieure 1. Or c'est malheureusement ce que fera une régression linéaire…

Pour cela nous avons démontré qu'il était possible d'utiliser une autre technique mathématique qui nous assurait que la condition susmentionnée soit satisfaite. Nous allons reprendre le même exemple mais au lieu de faire les calculs à la main, nous les faisons donc avec R (et les comparer accessoirement avec R).

Nous utiliserons le même jeu de données:



Nous utilisons la commande native **glm( )** et nous remarquons que nous obtenons les mêmes résultats que dans Minitab (avec la même différence que dans le cours théorique et pour les mêmes raisons):



```
> mydata<-read.csv("C:/RegressionLogistique.csv",header=T,sep=";")
> modele<-glm(Status~Montant.crédit,family=binomial("logit"),data=mydata)
> modele

Call:  glm(formula = Status ~ Montant.crédit, family = binomial("logit"),
    data = mydata)

Coefficients:
   (Intercept)  Montant.crédit
    -61.318316        0.002211

Degrees of Freedom: 135 Total (i.e. Null);  134 Residual
Null Deviance:      172.7
Residual Deviance: 123.1        AIC: 127.1
> summary(modele)

Call:
glm(formula = Status ~ Montant.crédit, family = binomial("logit"),
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0983  -0.7326   0.1286   0.7086   1.7009

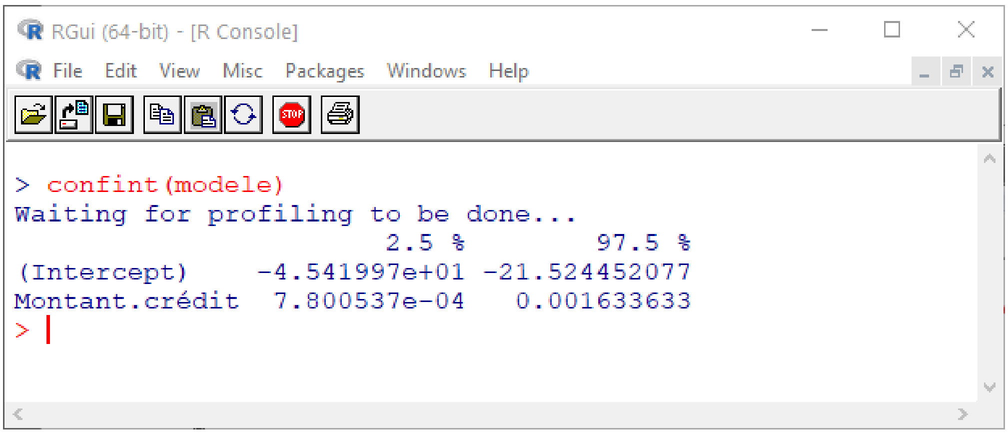
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.132e+01  1.202e+01  -5.101 3.38e-07 ***
Montant.crédit  2.211e-03  4.308e-04   5.132 2.86e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 172.67  on 135  degrees of freedom
Residual deviance: 123.10  on 134  degrees of freedom
AIC: 127.1

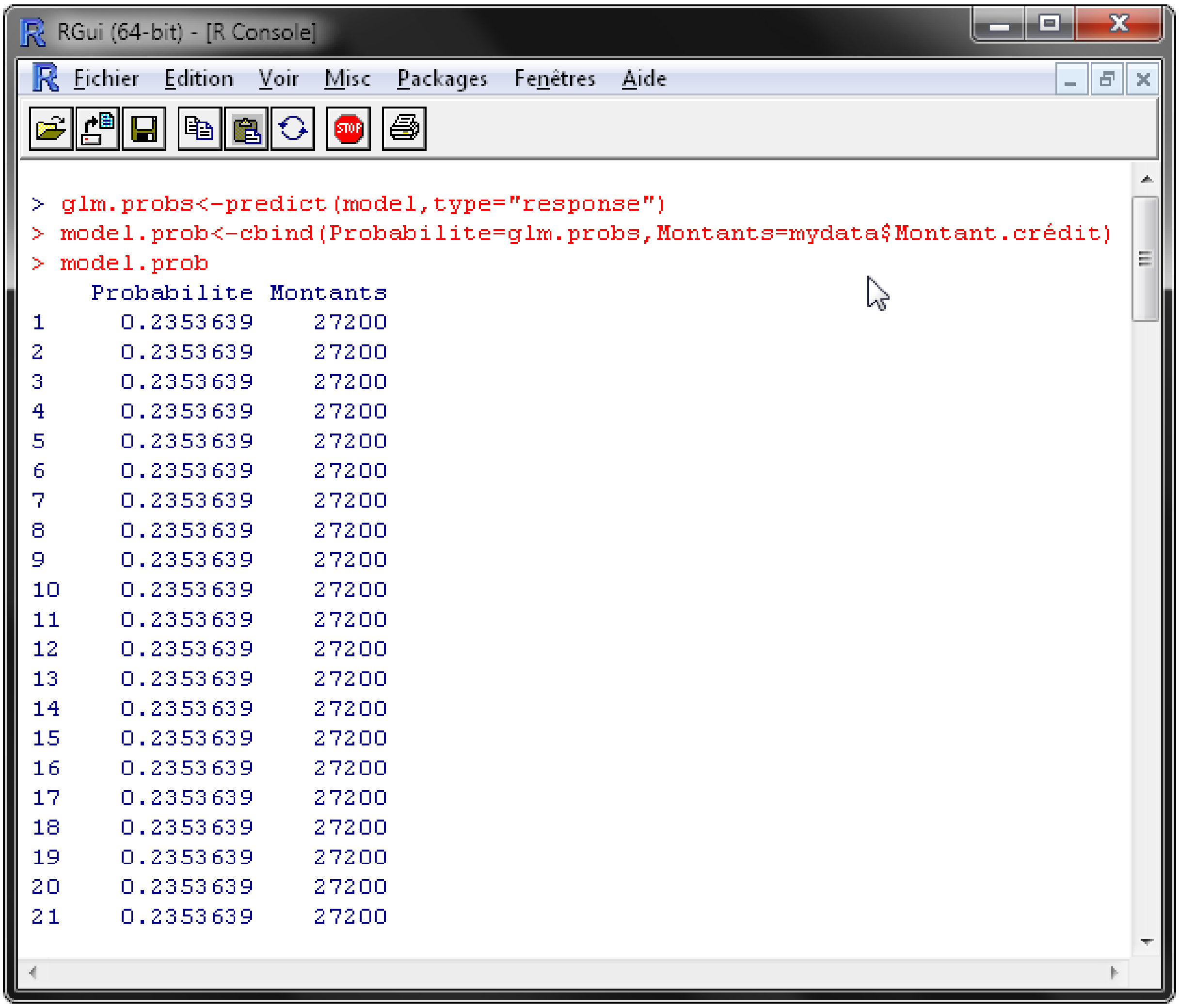
Number of Fisher Scoring iterations: 5
```

Nous pouvons calculer l'intervalle de confiance à 95% des coefficients comme suit:

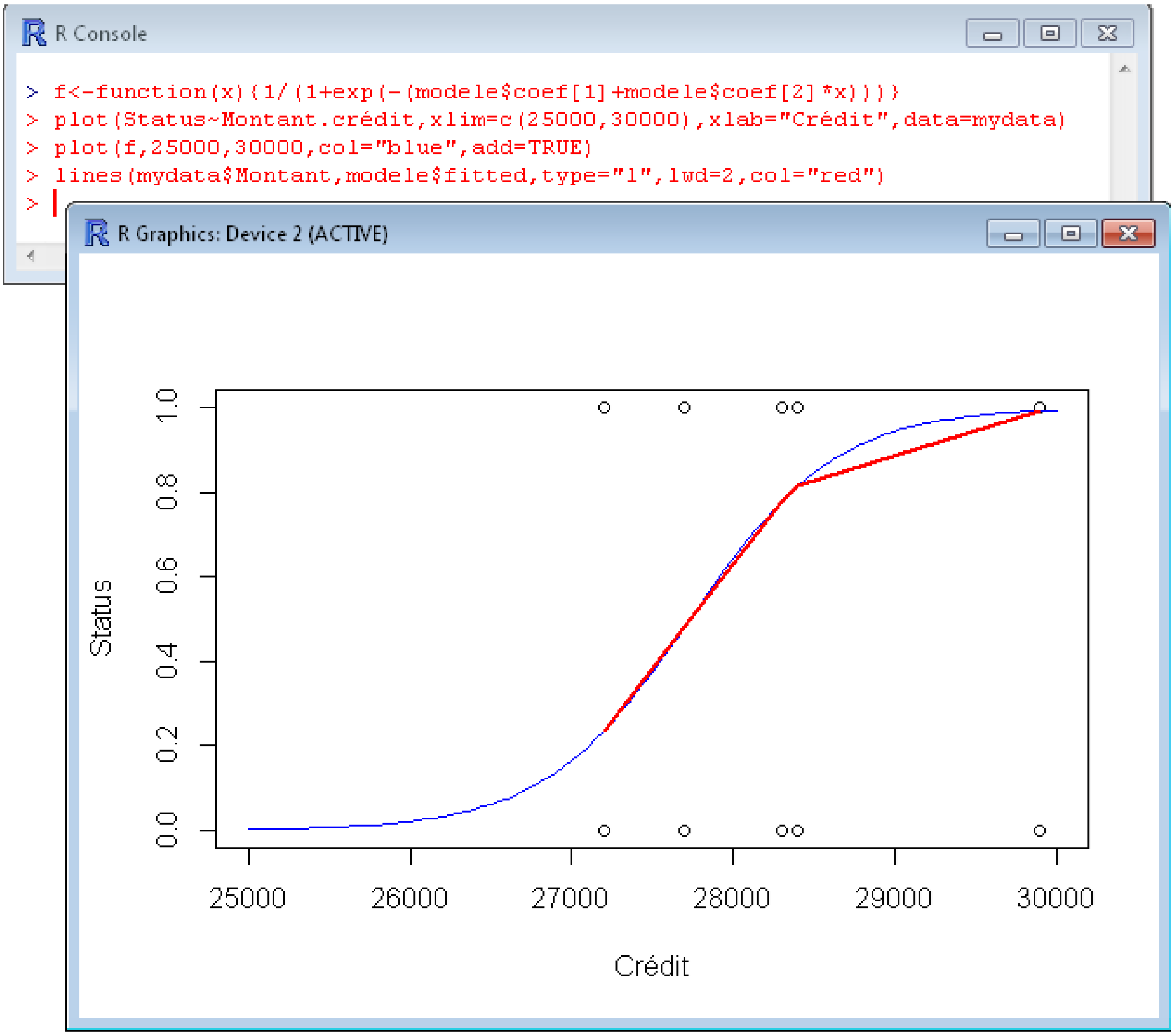


```
> confint(modele)
Waiting for profiling to be done...
                       2.5 %        97.5 %
(Intercept)    -4.541997e+01 -21.524452077
Montant.crédit  7.800537e-04   0.001633633
>
```

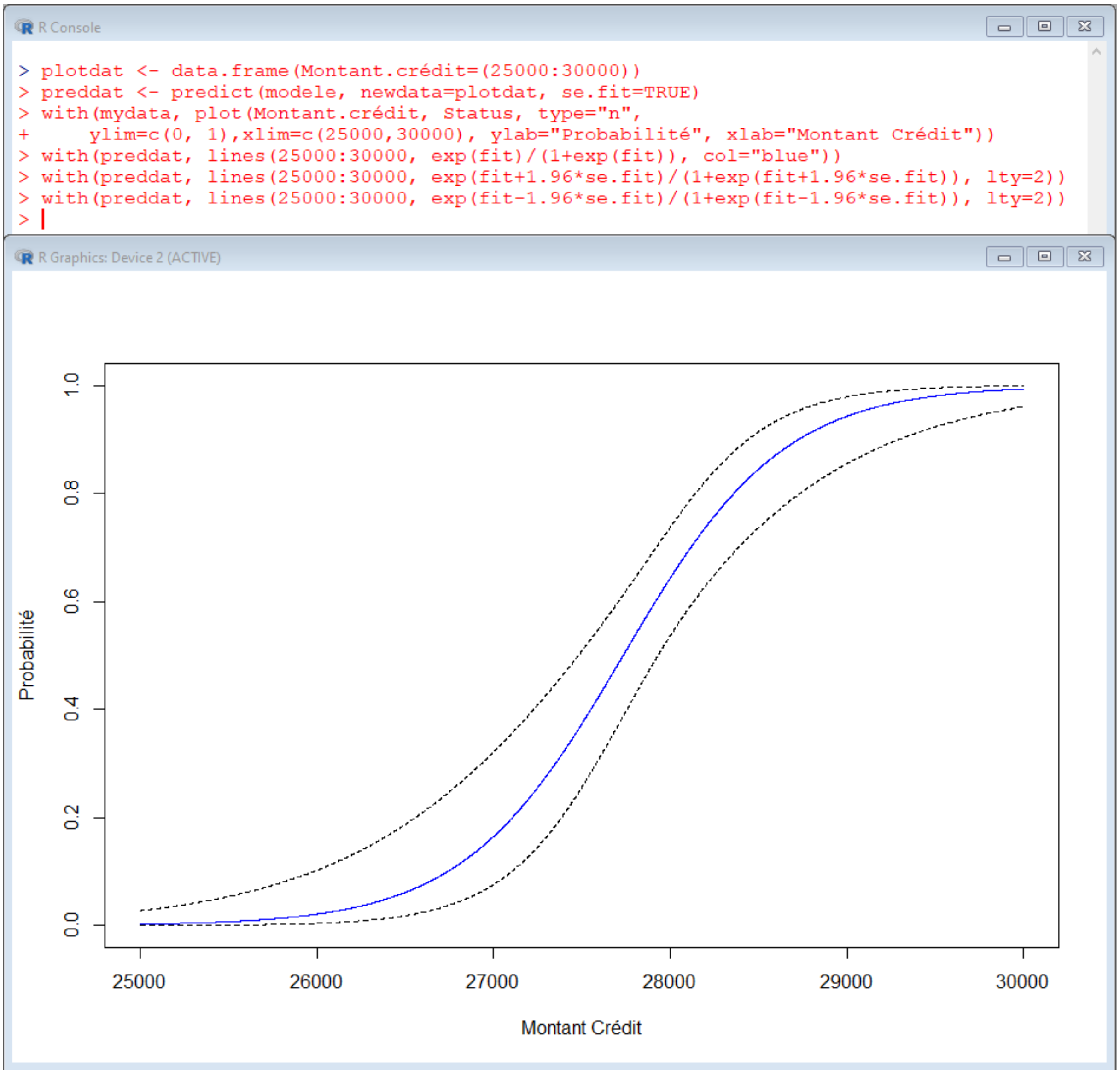
Nous pouvons construire un petit data frame avec les données du modèle:



Nous pouvons faire un plot des points mesurés, des points abscisses mesurées avec la théorie ainsi que d'un lissage:



On peut aussi calculer et afficher le non moins important intervalle de confiance:



Et obtenir les valeurs correspondantes:



```
> inverse_logit = function(x){
+   exp(x)/(1+exp(x))
+ }
>
> fit<-glm.probs$fit
> se.fit<-glm.probs$se.fit
> se_high<-inverse_logit(fit+1.96*se.fit)
> se_low<-inverse_logit(fit-1.96*se.fit)
> expected<-inverse_logit(fit)
> (cbind(mydata$Montant.crédit,expected,se_low,se_high))
           expected    se_low   se_high
1   27200 0.2353639 0.1288647 0.3904313
2   27200 0.2353639 0.1288647 0.3904313
3   27200 0.2353639 0.1288647 0.3904313
4   27200 0.2353639 0.1288647 0.3904313
5   27200 0.2353639 0.1288647 0.3904313
6   27200 0.2353639 0.1288647 0.3904313
7   27200 0.2353639 0.1288647 0.3904313
8   27200 0.2353639 0.1288647 0.3904313
9   27200 0.2353639 0.1288647 0.3904313
10  27200 0.2353639 0.1288647 0.3904313
11  27200 0.2353639 0.1288647 0.3904313
12  27200 0.2353639 0.1288647 0.3904313
13  27200 0.2353639 0.1288647 0.3904313
14  27200 0.2353639 0.1288647 0.3904313
15  27200 0.2353639 0.1288647 0.3904313
16  27200 0.2353639 0.1288647 0.3904313
17  27200 0.2353639 0.1288647 0.3904313
18  27200 0.2353639 0.1288647 0.3904313
19  27200 0.2353639 0.1288647 0.3904313
20  27200 0.2353639 0.1288647 0.3904313
21  27200 0.2353639 0.1288647 0.3904313
22  27200 0.2353639 0.1288647 0.3904313
23  27200 0.2353639 0.1288647 0.3904313
24  27200 0.2353639 0.1288647 0.3904313
25  27200 0.2353639 0.1288647 0.3904313
26  27200 0.2353639 0.1288647 0.3904313
27  27200 0.2353639 0.1288647 0.3904313
28  27700 0.4818202 0.3685360 0.5970046
29  27700 0.4818202 0.3685360 0.5970046
30  27700 0.4818202 0.3685360 0.5970046
31  27700 0.4818202 0.3685360 0.5970046
32  27700 0.4818202 0.3685360 0.5970046
33  27700 0.4818202 0.3685360 0.5970046
34  27700 0.4818202 0.3685360 0.5970046
```

Courbe ROC et Matrice de Confusion

Nous pouvons également avoir la courbe ROC (Receiver Operating Characteristic) à l'aide de la commande **roc( )** du package **pROC( )**:

```
> (modele<-glm(Status~Montant.crédit,family=binomial("logit"),data=mydata))

Call:  glm(formula = Status ~ Montant.crédit, family = binomial("logit"),
    data = mydata)

Coefficients:
   (Intercept)  Montant.crédit
    -61.318316        0.002211

Degrees of Freedom: 135 Total (i.e. Null);  134 Residual
Null Deviance:      172.7
Residual Deviance: 123.1        AIC: 127.1
> prob<-predict(modele,type=c("response"))
> mydata$prob<-prob
> library("pROC")
> (g<-roc(Status~prob,data=mydata))
Setting levels: control = 0, case = 1
Setting direction: controls < cases

Call:
roc.formula(formula = Status ~ prob, data = mydata)

Data: prob in 45 controls (Status 0) < 91 cases (Status 1).
Area under the curve: 0.8193
> plot(g)
> |
```

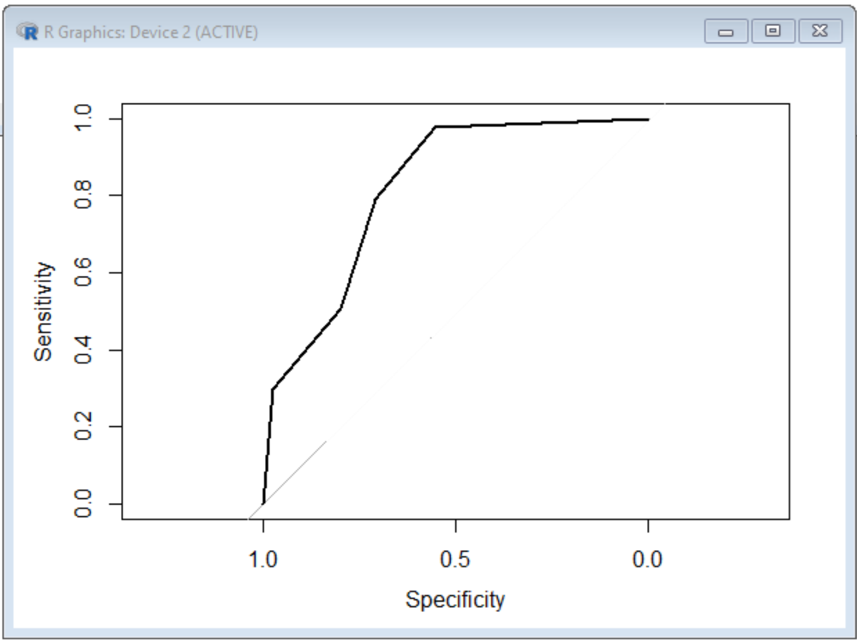


Figure 253 Graph d'une courbe ROC (Receiver operating characteristic)

La courbe ROC est identique à Minitab et à celle calculée dans le cours théorique à la différence que pour Minitab nous avons du laborieusement aller chercher une macro sur le site web de l'éditeur du logiciel.

Nous pouvons aussi avoir un intervalle de confiance à l'aide du bootstrapping avec la commande **ci.se( )** du même package (voilà une fonction bien utile qu'il manque à bon nombre de logiciels statistiques pour la courbe ROC) et mélangé avec une fonction nommée **plot.roc( )**:

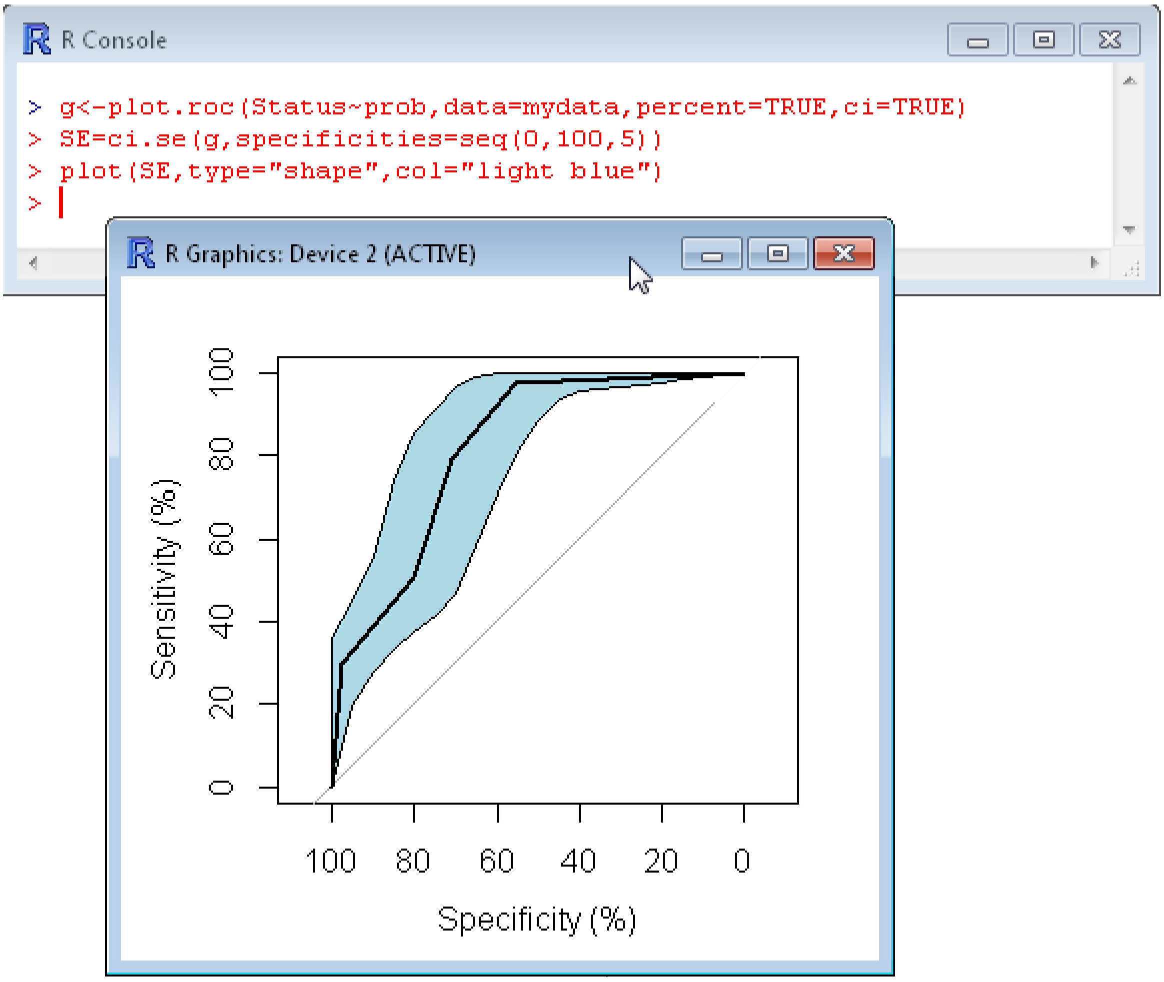
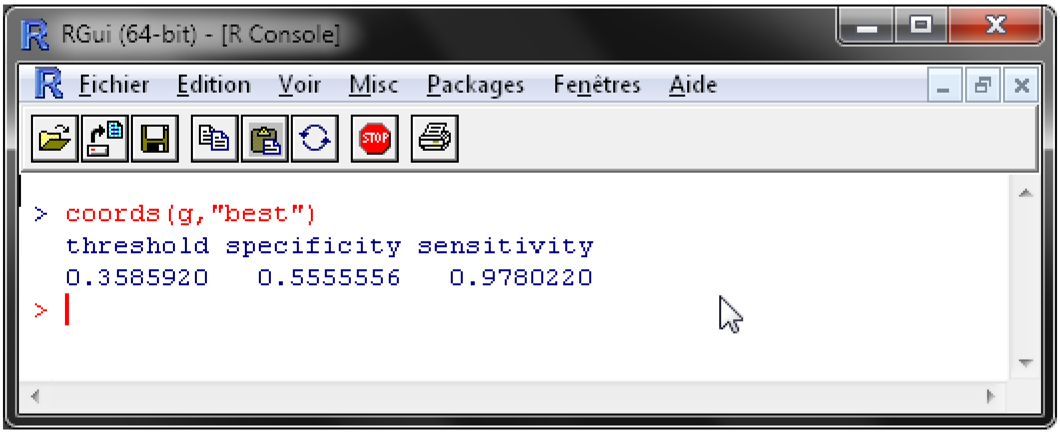
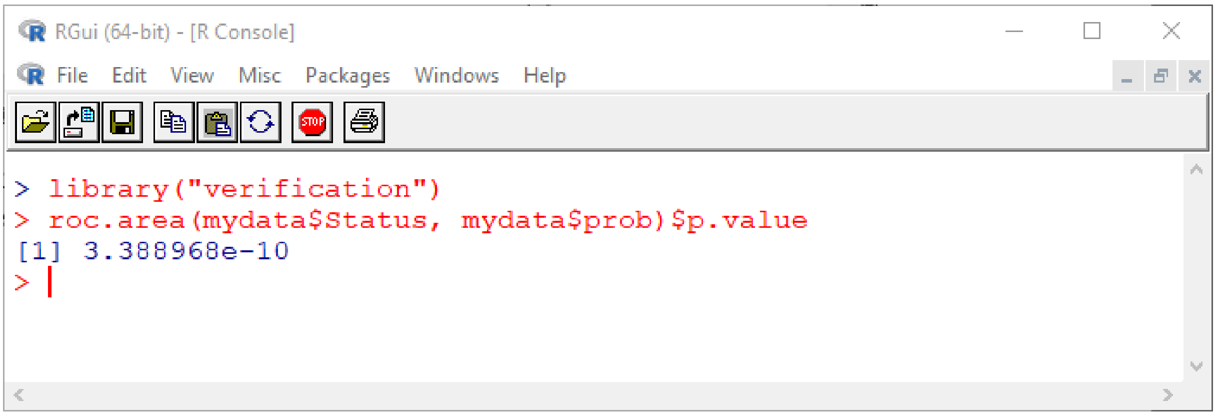


Figure 254 Graph d'une courbe ROC (Receiver operating characteristic) avec intervalle de confiance

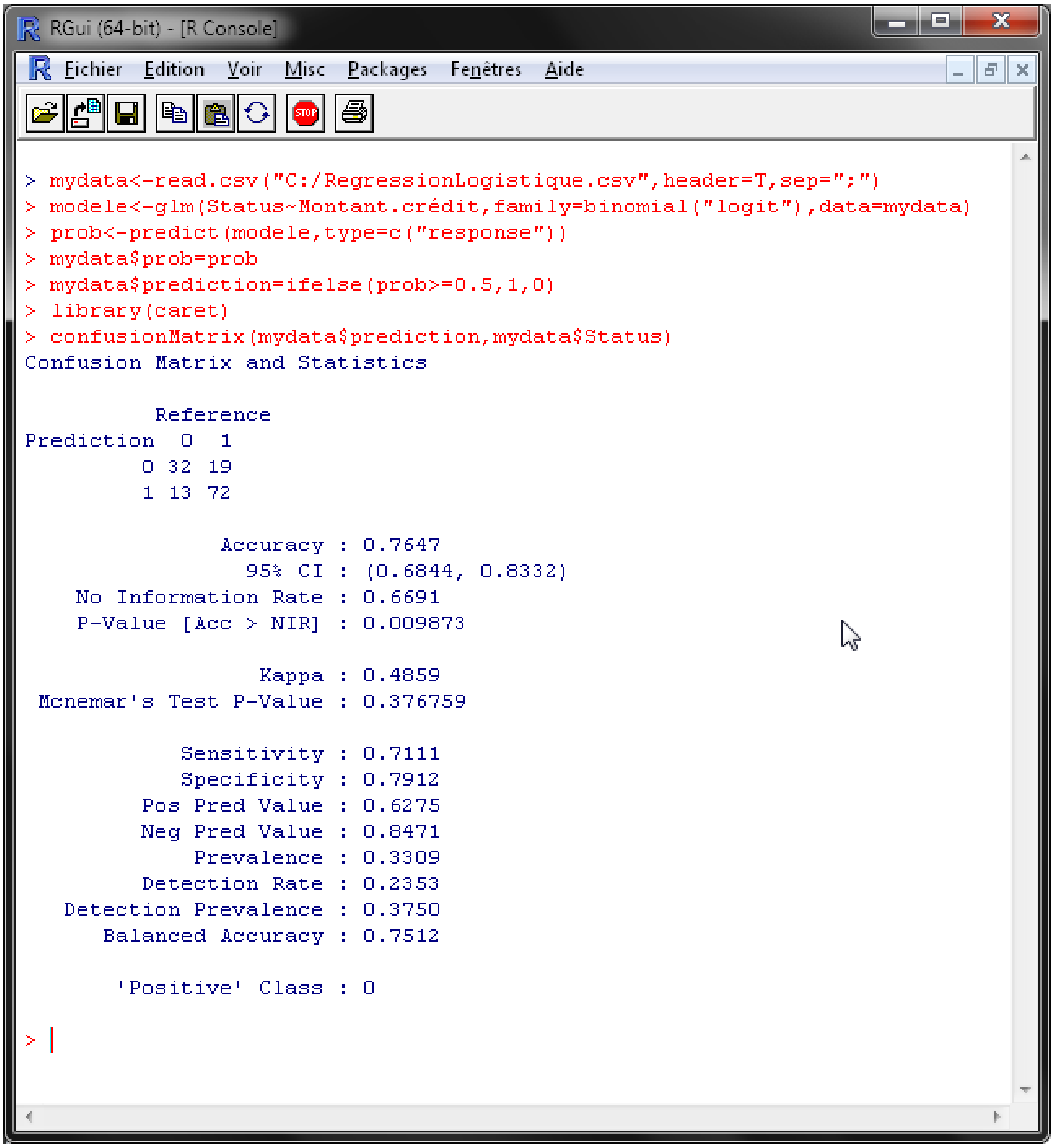
Avec la commande **coords( )** du package **pROC** nous pouvons obtenir le meilleur cut-off:



Et avec le package **verification** et sa fonction **roc.area( )** calculer la p-value que la courbe ROC est significativement différente de la diagonales (test des rangs de Wilcoxon):



Nous pouvons avec la commande **matrixConfusion( )** du package **caret** afficher la matrice de confusion et autres informations utiles comme la spécificité et la sensibilité:



Et enfin comme nous l'avons démontré mathématiquement aussi en détails dans le cours théorique, nous pouvons calculer le log-loss de tout classificateur binaire. Pour cela, nous allons utiliser la fonction **LogLoss( )** du package **MLmetrics**:

```
> library("MLmetrics")
> LogLoss(y_pred=modele$fitted.value,y_true=mydata$Status)
[1] 0.4525754
>
```

Nous pouvons tout inclure dans une seule fonction bien sûr et ajouter le score F1, l'indice de Rand et aussi le coefficient de Matthew ou autre…:

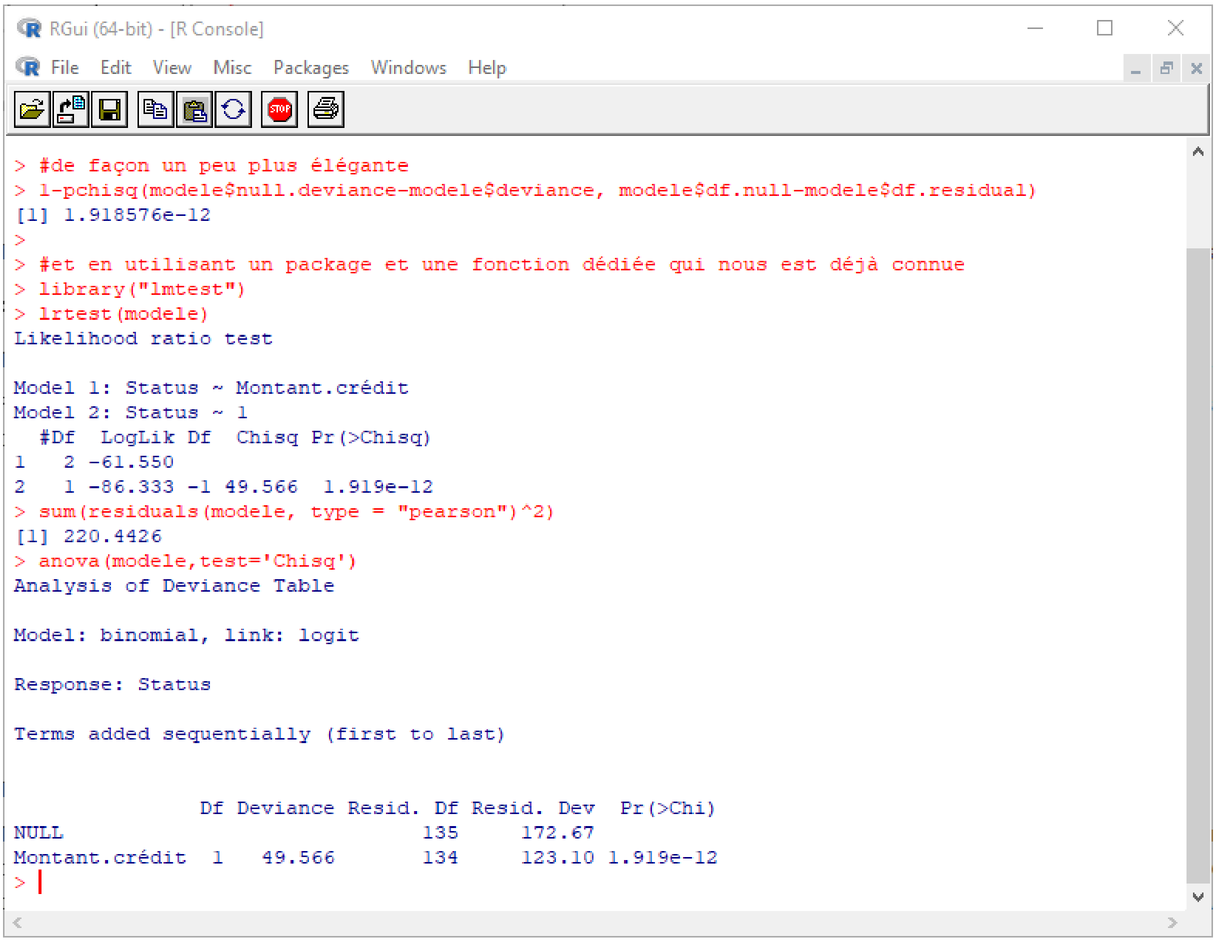
```
> mydata<-read.csv ("C:/tmp/RegressionLogistique.csv", header=T, sep=";")
> modele<-glm(Status~Montant.crédit, family=binomial ("logit"), data=mydata)
> prob<-predict (modele, type=c ("response"))
> mydata$prob=prob
> mydata$prediction=ifelse(prob>=0.5,1,0)
> library("caret")
> CM<-confusionMatrix(as.factor(mydata$prediction), as.factor(mydata$Status))
>
> err_metric=function(CM)
+ {
+   TN =CM[1,1]
+   TP =CM[2,2]
+   FP =CM[1,2]
+   FN =CM[2,1]
+   precision =(TP)/(TP+FP)
+   recall_score =(FP)/(FP+TN)
+
+   f1_score=2*((precision*recall_score)/(precision+recall_score))
+   accuracy_model  =(TP+TN)/(TP+TN+FP+FN)
+   False_positive_rate =(FP)/(FP+TN)
+   False_negative_rate =(FN)/(FN+TP)
+
+   print(paste("Precision value of the model: ",round(precision,2)))
+   print(paste("Accuracy (Rand Index) of the model: ",round(accuracy_model,2)))
+   print(paste("Recall value of the model: ",round(recall_score,2)))
+   print(paste("False Positive rate of the model: ",round(False_positive_rate,2)))
+
+   print(paste("False Negative rate of the model: ",round(False_negative_rate,2)))
+
+   print(paste("f1 score of the model: ",round(f1_score,2)))
+ }
>
> err_metric(CM$table)
[1] "Precision value of the model:  0.79"
[1] "Accuracy (Rand Index) of the model:  0.76"
[1] "Recall value of the model:  0.37"
[1] "False Positive rate of the model:  0.37"
[1] "False Negative rate of the model:  0.15"
[1] "f1 score of the model:  0.51"
>
```

Nous obtenons donc la même chose qu'avec Tanagra et qu'avec les calculs effectués à la main dans le cours théorique.

Tests d'adéquations du ratio des vraisemblances, du khi-2 de Pearson et de Hosmer-Lemeshow

Nous pouvons utiliser plusieurs tests pour vérifier l'adéquation de notre modèles aux données. Concentrons-nous sur les trois seuls que nous avons démontré dans le cours théorique.

Commençons par le test du ratio des vraisemblances (ie des déviances comme nous l'avons vu dans le cours théorique):



Évidemment les trois résultats concordent et nous voyons que le modèle avec les variables explicatives est (sans surprises dans ce cas particulier) bien meilleur qui celui sans variables explicatives…

Maintenant passons au texte du khi-2 de Pearson pour les résidus:

```
> pchisq(sum((mydata$Status-modele$fitted.value)^2/modele$fitted.value),modele$df.residual)
[1] 2.04020557300298e-18
> #1- la valeur du dessus donne la p-value. Donc le modèle n'est pas rejeté de toute évidence...
>
```

Donc nous avons ici la même conclusion.

Maintenant voyons sir nous pouvons faire le test du chi-2 de données de comptage (bon d'avance on sait que cela va échouer car certaines valeurs attendues du tableau de contingences valent zéro):

```
> #on prépare les données
> glm.probs<-predict(modele,type="response")
> modele.prob<-data.frame(cbind(Probabilite=ifelse(glm.probs>=0.5,1,0),
+ Montants=mydata$Montant.crédit,Status=mydata$Status))
> (agg<-aggregate(cbind(modele.prob$Probabilite,modele.prob$Status),
+ by=list(modele.prob$Montants), FUN=sum))
  Group.1 V1 V2
1   27200  0  2
2   27700  0 17
3   28300 30 26
4   28400 27 19
5   29900 28 27
> #pas bon signe... y'a des valeurs nulles...
>
> #première analyse à la main pour la p-value directement
> #on observer d'abord la distance du chi-2
> sum((agg$V1-agg$V2)^2/agg$V1)
[1] Inf
> #c'est foutu mais par principe faisons comme si on avait rien vu...
> pchisq(sum((agg$V1-agg$V2)^2/agg$V1),nrow(agg)-1,lower.tail=FALSE)
[1] 0
>
> #et autrement (mais on a le même souci toujours à cause des valeurs nulles)
> chisq.test(agg$V2,p=agg$V1/sum(agg$V1))

        Chi-squared test for given probabilities

data:  agg$V2
X-squared = Inf, df = 4, p-value < 2.220446049e-16

Warning message:
In chisq.test(agg$V2, p = agg$V1/sum(agg$V1)) :
  Chi-squared approximation may be incorrect
>
```

Bref on peut rien dire dans ce cas particulier. Il faut oublier ce test là!

Et voyons ce que nous dit les test de Hosmwer-Lemeshow avec la fonction **hoslem.test( )** du package **ResourceSelection**:

```
> library("ResourceSelection")
> (hl<-hoslem.test(mydata$Status,fitted(modele)))

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  mydata$Status, fitted(modele)
X-squared = 6.21789238699, df = 8, p-value = 0.622839186513

> #comme p>0.05 nous ne rejettons pas l'hypothèse comme quoi notre modèle est bon
> #on peut afficher les groupements choisis par la fonction:
> cbind(hl$observed,hl$expected)
              y0 y1           yhat0           yhat1
[0.235,0.482] 32 19 33.08149022516533 17.9185097748347
(0.482,0.778]  4 26  6.66094240856021 23.3390575914398
(0.778,0.814]  8 19  5.02708448292803 21.9729155170720
(0.814,0.992]  1 27  0.23048297337520 27.7695170266248
> 
```

Et enfin appliquons ce test sur un cas pratique que nous connaissons déjà bien:

```
> mydata<-read.csv("c:/tmp/RegressionLogistique.csv",header=T,sep=";")
> modele<-glm(Status~Montant.crédit,family=binomial("logit"),data=mydata)
> modele

Call:  glm(formula = Status ~ Montant.crédit, family = binomial("logit"),
    data = mydata)

Coefficients:
   (Intercept)  Montant.crédit
    -61.318316        0.002211

Degrees of Freedom: 135 Total (i.e. Null);  134 Residual
Null Deviance:      172.7
Residual Deviance: 123.1        AIC: 127.1
> hoslem.test(modele$y,fitted(modele),g=10)

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  modele$y, fitted(modele)
X-squared = 6.2179, df = 8, p-value = 0.6228

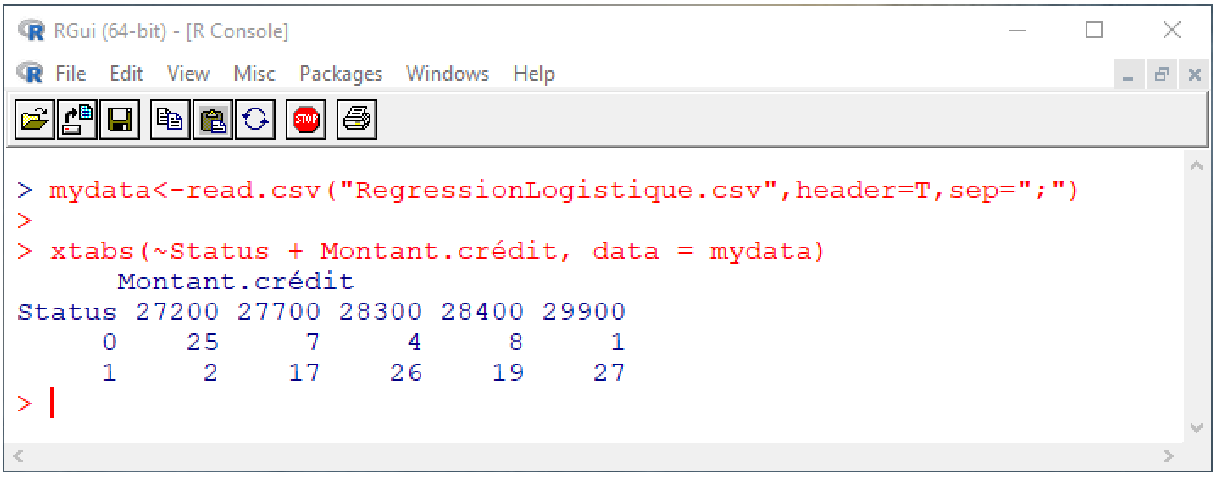
> 
```

Régression binomiale (binaire/scoring) probit

Nous allons reprendre le même jeu de données que ci-dessus et faire une régression probit univariée et ensuite la comparer à la régression logit ci-dessus mais sensiblement avec d'autres approches statistiques ET visuelles.

Rappelons avant cela que nous avons déjà mentionné les différences entre la régression logit et probit dans le cours théorique donc nous ne reviendrons pas dessus!

D'abord nous chargeons le jeu de données qui pour des raisons pédagogiques évidentes seront les mêmes que l'exemple précédent:



Ensuite nous lançons la régression en précisant bien que nous voulons le lien probit:

```
> model_probit <- glm(Status ~ Montant.crédit,
+                     family = binomial(link = "probit"),
+                     data = mydata)
> model_probit

Call:  glm(formula = Status ~ Montant.crédit, family = binomial(link = "probit"),
    data = mydata)

Coefficients:
   (Intercept)  Montant.crédit
    -32.679944        0.001178

Degrees of Freedom: 135 Total (i.e. Null);  134 Residual
Null Deviance:      172.7
Residual Deviance: 125.3        AIC: 129.3
> confint(model_probit)
Waiting for profiling to be done...
                       2.5 %        97.5 %
(Intercept)    -4.541997e+01 -21.524452077
Montant.crédit  7.800537e-04   0.001633633
>
> library("lmtest")
> coeftest(model_probit, type = "HC1")

z test of coefficients:

                 Estimate  Std. Error z value  Pr(>|z|)
(Intercept)    -3.2680e+01  6.4416e+00 -5.0733 3.910e-07
Montant.crédit  1.1782e-03  2.3051e-04  5.1113 3.199e-07

>
```

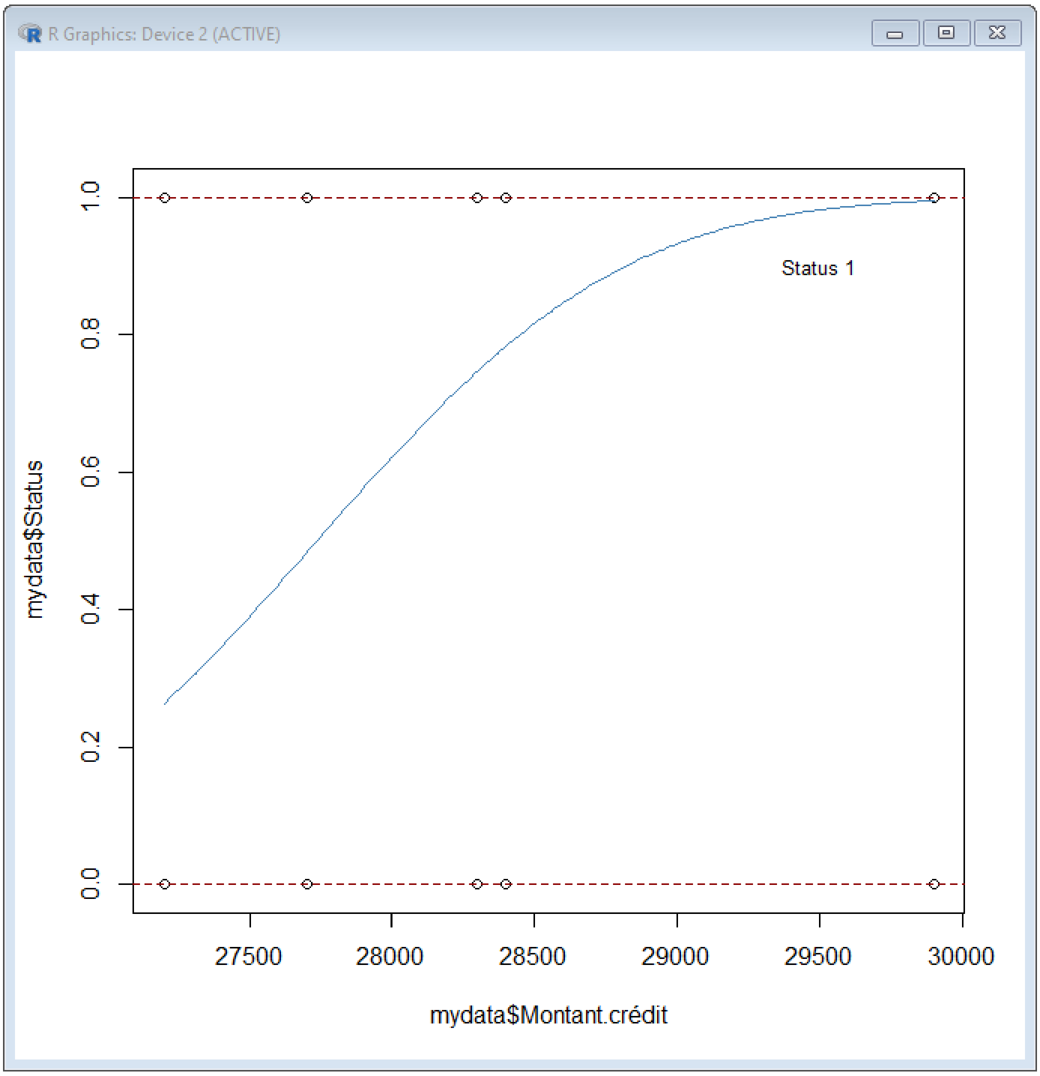
Nous pouvons maintenant faire une graphique d'un genre différente à celui fait dans l'exemple précédent:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> plot(x = mydata$Montant.crédit,
+      y = mydata$Status ,
+      main = "Probit Model of the Probability of Status given Credit",
+      xlab = "Credit",
+      ylab = "Status",
+      pch = 20,
+      ylim = c(-0.4, 1.4),
+      cex.main = 0.85)
>
> plot(x = mydata$Montant.crédit,
+      y = mydata$Status)
>
> abline(h = 1, lty = 2, col = "darkred")
> abline(h = 0, lty = 2, col = "darkred")
> text(29500, 0.9, cex = 0.8, "Status 1")
> text(29500, -0.1, cex= 0.8, "Status 0")
>
> x <- seq(min(mydata$Montant.crédit), max(mydata$Montant.crédit), 10)
> y <- predict(model_probit, list(Montant.crédit= x), type = "response")
>
> lines(x, y, lwd = 1.5, col = "steelblue")
> |
```

Ce qui donnera:



Maintenant voyons pour le plaisir la variation de probabilité lorsqu'on passe d'une valeur de crédit de 28'000.- à 29'000.-:

```
> #calculons la variation de probabilité lorsqu'on passe de 28000 à 29000
> predictions <- predict(model_probit,
+                       newdata = data.frame("Montant.crédit" = c(28000, 29000)),
+                       type = "response")
>
> diff(predictions)
        2
0.3099161
>
```

Enfin comparons avec le modèle de régression logit:

```
> (model_logit <- glm(Status ~ Montant.crédit,
+                     family = binomial(link = "logit"),
+                     data = mydata))

Call:  glm(formula = Status ~ Montant.crédit, family = binomial(link = "logit"),
    data = mydata)

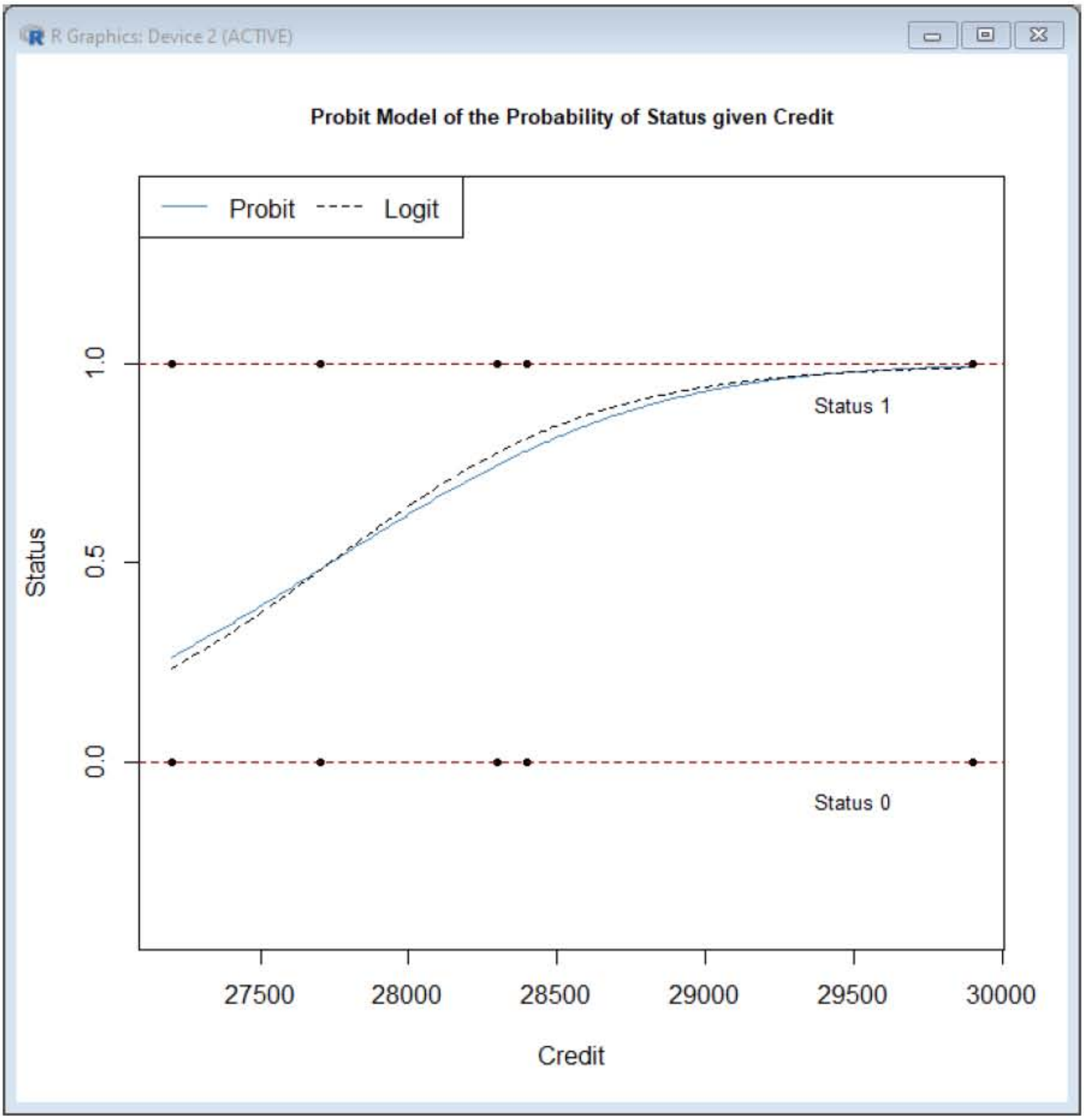
Coefficients:
   (Intercept)  Montant.crédit
    -61.318316        0.002211

Degrees of Freedom: 135 Total (i.e. Null);  134 Residual
Null Deviance:      172.7
Residual Deviance: 123.1        AIC: 127.1
>
```

Et encore mieux (!), comparons graphiquement:

```
> plot(x = mydata$Montant.crédit,
+      y = mydata$Status ,
+      main = "Probit Model of the Probability of Status given Credit",
+      xlab = "Credit",
+      ylab = "Status",
+      pch = 20,
+      ylim = c(-0.4, 1.4),
+      cex.main = 0.85)
>
> abline(h = 1, lty = 2, col = "darkred")
> abline(h = 0, lty = 2, col = "darkred")
> text(29500, 0.9, cex = 0.8, "Status 1")
> text(29500, -0.1, cex= 0.8, "Status 0")
>
> x <- seq(min(mydata$Montant.crédit), max(mydata$Montant.crédit), 10)
> y <- predict(model_probit, list(Montant.crédit= x), type = "response")
>
> lines(x, y, lwd = 1.5, col = "steelblue")
>
> y_logit <- predict(model_logit , list(Montant.crédit= x), type = "response")
> lines(x, y_logit, lwd = 1.5, col = "black", lty = 2)
>
> legend("topleft",
+        horiz = TRUE,
+        legend = c("Probit", "Logit"),
+        col = c("steelblue", "black"),
+        lty = c(1, 2))
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Probit Model of the Probability of Status given Credit
Probit
Logit
Status 1
Status 0
Status
0.0
0.5
1.0
27500
28000
28500
29000
29500
30000
Credit

Odds ratio et Risque Relatif

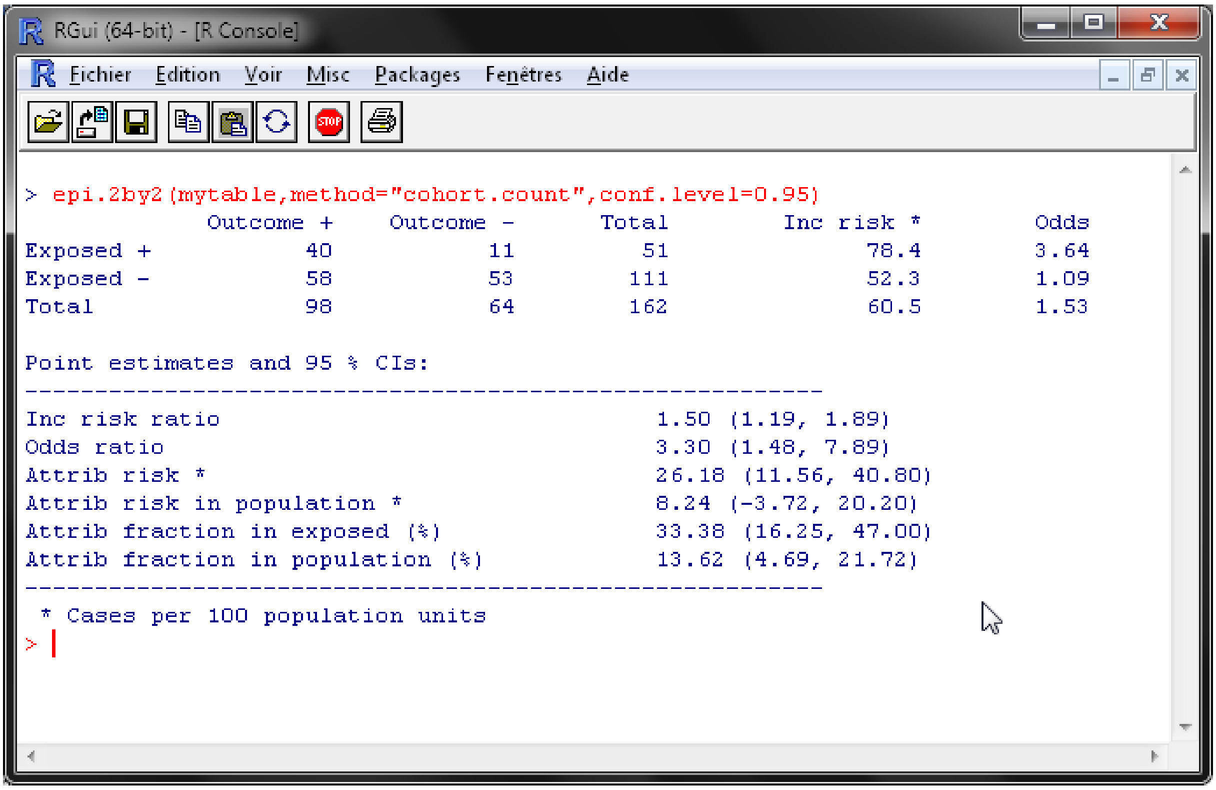
Dans le cours théorique de méthode numérique nous avons introduit le concept de rapport des cotes et de risque relatif. Dans le chapitre de Statistiques nous avons démontré comment obtenir leur intervalle de confiance.

Nous allons vérifier dans un premier temps si nous obtenons les mêmes résultats que ceux calculés à la main et dans un deuxième temps comparer l'odds ratio et son intervalle de confiance avec ce qui était donné par Minitab (ce dernier ne donnant aucune information sur le risque relatif) et le risque relatif et son intervalle de confiance à ce qui était donné par MedCalc.

Nous importons donc d'abord les mêmes données que dans le cours théorique, nous les mettons sous forme de table de contingence et en faisons un petit **barplot( )** classique:



Ensuite, nous utilisons la commande **epi.2by2( )** du package **epiR** pour faire une analyse du point de vue du placebo:



```
> epi.2by2(mytable,method="cohort.count",conf.level=0.95)
             Outcome +    Outcome -      Total        Inc risk *        Odds
Exposed +           40           11         51              78.4        3.64
Exposed -           58           53        111              52.3        1.09
Total               98           64        162              60.5        1.53

Point estimates and 95 % CIs:
---------------------------------------------------------
Inc risk ratio                             1.50 (1.19, 1.89)
Odds ratio                                 3.30 (1.48, 7.89)
Attrib risk *                              26.18 (11.56, 40.80)
Attrib risk in population *                8.24 (-3.72, 20.20)
Attrib fraction in exposed (%)             33.38 (16.25, 47.00)
Attrib fraction in population (%)          13.62 (4.69, 21.72)
---------------------------------------------------------
 * Cases per 100 population units
>
```

Rappel: Un odds ratio de 3.30 signifie pour rappel dans le cas présent que ceux traités au placebo présentent un facteur de 3.30 d'exposition supplémentaire au risque cardiaque par rapport à ceux traités à l'aspirine.

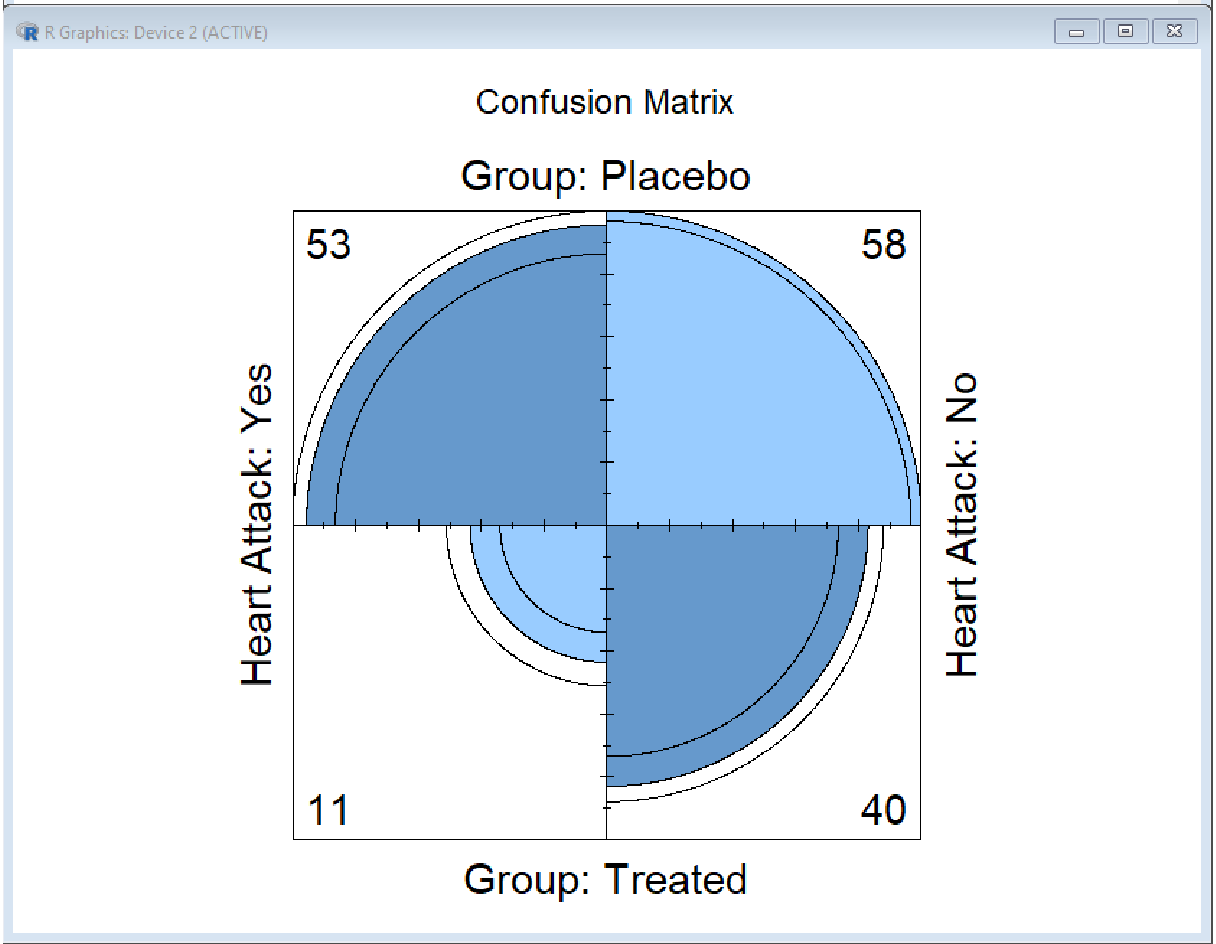
Pour le Inc risk ratio qui n'est en fait que le risque relatif, nous avons donc une valeur de 1.5 (au lieu du 2.21 tel que calculé à la main et avec MedCalc) et un intervalle de confiance de [1.19,1.89] (au lieu de [1.266,3.869] tel que calculé à la main et avec MedCalc). La raison est simple, pour obtenir les mêmes résultats il faut faire l'analyse par rapport à un autre groupe:

```
> table<-matrix(c(53,11,58,40),nrow=2,dimnames=list("Group"=c("Placebo", "Treated"),
+ "Heart Attack"=c("Yes","No")))
> table
         Heart Attack
Group     Yes No
  Placebo  53 58
  Treated  11 40
> library("epiR")
> epi.2by2(table,conf.level=0.95)
             Outcome +    Outcome -      Total        Inc risk *        Odds
Exposed +           53           58        111              47.7       0.914
Exposed -           11           40         51              21.6       0.275
Total               64           98        162              39.5       0.653

Point estimates and 95% CIs:
-------------------------------------------------------------------
Inc risk ratio                               2.21 (1.27, 3.87)
Odds ratio                                   3.32 (1.55, 7.14)
Attrib risk *                                26.18 (11.56, 40.80)
Attrib risk in population *                  17.94 (4.37, 31.51)
Attrib fraction in exposed (%)               54.83 (21.05, 74.16)
Attrib fraction in population (%)            45.40 (13.10, 65.70)
-------------------------------------------------------------------
 Test that OR = 1: chi2(1) = 10.021 Pr>chi2 = 0.00
 Wald confidence limits
 CI: confidence interval
 * Outcomes per 100 population units
> 
```

On peut afficher le graphique fourfold correspondant:

```
> fourfoldplot(table,std ="ind.max",conf.level = 0.95,main="Confusion Matrix")
> sqrt(table/max(table))
          Heart Attack
Group            Yes        No
  Placebo 0.9559253 1.0000000
  Treated 0.4354942 0.8304548
> (or<-(table[1, 1]*table[2, 2])/(table[1, 2]*table[2, 1]))
[1] 3.322884
> (se<-sqrt(sum(1/table)))
[1] 0.3898954
>
```



On peut aussi faire le cacul de façon plus directe et ludique pour le risk ratio mais avec un autre point de vue (comme discuté dans le cours théorique!) – celui de l'aspirine! - en utilisant le package **fmsb** et sa fonction **riskratio( )** qui donne en plus la *p*-value:

```
> library("fmsb")
> oddsratio(a=11, b=53, c=40, d=58, conf.level=0.95, p.calc.by.independence=TRUE)
           Disease Nondisease Total
Exposed         11         40    51
Nonexposed      53         58   111
Total           64         98   162

        Odds ratio estimate and its significance probability

data:  11 53 40 58
p-value = 0.001601
95 percent confidence interval:
 0.1401539 0.6461961
sample estimates:
[1] 0.3009434

>
```

Nous avons donc avec le package **epiR** un odds ratio de 3.30 au lieu du 3.32 tel que calculé à la main et avec Minitab et un intervalle de confiance de [1.48,7.89] au lieu de [1.55,7.14] tel que calculé à la main et avec Minitab. La différence est donc raisonnable...

Ou on peut ré-obtenir le résultat du début:

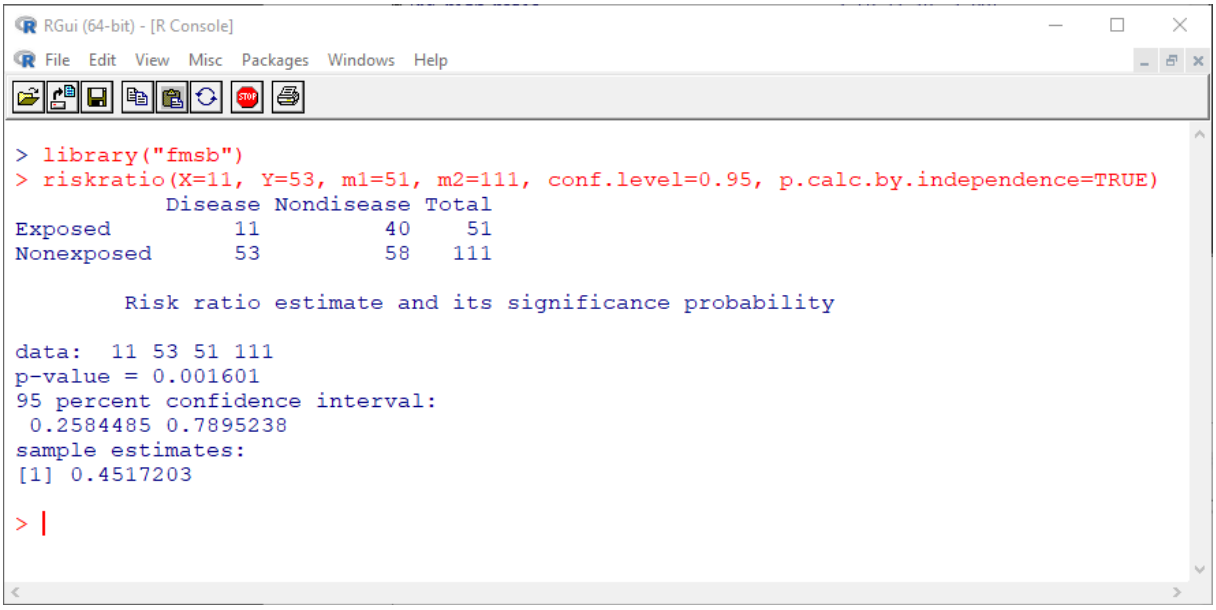
```
> oddsratio(a=40, b=58, c=11, d=53, conf.level=0.95, p.calc.by.independence=TRUE)
           Disease Nondisease Total
Exposed         40         11    51
Nonexposed      58         53   111
Total           98         64   162

        Odds ratio estimate and its significance probability

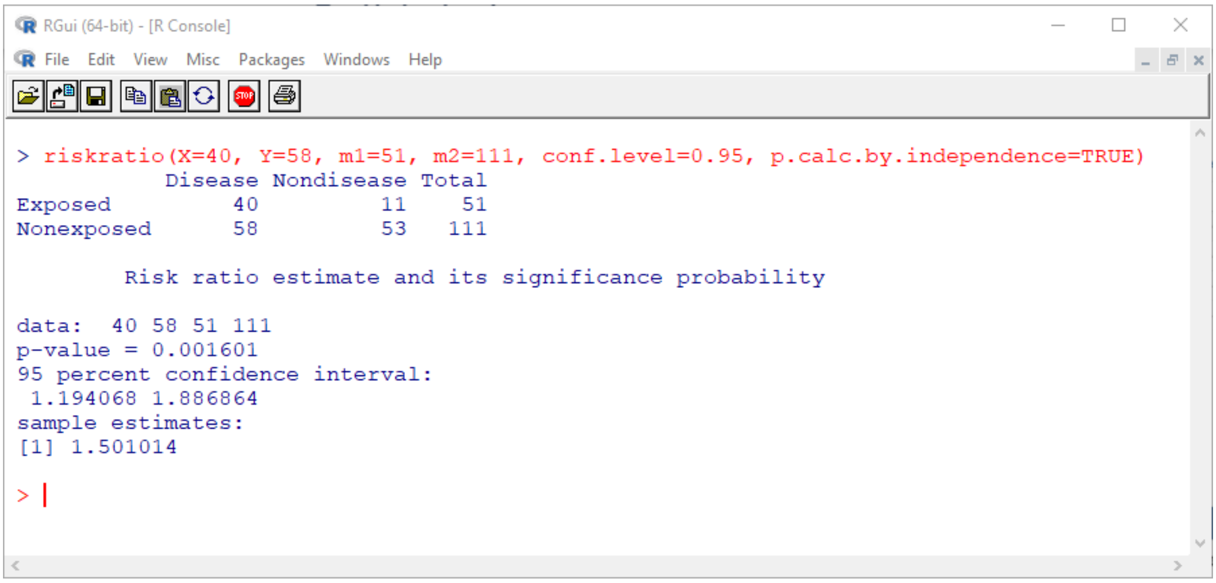
data:  40 58 11 53
p-value = 0.001601
95 percent confidence interval:
 1.547518 7.135012
sample estimates:
[1] 3.322884

>
```

On peut aussi faire le cacul de façon plus directe et ludique pour le risk ratio mais avec un autre point de vue (comme mention dans le cours théorique) – celui de l'aspirine - en utilisant le package **fmsb** et sa fonction **oddsratio( )** qui donne aussi en plus la *p*-value:



Ou on peut ré-obtenir le résultat du début:

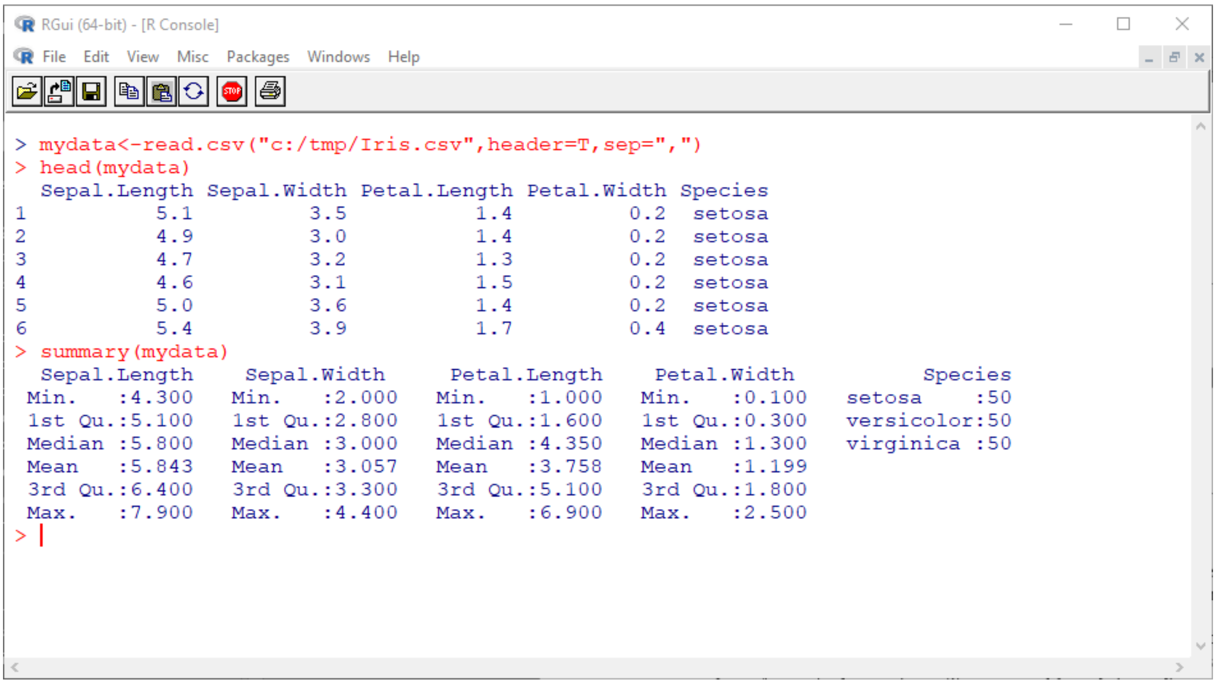


Régression multinomiale (nominale)

Nous avons démontré (non sans mal…) dans le cours théorique que la régression logistique multinomiale était une généralisation de la régression logistique binomiale.

Cependant au vu de la lourdeur des calculs, nous nous sommes abstenus de faire tout calcul à la main ou même dans Microsoft Excel. Le but ici va donc être de vérifier si nous obtenons au moins les mêmes résultats qu'avec Tanagra ou Minitab (pour l'approche utilisant les réseaux de neurones, nous renvoyons le lecteur au chapitre sur la Machine Learning .

Pour faire original (…) nous allons nous baser sur le jeu de données *Iris* de Fisher:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mydata<-read.csv("c:/tmp/Iris.csv",header=T,sep=",")
> head(mydata)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> summary(mydata)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> 
```

D'abord voyons ce que nous pouvons obtenir avec la fonction **vglm( )** du package **VGAM**:

```
> library("VGAM")
> fit <- vglm(Species~.,family=multinomial(refLevel=1), data=mydata )
There were 33 warnings (use warnings() to see them)
> summary(fit)

Call:
vglm(formula = Species ~ ., family = multinomial(refLevel = 1),
    data = mydata)


Pearson residuals:
                       Min         1Q     Median        3Q   Max
log(mu[,2]/mu[,1]) -1.393 -0.0002585 -6.687e-06 0.0004556 1.810
log(mu[,3]/mu[,1]) -1.810 -0.0003211  1.265e-09 0.0002391 1.393

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept):1      7.272  19432.034   0.000    1.000
(Intercept):2    -35.365  19432.051  -0.002    0.999
Sepal.Length:1    -6.777   5790.843      NA       NA
Sepal.Length:2    -9.242   5790.844  -0.002    0.999
Sepal.Width:1     -5.889   2888.838      NA       NA
Sepal.Width:2    -12.569   2888.841  -0.004    0.997
Petal.Length:1    13.201   4446.219   0.003    0.998
Petal.Length:2    22.630   4446.221      NA       NA
Petal.Width:1     15.633   7133.169   0.002    0.998
Petal.Width:2     33.919   7133.176   0.005    0.996

Number of linear predictors:  2

Names of linear predictors: log(mu[,2]/mu[,1]), log(mu[,3]/mu[,1])

Residual deviance: 11.8985 on 290 degrees of freedom

Log-likelihood: -5.9493 on 290 degrees of freedom

Number of iterations: 21

Warning: Hauck-Donner effect detected in the following estimate(s):
'Sepal.Length:1', 'Sepal.Width:1', 'Petal.Length:2'

Reference group is level  1  of the response
>
```

Nous retrouvons donc bien des valeurs très similaires à celles de Minitab. Par contre nous sommes loin du compte relativement à Tanagra (mais nous savons qu'il y a un problème dans ce dernier).

Avec la fonction habituelle predict( ) et du formatage, nous pouvons voir comment chaque ligne du jeu de données initial serait classée à posteriori (nous pourrions refaire une matrice de confusion et autres indicateurs statistiques):

```
> noquote(format(round(predict(fit,mydata[,-5],type = "response"),3), format="f"))
   setosa versicolor virginica
1  1.000  0.000      0.000
2  1.000  0.000      0.000
3  1.000  0.000      0.000
4  1.000  0.000      0.000
5  1.000  0.000      0.000
6  1.000  0.000      0.000
7  1.000  0.000      0.000
8  1.000  0.000      0.000
9  1.000  0.000      0.000
10 1.000  0.000      0.000
11 1.000  0.000      0.000
12 1.000  0.000      0.000
13 1.000  0.000      0.000
14 1.000  0.000      0.000
15 1.000  0.000      0.000
16 1.000  0.000      0.000
17 1.000  0.000      0.000
18 1.000  0.000      0.000
19 1.000  0.000      0.000
20 1.000  0.000      0.000
21 1.000  0.000      0.000
22 1.000  0.000      0.000
23 1.000  0.000      0.000
24 1.000  0.000      0.000
25 1.000  0.000      0.000
26 1.000  0.000      0.000
27 1.000  0.000      0.000
28 1.000  0.000      0.000
29 1.000  0.000      0.000
30 1.000  0.000      0.000
31 1.000  0.000      0.000
32 1.000  0.000      0.000
33 1.000  0.000      0.000
34 1.000  0.000      0.000
35 1.000  0.000      0.000
36 1.000  0.000      0.000
37 1.000  0.000      0.000
38 1.000  0.000      0.000
39 1.000  0.000      0.000
40 1.000  0.000      0.000
41 1.000  0.000      0.000
42 1.000  0.000      0.000
```

Évidemment, la même chose s'appliquerait avec de nouvelles données!

On peut vérifier avec un deuxième package qui a priori à une approche indépendante basée sur la fonction **mlogit( )** du package… **mlogit**:

```
> library("mlogit")
> adjusted_data <- mlogit.data(mydata, choice="Species", shape="wide")
> fit<-mlogit(Species ~ 0 | Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=adjusted_data)
> fit

Call:
mlogit(formula = Species ~ 0 | Sepal.Length + Sepal.Width + Petal.Length +     Petal.Width, data = adjusted_data, method = "nr", print.level = 0)

Coefficients:
 versicolor:(intercept)    virginica:(intercept)  versicolor:Sepal.Length
                 7.7935                 -34.8443                  -7.7171
 virginica:Sepal.Length   versicolor:Sepal.Width    virginica:Sepal.Width
               -10.1823                  -5.9009                 -12.5818
versicolor:Petal.Length   virginica:Petal.Length   versicolor:Petal.Width
                14.6912                  24.1206                  16.7926
  virginica:Petal.Width
                35.0787

> |
```

On retrouve bien le même ordre de grandeur pour les coefficients. Cependant pour les *p*-values on repassera…

Régression ordinale

Nous n'allons pas faire dans l'originalité ici mais compiler les trois publications Internet suivantes qui se complètenet toutes trois dans leur approche et sont basées toutes sur le même jeu de données (il ne nous a pas été possible d'identifier l'ordre chronologique dans lequel ils ont été publié):

- https://stats.idre.ucla.edu/stata/dae/ordered-logistic-regression/
- https://www.analyticsvidhya.com/blog/2016/02/multinomial-ordinal-logistic-regression/
- https://stats.idre.ucla.edu/r/dae/ordinal-logistic-regression/

Chargeons d'abord les packages qui vont être nécessaires ainsi que le jeu données qui va nous accompagner:

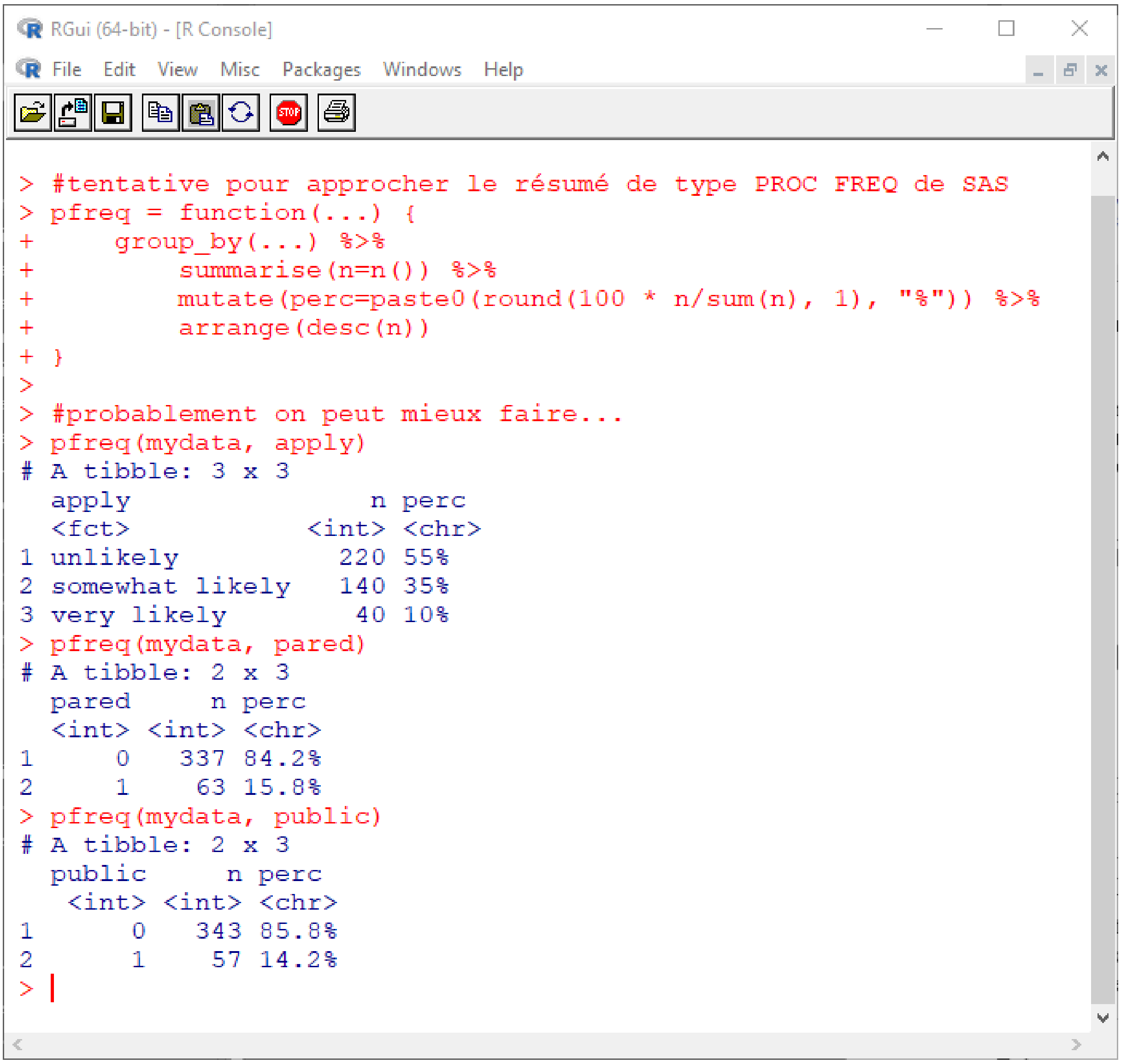
```
> library("foreign")
> library("ggplot2")
> library("MASS")
> library("Hmisc")
> library("reshape2")
>
> mydata <- foreign::read.dta("c:/tmp/ologit.dta")
> head(mydata,10)
             apply pared public  gpa
1      very likely     0      0 3.26
2  somewhat likely     1      0 3.21
3         unlikely     1      1 3.94
4  somewhat likely     0      0 2.81
5  somewhat likely     0      0 2.53
6         unlikely     0      1 2.59
7  somewhat likely     0      0 2.56
8  somewhat likely     0      0 2.73
9         unlikely     0      0 3.00
10 somewhat likely     1      0 3.50
> str(mydata)
'data.frame':   400 obs. of  4 variables:
 $ apply : Factor w/ 3 levels "unlikely","somewhat likely",..: 3 2 1 2 2 1 2 2 1 2 ...
 $ pared : int  0 1 1 0 0 0 0 0 0 1 ...
 $ public: int  0 0 1 0 0 1 0 0 0 0 ...
 $ gpa   : num  3.26 3.21 3.94 2.81 2.53 ...
 - attr(*, "datalabel")= chr ""
 - attr(*, "time.stamp")= chr " 3 Aug 2010 10:45"
 - attr(*, "formats")= chr  "%15.0g" "%9.0g" "%9.0g" "%9.0g"
 - attr(*, "types")= int  251 251 251 254
 - attr(*, "val.labels")= chr  "apply" "" "" ""
 - attr(*, "var.labels")= chr  "" "" "" ""
 - attr(*, "version")= int 12
 - attr(*, "label.table")=List of 1
  ..$ apply: Named int  0 1 2
  .. ..- attr(*, "names")= chr  "unlikely" "somewhat likely" "very likely"
> 
```

Pour les francophones, voici un explication du jeu de données:

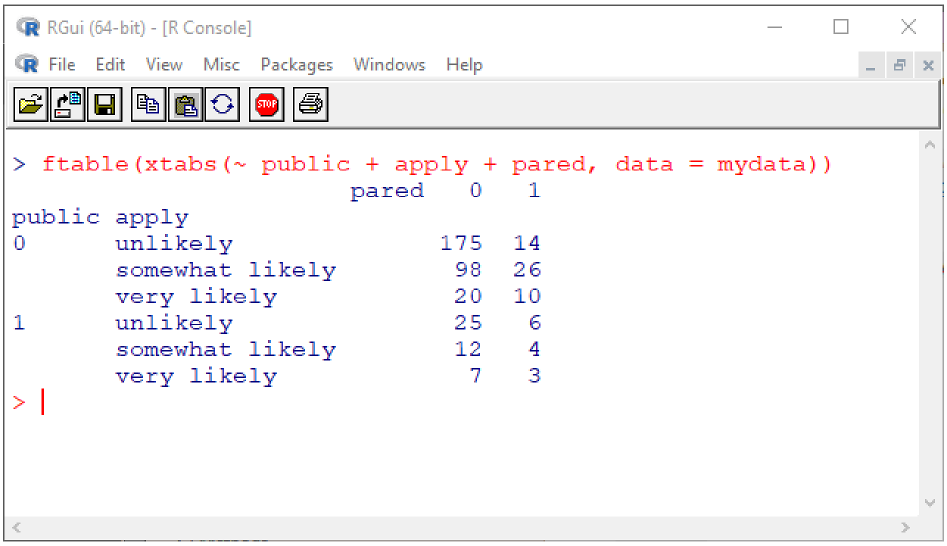
Une étude souhaite examiner les facteurs qui influencent la décision de postuler ou non à des études supérieures. On demande à des lycéens s'il est peu probable (*unlikely*), assez probable (*likely*) ou très probable (*very likely*) qu'ils postulent à des études supérieures. Par conséquent, notre variable de résultat comporte trois catégories ordonnées.

Des données sur le statut éducatif des parents, qu'un des deux au moins a suivi une formation supérieure (variable nommée *pared* pour l'abréviation de "parent education" dont la valeur est 0/1), et une donnée si le lycée de l'élève est privé ou publique (variable nommée *public* dont la valeur est 0/1), et le GPA (score scolaire aux USA) des lycéens sont collectés. Les chercheurs ont des raisons de croire que les "distances" entre ces trois points ne sont pas égales. Par exemple, la "distance" entre «peu probable» et «assez probable» peut être plus courte que la distance entre «assez probable» et «très probable».

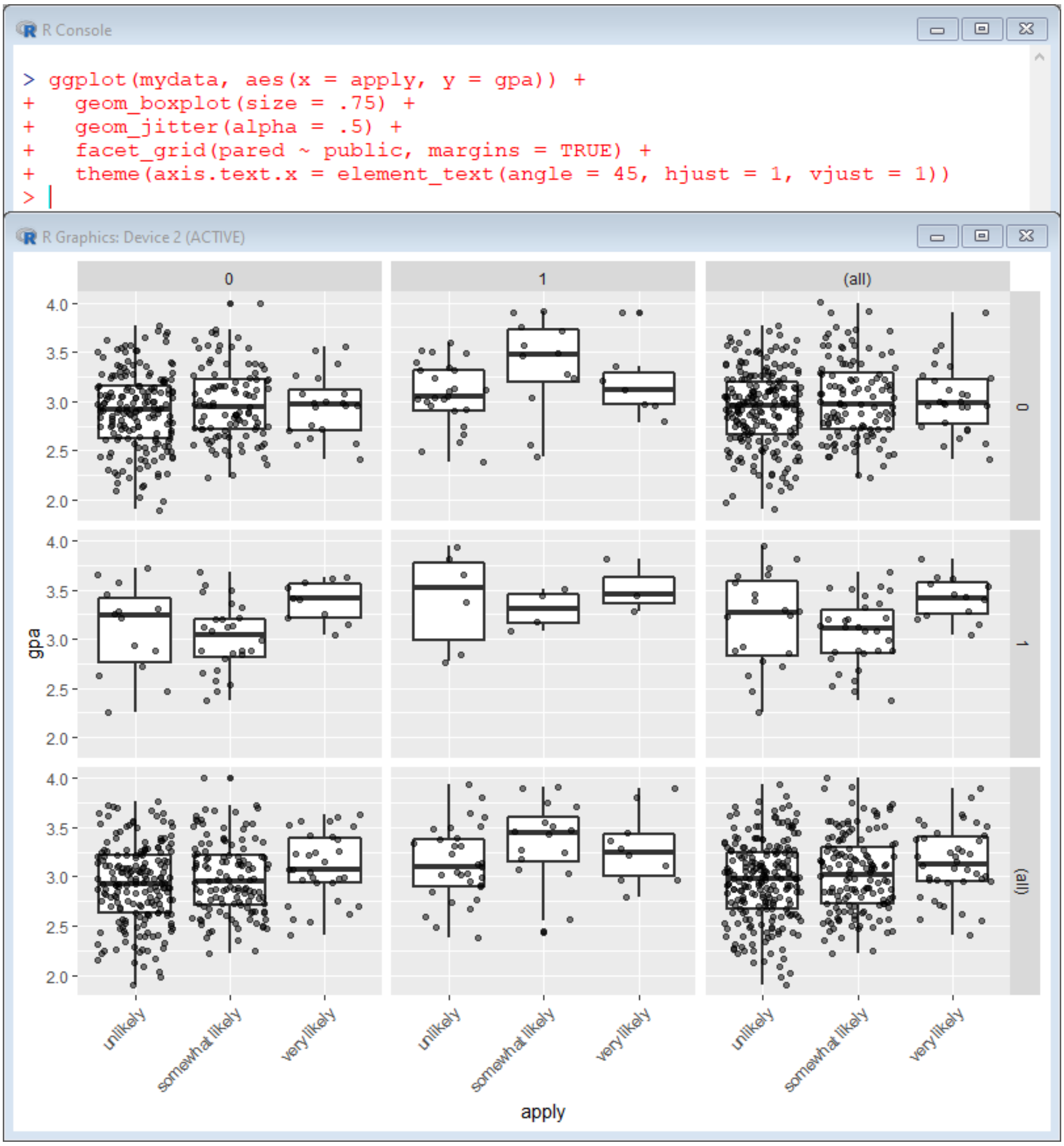
Faisons avant de poursuivre, une autre résumé statistique de jeu de données:



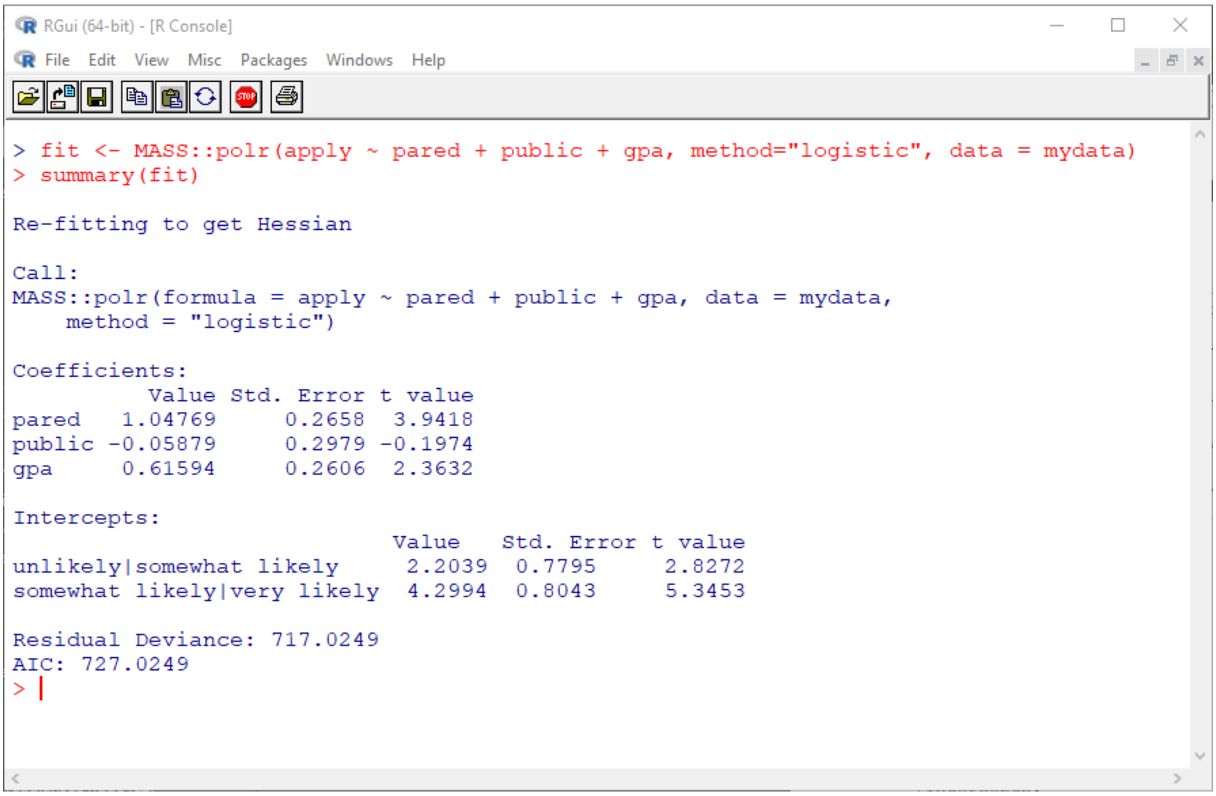
Ou encore une autre résumé pour aider à comprendre le jeu de données:



Ou pour ceux qui préfèrent les visuels:



Nous lançons la régression ordinale simple (sans interaction et d'ordre 1) avec la fonction **polr( )** du package MASS:



En d'autres termes:

$$
\begin{aligned}
\operatorname{logit}(P(Y \leq 1)) &= 2.20 - 1.05 \cdot \text{PARED} - (-0.06) \cdot \text{PUBLIC} - 0.616 \cdot \text{GPA} \\
\operatorname{logit}(P(Y \leq 2)) &= 4.30 - 1.05 \cdot \text{PARED} - (-0.06) \cdot \text{PUBLIC} - 0.616 \cdot \text{GPA}
\end{aligned}
$$

Nous pouvons calculer les p-values des coefficients de différentes manières. D'abord manuellement:

```
> (coeff_table <- coef(summary(fit)))

Re-fitting to get Hessian

                              Value Std. Error    t value
pared                    1.04769010  0.2657894  3.9418053
public                  -0.05878572  0.2978611 -0.1973595
gpa                      0.61594057  0.2606337  2.3632420
unlikely|somewhat likely 2.20391473  0.7795447  2.8271820
somewhat likely|very likely 4.29936315  0.8043260  5.3452991
> #approximation normale valable que pour les grands échantillons
> p_normal <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
> #ou mieux en utilisant la loi de student "pt"
> p_student <- pt(abs(coeff_table[, "t value"]), df=fit$df.residual, lower.tail = FALSE) * 2
> (p_table <- cbind(coeff_table, "p value Approx" = p_normal,"p value Exact" = p_student))
                              Value Std. Error    t value p value Approx p value Exact
pared                    1.04769010  0.2657894  3.9418053   8.087062e-05  9.564431e-05
public                  -0.05878572  0.2978611 -0.1973595   8.435462e-01  8.436477e-01
gpa                      0.61594057  0.2606337  2.3632420   1.811583e-02  1.859943e-02
unlikely|somewhat likely 2.20391473  0.7795447  2.8271820   4.695963e-03  4.934827e-03
somewhat likely|very likely 4.29936315  0.8043260  5.3452991   9.026786e-08  1.529727e-07
>
```

Ou avec la fonction **clm( )** du package **ordinal**:

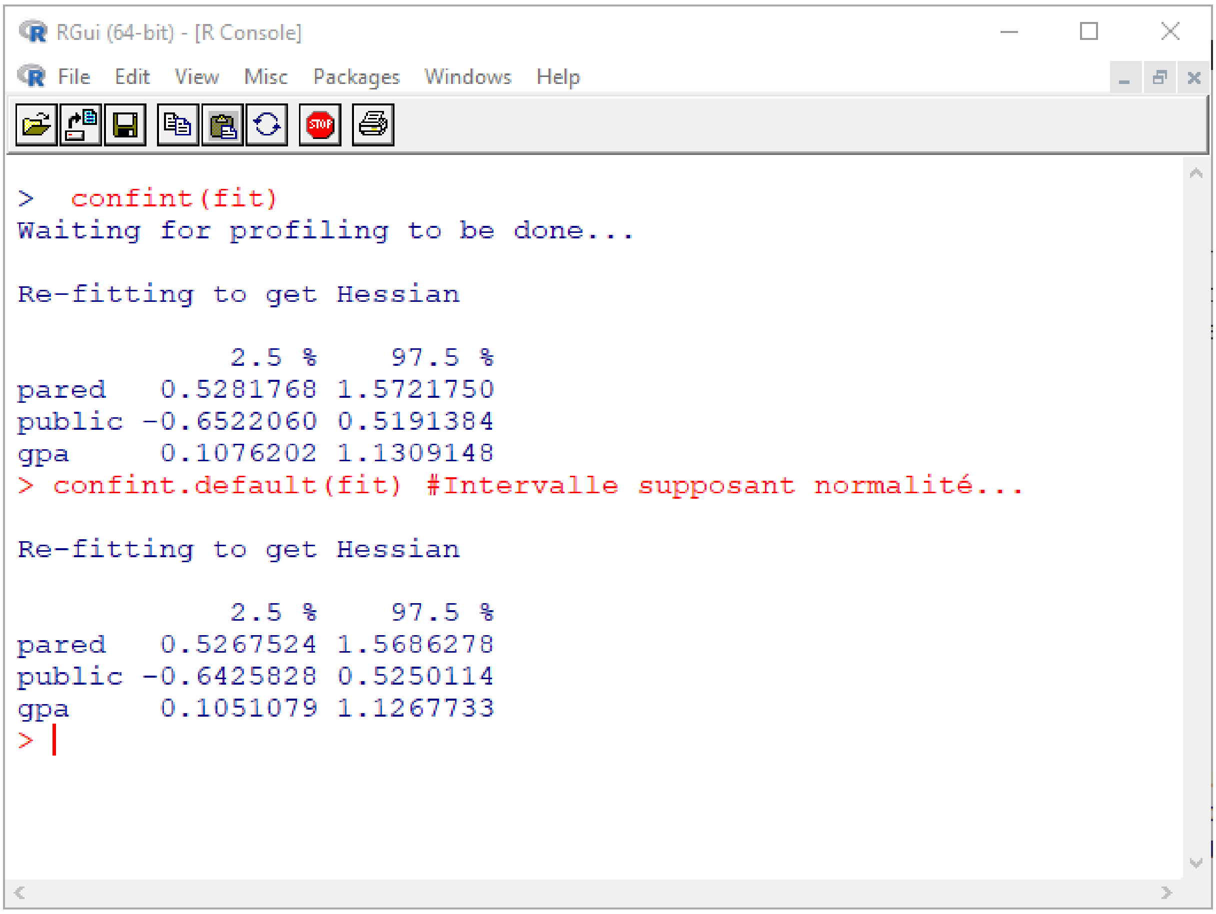
```
> library("ordinal")
> fit_alternative <- clm(apply ~ pared + public + gpa, data = mydata)
> summary(fit_alternative)
formula: apply ~ pared + public + gpa
data:    mydata

 link  threshold nobs logLik  AIC    niter max.grad cond.H
 logit flexible  400  -358.51 727.02 5(0)  1.63e-10 1.3e+03

Coefficients:
       Estimate Std. Error z value Pr(>|z|)
pared   1.04766    0.26579   3.942 8.09e-05
public -0.05868    0.29786  -0.197   0.8438
gpa     0.61575    0.26063   2.363   0.0182

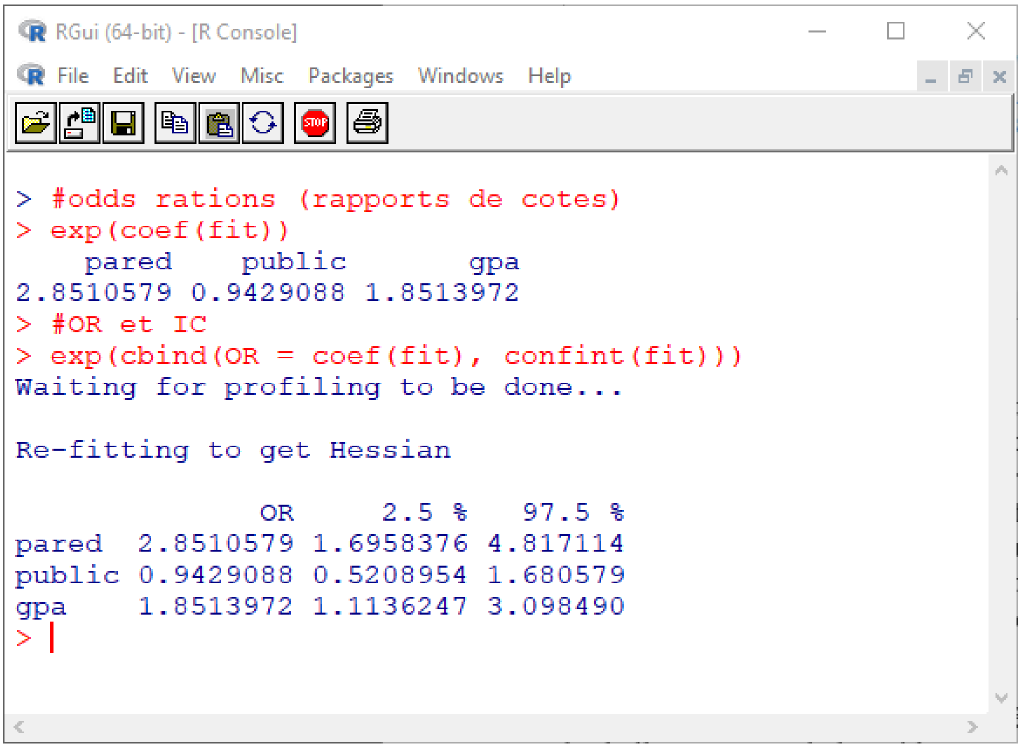
Threshold coefficients:
                            Estimate Std. Error z value
unlikely|somewhat likely      2.2033     0.7795   2.826
somewhat likely|very likely   4.2988     0.8043   5.345
>
```

Nous pouvons aussi obtenir les intervalles de confiance des coefficients:



Les IC à la fois pour *pared* et *gpa* ne comprennent pas 0; *public* le fait. Les estimations des sortie sont données en unités de logits ordonnés, ou en logarithmes des cotes ordonnés. Donc, pour la variable *pared*, nous dirions que pour une augmentation d'une unité de *pared* (c.-à-d. allant de 0 à 1), nous nous attendons à une augmentation de 1.05 de la valeur attendue de *apply* sur l'échelle log odds, compte tenu de toutes les autres variables modèle sont maintenues constantes. Pour *gpa*, nous dirions que pour une augmentation d'une unité de *gpa*, nous nous attendrions à une augmentation de 0.62 de la valeur attendue de *appply* dans l'échelle log odds, étant donné que toutes les autres variables du modèle sont maintenues constantes.

Les coefficients du modèle peuvent être quelque peu difficiles à interpréter car ils sont mis à l'échelle en termes de logarithmes. Une autre façon d'interpréter les modèles de régression logistique consiste à convertir les coefficients en odds ratios. Pour obtenir les OR et les intervalles de confiance, nous exponentions simplement les estimations et les intervalles de confiance:



Ces coefficients sont les rapports de cotes proportionnels et nous les interpréterions à peu près comme nous le ferions les rapports de cotes d'une régression logistique binaire.

Il existe de nombreuses interprétations équivalentes du rapport de cotes en fonction de la façon dont la probabilité est définie et de la direction des cotes[28] (le symbole * dénote la méthode la plus simple à interpréter).

Pared (Parental Education):

1. * Pour les étudiants dont les parents ont fréquenté le collège, les chances d'être plus susceptibles (c.-à-d. Très ou assez probable versus peu probable) de postuler sont de 2.85 fois celles des étudiants dont les parents ne sont pas allés au collège, en maintenant constantes toutes les autres variables.

2. Pour les étudiants dont les parents n'ont pas fréquenté le collège, les chances d'être moins susceptibles de postuler (c.-à-d. Peu probable ou plutôt ou très probable) sont 2.85 fois supérieures à celles des étudiants dont les parents sont allés au collège, en maintenant constantes toutes les autres variables.

School Type (public)

- Pour les élèves des écoles publiques, les chances d'être plus susceptibles (c.-à-d. Très ou plutôt probables contre peu probables) de présenter une demande sont de 5.71% inférieures [c.-à-d. (1 -0.943) x 100%] que les élèves des écoles privées, , en maintenant constantes toutes les autres variables.

[28] Pour les détails voir https://stats.idre.ucla.edu/r/faq/ologit-coefficients/

    - * Pour les élèves des écoles privées, les chances d'être plus susceptibles de postuler sont de 1.06 fois [soit 1 / 0.943] que celles des élèves des écoles publiques, en maintenant constantes toutes les autres variables (rapport de cotes positif).

- Pour les élèves des écoles privées, les chances d'être moins susceptibles de postuler (c.-à-d. Improbables contre plutôt ou très probables) sont de 5.71% inférieures à celles des élèves des écoles publiques, en maintenant constantes toutes les autres variables.

    - Pour les élèves des écoles publiques, la probabilité d'être moins susceptible de postuler est de 1.06 fois celle des élèves des écoles privées, en maintenant constantes toutes les autres variables (rapport de cotes positif).

GPA

1. * Pour chaque augmentation d'une unité de la moyenne pondérée cumulative des étudiants, les chances d'être plus susceptibles de postuler (très ou plutôt probable par rapport à peu probable) sont multipliées par 1.85 fois (c.-à-d., augmente de 85%), en maintenant constantes toutes les autres variables.

2. Pour chaque diminution d'une unité de la moyenne pondérée cumulative des étudiants, les chances d'être moins susceptibles de postuler (peu probable ou plutôt ou très probable) sont multipliées par 1.85 fois, en maintenant constantes toutes les autres variables.

Comme mentionné dans le cours théorique, tester l'hypothèse des cotes proportionnels semble délicat. Nous allons donc ici utiliser la méthode proposée par Harrell. Rappelons que ce dernier propose une méthode graphique pour valider l'hypothèse.

Les valeurs affichées dans le graphique proposées par Harrell sont essentiellement des prédictions (linéaires) d'un modèle logit, utilisées pour modéliser la probabilité que y soit supérieur ou égal à une valeur donnée (pour chaque niveau de *y*), en utilisant une variable prédictive (*x*) à un temps. Pour créer ce graphique, nous aurons besoin de la libraire **Hmisc**.

Le code ci-dessous contient deux parties et est utilisé pour créer le graphique de Harrell pour tester l'hypothèse de cotes proportionnelles. Fondamentalement, nous allons représenter graphiquement les logits prédits à partir de régressions logistiques individuelles avec un seul prédicteur où les groupes de résultats sont définis soit par *apply* = 2 et par *apply* = 3. Si la différence entre les logits prédits pour différents niveaux d'un prédicteur, disons pared, est même si le résultat est défini par *apply*> = 2 ou *apply*> = 3, alors nous pouvons être a peu près confiants que l'hypothèse de scotes proportionnelles est vraie. En d'autres termes, si la différence entre les logits pour *pared* = 0 et *pared* = 1 est la même lorsque le résultat est *apply*> = 2 que la différence lorsque le résultat est *apply*> = 3, alors l'hypothèse de cotes proportionnelles tient probablement.

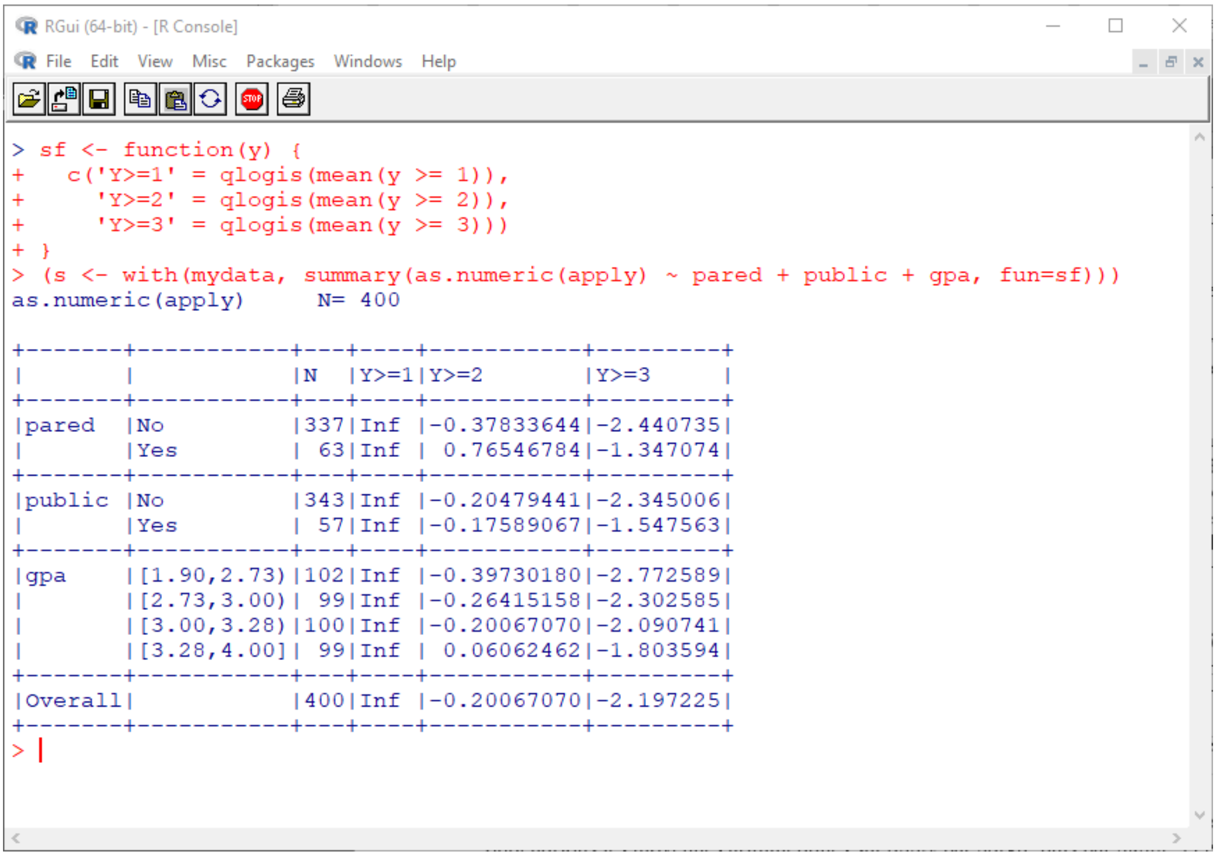
La première partie crée la fonction qui estime les valeurs qui seront représentées graphiquement. La première ligne de cette commande indique à R que **sf( )** sera une fonction, et que cette fonction prend un argument, que nous nommons *y*. La fonction **sf( )** calculera la probabilité logarithmique d'être supérieure ou égale à chaque valeur de la variable cible. Pour nos besoins, nous aimerions que les log des cotes de *apply* soient supérieures ou égales à 2,

puis supérieures ou égales à 3. Selon le nombre de catégories devotre variable dépendante et le codage de vos variables, vous pouvez si nécesseaire modifier cette fonction.

Remarque: Ci-dessous, la fonction est configurée pour une variable *y* à trois niveaux, 1, 2, 3. Si votre variable dépendante a 4 niveaux, étiquetés 1, 2, 3, 4, vous devrez ajouter 'Y> = 4' = qlogis (mean(y> = 4)) à l'intérieur du premier ensemble de parenthèses. Si votre variable dépendante était codée 0, 1, 2 au lieu de 1, 2, 3, vous devez modifier le code, en remplaçant chaque instance de 1 par 0, 2 par 1, etc.

À l'intérieur de la fonction **sf( )**, nous trouvons la fonction **qlogis( )**, qui transforme une probabilité en logit. Donc, nous allons essentiellement alimenter les probabilités de *apply* supérieures à 2 ou 3 à qlogis, et cela retournera les transformations logit de ces probabilités. À l'intérieur de la fonction qlogis, nous voyons que nous voulons que les logarithmes des cotes de la moyenne de *y*> = 2. Lorsque nous fournissons un argument tel que *apply*, à la fonction *apply* , *y*> = 2 sera évalué à 0/1 (FAUX / VRAI), et en prenant la moyenne de ce vecteur, vous obtiendrez la proportion ou la probabilité que *apply*> = 2.

La deuxième commande ci-dessous avec la fonction **with( )** appelle la fonction **sf( )** sur plusieurs sous-ensembles des données définies par les prédicteurs. Dans cette commande, nous voyons la fonction **summary( )** avec une formule fournie comme premier argument. Lorsque R voit un appel à **summary( )** avec un argument de formule, il calcule les statistiques descriptives de la variable sur le côté gauche de la formule par groupes sur le côté droit de la formule et renvoie les résultats dans un joli tableau. Par défaut, la fonction **summary( )** calcule la moyenne de la variable de gauche. Donc, si nous avions utilisé le code **summary(as.numeric (apply) ~ pared + public + gpa)** sans l'argument *fun*, nous obtiendrions les moyennes arithmétiques sur *apply* par *pared*, puis par *public*, et enfin par *gpa* divisé en 4 égaux groupes. Cependant, nous pouvons remplacer le calcul de la moyenne en fournissant notre propre fonction, à savoir **sf( )** à l'argument *fun* =.



```
> sf <- function(y) {
+   c('Y>=1' = qlogis(mean(y >= 1)),
+     'Y>=2' = qlogis(mean(y >= 2)),
+     'Y>=3' = qlogis(mean(y >= 3)))
+ }
> (s <- with(mydata, summary(as.numeric(apply) ~ pared + public + gpa, fun=sf)))
as.numeric(apply)     N= 400

+-------+-----------+---+----+-----------+---------+
|       |           |N  |Y>=1|Y>=2       |Y>=3     |
+-------+-----------+---+----+-----------+---------+
|pared  |No         |337|Inf |-0.37833644|-2.440735|
|       |Yes        | 63|Inf | 0.76546784|-1.347074|
+-------+-----------+---+----+-----------+---------+
|public |No         |343|Inf |-0.20479441|-2.345006|
|       |Yes        | 57|Inf |-0.17589067|-1.547563|
+-------+-----------+---+----+-----------+---------+
|gpa    |[1.90,2.73)|102|Inf |-0.39730180|-2.772589|
|       |[2.73,3.00)| 99|Inf |-0.26415158|-2.302585|
|       |[3.00,3.28)|100|Inf |-0.20067070|-2.090741|
|       |[3.28,4.00]| 99|Inf | 0.06062462|-1.803594|
+-------+-----------+---+----+-----------+---------+
|Overall|           |400|Inf |-0.20067070|-2.197225|
+-------+-----------+---+----+-----------+---------+
> 
```

Le tableau ci-dessus affiche les valeurs prédites (linéaires) que nous obtiendrions si nous régressions notre variable dépendante sur nos variables prédictives une à la fois, sans l'hypothèse de pentes parallèles.

Nous pouvons évaluer l'hypothèse des pentes parallèles en exécutant une série de régressions logistiques binaires avec différents points de coupure sur la variable dépendante et en vérifiant l'égalité des coefficients entre les points de coupure. Nous assouplissons ainsi l'hypothèse des pentes parallèles pour vérifier sa tenabilité.

Pour ce faire, nous transformons la variable dépendante ordinale d'origine en une nouvelle variable dépendante binaire qui est égale à zéro si la variable dépendante ordinale d'origine (ici *apply*) est inférieure à une valeur donnée *a* et 1 si la variable ordinale est supérieur ou égal à cette même valeur donnée *a* (notez que c'est également ce que représentent les coefficients du modèle de régression ordinale).

Ceci est fait pour *k*-1 niveaux de la variable ordinale et est effectué par le code **as.numeric(*apply*)> =** *a* ci-dessous.

La première ligne de code ci-dessous estime l'effet de *pared* sur le choix de «unlikely» par rapport à «somewhat likely» ou «very likely» du la variable *apply*. La deuxième ligne de code estime l'effet de *pared* sur le choix de «unlikely» ou «somewhat likely» par rapport à «very likely» de la variable *apply*.

En regardant l'ordonnée à l'origine de ce modèle (-0.3783), nous voyons qu'il correspond à la valeur prédite dans la cellule pour *pared* égal à «no» dans la colonne pour $Y >= 1$, la valeur en dessous, pour *pared* égal à «yes» est égal eà l'ordonnée à l'origine plus le coefficient de *pared* (c'est-à-dire -0.3783 + 1.1438 = 0.765).

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> glm(I(as.numeric(apply) >= 2) ~ pared, family="binomial", data = mydata)

Call:  glm(formula = I(as.numeric(apply) >= 2) ~ pared, family = "binomial",
    data = mydata)

Coefficients:
(Intercept)        pared
    -0.3783       1.1438

Degrees of Freedom: 399 Total (i.e. Null);  398 Residual
Null Deviance:      550.5
Residual Deviance: 534.1        AIC: 538.1
> glm(I(as.numeric(apply) >= 3) ~ pared, family="binomial", data = mydata)

Call:  glm(formula = I(as.numeric(apply) >= 3) ~ pared, family = "binomial",
    data = mydata)

Coefficients:
(Intercept)        pared
     -2.441        1.094

Degrees of Freedom: 399 Total (i.e. Null);  398 Residual
Null Deviance:      260.1
Residual Deviance: 252.2        AIC: 256.2
> 
```

Nous pouvons utiliser les valeurs du tableau précédent pour nous aider à évaluer si l'hypothèse de cotes proportionnelles est raisonnable pour notre modèle (on reproduit la table encore une fois ci-dessous par commodité):

```
> sf <- function(y) {
+   c('Y>=1' = qlogis(mean(y >= 1)),
+     'Y>=2' = qlogis(mean(y >= 2)),
+     'Y>=3' = qlogis(mean(y >= 3)))
+ }
> (s <- with(mydata, summary(as.numeric(apply) ~ pared + public + gpa, fun=sf)))
as.numeric(apply)     N= 400

+-------+-----------+---+----+-----------+---------+
|       |           |N  |Y>=1|Y>=2       |Y>=3     |
+-------+-----------+---+----+-----------+---------+
|pared  |No         |337|Inf |-0.37833644|-2.440735|
|       |Yes        | 63|Inf | 0.76546784|-1.347074|
+-------+-----------+---+----+-----------+---------+
|public |No         |343|Inf |-0.20479441|-2.345006|
|       |Yes        | 57|Inf |-0.17589067|-1.547563|
+-------+-----------+---+----+-----------+---------+
|gpa    |[1.90,2.73)|102|Inf |-0.39730180|-2.772589|
|       |[2.73,3.00)| 99|Inf |-0.26415158|-2.302585|
|       |[3.00,3.28)|100|Inf |-0.20067070|-2.090741|
|       |[3.28,4.00]| 99|Inf | 0.06062462|-1.803594|
+-------+-----------+---+----+-----------+---------+
|Overall|           |400|Inf |-0.20067070|-2.197225|
+-------+-----------+---+----+-----------+---------+
>
```

Par exemple, lorsque *pared* est égal à «No», la différence entre la valeur prévue pour *apply* supérieur ou égal à deux et *apply* supérieur ou égal à trois est à peu près 2:

$$-0.378 - -2.440 = 2.062$$

Pour un *pared* égal à «Yes», la différence des valeurs prévues pour *apply* supérieur ou égal à deux et appliquer *apply* ou égal à trois est également d'environ 2:

$$0.765 - -1.347 = 2.112$$

Cela suggère que l'hypothèse des pentes parallèles est raisonnable (ces différences sont ce que le graphique ci-dessous trace).

Tournons notre attention vers les prédictions avec *public* comme variable prédictive, nous voyons que lorsque *public* est réglé sur «No», la différence de prédictions pour *apply* supérieur ou égal à deux, par rapport à *apply* supérieur ou égal à trois est d'environ 214:

$$- 0.204 - -2.345 = 2.141$$

Lorsque public est défini sur «Yes», la différence entre les coefficients est d'environ 1.37:

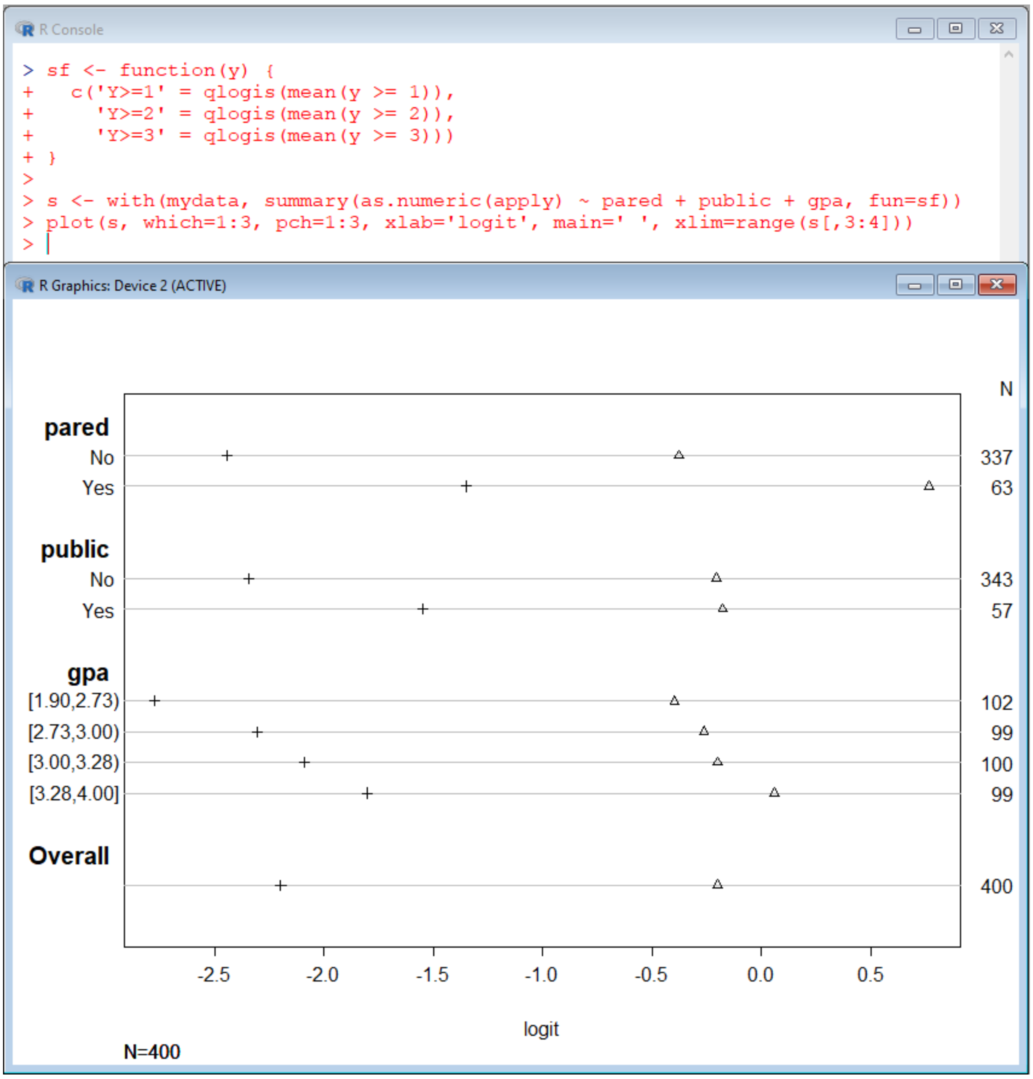
$$-0.175 - -1.547 = 1.372$$

Les différences de distance entre les deux ensembles de coefficients (2.14 contre 1.37) peuvent suggérer que l'hypothèse des pentes parallèles ne s'applique pas au prédicteur public!

Cela indiquerait que l'effet de la fréquentation d'une école publique par rapport à une école privée est différent pour la transition de «very likely» à «something likely» et «something likely» à «very likely».

La commande **plot( )** ci-dessous indique à R que l'objet que nous souhaitons tracer est l'objet *s*. La commande which = 1: 3 est une liste de valeurs indiquant les niveaux de *y* qui doit être inclus dans le tracé. Si votre variable dépendante avait plus de trois niveaux, vous devez remplacer le 3 par le nombre de catégories (par exemple, 4 pour une variable à quatre catégories, même si elle est numérotée 0, 1, 2, 3). La commande pch = 1: 3 sélectionne les marqueurs à utiliser et est facultative, tout comme xlab = 'logit' qui étiquette l'axe des *x* et main = ' ' qui définit l'étiquette principale du graphique à vide.

Si l'hypothèse de cotes proportionnelles est vérifiée, pour chaque variable prédictive, la distance entre les symboles pour chaque ensemble de catégories de la variable dépendante devrait rester similaire. Pour aider à le démontrer, nous avons normalisé tout le premier ensemble de coefficients à zéro, donc il y a un point de référence commun. En regardant les coefficients de la variable *pared*, nous voyons que la distance entre les deux ensembles de coefficients est similaire. En revanche, les distances entre les estimations pour *public* sont différentes (c'est-à-dire que les marqueurs sont beaucoup plus éloignés sur la deuxième ligne que sur la première), ce qui suggère que l'hypothèse de cotes proportionnelles peut ne pas tenir!



Sinon nous pouvons exécuter la test de Brant avec la fonction **brant( )** avec package **brant** ou avec la fontion **poTest( )** du package **car**:

```
> #test des cotes proportionnels:
> library("brant")
> brant(fit)
--------------------------------------------
Test for        X2      df      probability
--------------------------------------------
Omnibus         4.34    3       0.23
pared           0.13    1       0.72
public          3.44    1       0.06
gpa             0.18    1       0.67
--------------------------------------------

H0: Parallel Regression Assumption holds               X2 df probability
Omnibus 4.3429046  3  0.22673821
pared   0.1323413  1  0.71601675
public  3.4433562  1  0.06350656
gpa     0.1791233  1  0.67212771
> #aussi le test de Brant mais sous une autre forme
> library("car")
> poTest(fit)

Tests for Proportional Odds
MASS::polr(formula = apply ~ pared + public + gpa, data = mydata,
    method = "logistic")

        b[polr] b[>unlikely] b[>somewhat likely] Chisquare df Pr(>Chisq)
Overall                                               4.34  3      0.227
pared    1.0477       1.0596              0.9156      0.13  1      0.716
public  -0.0588      -0.2006              0.5351      3.44  1      0.064
gpa      0.6159       0.5482              0.7363      0.18  1      0.672
> 
```

Une fois que nous avons terminé d'évaluer si les hypothèses de notre modèle se vérifient, nous pouvons calculer des prédictins de probabilités, qui sont généralement plus faciles à comprendre que les coefficients ou les rapports de cotes. Par exemple, nous pouvons faire varier le *gpa* pour chaque niveau de *pared* et *public* et calculer la probabilité d'être dans chaque catégorie de *apply*. Pour ce faire, nous créons un petit nouvel ensemble de 20 données de toutes les valeurs à utiliser pour la prédiction:

```
> (nouvelles_donnees_test <- data.frame(
+   pared = rep(0:1, each = 5),
+   public = rep(0:1, each = 10),
+   gpa = rep(seq(from = 1.9, to = 4, length.out = 5), 4)))
   pared public   gpa
1      0      0 1.900
2      0      0 2.425
3      0      0 2.950
4      0      0 3.475
5      0      0 4.000
6      1      0 1.900
7      1      0 2.425
8      1      0 2.950
9      1      0 3.475
10     1      0 4.000
11     0      1 1.900
12     0      1 2.425
13     0      1 2.950
14     0      1 3.475
15     0      1 4.000
16     1      1 1.900
17     1      1 2.425
18     1      1 2.950
19     1      1 3.475
20     1      1 4.000
> 
```

Et allons-y pour la prédiction:

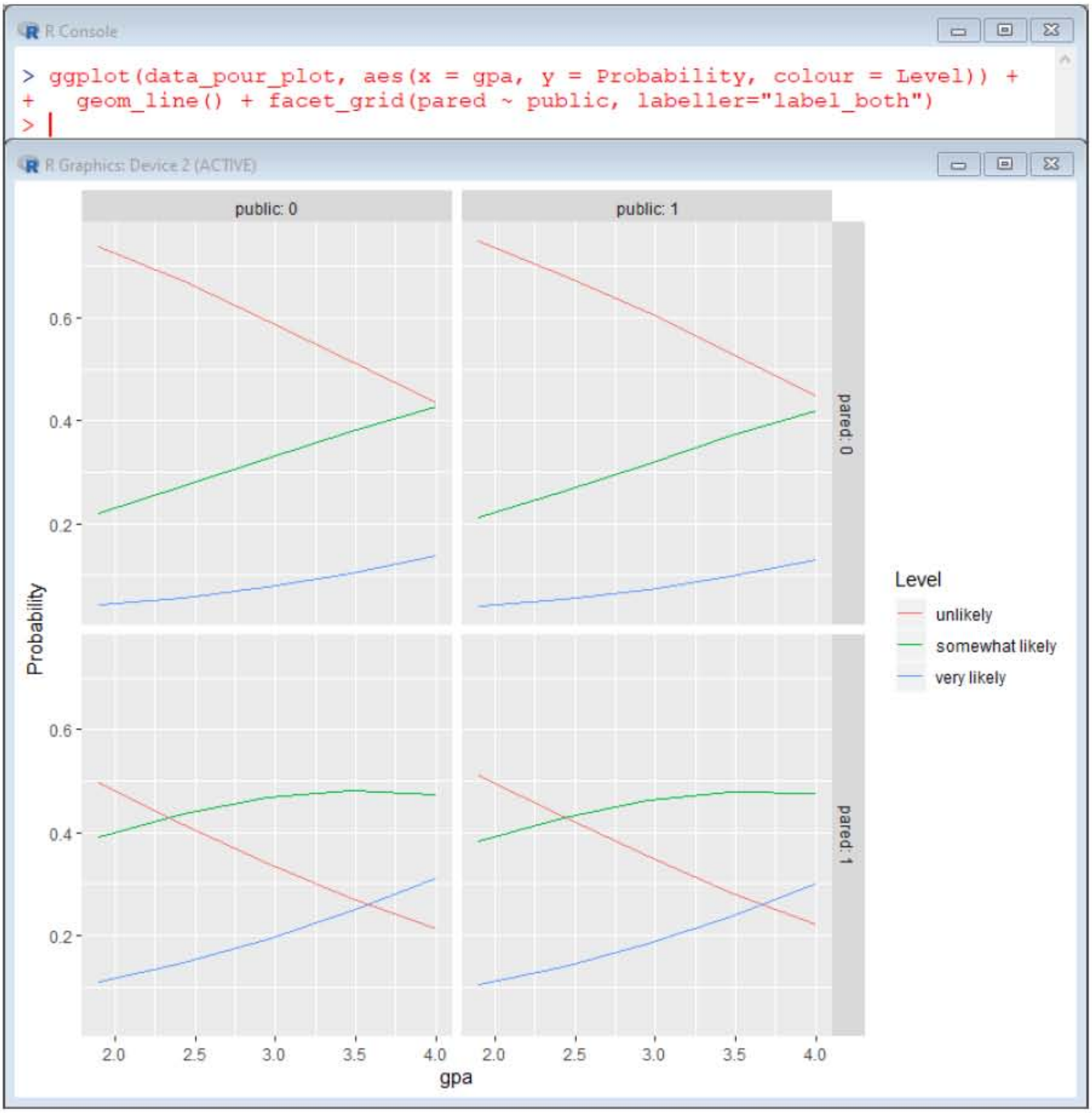
```
> (predictions<-cbind(nouvelles_donnees_test, predict(fit, nouvelles_donnees_test, type = "probs")))
   pared public   gpa  unlikely somewhat likely very likely
1      0      0 1.900 0.7376186       0.2204577  0.04192370
2      0      0 2.425 0.6704584       0.2725251  0.05701654
3      0      0 2.950 0.5955338       0.3273606  0.07710563
4      0      0 3.475 0.5158750       0.3806290  0.10349601
5      0      0 4.000 0.4354012       0.4270264  0.13757240
6      1      0 1.900 0.4964844       0.3925963  0.11091921
7      1      0 2.425 0.4164336       0.4365276  0.14703885
8      1      0 2.950 0.3405599       0.4670646  0.19237545
9      1      0 3.475 0.2720654       0.4803022  0.24763242
10     1      0 4.000 0.2128992       0.4744834  0.31261744
11     0      1 1.900 0.7488359       0.2115390  0.03962507
12     0      1 2.425 0.6833142       0.2627488  0.05393697
13     0      1 2.950 0.6096106       0.3173643  0.07302504
14     0      1 3.475 0.5305387       0.3712940  0.09816734
15     0      1 4.000 0.4499032       0.4193517  0.13074512
16     1      1 1.900 0.5111790       0.3835678  0.10525322
17     1      1 2.425 0.4307859       0.4293962  0.13981794
18     1      1 2.950 0.3538830       0.4627102  0.18340685
19     1      1 3.475 0.2838623       0.4792945  0.23684318
20     1      1 4.000 0.2229163       0.4769574  0.30012630
> 
```

Maintenant, nous pouvons remodeler les données avec le package **reshape2** et sa fonction **melt( )**:

```
> (data_pour_plot <- melt(predictions, id.vars = c("pared", "public", "gpa"),
+   variable.name = "Level", value.name="Probability"))
   pared public   gpa           Level Probability
1      0      0 1.900        unlikely  0.73761859
2      0      0 2.425        unlikely  0.67045835
3      0      0 2.950        unlikely  0.59553382
4      0      0 3.475        unlikely  0.51587497
5      0      0 4.000        unlikely  0.43540118
6      1      0 1.900        unlikely  0.49648444
7      1      0 2.425        unlikely  0.41643359
8      1      0 2.950        unlikely  0.34055991
9      1      0 3.475        unlikely  0.27206538
10     1      0 4.000        unlikely  0.21289918
11     0      1 1.900        unlikely  0.74883590
12     0      1 2.425        unlikely  0.68331418
13     0      1 2.950        unlikely  0.60961062
14     0      1 3.475        unlikely  0.53053868
15     0      1 4.000        unlikely  0.44990320
16     1      1 1.900        unlikely  0.51117895
17     1      1 2.425        unlikely  0.43078587
18     1      1 2.950        unlikely  0.35388298
19     1      1 3.475        unlikely  0.28386232
20     1      1 4.000        unlikely  0.22291626
21     0      0 1.900 somewhat likely  0.22045771
22     0      0 2.425 somewhat likely  0.27252510
23     0      0 2.950 somewhat likely  0.32736055
24     0      0 3.475 somewhat likely  0.38062901
25     0      0 4.000 somewhat likely  0.42702641
26     1      0 1.900 somewhat likely  0.39259634
27     1      0 2.425 somewhat likely  0.43652756
28     1      0 2.950 somewhat likely  0.46706464
29     1      0 3.475 somewhat likely  0.48030221
30     1      0 4.000 somewhat likely  0.47448338
31     0      1 1.900 somewhat likely  0.21153903
32     0      1 2.425 somewhat likely  0.26274885
33     0      1 2.950 somewhat likely  0.31736434
34     0      1 3.475 somewhat likely  0.37129397
35     0      1 4.000 somewhat likely  0.41935168
36     1      1 1.900 somewhat likely  0.38356783
37     1      1 2.425 somewhat likely  0.42939618
38     1      1 2.950 somewhat likely  0.46271017
39     1      1 3.475 somewhat likely  0.47929450
40     1      1 4.000 somewhat likely  0.47695744
41     0      0 1.900     very likely  0.04192370
42     0      0 2.425     very likely  0.05701654
43     0      0 2.950     very likely  0.07710563
44     0      0 3.475     very likely  0.10349601
45     0      0 4.000     very likely  0.13757240
46     1      0 1.900     very likely  0.11091921
47     1      0 2.425     very likely  0.14703885
48     1      0 2.950     very likely  0.19237545
```

et tracer toutes les probabilités prédites pour les différentes conditions. Nous traçons les probabilités prédites, reliées à une ligne, colorées par niveau du résultat, *apply* et facettées par niveau de *pared* et *public*. Nous utilisons également une fonction d'étiquette personnalisée

pour ajouter des étiquettes plus claires montrant ce que chaque colonne et ligne du tracé représente:

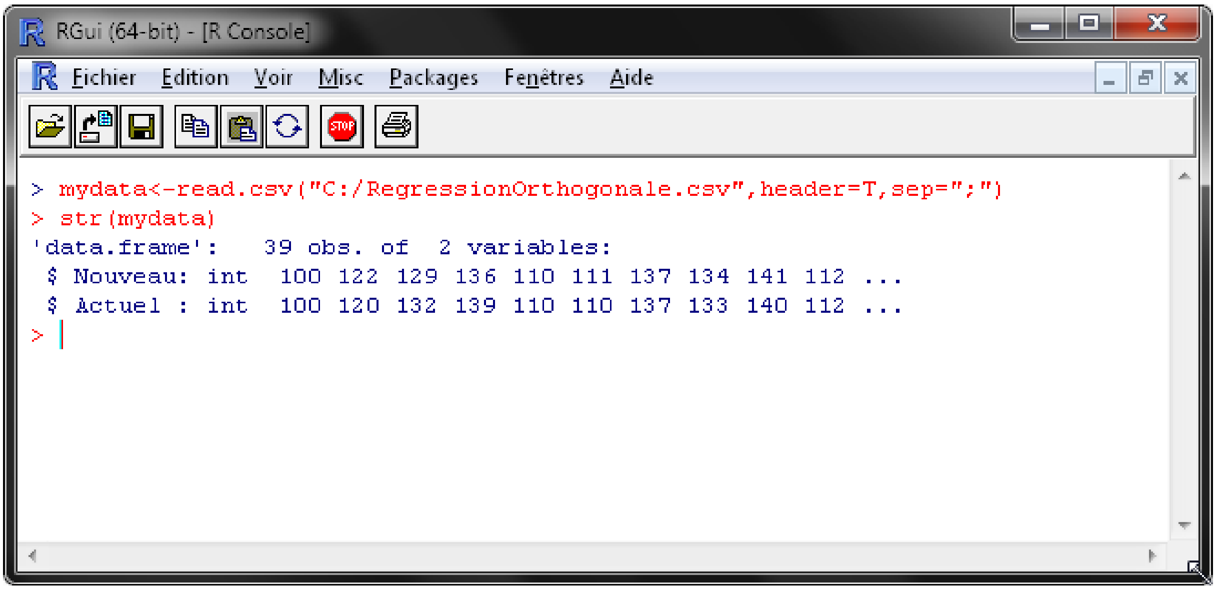


Exemple 273.: Régression orthogonale (régression de Deming)

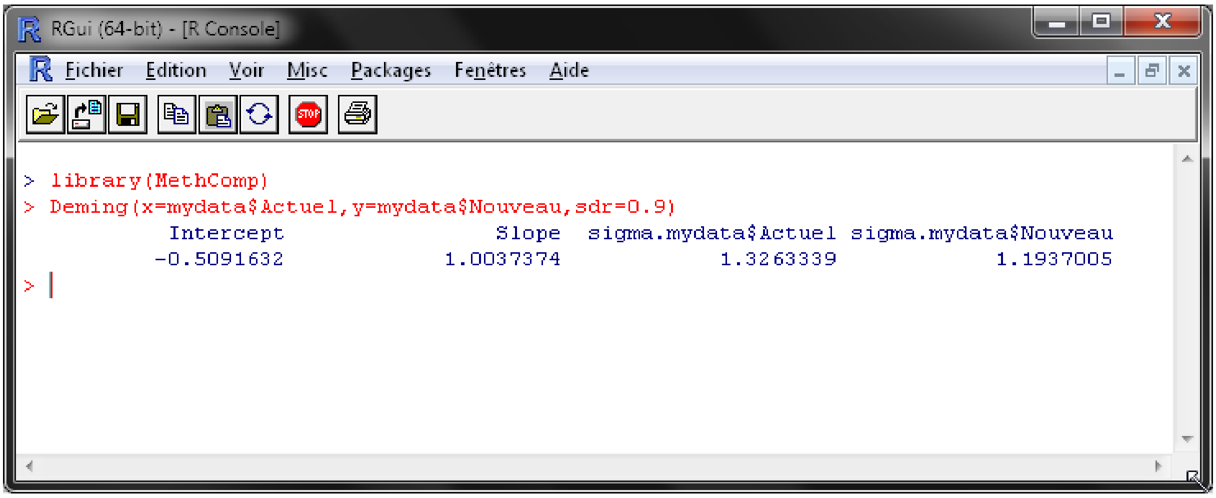
R 3.0.2

Nous allons encore une fois vérifier ici si les calculs effectués à la main et dans Microsoft Excel suite aux démonstrations mathématiques d'une partie des éléments de la régression de Deming effectuées dans le cours théorique.

Pour cela, nous utiliserons les 39 lignes suivantes (prises en partie de l'aide Minitab) qui correspondent à la mesure de la pression sanguine avec deux instruments de 39 individus (chaque individu ayant mesuré sa pression sanguine une fois avec l'ancien et une fois avec le nouvel instrument):



En utilisant alors la commande **Deming( )** du package **MethComp**:



Nous voyons alors que bien que la pente soit très proche de ce que nous avons obtenu avec Minitab, au niveau de l'*Intercept* il y a près de 20% de différence!

De plus, nous n'avons pas d'intervalle de confiance ce qui enlève tout utilité à l'analyse telle quelle est présentée ci-dessus. Heureusement, les auteurs proposent une régression orthogonale par bootstrapping. Nous avons alors:

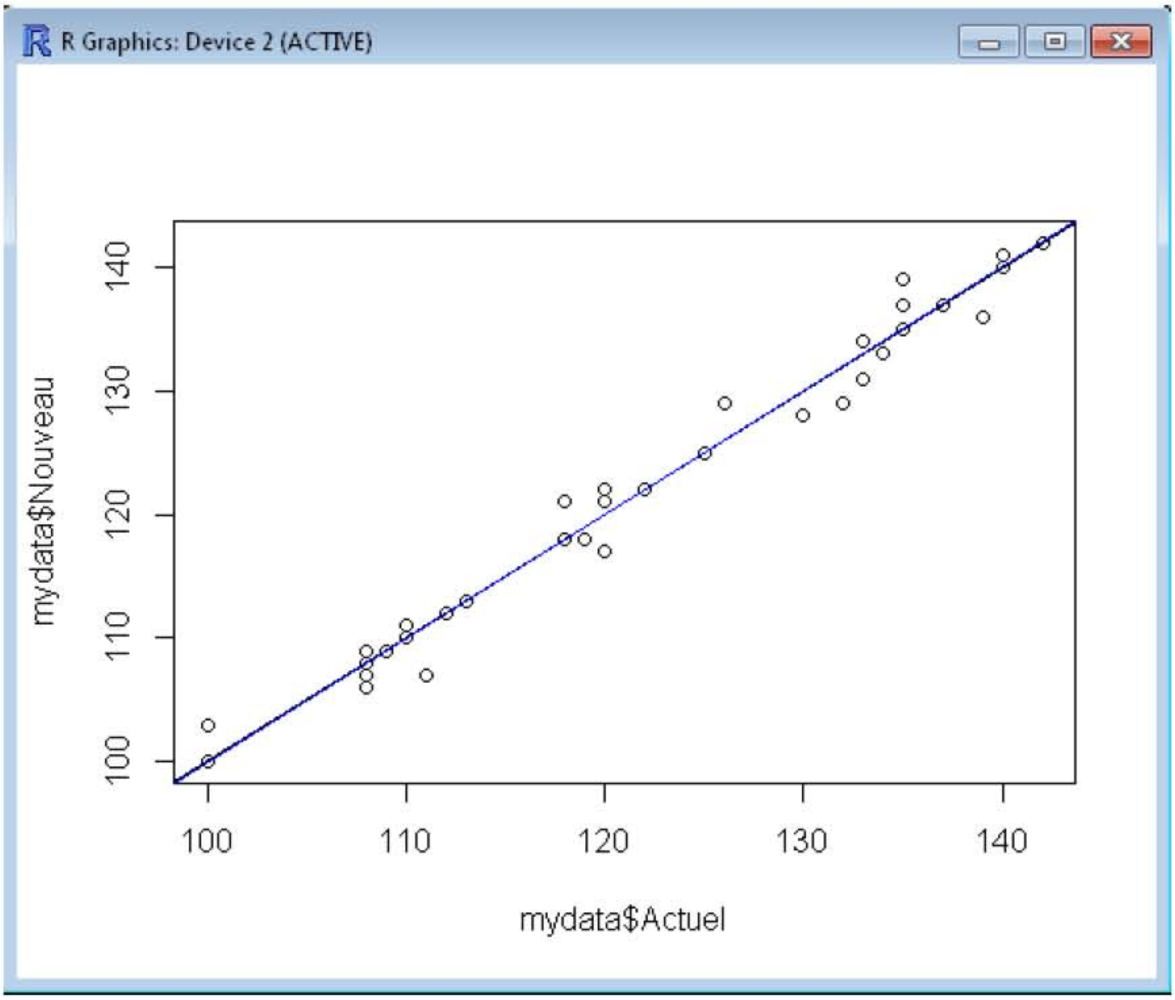
```
> Deming(x=mydata$Actuel,y=mydata$Nouveau,boot=TRUE)
mydata$Nouveau  = alpha + beta* mydata$Actuel
                         Estimate S.e.(boot)        50%       2.5%    97.5%
Intercept              -0.3867358 2.59070595 -0.5122269 -5.4234882 4.559345
Slope                   1.0027381 0.02131253  1.0039809  0.9620726 1.043659
sigma.mydata$Actuel     1.2619752 0.14422966  1.2263183  0.9209491 1.510403
sigma.mydata$Nouveau    1.2619752 0.14422966  1.2263183  0.9209491 1.510403
>
```

Et là nous sommes beaucoup plus proches des résultats donnés par Minitab ce qui est heureux et de plus, nous avons les intervalles de confiance avec donc les mêmes conclusions.

Nous pouvons aussi ploter les trois modèles de régressions (pour voir les différences quand celles-ci sont éventuellement significatives):

```
> library(MethComp)
> orthClassic<-Deming(x=mydata$Actuel,y=mydata$Nouveau,sdr=0.9)
> orthBoot<-Deming(x=mydata$Actuel,y=mydata$Nouveau,boot=TRUE)
mydata$Nouveau  = alpha + beta* mydata$Actuel
> #on met les points
> plot(x=mydata$Actuel,y=mydata$Nouveau)
> #on montre une régression classique (moindres carrés)
> abline(lm(mydata$Nouveau~mydata$Actuel))
> #on montre la régressin de Deming classique
> abline(orthClassic[1:2],col="red")
> #on montre la régression de Deming par Boostrap
> abline(orthBoot[1:2],col="blue")
>
```

Ce qui nous donne:


R Graphics: Device 2 (ACTIVE)
mydata$Nouveau
mydata$Actuel
100
110
120
130
140
100
110
120
130
140

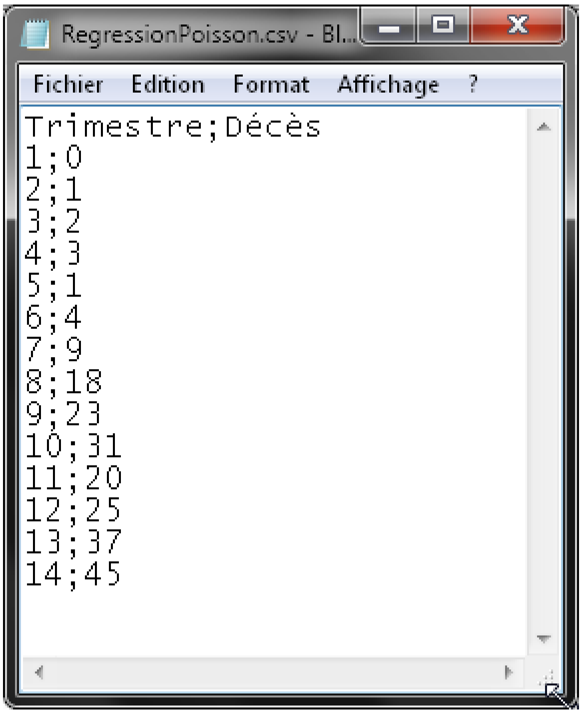
Exemple 274.: Régressions linéaires généralisées (GLM)

R 3.0.2

Nous allons ici vérifier si nous retrouvons bien les différents résultats calculés à la main et obtenus dans le cours théorique après avoir fait la démonstration détaillée de certains modèles GLM particuliers (nous ne reviendrons donc pas sur le cas gaussien déjà traité plus haut).

Régression de Poisson log-linéaire (régression des événements rares)

Nous allons reprendre ici l'exemple fait dans le cours théorique utilisant le même jeu de donnée que dans le cours Microsoft Excel et Minitab et évidemment avec les mêmes critiques et hypothèses:

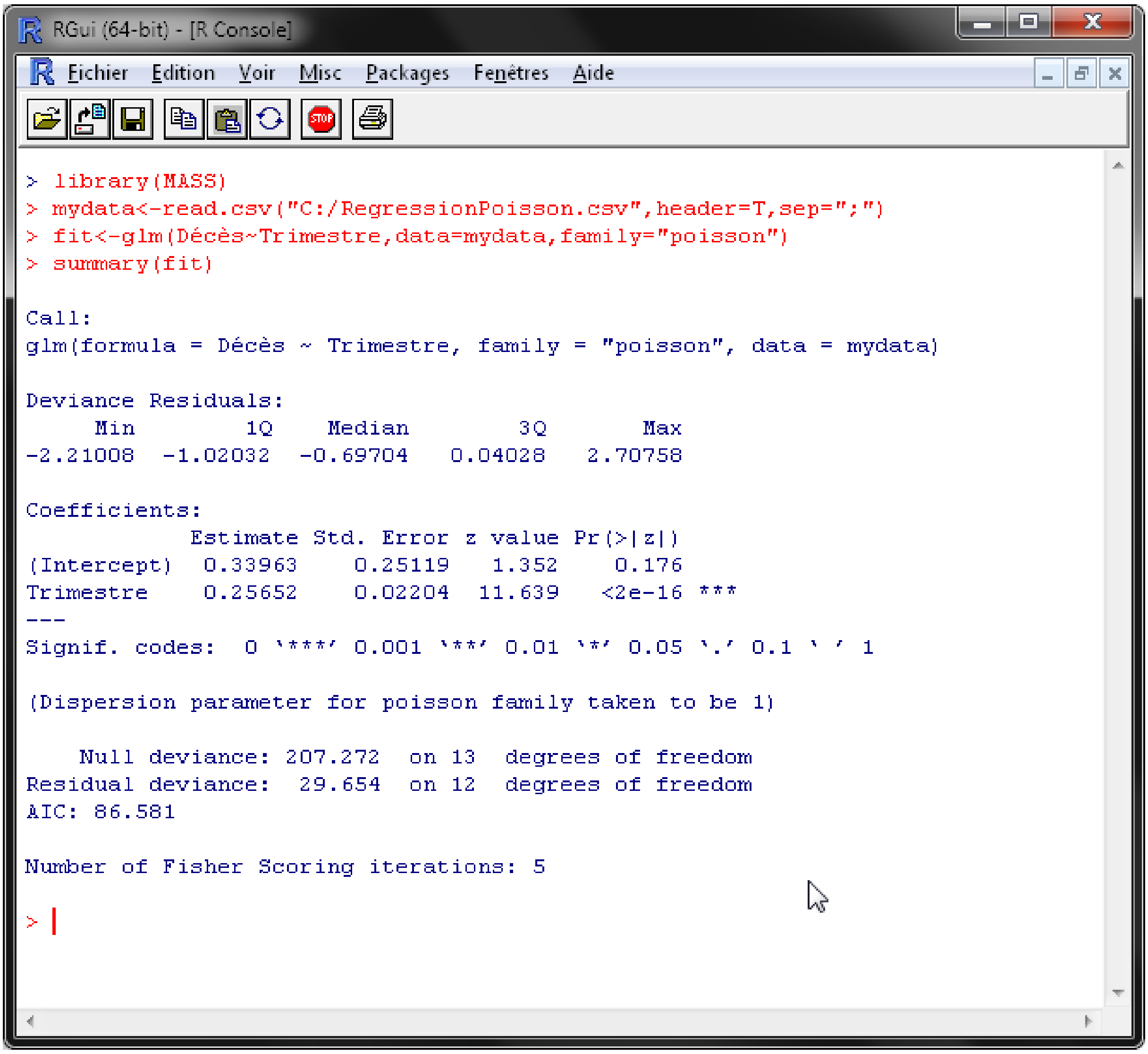


RegressionPoisson.csv - Bl...

Fichier Edition Format Affichage ?

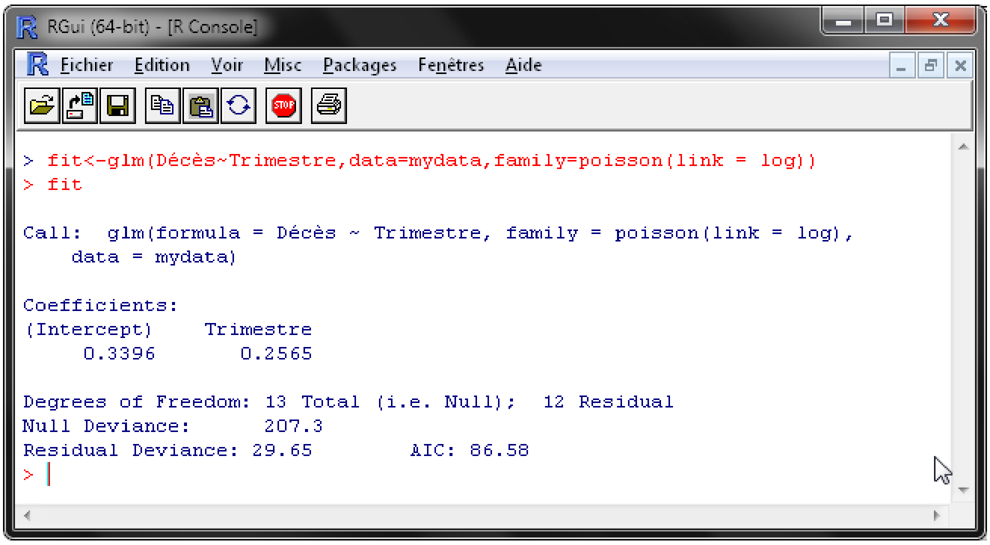
```
Trimestre;Décès
1;0
2;1
3;2
4;3
5;1
6;4
7;9
8;18
9;23
10;31
11;20
12;25
13;37
14;45
```

Ensuite, nous utilisons la commande native **glm( )** de R:

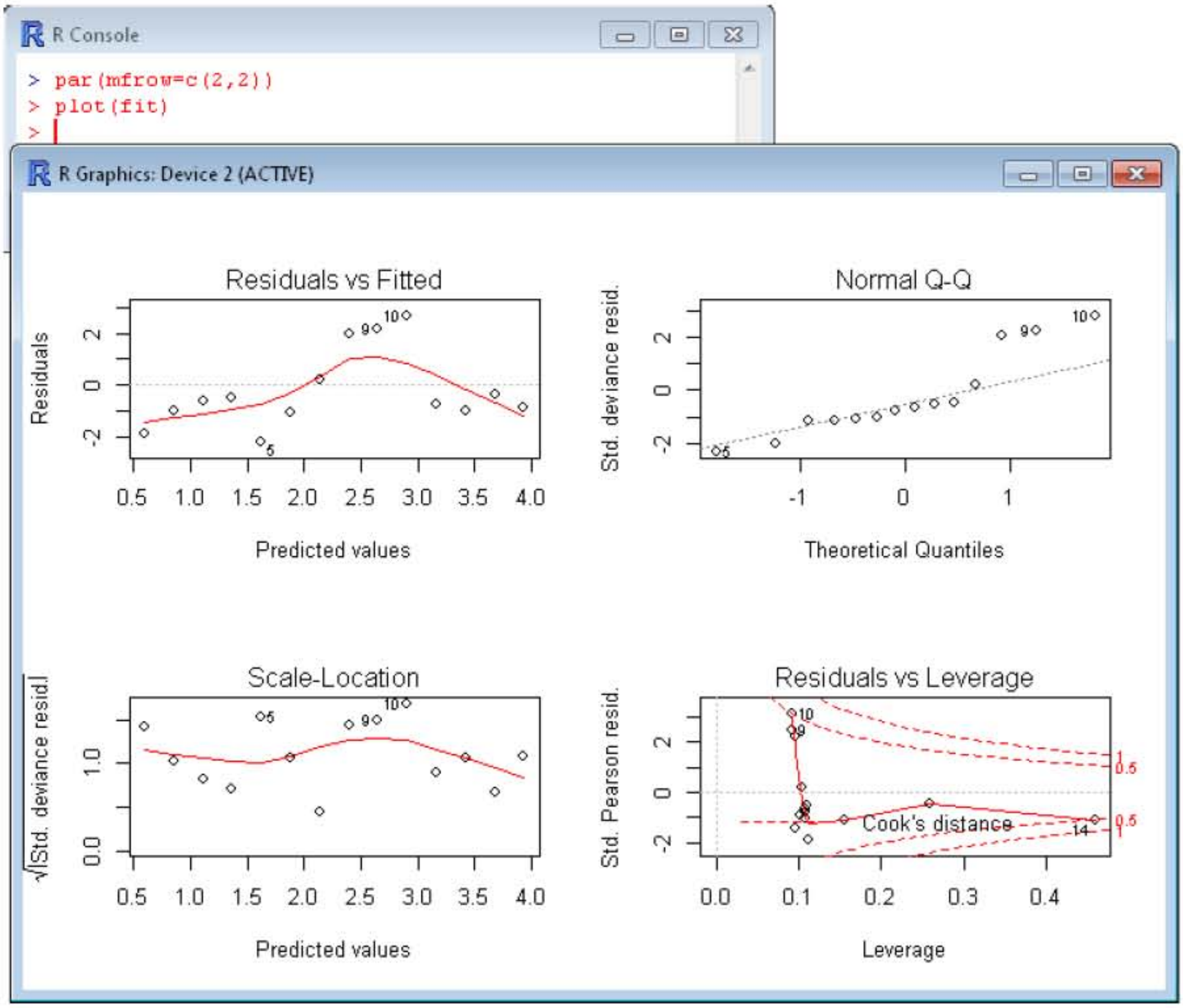


Donc en ce qui concerne les coefficients nous obtenons les mêmes valeurs que dans le cours théorique. Pour le reste... nous ferons confiance à R...

Nous pouvons si besoin être plus précis en indiquant le type de lien.

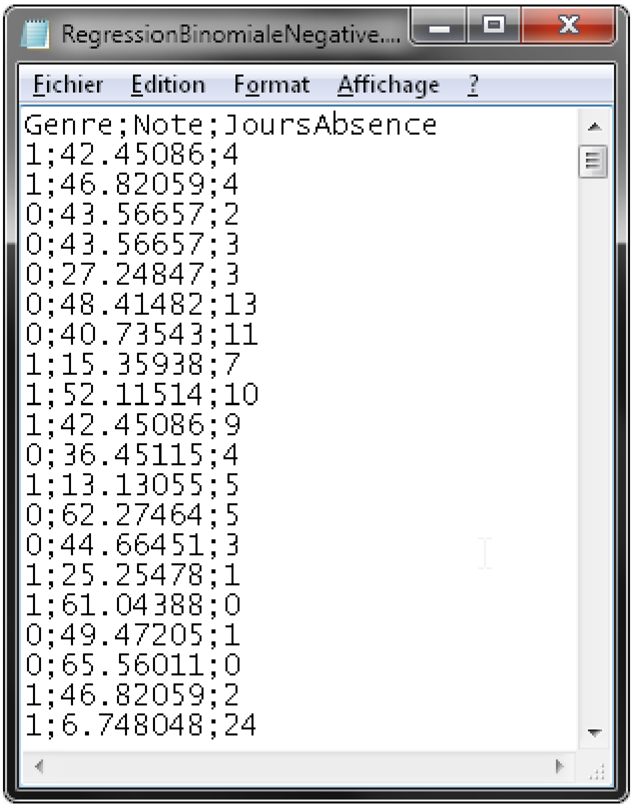


Et nous pouvons plotter un petit résumé important concernant la validité du modèle:



Régression GLM Négative binomiale (NB-2)

Nous prenons donc le même jeu de données que dans le cours théorique et Microsoft Excel dont voici un échantillon:




```
Genre;Note;JoursAbsence
1;42.45086;4
1;46.82059;4
0;43.56657;2
0;43.56657;3
0;27.24847;3
0;48.41482;13
0;40.73543;11
1;15.35938;7
1;52.11514;10
1;42.45086;9
0;36.45115;4
1;13.13055;5
0;62.27464;5
0;44.66451;3
1;25.25478;1
1;61.04388;0
0;49.47205;1
0;65.56011;0
1;46.82059;2
1;6.748048;24
```

et ensuite nous utilisons la command **glm.nb( )** du package **MASS** pour obtenir:

```
> library(MASS)
> mydata<-read.csv("C:/RegressionBinomialeNegative.csv",header=T,sep=";")
> fit<-glm.nb(JoursAbsence~Note+Genre,data=mydata)
> summary(fit)

Call:
glm.nb(formula = JoursAbsence ~ Note + Genre, data = mydata,
    init.theta = 0.775738356, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0040  -1.0659  -0.4137   0.2747   2.7043

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.703440   0.224672  12.033  < 2e-16 ***
Note        -0.015649   0.003883  -4.030 5.57e-05 ***
Genre       -0.431207   0.139017  -3.102  0.00192 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.7757) family taken to be 1)

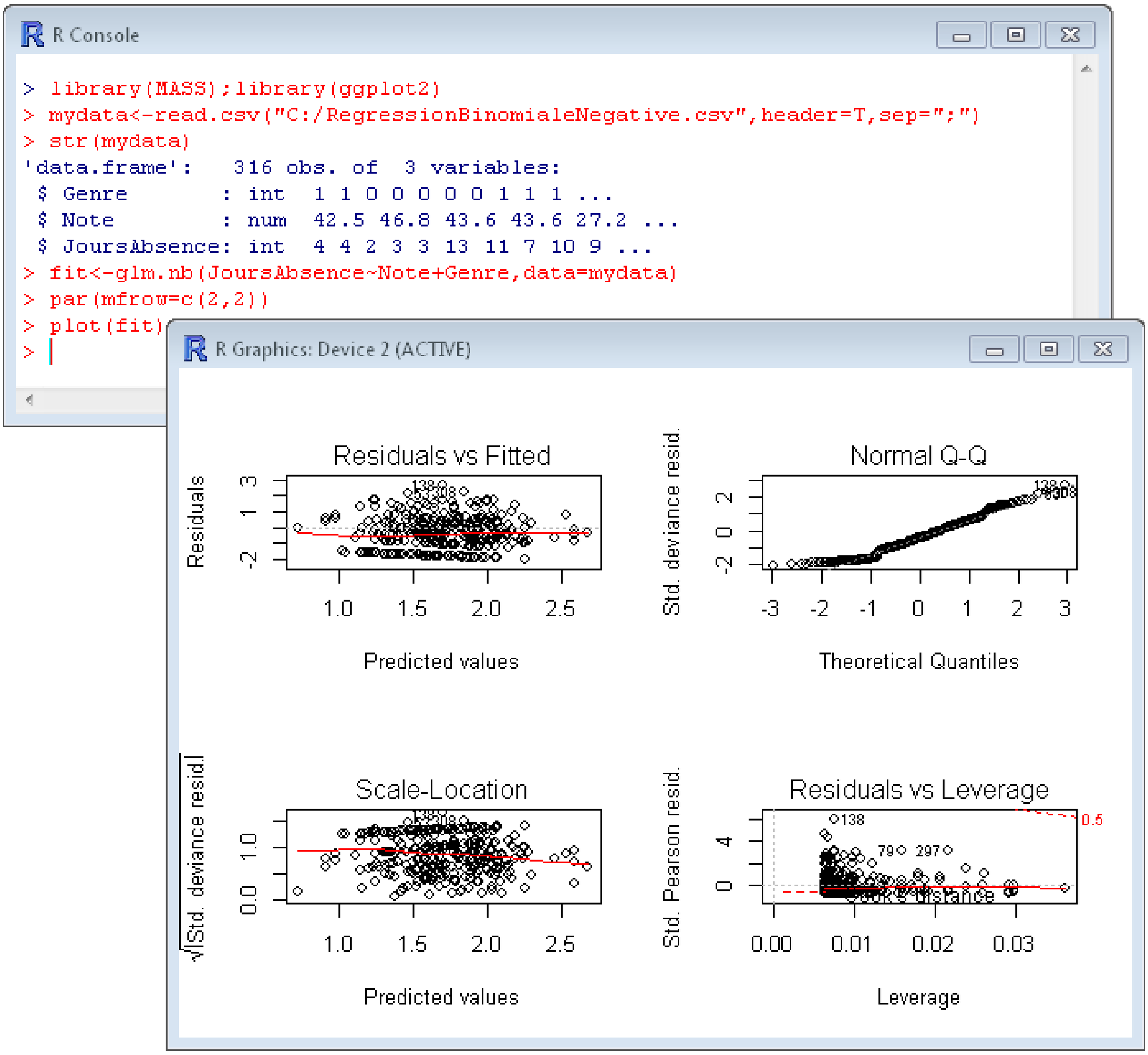
    Null deviance: 378.29  on 315  degrees of freedom
Residual deviance: 356.90  on 313  degrees of freedom
AIC: 1769.9

Number of Fisher Scoring iterations: 1

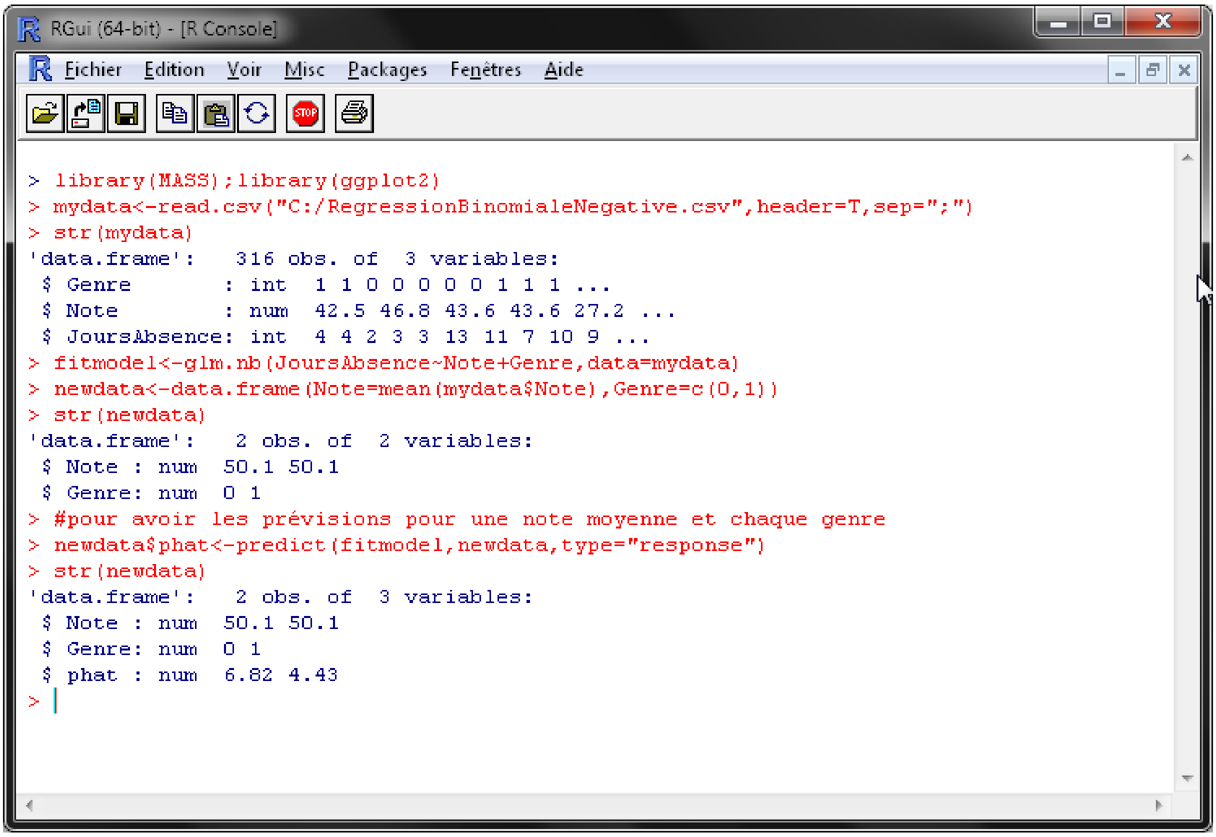

              Theta:  0.7757
          Std. Err.:  0.0741

 2 x log-likelihood:  -1761.8550
>
```

Nous nous retrouvons donc avec les mêmes résultats des coefficients et de l'ordonnée à l'origine que dans le cours théorique et qu'avec SAS. Après pour les autres informations... c'est une autre histoire!



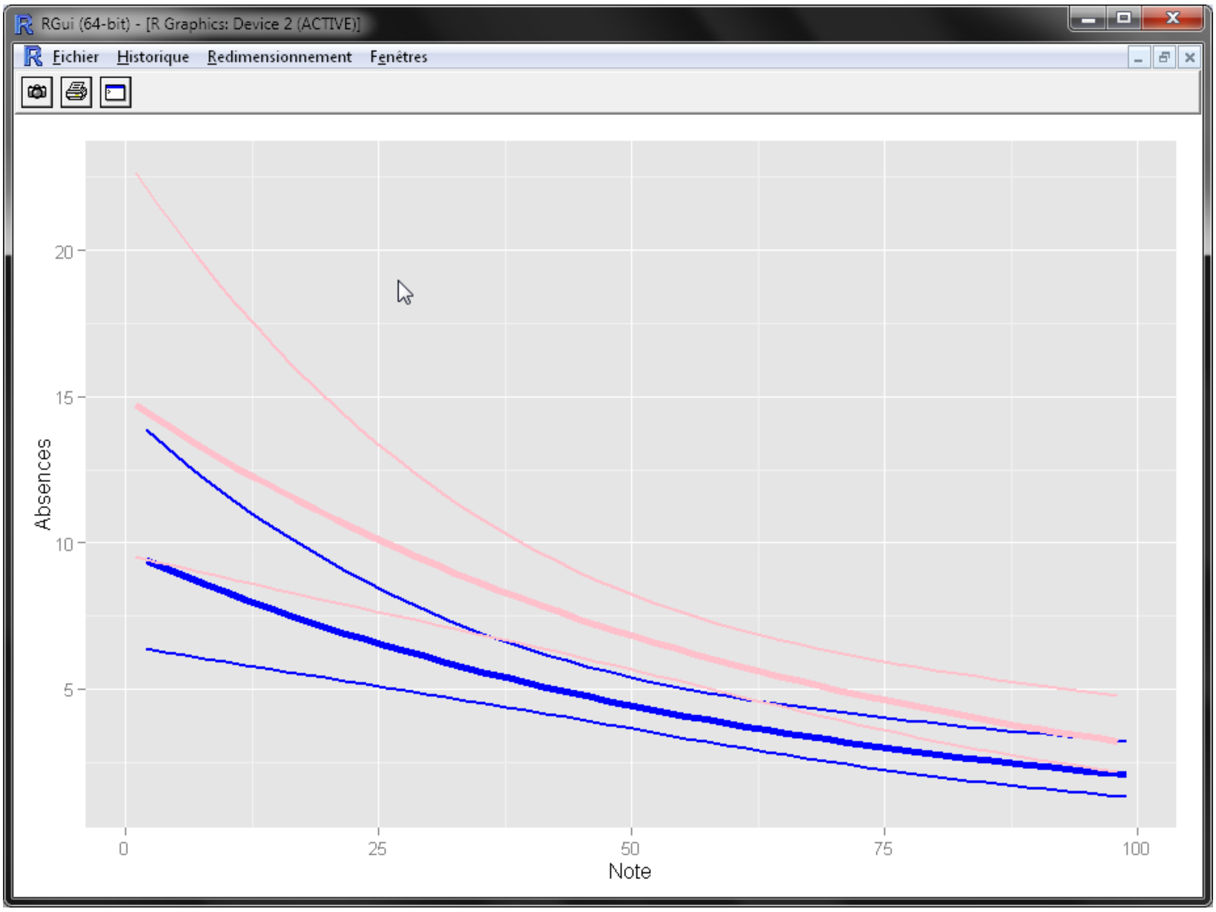
On peut faire un plot du modèle. Voyons cela (si vous avez des propositions d'amélioration, n'hésitez pas à me contacter!). D'abord montrons que nous pouvons obtenir les valeurs des moyennes d'absence du modèle théorique pour chaque genre pour les notes moyennes:



Ensuite, nous pouvons faire cela pour toute l'étendue par genre et par note avec un graphe:

```
> newdata<-data.frame(Note=rep(seq(from=min(mydata$Note),to=max(mydata$Note),length.out=100),2),Genre=c(0,1))
> newdata<-cbind(newdata,predict(fitmodel,newdata,type="link",se.fit=TRUE))
> str(newdata)
'data.frame':   200 obs. of  5 variables:
 $ Note          : num  1.01 2 2.99 3.98 4.97 ...
 $ Genre         : num  0 1 0 1 0 1 0 1 0 1 ...
 $ fit           : num  2.69 2.24 2.66 2.21 2.63 ...
 $ se.fit        : num  0.221 0.198 0.214 0.192 0.207 ...
 $ residual.scale: num  1 1 1 1 1 1 1 1 1 1 ...
> newdata<- within(newdata, {
+   Absences <- exp(fit)
+   LL <- exp(fit- 1.96 * se.fit)
+   UL <- exp(fit+ 1.96 * se.fit)
+ })
> str(newdata)
'data.frame':   200 obs. of  8 variables:
 $ Note          : num  1.01 2 2.99 3.98 4.97 ...
 $ Genre         : num  0 1 0 1 0 1 0 1 0 1 ...
 $ fit           : num  2.69 2.24 2.66 2.21 2.63 ...
 $ se.fit        : num  0.221 0.198 0.214 0.192 0.207 ...
 $ residual.scale: num  1 1 1 1 1 1 1 1 1 1 ...
 $ UL            : num  22.7 13.9 21.7 13.3 20.7 ...
 $ LL            : num  9.53 6.37 9.36 6.26 9.2 ...
 $ Absences      : num  14.7 9.4 14.25 9.12 13.81 ...
> newdataG<-newdata[which(newdata$Genre==1),]
> newdataF<-newdata[which(newdata$Genre==0),]
> ggplot()+
+ geom_line(data=newdataG,aes(x=Note,y=Absences),colour="blue",size=2)+
+ geom_line(data=newdataG,aes(x=Note,y=UL),colour="blue",size=1)+
+ geom_line(data=newdataG,aes(x=Note,y=LL),colour="blue",size=1)+
+ geom_line(data=newdataF,aes(x=Note,y=Absences),colour="pink",size=2)+
+ geom_line(data=newdataF,aes(x=Note,y=UL),colour="pink",size=1)+
+ geom_line(data=newdataF,aes(x=Note,y=LL),colour="pink",size=1)
>
```

Ce qui donnera:



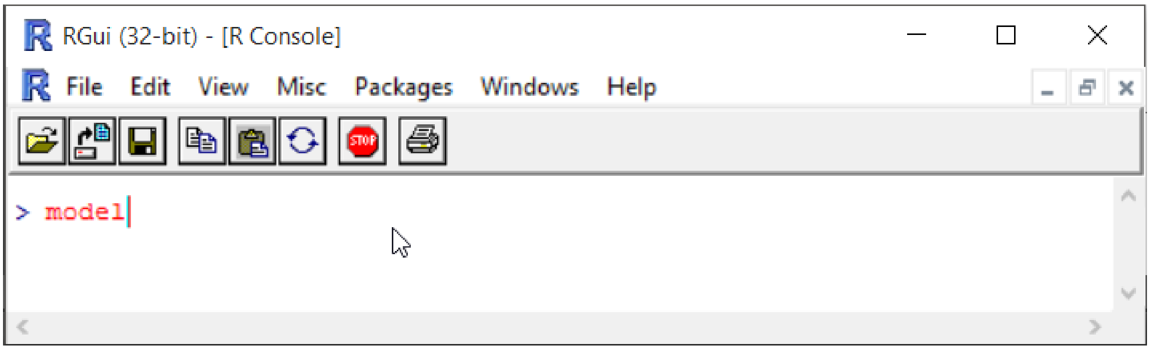
Donc il est rassurant de voir que plus les absences sont faibles, meilleure est la note...

Recherche automatique de la meilleure régression de Poisson

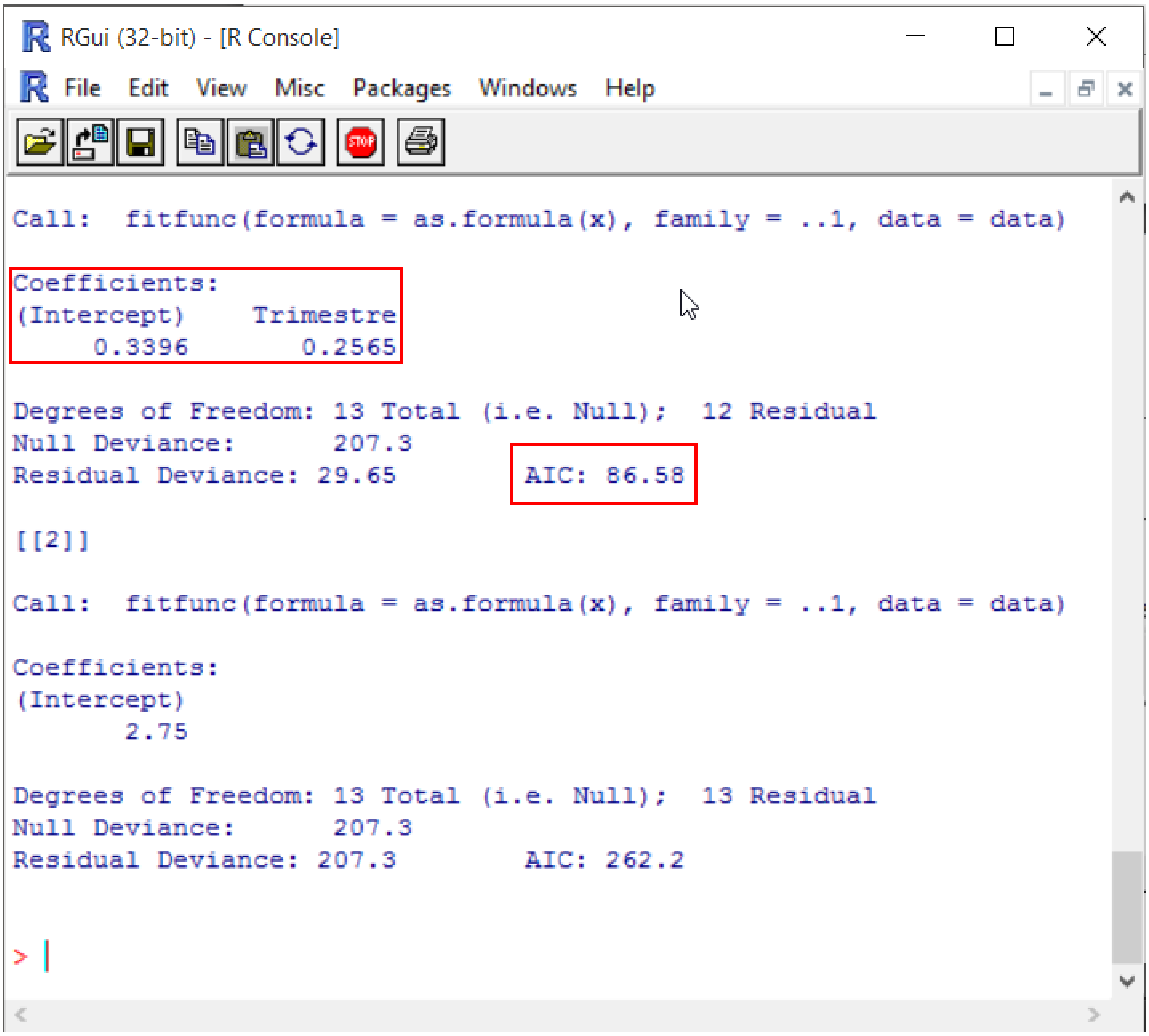
Avec le package **glmulti** et la fonction du même nom nous pouvons dans la version **x86** de R demander à chercher le meilleur modèle de Poisson (ce package ne fait pas encore le modèle binomial négatif mais il fait par contre le modèle logit!). Pour cela, nous écrivons:



En écrivant simplement:



Nous obtenons une longue liste de résultats, dont:

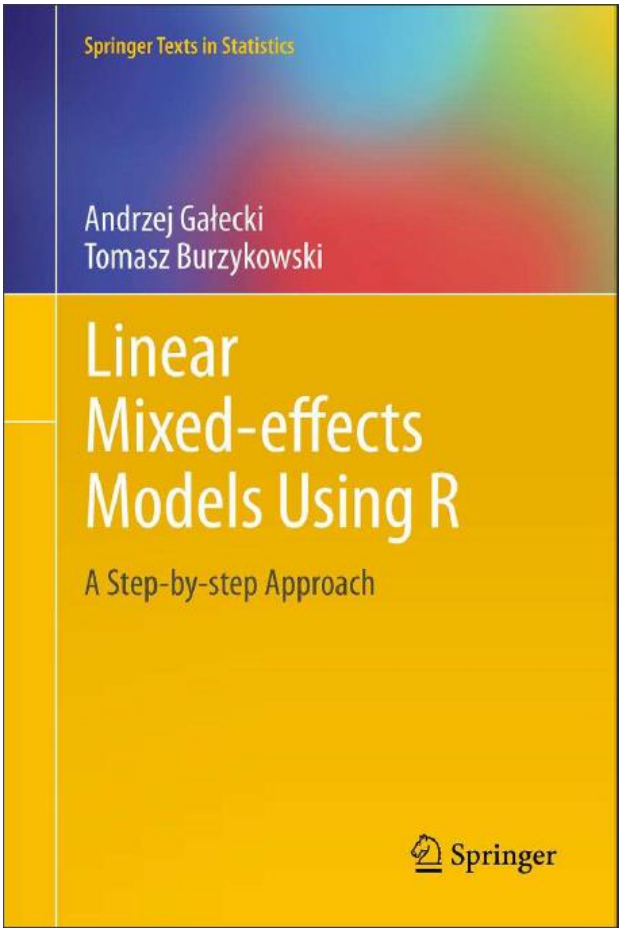


Exemple 275.: Régressions linéaires généralisées mixtes/longitudinales (GLMM)

R 3.6.1

Nous avons vu que le sujet des GLMM est extrêmement vase, tout autant que les ANOVA… 😉

Par ailleurs il y a un ouvrage de presque 600 pages dédié au sujet avec R:

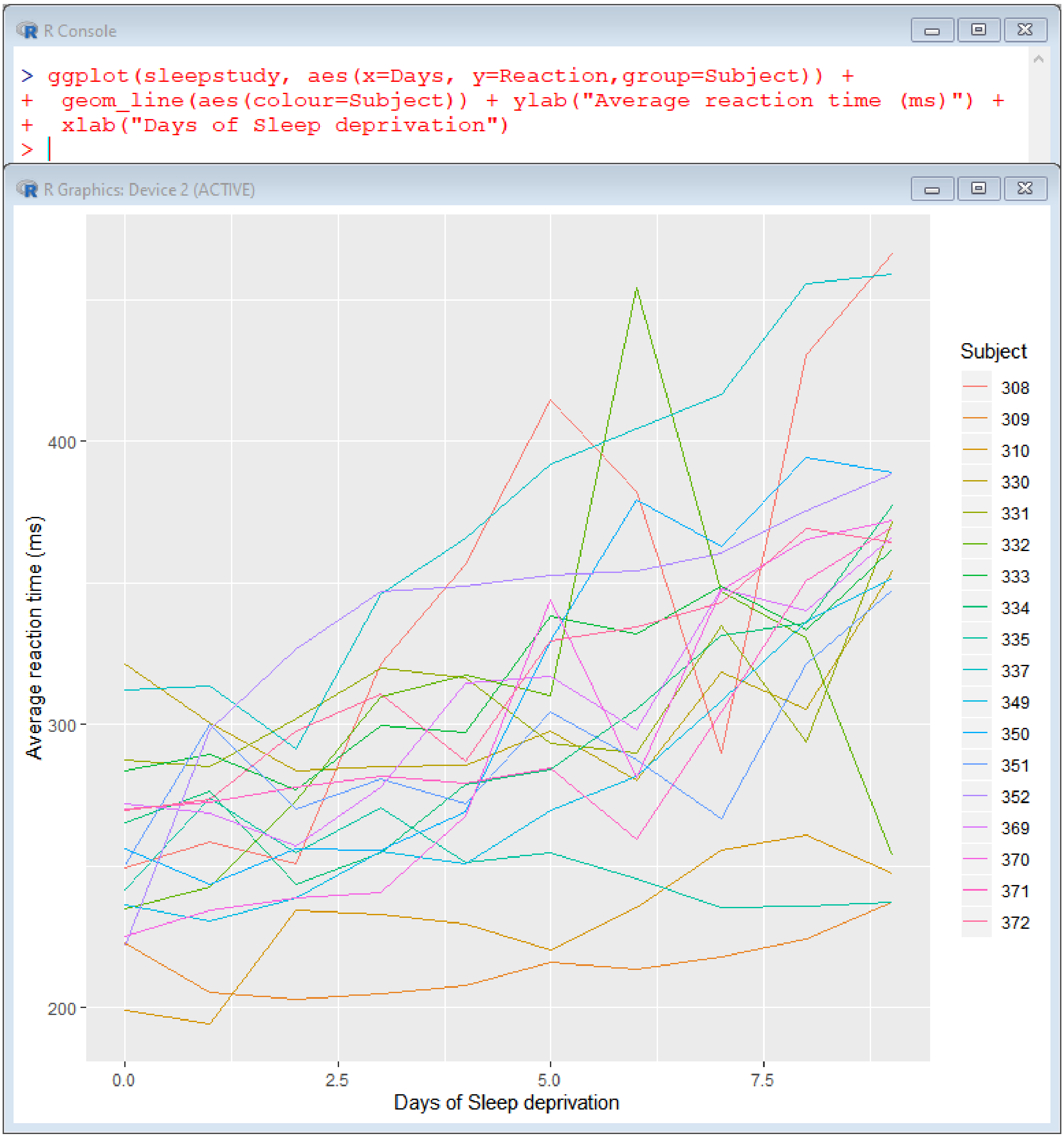


Il va donc sans dire qu'ici nous ne ferons de survoler le sujet avec un cas extrêmement simple!

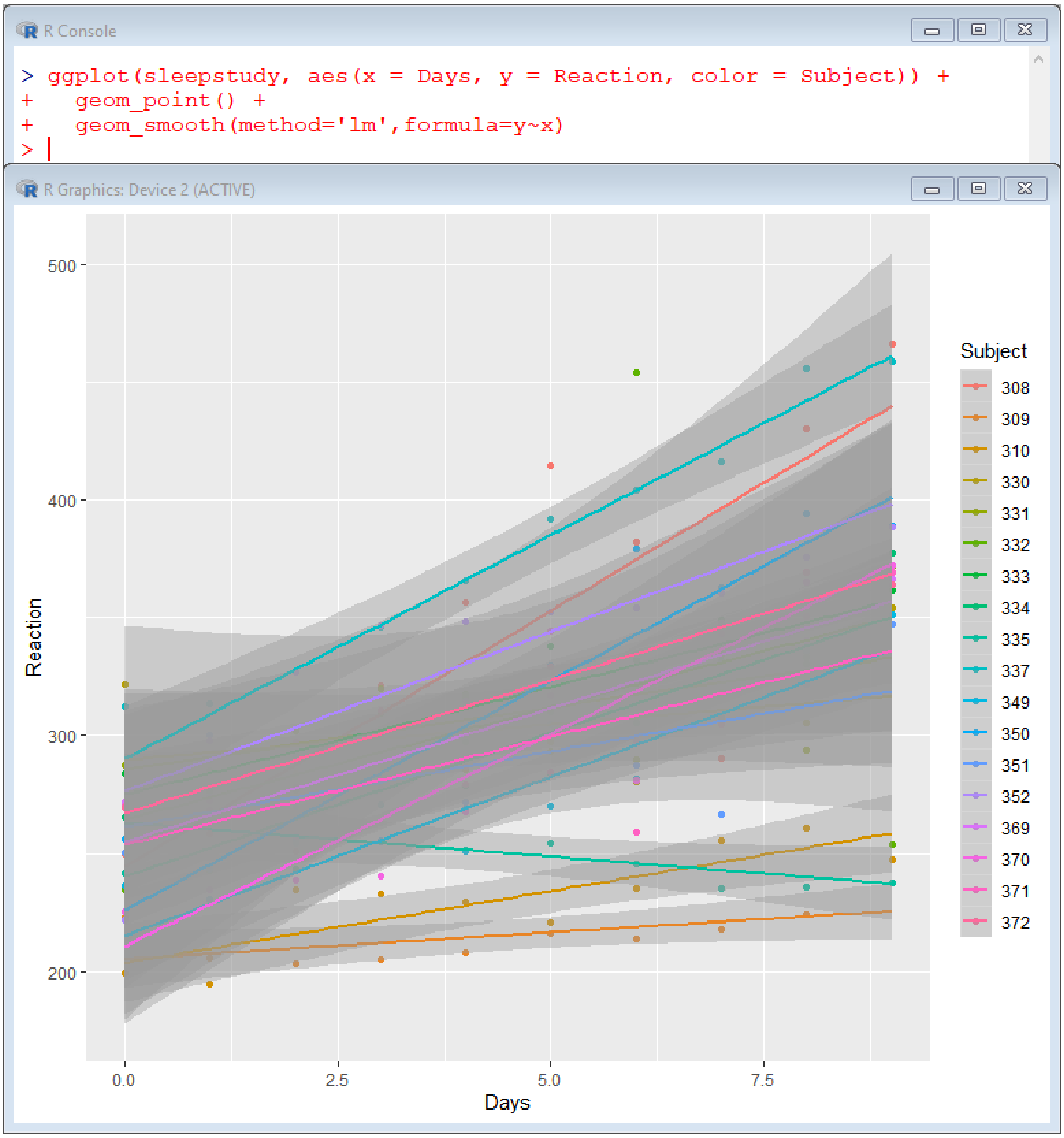
Pour cela on va utiliser le jeu de données *sleestudy* suivant (il s'agit de 18 personnes, qu'on a laissé dormir 8 heures par jour pendant une longue période et ensuite dont on a retiré 3 heures de sommeil – donc plus que 5 heures… - et dont on a mesuré la vitesse des réflexes en millisecondes sur 9 jours):

```
> library("lattice")
> data("sleepstudy")
> str(sleepstudy)
'data.frame':   180 obs. of  3 variables:
 $ Reaction: num  250 259 251 321 357 ...
 $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
 $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1 1 1 1 1 1 1 1 1 ...
> sleepstudy
    Reaction Days Subject
1        250    0     308
2        259    1     308
3        251    2     308
4        321    3     308
5        357    4     308
6        415    5     308
7        382    6     308
8        290    7     308
9        431    8     308
10       466    9     308
11       223    0     309
12       205    1     309
13       203    2     309
14       205    3     309
15       208    4     309
16       216    5     309
17       214    6     309
18       218    7     309
19       224    8     309
20       237    9     309
21       199    0     310
22       194    1     310
23       234    2     310
24       233    3     310
25       229    4     310
26       220    5     310
27       235    6     310
28       256    7     310
29       261    8     310
30       248    9     310
31       322    0     330
32       300    1     330
33       284    2     330
34       285    3     330
35       286    4     330
36       298    5     330
37       280    6     330
38       318    7     330
39       305    8     330
40       354    9     330
```

Faisons quelques graphiques pour afficher d'abord l'état des choses:

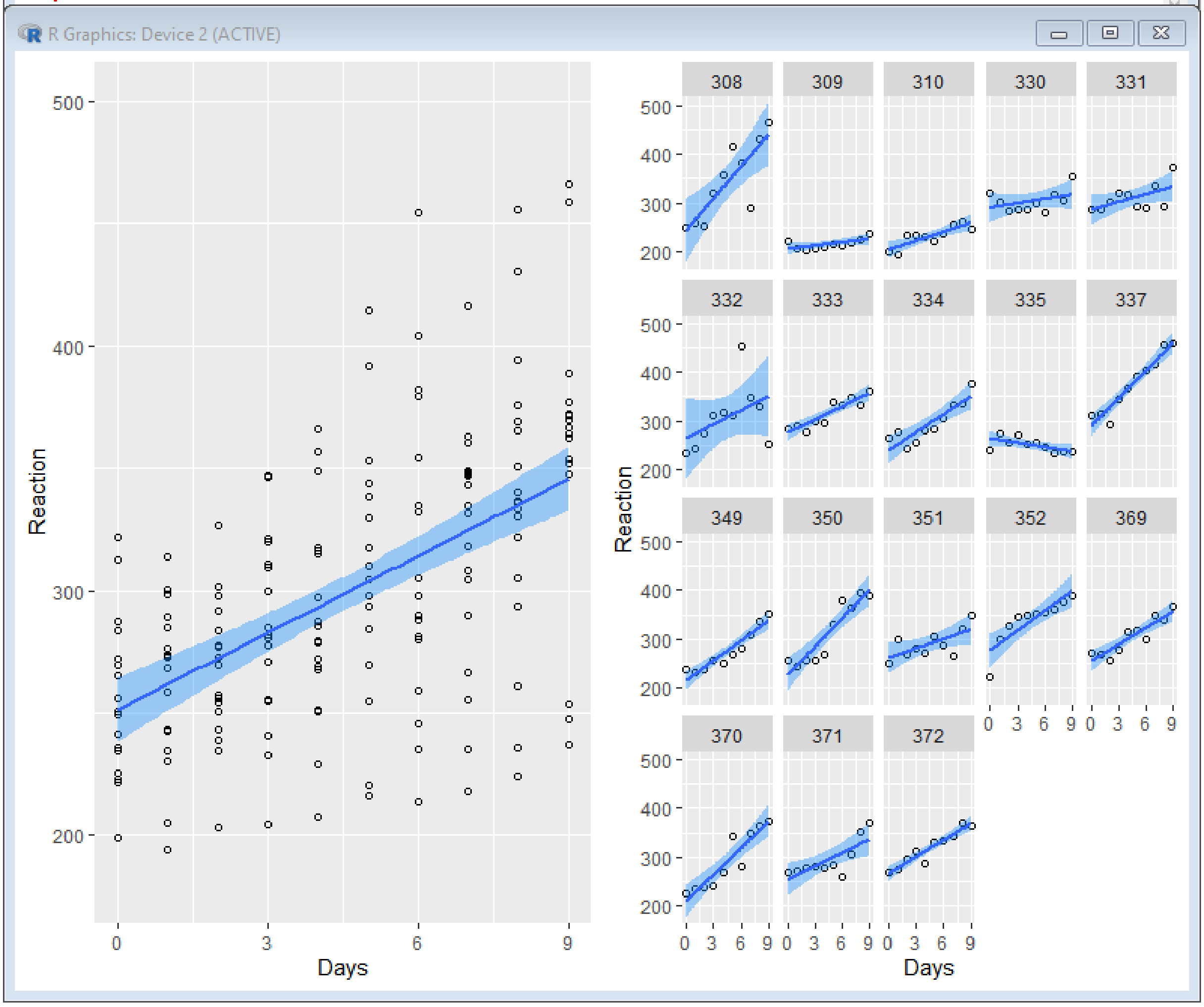


Faisons des régressions simples multiples pour chacun des 18 individus:

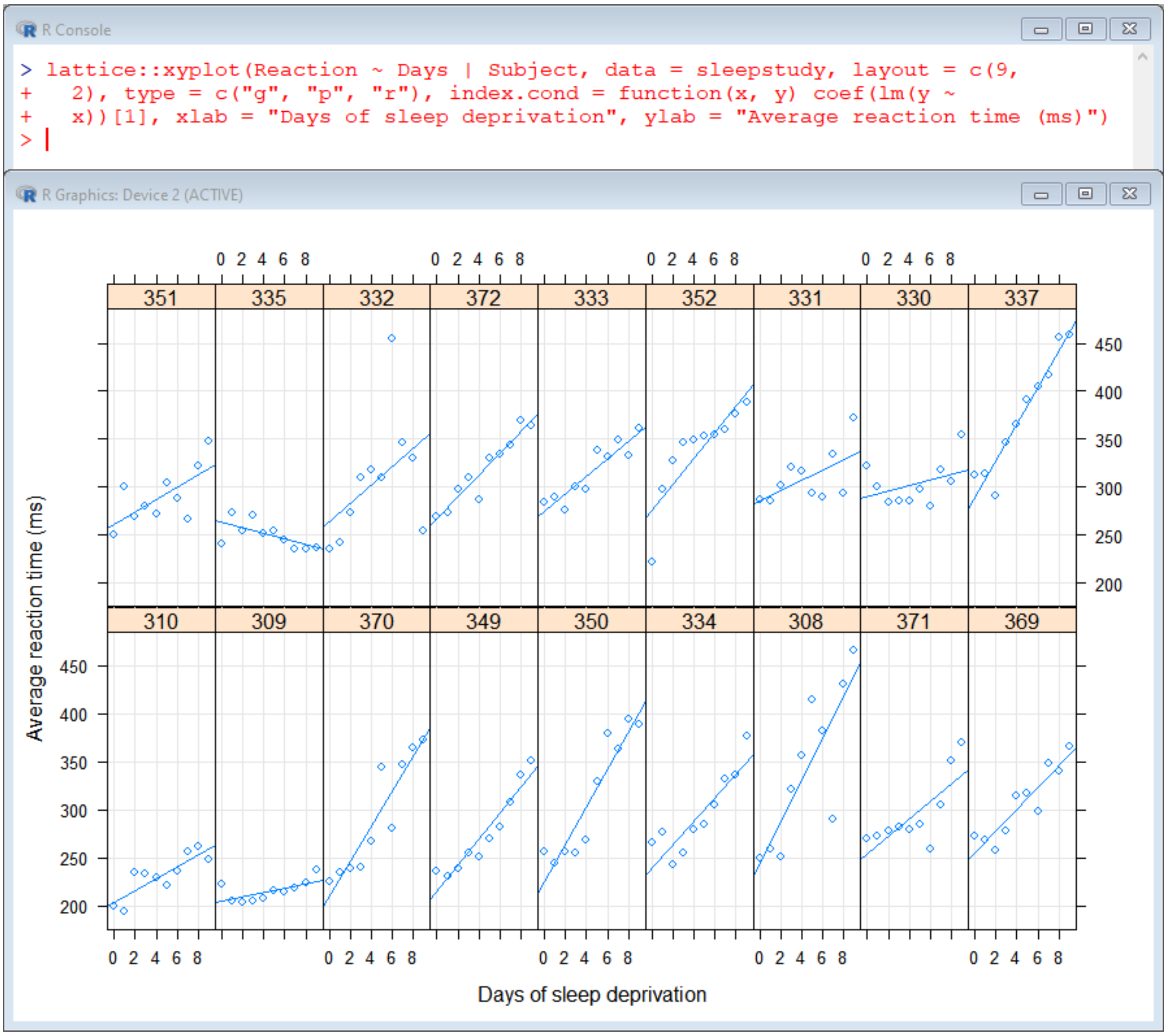


Ou dans un autre style beaucoup plus intéressant:

```
> library("gridExtra")
> p1 <- ggplot(sleepstudy, aes(x = Days, y = Reaction)) +
+     geom_point(shape = 1) +
+     geom_smooth(method = "lm", fill = "dodgerblue", level = .95) +
+     scale_x_continuous(breaks = c(0, 3, 6, 9)) +
+     coord_cartesian(ylim = c(180, 500))
> p2 <- p1 + facet_wrap(~Subject, nrow = 4)
> gridExtra::grid.arrange(p1, p2, ncol = 2)
>
```

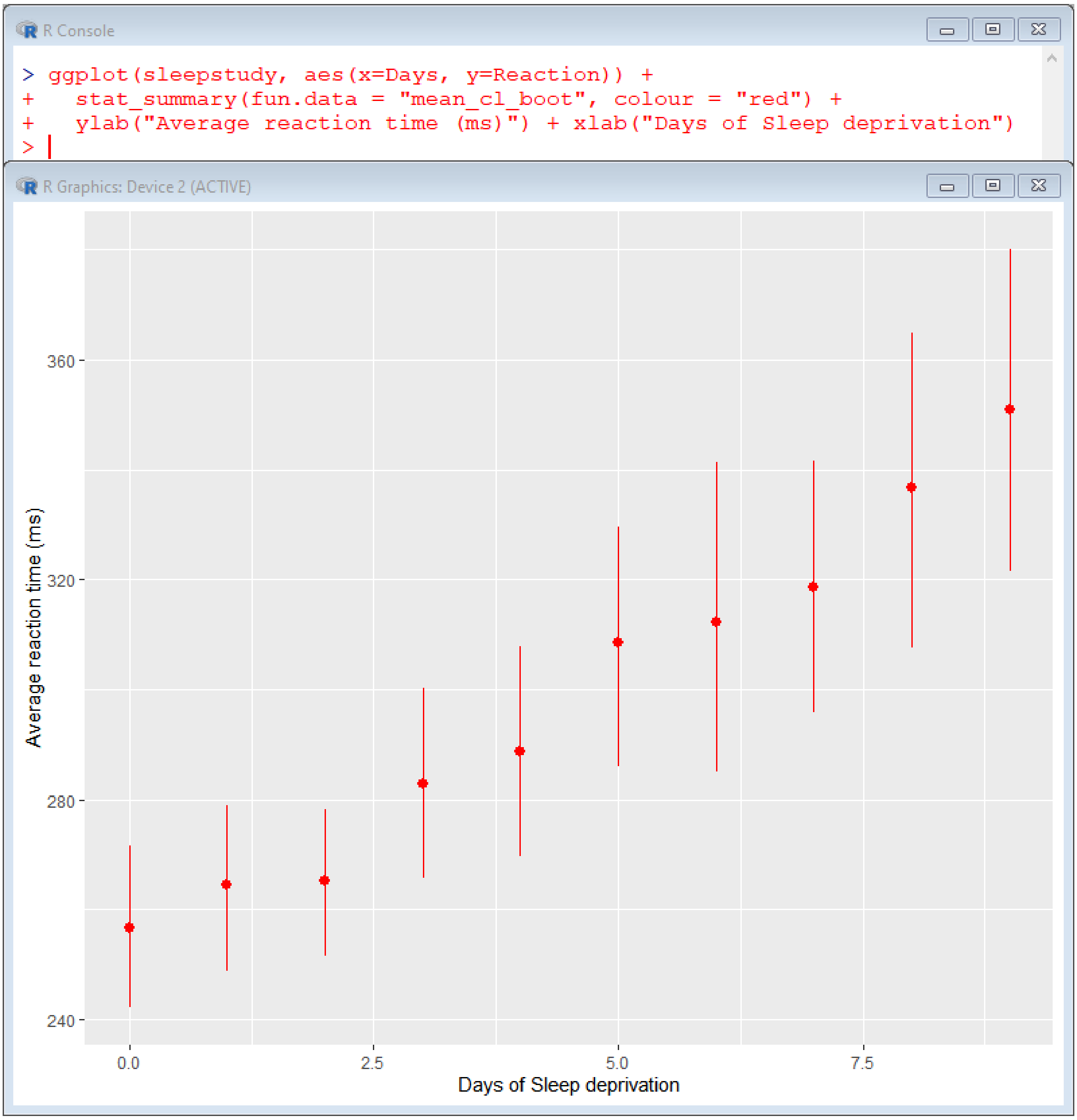


Ou dans un style plus académique:

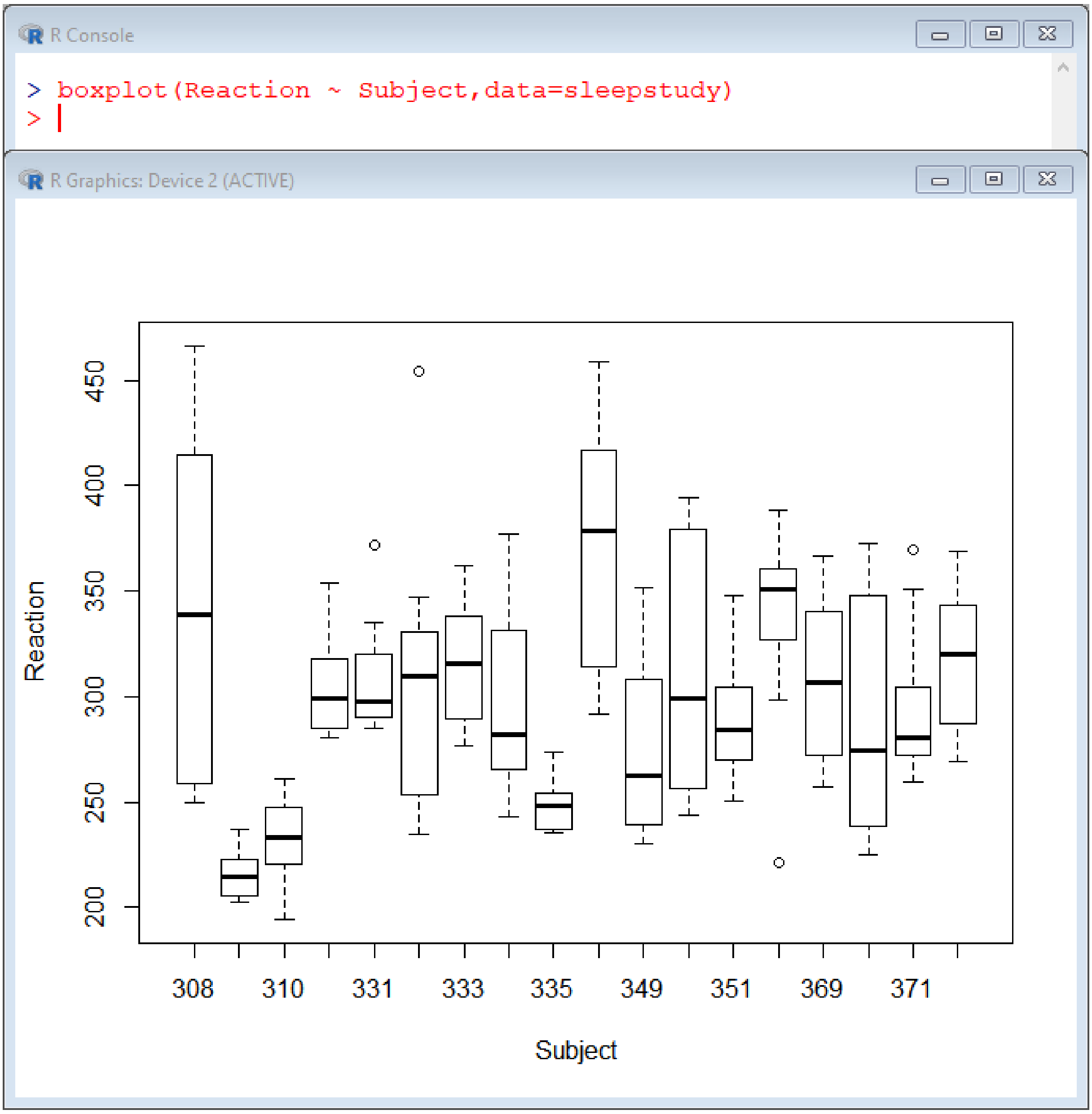


On voit clairement que faire une régression GLM vu les tendances qui s'inversent de ci de là… va pas être la meilleure solution…!

Faisons aussi un plot de l'évolution des temps de réflexes avec un intervalle de confiance utilisant le bootstrapping:


R Console
> ggplot(sleepstudy, aes(x=Days, y=Reaction)) +
+ stat_summary(fun.data = "mean_cl_boot", colour = "red") +
+ ylab("Average reaction time (ms)") + xlab("Days of Sleep deprivation")
>
R Graphics: Device 2 (ACTIVE)
Average reaction time (ms)
360
320
280
240
0.0
2.5
5.0
7.5
Days of Sleep deprivation

Ou encore:



Faisons maintenant une régression linéaire simple (même si nous savons déjà que ce n'est pas une mauvaise idée):

```
> mod <- lm(Reaction ~ Days, sleepstudy)
> summary(mod)

Call:
lm(formula = Reaction ~ Days, data = sleepstudy)

Residuals:
    Min      1Q  Median      3Q     Max 
-110.85  -27.48    1.55   26.14  139.95 

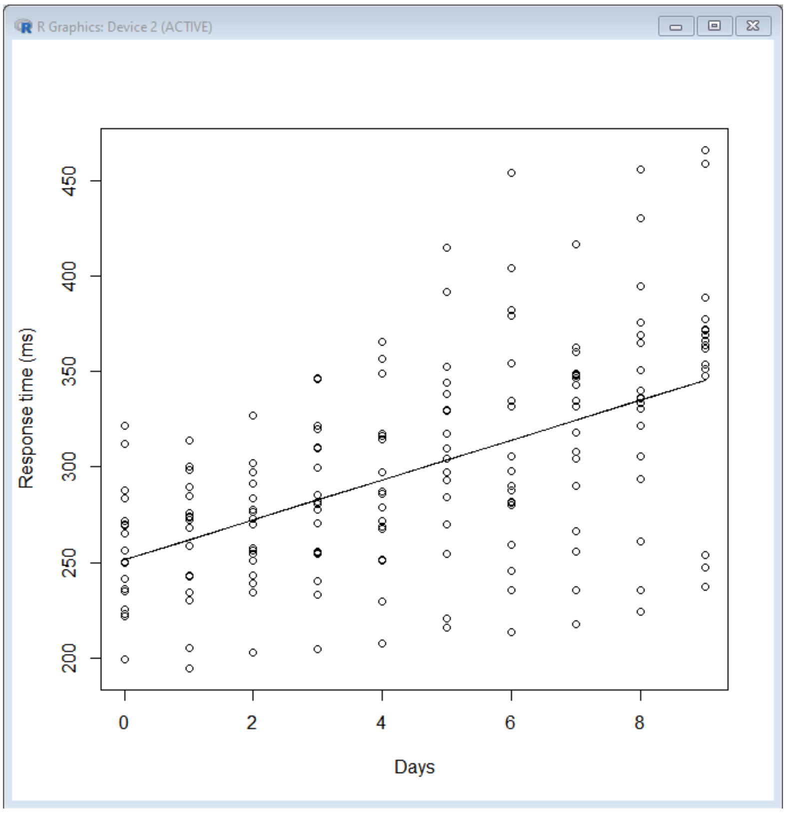
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   251.41       6.61   38.03  < 2e-16
Days           10.47       1.24    8.45  9.9e-15

Residual standard error: 47.7 on 178 degrees of freedom
Multiple R-squared:  0.286,	Adjusted R-squared:  0.282 
F-statistic: 71.5 on 1 and 178 DF,  p-value: 9.89e-15

> plot(sleepstudy$Days,sleepstudy$Reaction,
+ xlab = "Days", ylab = "Response time (ms)")
> lines(sleepstudy$Days,mod$fitted.values)
> 
```

On voit que le coefficient de corrélation est terriblement mauvais…

Ce qui donne graphiquement:



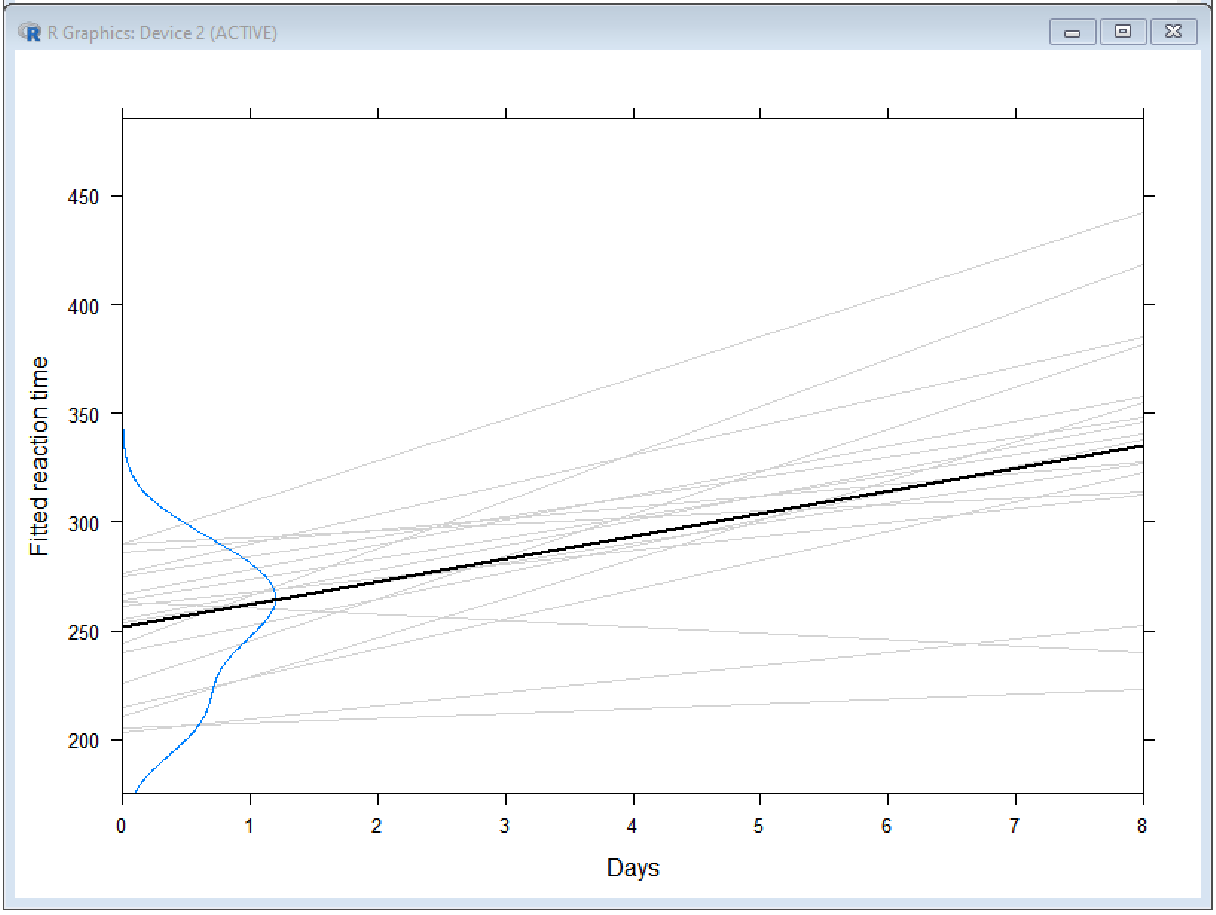
Faisons des multiples régressions linéaires pour chacun des sujets (toujours dans un objectif de diagnostique) à l'aide de la fonction **lmList( )** du package **lme4**:

```
> #une régression linéaire pour chaque sujet
> (reg.subj <- lmList(Reaction ~ Days | Subject, data=sleepstudy))
Call: lmList(formula = Reaction ~ Days | Subject, data = sleepstudy)
Coefficients:
    (Intercept)  Days
308         244 21.76
309         205  2.26
310         203  6.11
330         290  3.01
331         286  5.27
332         264  9.57
333         275  9.14
334         240 12.25
335         263 -2.88
337         290 19.03
349         215 13.49
350         226 19.50
351         261  6.43
352         276 13.57
369         255 11.35
370         210 18.06
371         254  9.19
372         267 11.30

Degrees of freedom: 180 total; 144 residual
Residual standard error: 25.6
> reg.subj.df <- data.frame(lapply(reg.subj, coef))
> #avec l'intervalle de fluctuation de l'ensemble au 1er et 3ème quartile
> apply(reg.subj.df, 1, quantile, prob = c(0.25, 0.75))
    (Intercept)  Days
25%         229  6.19
75%         273 13.55
> |
```

Et on peut faire un graphique des plus intéressants à l'aide des variables précédemment créées:

```
> intcpt.dens <- density(as.numeric(reg.subj.df[1, ]), adj = 1.2)
> len <- length(intcpt.dens$x)
> xyplot(Reaction ~ Days, data = sleepstudy, xlim = c(0, 8), ylab = "Fitted $
+   scales = list(x = list(at = seq(0, 8, by = 1))), panel = function(x,
+     y) {
+     apply(reg.subj.df, 2, panel.abline, col = "lightgrey")
+     panel.lines(x = c(intcpt.dens$y * 100, rep(0, len)),
+       y = c(intcpt.dens$x, rev(intcpt.dens$x)))
+     panel.lmline(x, y, col = "black", lwd = 2)
+   })
> |
```



Faisons la régression linéaire mixte maintenant avec la fonction **lmer( )** du package **lme4**:

```
> lmerfit <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
> summary(lmerfit )
Linear mixed model fit by REML ['lmerMod']
Formula: Reaction ~ Days + (Days | Subject)
   Data: sleepstudy

REML criterion at convergence: 1744

Scaled residuals:
   Min      1Q Median      3Q     Max
-3.954 -0.463  0.023  0.463  5.179

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Subject  (Intercept) 611.9    24.74
          Days         35.1     5.92    0.07
 Residual             654.9    25.59
Number of obs: 180, groups:  Subject, 18

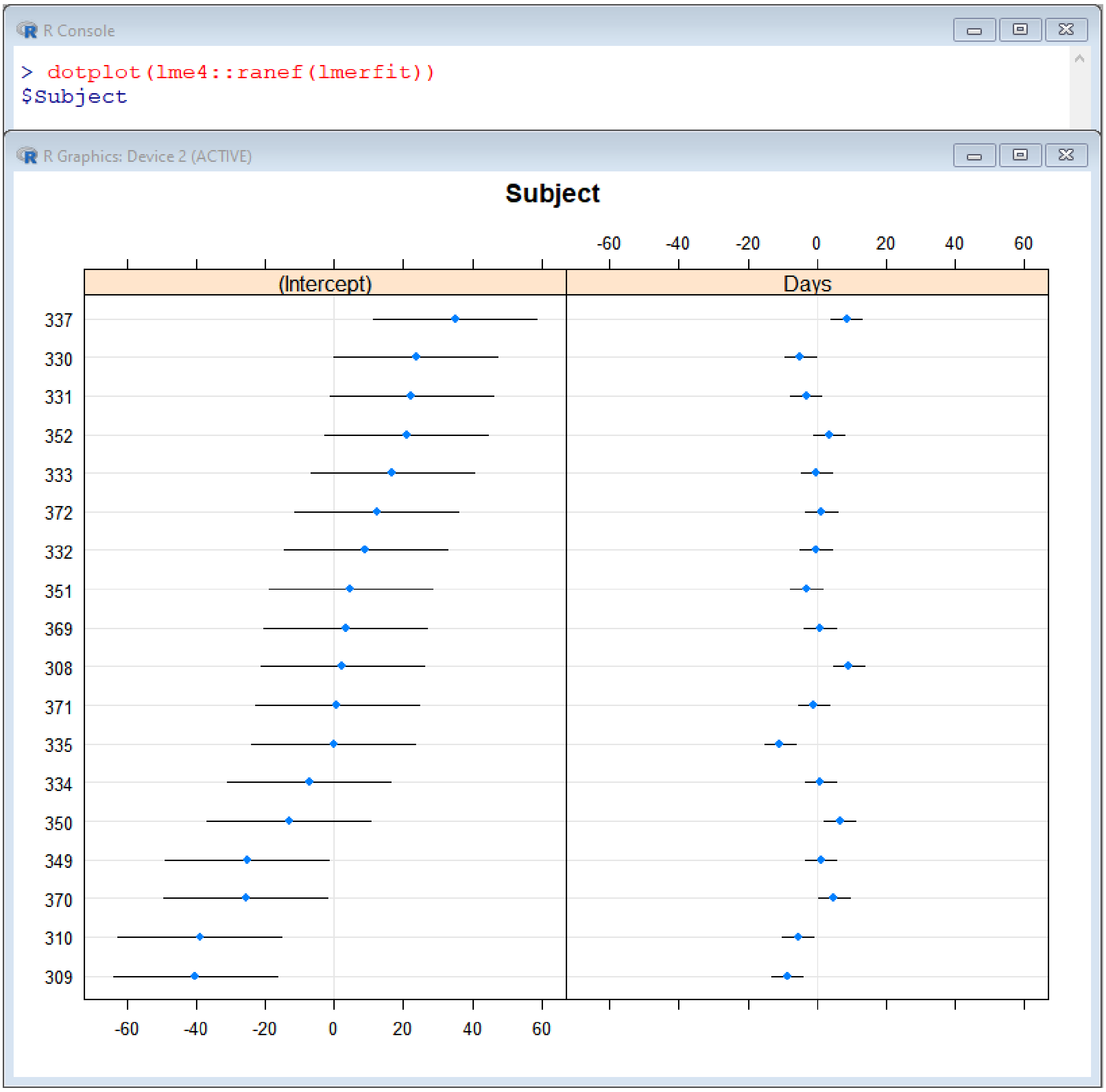
Fixed effects:
            Estimate Std. Error t value
(Intercept)   251.41       6.82   36.84
Days           10.47       1.55    6.77

Correlation of Fixed Effects:
     (Intr)
Days -0.138
>
```

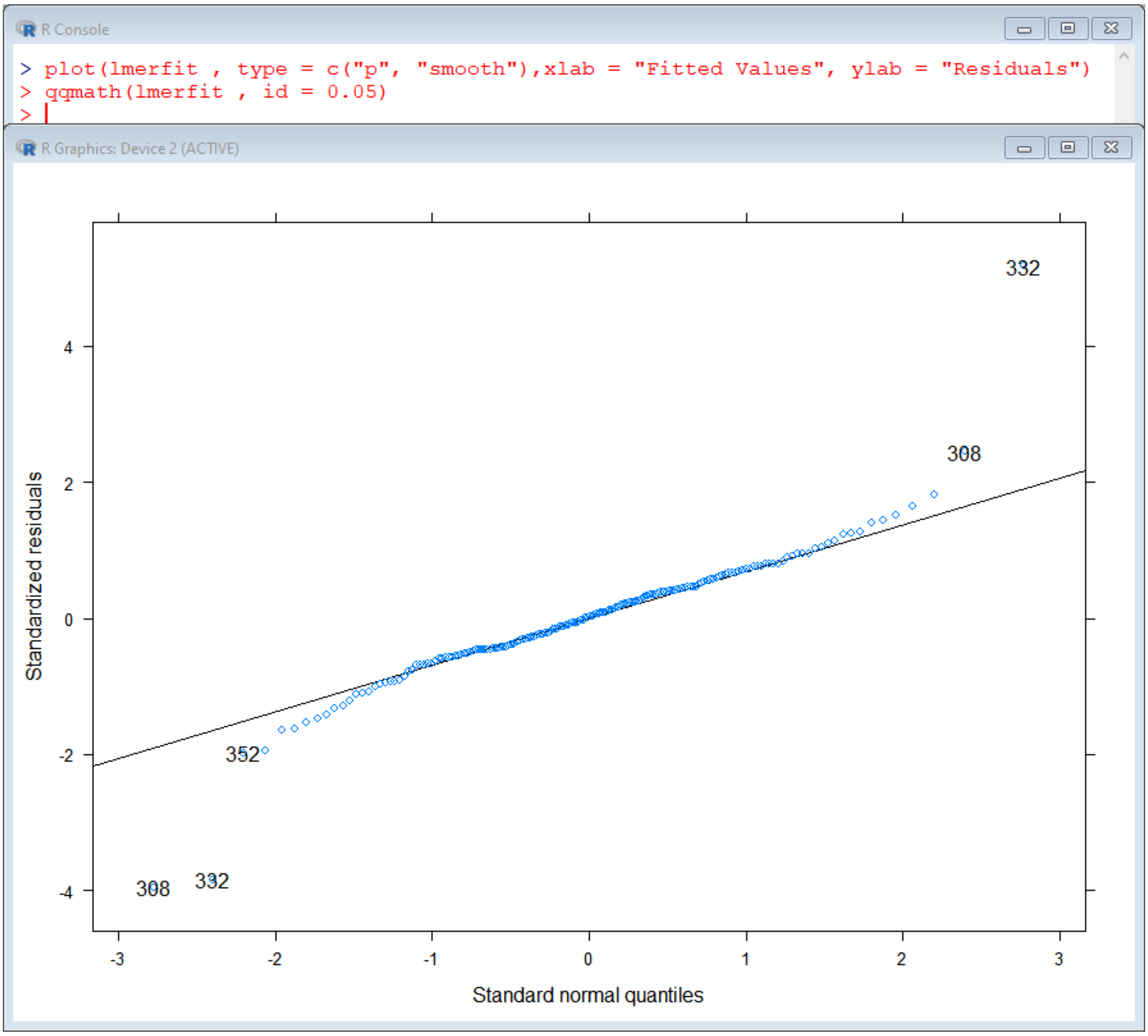
Pour ceux qui veulent l'équation du modèle en LaTeX cela est possible avec le package **equatiomatic** et sa fonction **extract_eq( )**:

```
> library("equatiomatic")
> fit <- lmer(Reaction~Days + (Days|Subject), sleepstudy)
> extract_eq(fit, use_coefs=TRUE, wrap = TRUE)
```

Nous pouvons faire un graphique de chacune des pentes et abscisses normalisés:

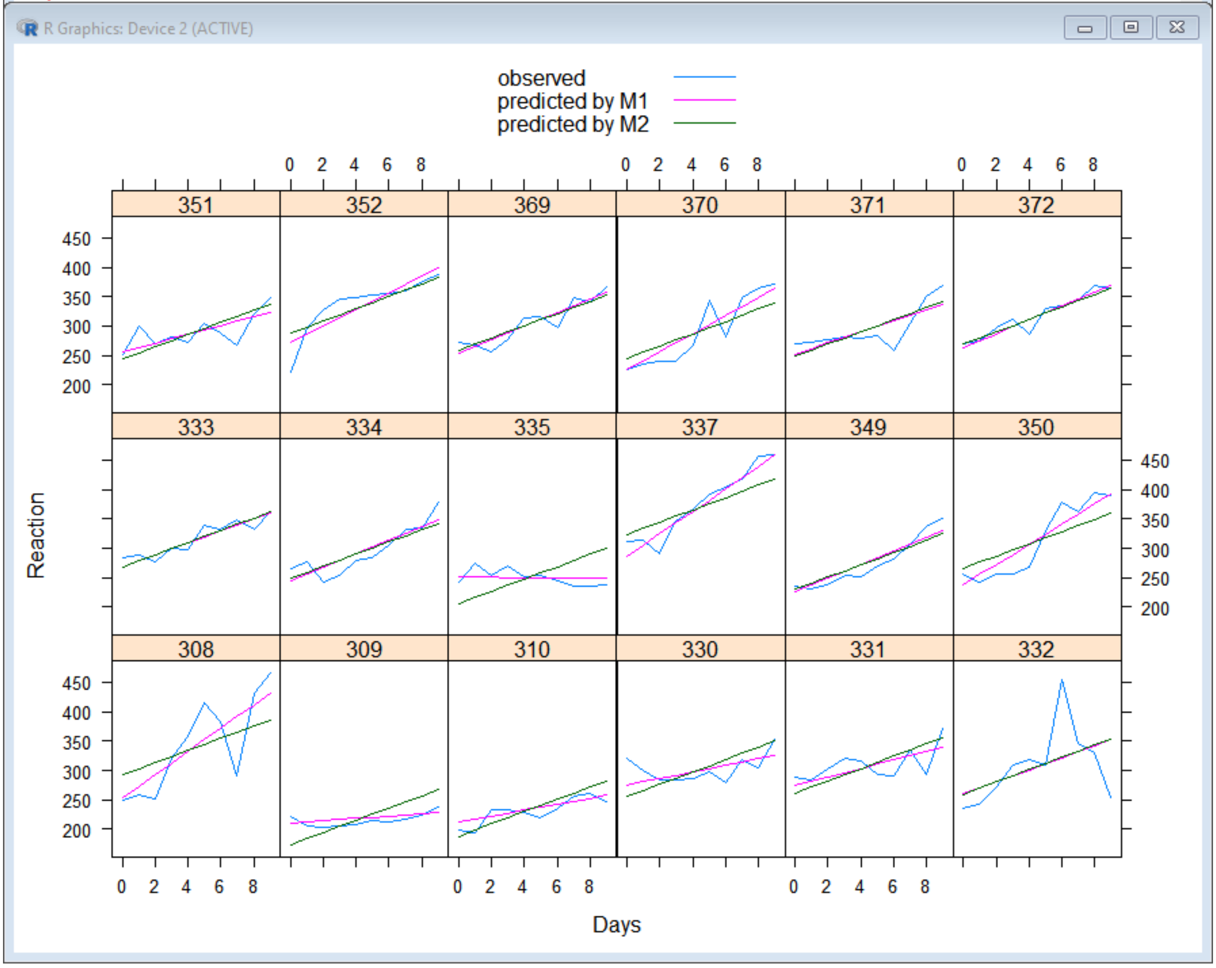


Faisons un analyse qualitative des résidus:

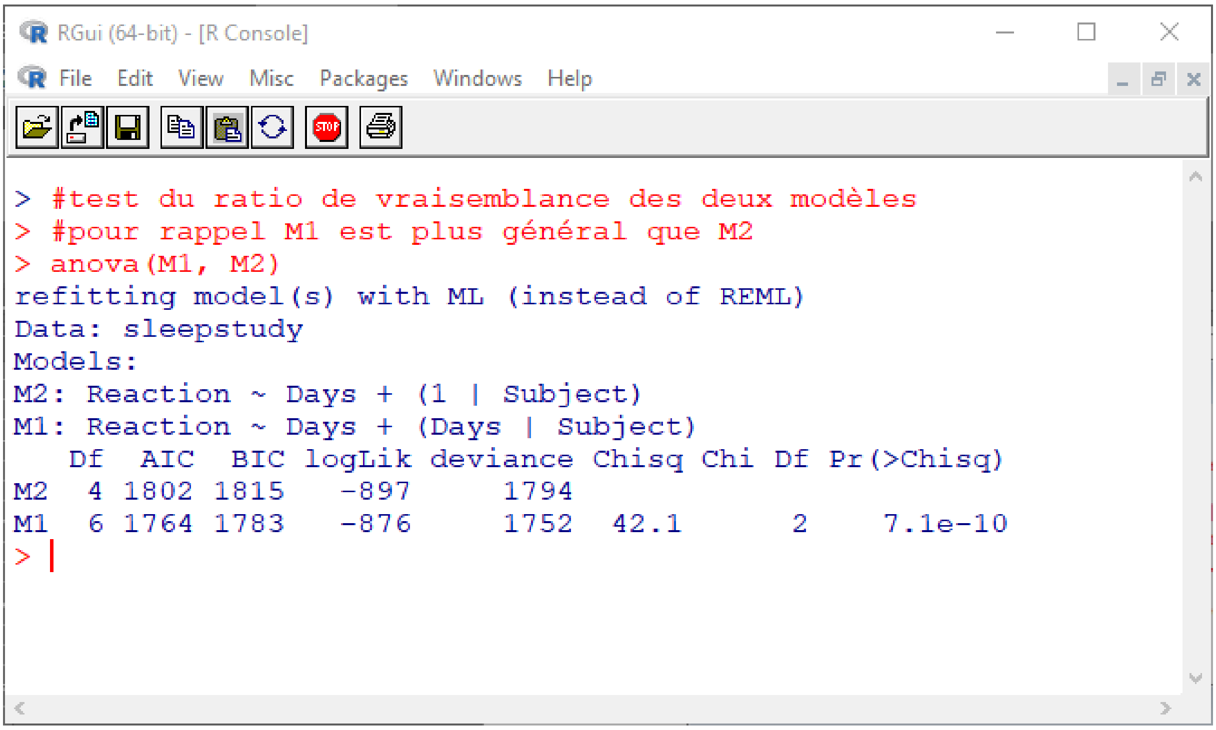


Comparons maintenant deux modèles linéaires mixtes différents avec les valeurs observées:

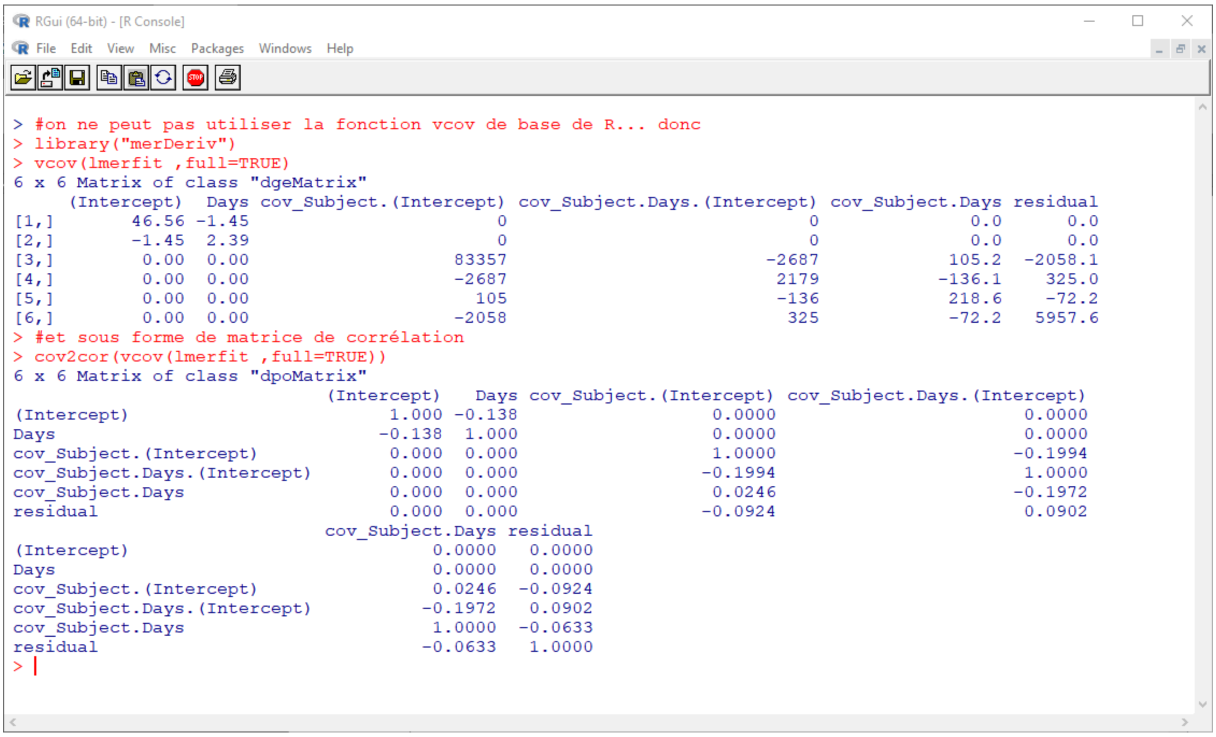
```
> #modèle complet mixte
> M1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
> #modèle où les effets aléatoires ont leur abscisse forcée à l'origine
> M2 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
> df <- cbind(sleepstudy, data.frame(yhat.M1 = predict(M1), yhat.M2 = predict(M2)))
> xyplot(Reaction + yhat.M1 + yhat.M2 ~ Days | Subject, data = df, type = "l",
+        ylab = "Reaction", layout = c(6, 3),
+        auto.key = list(text = c("observed", "predicted by M1", "predicted by M2"),
+   points = FALSE, lines = TRUE))
>
```



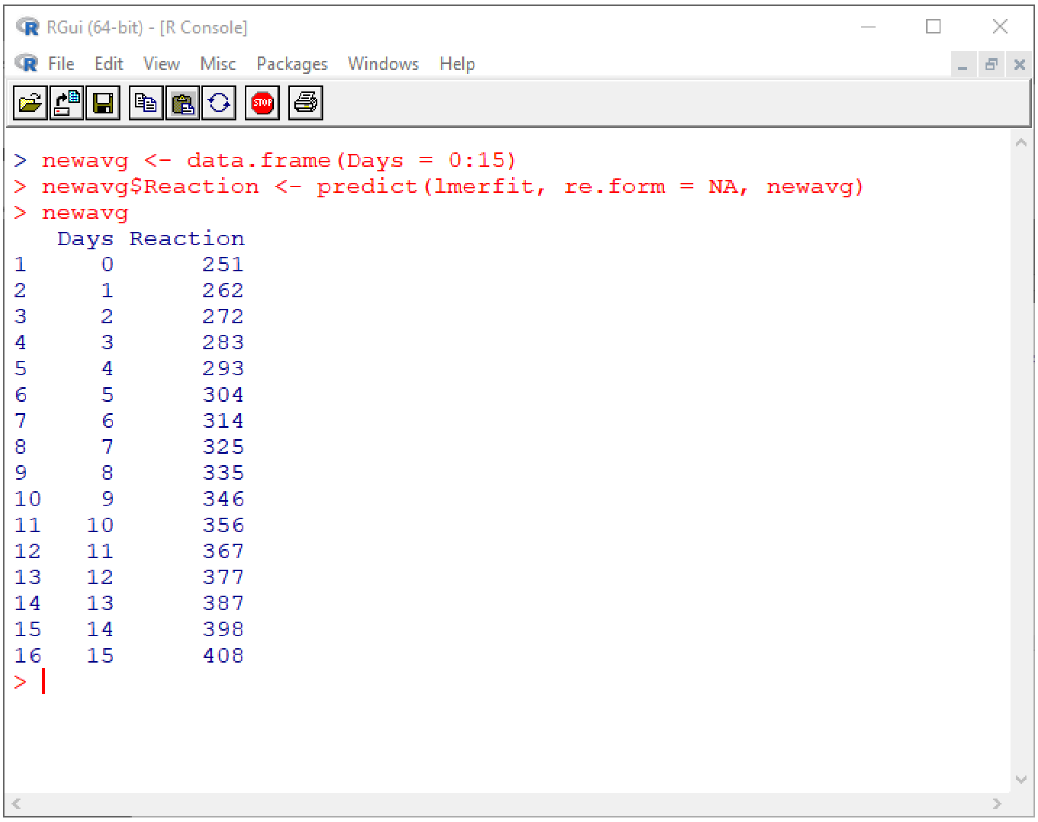
On peut faire un test du maximum de vraisemblance entre les deux modèles:



On sait que la matrice des variances-covariances est un peu le nerf de la gère des GLMM. Voyons comment l'obtenir:



Enfin, on peut faire des prédictions pour terminer:



Voilà ce sera tout pour cette petit introduction aux GLMM.

Equivalence entre le test de Student et le le test de student de l'effet fixe

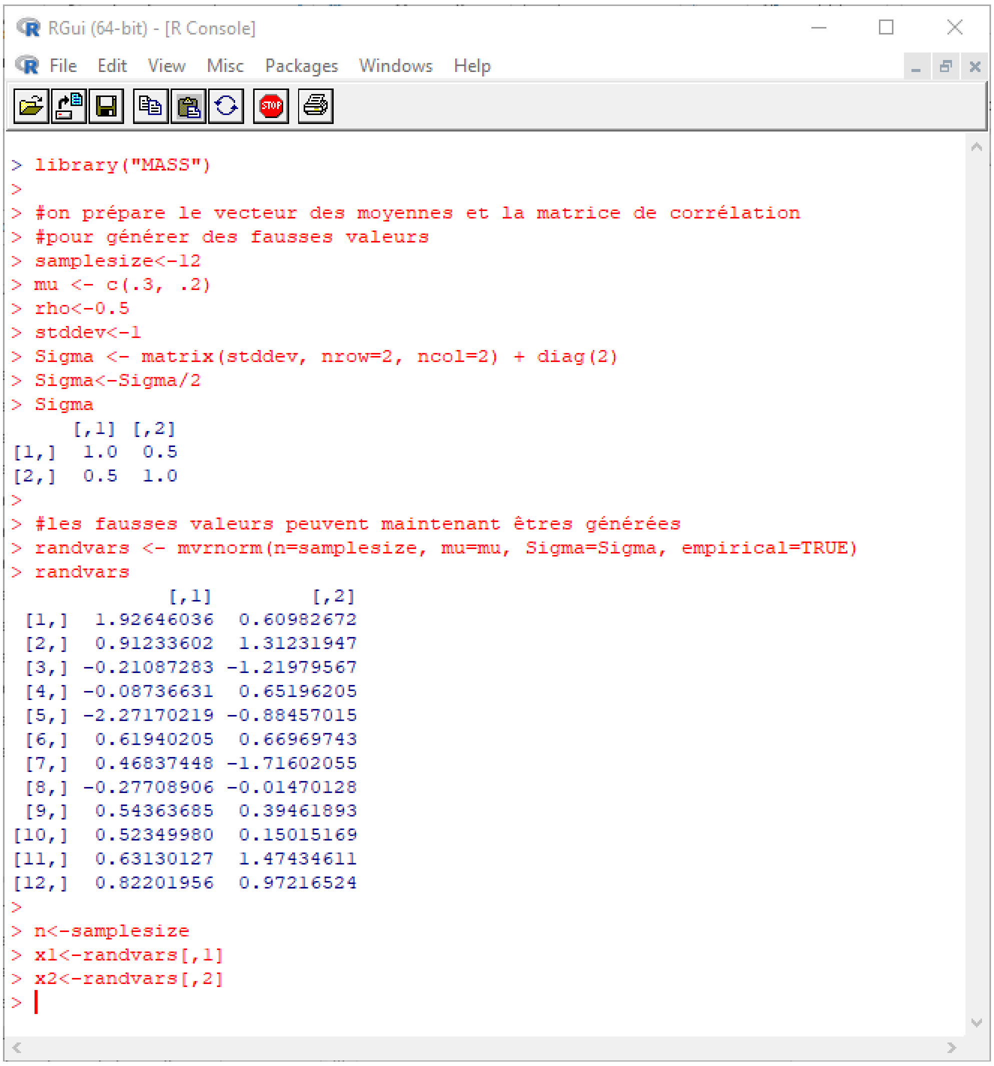
Comme mentionné dans le cours théorique et selon l'excellent article de Shravan Vasishth:

https://vasishth-statistics.blogspot.com/2018/04/a-little-known-fact-paired-t-test-is.html

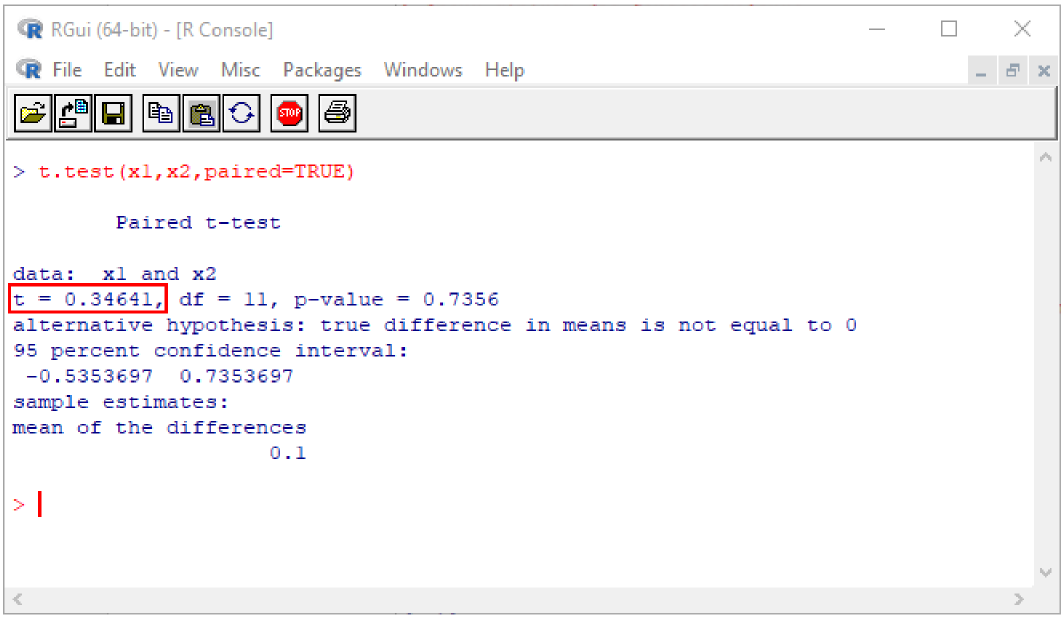
Le test T apparié est équivalent à un modèle mixte linéaire avec interceptions variables!

Vérifions cela en pratique (après l'avoir démontré dans le cours théorique bien évidemment!):

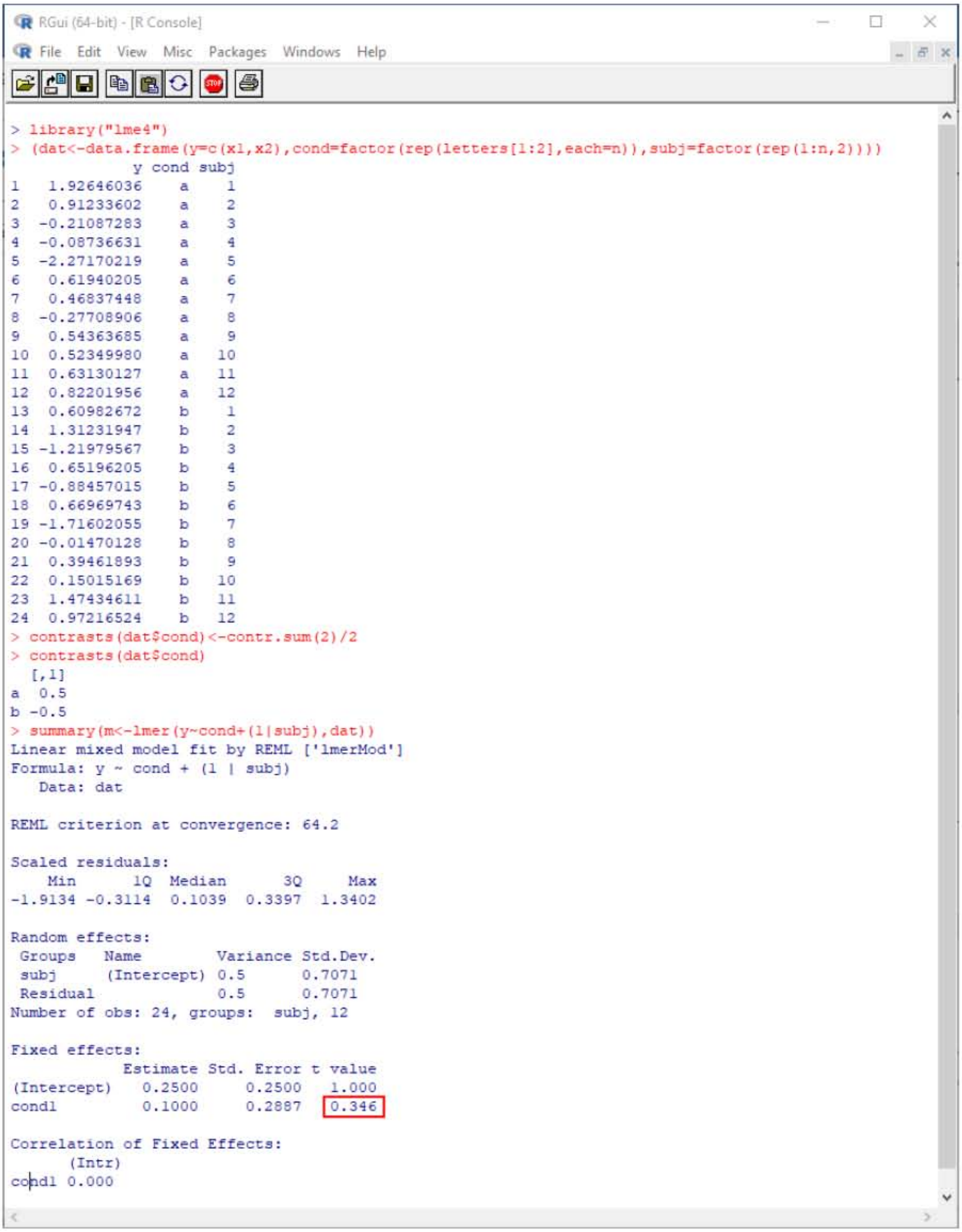
On prépare d'abord les données:



Exécutons maintenant un test apparié de Student:



Comaprons donc la t-valeur à celle obtenu en faisaint une analyse par modèles linéaires mixtes:



```
> library("lme4")
> (dat<-data.frame(y=c(x1,x2),cond=factor(rep(letters[1:2],each=n)),subj=factor(rep(1:n,2))))
             y cond subj
1   1.92646036    a    1
2   0.91233602    a    2
3  -0.21087283    a    3
4  -0.08736631    a    4
5  -2.27170219    a    5
6   0.61940205    a    6
7   0.46837448    a    7
8  -0.27708906    a    8
9   0.54363685    a    9
10  0.52349980    a   10
11  0.63130127    a   11
12  0.82201956    a   12
13  0.60982672    b    1
14  1.31231947    b    2
15 -1.21979567    b    3
16  0.65196205    b    4
17 -0.88457015    b    5
18  0.66969743    b    6
19 -1.71602055    b    7
20 -0.01470128    b    8
21  0.39461893    b    9
22  0.15015169    b   10
23  1.47434611    b   11
24  0.97216524    b   12
> contrasts(dat$cond)<-contr.sum(2)/2
> contrasts(dat$cond)
  [,1]
a  0.5
b -0.5
> summary(m<-lmer(y~cond+(1|subj),dat))
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ cond + (1 | subj)
   Data: dat

REML criterion at convergence: 64.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.9134 -0.3114  0.1039  0.3397  1.3402

Random effects:
 Groups   Name        Variance Std.Dev.
 subj     (Intercept) 0.5      0.7071
 Residual             0.5      0.7071
Number of obs: 24, groups:  subj, 12

Fixed effects:
            Estimate Std. Error t value
(Intercept)   0.2500     0.2500   1.000
cond1         0.1000     0.2887   0.346

Correlation of Fixed Effects:
      (Intr)
cond1 0.000
```

On retrouve donc bien la même t-valeur!

Exemple 276.: Régression en composantes principales

R 3.4.2

Cette régression pour venir à bout de la colinéarité, utilise les composantes principales issues d'une ACP plutôt que les variables brutes du jeu de données.

Pour cela on va travailler avec le jeu de données suivant:

```
> data(vehicles,package="plsdepot")
> str(vehicles)
'data.frame':   30 obs. of  16 variables:
 $ diesel      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ turbo       : int  0 0 0 0 1 0 0 0 0 0 ...
 $ two.doors   : int  1 0 1 0 1 0 1 0 0 0 ...
 $ hatchback   : int  1 0 0 0 1 1 1 0 0 0 ...
 $ wheel.base  : num  94.5 105.8 101.2 94.5 93.7 ...
 $ length      : num  171 193 177 159 157 ...
 $ width       : num  65.5 71.4 64.8 63.6 63.8 63.8 64 65.2 61.8 69.6 ...
 $ height      : num  52.4 55.7 54.3 52 50.8 50.6 52.6 54.1 53.5 52.8 ...
 $ curb.weight: int  2823 2844 2395 1909 2128 1967 1956 2304 2337 4066 ...
 $ eng.size    : int  152 136 108 90 98 90 92 110 111 258 ...
 $ horsepower : int  154 110 101 70 102 68 76 86 78 176 ...
 $ peak.rpm    : int  5000 5500 5800 5400 5500 5500 6000 5800 4800 4750 ...
 $ price       : int  16500 17710 16430 6575 7957 6229 7129 8845 6785 35550 ...
 $ symbol      : int  1 1 2 0 1 1 1 0 0 0 ...
 $ city.mpg    : int  19 19 23 38 24 31 30 27 24 15 ...
 $ highway.mpg: int  26 25 29 43 30 38 34 33 29 19 ...
>
```

R propose, plusieurs fonctions pour réaliser une ACP, dont l'une d'elle **prcomp( )** offre les arguments comme ***center*** et ***scale*** pour centrer et réduire les prédicteurs ce qui est obligatoire pour mettre ces derniers à la même échelle :

```
> ACP<-prcomp(data.matrix(vehicles[,-13]),
+ scale=TRUE,center=TRUE)
> summary(ACP)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     2.7088 1.7385 1.18090 0.95326 0.85081 0.68018 0.63435
Proportion of Variance 0.4892 0.2015 0.09297 0.06058 0.04826 0.03084 0.02683
Cumulative Proportion  0.4892 0.6906 0.78361 0.84419 0.89245 0.92330 0.95012
                           PC8     PC9    PC10    PC11    PC12    PC13   PC14
Standard deviation     0.52882 0.43413 0.33680 0.31055 0.18198 0.14038 0.1094
Proportion of Variance 0.01864 0.01256 0.00756 0.00643 0.00221 0.00131 0.0008
Cumulative Proportion  0.96877 0.98133 0.98889 0.99532 0.99753 0.99884 0.9996
                          PC15
Standard deviation     0.07337
Proportion of Variance 0.00036
Cumulative Proportion  1.00000
> 
```

On obtient en suite une liste de 15 composantes, donc de mêmes nombres que les variables de bases mais dont chacune est une combinaison linéaire des variables de bases. On peut afficher le poids de chaque variable pour toutes les composantes comme ceci :

```
> #les poids de chaque variables dans les composantes
> library("dplyr")
> ACP %>% print(digits=2)
Standard deviations (1, .., p=15):
 [1] 2.709 1.738 1.181 0.953 0.851 0.680 0.634 0.529 0.434 0.337 0.311 0.182
[13] 0.140 0.109 0.073

Rotation (n x k) = (15 x 15):
              PC1     PC2     PC3    PC4     PC5    PC6    PC7    PC8    PC9
diesel       0.14  0.3912  0.3457  0.217 -0.1293 -0.310  0.378 -0.037  0.558
turbo        0.20  0.1267  0.4687 -0.384  0.3320 -0.423  0.023 -0.250 -0.324
two.doors   -0.14 -0.2770  0.5117  0.355  0.2072 -0.027 -0.115  0.548 -0.046
hatchback   -0.17 -0.2238  0.4948 -0.371 -0.0501  0.536 -0.074 -0.106  0.294
wheel.base   0.34  0.1167  0.0860 -0.044  0.0974  0.351  0.123  0.060 -0.132
length       0.35  0.0353 -0.1073  0.171  0.0429  0.183 -0.049 -0.107 -0.081
width        0.33 -0.0045  0.0601  0.031  0.1506  0.378  0.373 -0.145 -0.171
height       0.24  0.2361 -0.0426  0.339  0.5216  0.115 -0.374  0.119  0.119
curb.weight  0.35 -0.0895  0.0693  0.051 -0.1631  0.073  0.122  0.103  0.083
eng.size     0.29 -0.2402  0.0043  0.119 -0.4196 -0.033  0.199  0.246 -0.033
horsepower   0.22 -0.4172  0.0595 -0.022 -0.0488 -0.280  0.048  0.104 -0.315
peak.rpm    -0.16 -0.3213 -0.2844 -0.105  0.5567 -0.050  0.582  0.116  0.239
symbol      -0.19 -0.2893  0.1651  0.587  0.0053  0.018  0.084 -0.654 -0.097
city.mpg    -0.26  0.3630  0.1080  0.061 -0.0723  0.142  0.262  0.197 -0.288
highway.mpg -0.30  0.2738  0.0411  0.132 -0.0357  0.123  0.267  0.112 -0.414
               PC10   PC11   PC12   PC13    PC14    PC15
diesel      -0.1458  0.041 -0.133  0.168  0.0147 -0.1486
turbo        0.1566 -0.105  0.053 -0.282 -0.0207  0.0096
two.doors    0.1958  0.321 -0.094 -0.045 -0.0298  0.0469
hatchback   -0.2199 -0.263 -0.095 -0.050 -0.0549 -0.1157
wheel.base   0.5254 -0.112  0.072  0.513  0.2723 -0.2513
length       0.1228 -0.006 -0.670 -0.206 -0.4839 -0.1912
width       -0.3691  0.566  0.230 -0.112  0.0065  0.0994
height      -0.3279 -0.372  0.230 -0.090  0.0685 -0.0576
curb.weight  0.0880 -0.300 -0.122 -0.059  0.1163  0.8154
eng.size     0.0648 -0.248  0.341 -0.481  0.0242 -0.3861
horsepower  -0.4965 -0.186 -0.138  0.510 -0.0811 -0.1070
peak.rpm     0.1017 -0.189 -0.056 -0.070 -0.0582 -0.0320
symbol       0.1383 -0.163  0.123  0.045  0.0079  0.0255
city.mpg    -0.0021 -0.245  0.275  0.141 -0.6298  0.1298
highway.mpg -0.2116 -0.188 -0.397 -0.197  0.5115 -0.0740
> |
```

Ainsi les nouvelles variables Z s'obtiennent par le produit des points W (qu'on peut extraire de **$rotation**), à la matrice X des prédicteurs centrés et réduits :

Soit $\Sigma = X^\top X$, la matrice de covariance des prédicteurs X et W, respectivement λ, les vecteurs propres et valeurs prores de Σ :

de telle sorte que : $W^\top \Sigma W = \lambda$

Alors les nouvelles variables Z s'obtiennent comme suit :

$$Z = W^\top X$$

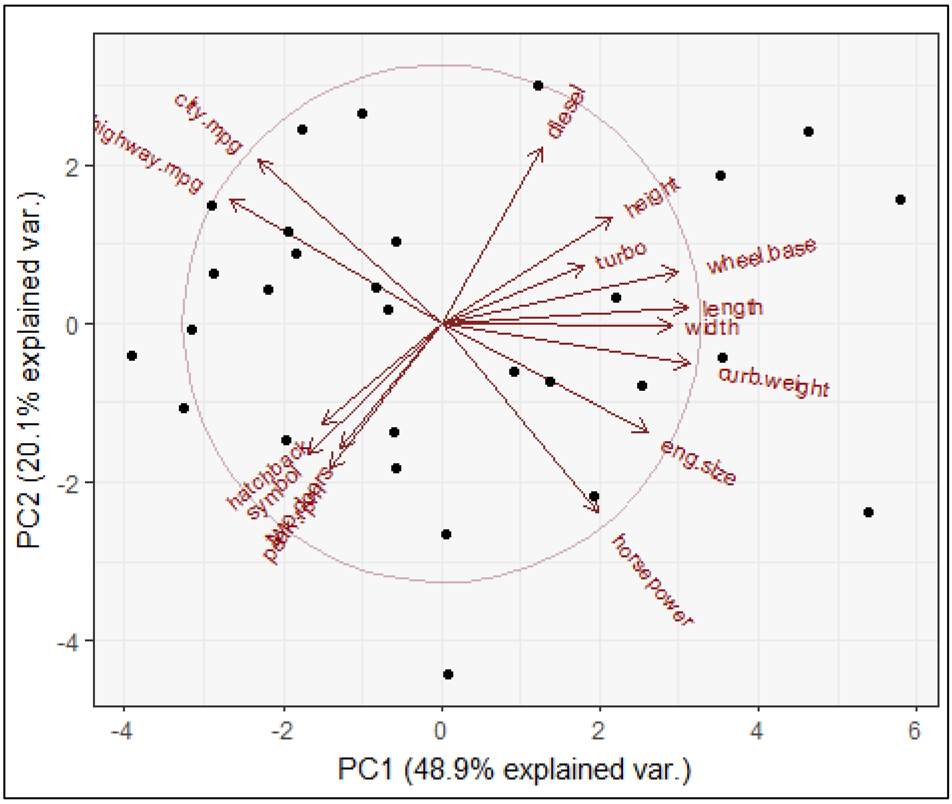
Mais tout l'art de l'ACP est justement de considérer seulement un nombre p de composantes qu'on juge suffisant selon la proportion d'information I% qu'elles expliquent :

$$I(\%) = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_p}{\lambda_1 + \lambda_2 + \cdots + \lambda_m}$$

Par exemple il est possible de projeter les données dans un espace réduit à deux dimensions en considérant les deux premières composantes(Z_1 et Z_2) comme principales :

```
> # installation de ggbiplot
> # devtools::install_github("vqv/ggbiplot")
> library(ggbiplot)
> # projection graphique des points dans un espace p = 2
> ggbiplot(ACP, obs.scale = 1,
+ var.scale = 1, circle = TRUE) +
+                theme_bw() +
+ theme(panel.background = element_rect(fill = "gray97"))
>
```

On obtient le graphique suivant communément appelé biplot avec la fonction **ggbiplot( )** du package **ggbiplot** :



En se basant sur la formule de I% plus haut, R a estimé que la première composante explique 48.9% de la variance et la seconde composante principale 20.1%, soit un cumul de 69% de la variance ou de l'information. Pour obtenir ces résultats numériques on utilise la **summary( )** sur le résultat de la fonction **prcomp( )** :

```
> summary(ACP)
Importance of components%s:
                          PC1    PC2     PC3     PC4
Standard deviation     2.7088 1.7385 1.18090 0.95326
Proportion of Variance 0.4892 0.2015 0.09297 0.06058
Cumulative Proportion  0.4892 0.6906 0.78361 0.84419
                           PC5     PC6     PC7     PC8
Standard deviation     0.85081 0.68018 0.63435 0.52882
Proportion of Variance 0.04826 0.03084 0.02683 0.01864
Cumulative Proportion  0.89245 0.92330 0.95012 0.96877
                           PC9    PC10    PC11    PC12
Standard deviation     0.43413 0.33680 0.31055 0.18198
Proportion of Variance 0.01256 0.00756 0.00643 0.00221
Cumulative Proportion  0.98133 0.98889 0.99532 0.99753
                          PC13   PC14    PC15
Standard deviation     0.14038 0.1094 0.07337
Proportion of Variance 0.00131 0.0008 0.00036
Cumulative Proportion  0.99884 0.9996 1.00000
>
```

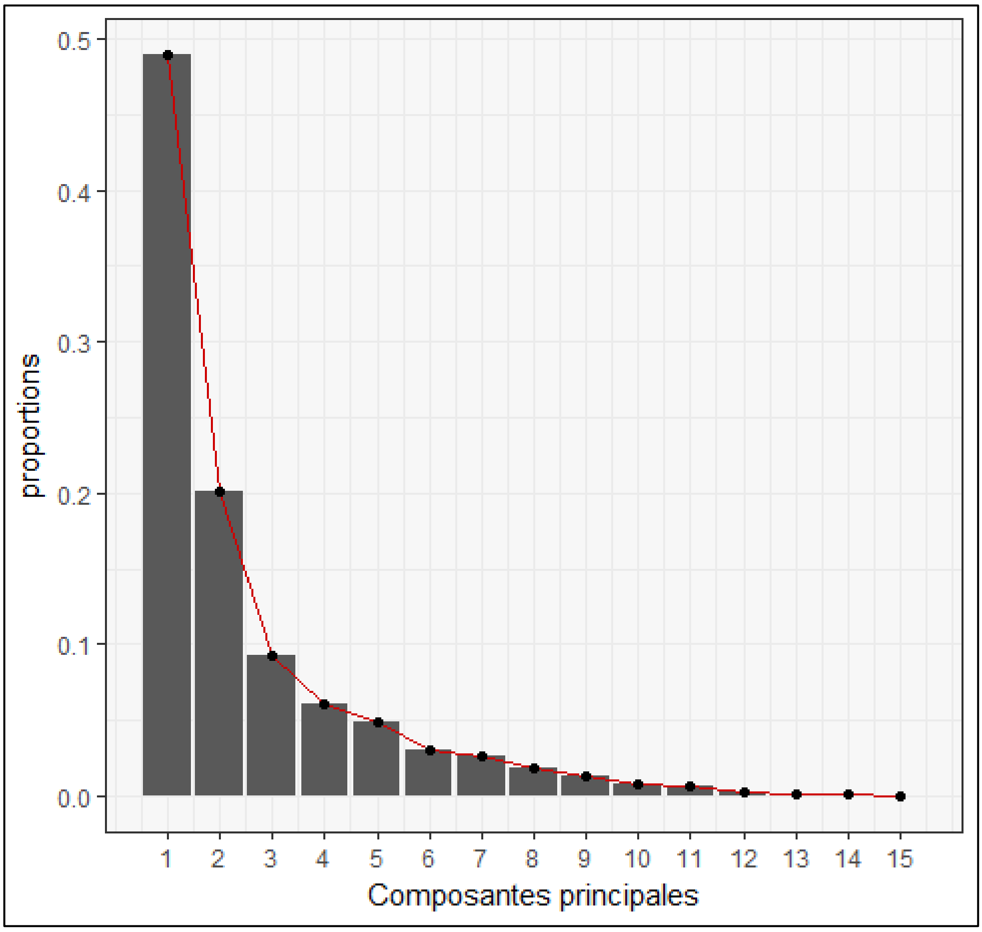
Manuellement, les proportions sont obtenues en rapportant la variance de chaque composante sur la variance totale des composantes :

```
> # la variance expliquée par chaque composante
> ACP$sdev^ 2
 [1] 7.337439966 3.022254899 1.394515436 0.908695829
 [5] 0.723883012 0.462642165 0.402394650 0.279650089
 [9] 0.188465318 0.113436032 0.096440195 0.033118235
[13] 0.019705897 0.011975460 0.005382818
> # proportion expliquée par chaque composante (%)
> ACP$sdev ^ 2 / sum(ACP$sdev ^ 2)
 [1] 0.4891626644 0.2014836599 0.0929676957 0.0605797219
 [5] 0.0482588675 0.0308428110 0.0268263100 0.0186433393
 [9] 0.0125643545 0.0075624021 0.0064293463 0.0022078823
[13] 0.0013137265 0.0007983640 0.0003588545
>    # afficher sous format pour centage
> # la porportion expliquée cumulée (%)
> cumsum(ACP$sdev ^ 2) / sum(ACP$sdev ^ 2)
 [1] 0.4891627 0.6906463 0.7836140 0.8441937 0.8924526
 [6] 0.9232954 0.9501217 0.9687651 0.9813294 0.9888918
[11] 0.9953212 0.9975291 0.9988428 0.9996411 1.0000000
>
```

Alors combien de composantes peut-on qualifier de principales ? souvent en pratique on s'aide d'un graphique qui retrace les proportions des variances expliquées en fonction des composantes :

```
> ggplot(data = data.frame(CP = 1:15,
+    proportions, stringsAsFactors = F),
+    aes(x = CP, y = proportions)) +
+    geom_bar(stat = "identity") +
+              geom_line(color = "red3") +
+              geom_point() +
+ scale_x_continuous(name = "Composantes principales",
+                breaks = 1:15) +
+                theme_bw() +
+ theme(panel.background = element_rect(fill = "gray97"))
>
```

Du code ci-dessus, on obtient le graphique suivant dénommé **screeplot** :



Pour répondre à la question posée plus haut, on sélectionne conventionnellement les p composantes pour lesquelles la courbe en rouge présente un point d'inflexion (c'est plus un point de rupture avec la tendance originale) là où la courbe forme une sorte de coude ici on peut considérer.

Ici, on pourrait se contenter des deux premières composantes soit une variance expliquée cumulée de 69.0% pour PC1 et PC2. Donc on est passé d'un espace m = 15 à un espace p = 2 avec une perte contrôler d'informations de 21.64% (en termes de variance expliquée). Les nouvelles variables ou Z sont obtenues avec la fonction **predict( )**:

```
> predict(ACP)[, 1:2] %>%
+    psych::headTail(digits = 4)
        PC1     PC2
1    0.0688 -2.6559
2    2.5304  -0.799
3   -0.5991  -1.355
4   -2.9106  1.4945
...     ...     ...
27   1.3769 -0.7402
28   2.1967  0.3292
29   3.5418  -0.425
30   3.5252   1.858
>
```

- Régression sur composantes principales

Maintenant que l'on sait ce qu'est une ACP, on peut réaliser une régression sur les composantes principales avec la fonction **pcr( )** du package **pls**. Cette fonction propose plusieurs techniques de validation de modèle à travers l'argument ***validation,*** on va utiliser celle vue plus haut la méthode de validation croisée :

```
> library(pls)
> reg_acp <- pcr(log(price) ~ .,
+                data = vehicles,
+                scale = TRUE,
+               validation = "CV")
> # sommaire
> summary(reg_acp)
Data:    X dimension: 30 15
   Y dimension: 30 1
Fit method: svdpc
Number of components considered: 15

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps
CV           0.566   0.3203    0.258   0.2626
adjCV        0.566   0.3195    0.255   0.2597
       4 comps  5 comps  6 comps  7 comps  8 comps
CV      0.2775   0.2901   0.2820   0.2456   0.2457
adjCV   0.2736   0.2872   0.2792   0.2352   0.2352
       9 comps  10 comps  11 comps  12 comps
CV      0.1754    0.1813    0.1754    0.1920
adjCV   0.1708    0.1762    0.1712    0.1871
       13 comps  14 comps  15 comps
CV       0.2182    0.2323    0.2325
adjCV    0.2117    0.2247    0.2251
```

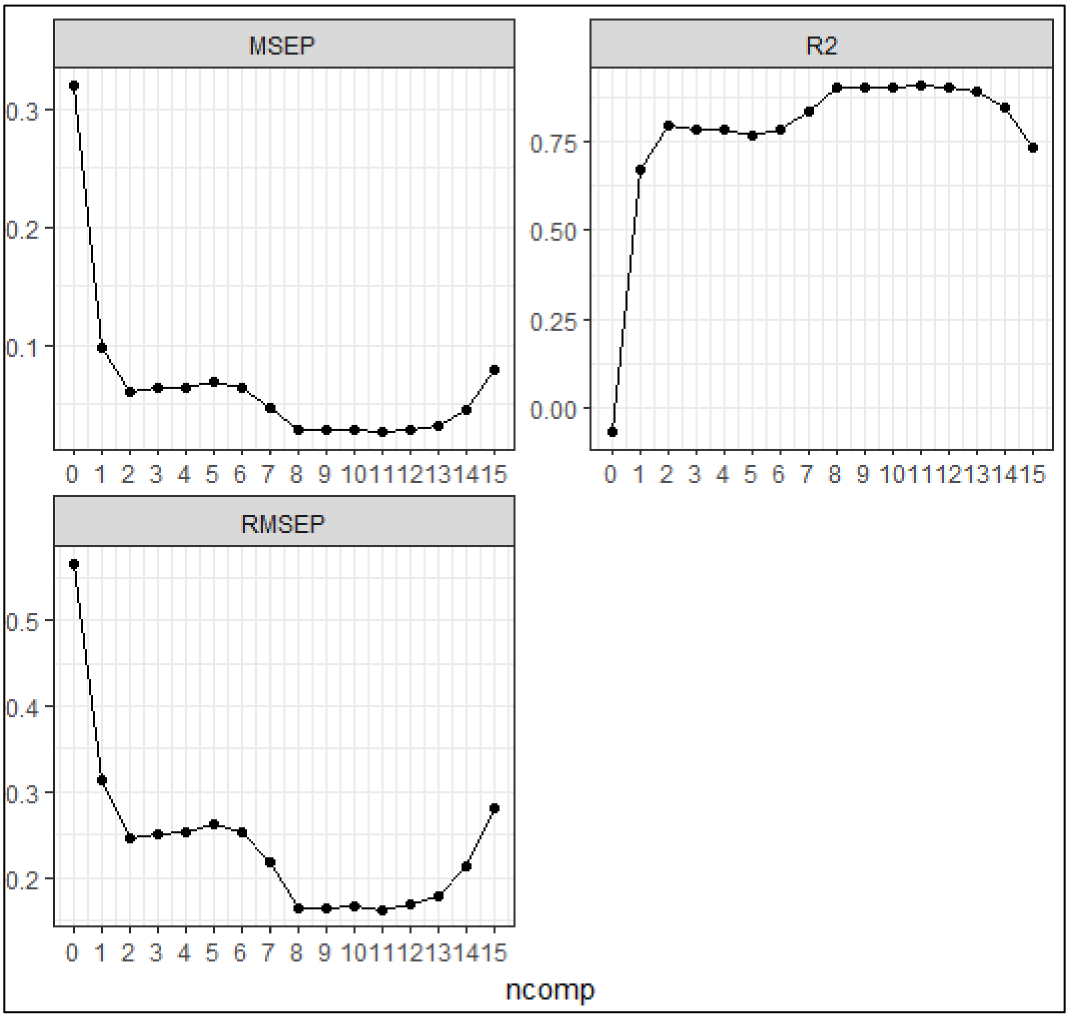
Combinée à la fonction **summary( )** on extrait les résultats de la validation croisée (10 partitionnements du jeu de données et la mesure d'erreur par la racine carrée de moyennes quadratique des erreurs de prédiction qu'on peut extraire avec la fonction **RMSEP( )**) ci-dessus et également on obtient le résultat de la variance expliquée comme ceci :

```
TRAINING: % variance explained
            1 comps  2 comps  3 comps  4 comps
X             48.92    69.06    78.36    84.42
log(price)    70.31    85.74    85.77    87.82
            5 comps  6 comps  7 comps  8 comps
X             89.25    92.33    95.01    96.88
log(price)    87.90    91.21    96.09    96.25
            9 comps  10 comps  11 comps  12 comps
X             98.13     98.89     99.53     99.75
log(price)    96.25     96.43     96.57     96.57
            13 comps  14 comps  15 comps
X              99.88     99.96    100.00
log(price)     96.91     96.98     97.01
>
```

Pour déterminer le meilleur nombre composant on peut utiliser plusieurs critères d'évaluation notamment **R2( )** et **MSEP( )** ou **RMSEP( )** qui désignent respectivement le R^2 et l'erreur quadratique de prédiction (le R de RMSEP veut dire tout simplement qu'on prend la racine carrée de MSEP :

```
> # extraire les différentes
> # mesures de performance du CV
> data.frame(
+ MSEP = MSEP(reg_acp)$val["CV",,],
+ R2 = R2(reg_acp)$val["CV",,],
+ RMSEP = RMSEP(reg_acp)$val["CV",,],
+ ncomp = 0:15) -> out
> # extrait
> out %>% psych::headTail()
> # graphique des différents mesures de performance du CV
            MSEP    R2 RMSEP ncomp
(Intercept) 0.32 -0.07  0.57     0
1 comps      0.1  0.67  0.31     1
2 comps     0.06   0.8  0.25     2
3 comps     0.06  0.79  0.25     3
...          ...   ...   ...   ...
12 comps    0.03   0.9  0.17    12
13 comps    0.03  0.89  0.18    13
14 comps    0.05  0.85  0.21    14
15 comps    0.08  0.73  0.28    15
> tidyr::gather(out, variables, valeurs, - ncomp) %>%
+    ggplot(., aes(x = ncomp, y = valeurs)) +
+    geom_line() + geom_point() +
+    facet_wrap(~variables, ncol = 2, scales = "free") +
+    theme_bw() + scale_x_continuous(breaks = 0:15)
```

On obtient le graphique suivant



Extraire le nombre de composantes principales les plus intéressantes selon les différents critères :

```
> # meilleure p pour R2
> out %>% filter(R2 == max(R2))
        MSEP        R2     RMSEP ncomp
1 0.02663294 0.9110266 0.163196    11
> # meilleure p pour MSEP
> out %>% filter(MSEP == min(MSEP))
        MSEP        R2     RMSEP ncomp
1 0.02663294 0.9110266 0.163196    11
> # meilleure p pour RMSEP
> out %>% filter(RMSEP == min(RMSEP))
        MSEP        R2     RMSEP ncomp
1 0.02663294 0.9110266 0.163196    11
>
```

Finalement on peut utiliser **predict( )** pour réaliser des prédictions pour les 11 composantes principales et comparés le résultat de MSE avec ceux des autres modèles :

```
> # prédiction sur les 11 composantes
> fitted_prc <- predict(reg_acp, ncomp = 11)
> # calul de la MSE = SSE/n regression sur CP
> mean((log(vehicles$price) - fitted_prc) ^ 2)
[1] 0.01026775
> # calcul de la MSE = SSE/n regression lasso
> mean((log(vehicles$price) - fitted_lasso) ^ 2)
[1] 0.01217806
> # calcul de la MSE = SSE/n regression ridge
> mean((log(vehicles$price) - fitted) ^ 2)
[1] 0.03131274
> # calcul de la MSE = SSE/n regression ordinaire
> mean((log(vehicles$price) - fitted(reg_lm)) ^ 2)
[1] 0.008963656
>
```

Exemple 277.: Régression (linéaire) par moindres carrés partiels (régression linéaire PLS univariée: PLS1)

R 3.0.2

Le but va être ici de vérifier si nous obtenons ou pas les résultats des calculs vus dans le cours théorique lors de la lecture du l'ouvrage de M. Tenenhaus[29] sur la régression PLS univariée (PLS1), c'est-à-dire la régression sur des variables explicatives corrélées avec une unique variable à expliquer.

Nous utiliserons les mêmes données que les calculs effectués à la main et que dans le cours Minitab:

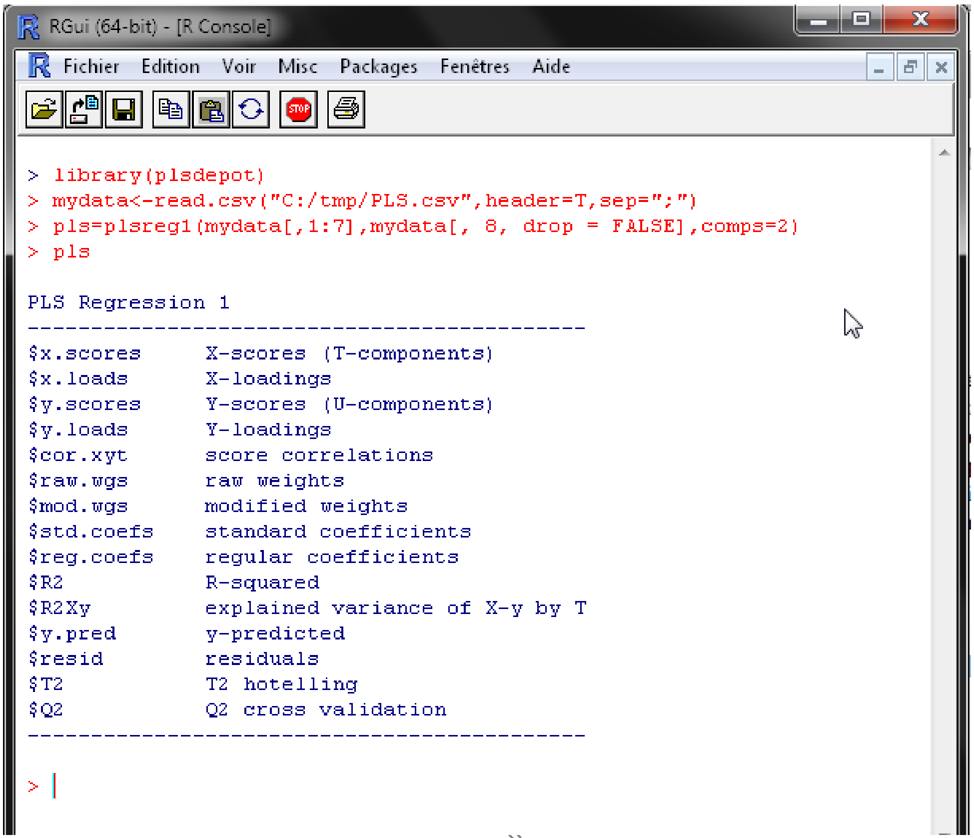
PLS.csv - Bloc-notes

Fichier Edition Format Affichage ?

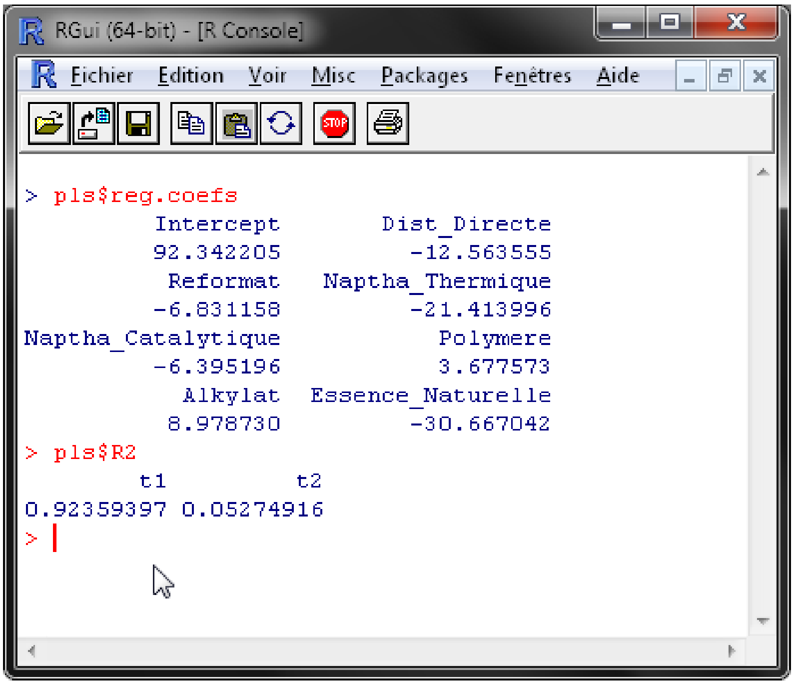
```
Dist_Directe;Reformat;Naptha_Thermique;Naptha_Catalytique;Polymere;Alkylat;Essence_Naturelle;Réponse
0.00;0.23;0.00;0.00;0.00;0.74;0.03;98.70
0.00;0.10;0.00;0.00;0.12;0.74;0.04;97.80
0.00;0.00;0.00;0.10;0.12;0.74;0.04;96.60
0.00;0.49;0.00;0.00;0.12;0.37;0.02;92.00
0.00;0.00;0.00;0.62;0.12;0.18;0.08;86.6
0.00;0.62;0.00;0.00;0.00;0.37;0.01;91.20
0.17;0.27;0.10;0.38;0.00;0.00;0.08;81.90
0.17;0.19;0.10;0.38;0.02;0.06;0.08;83.10
0.17;0.21;0.10;0.38;0.00;0.06;0.08;82.40
0.17;0.15;0.10;0.38;0.02;0.10;0.08;83.20
0.21;0.36;0.12;0.25;0.00;0.00;0.06;81.40
0.00;0.00;0.00;0.55;0.00;0.37;0.08;88.10
```

[29] Michel Tenenhaus, *Régression PLS, Édition Technip, ISBN 2-7108-0735-1*, Pages 75-83

Nous utilisons ensuite la fonction **plsreg1( )** du package **plsdepot** qui comme nous pouvons le voir ci-dessous nous laisse accès à un certain nombre d'information pertinentes:



Nous allons demander à voir celles qui correspondent aux calculs faits à la main dans le cours théorique et par Minitab:



Tout correspond bien!

Exemple 278.: Modélisation par équations structurelles (SEM)

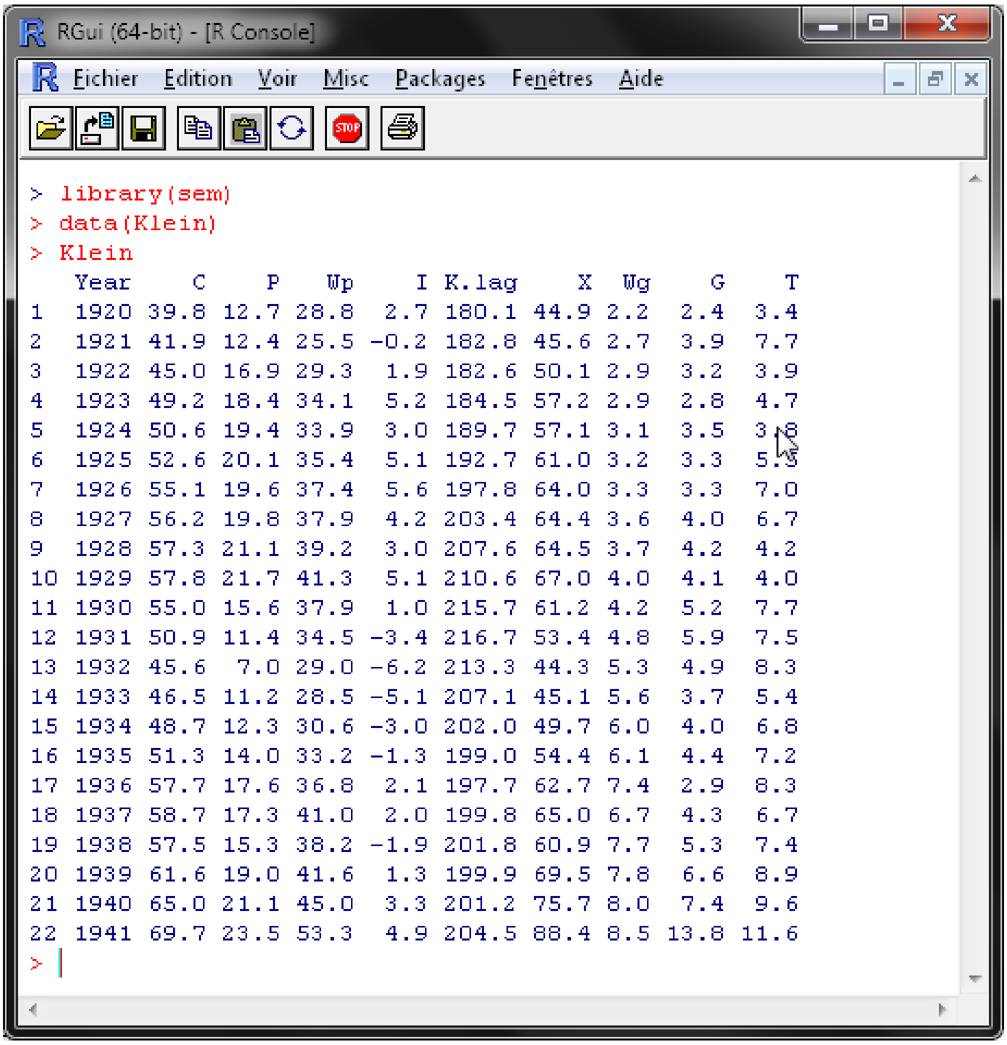
R 3.1.1

Il faut considérer cet exercice comme un brouillon à prendre avec des pincettes. Il existe effectivement de nombreuses méthodes numériques et plusieurs packages pour résoudre les SEM.

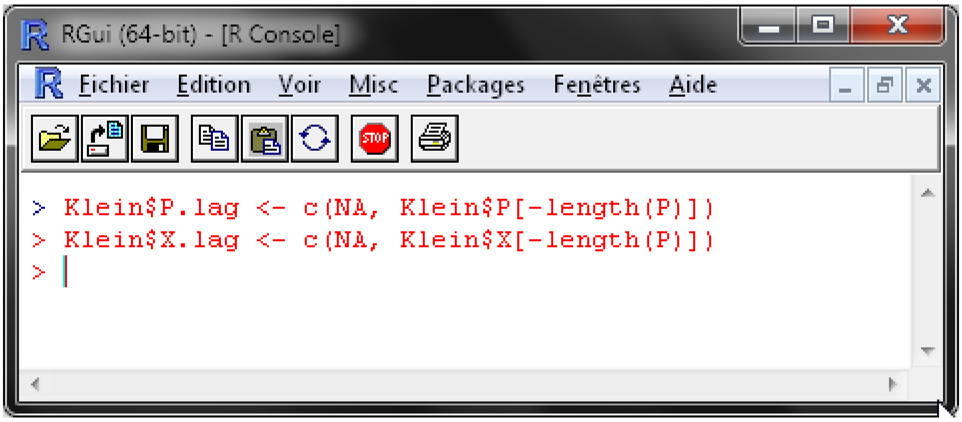
Le but ici va être de comparer les résultats des packages **sem** et **lavaan** par rapport au solveur Microsoft Excel en se basant sur le modèle de Klein qui pour rappel est:

$$\begin{aligned}
C_t &= \gamma_{10} + \gamma_{11}P_t + \gamma_{12}P_{t-1} + \beta_{11}\left(W_t^p + W_t^g\right) + \varepsilon_{1t} \\
I_t &= \gamma_{20} + \gamma_{21}P_t + \gamma_{22}P_{t-1} + \beta_{12}K_{t-1} + \varepsilon_{2t} \\
W_t^p &= \gamma_{30} + \gamma_{31}A_t + \beta_{31}X_t + \beta_{32}X_{t-1} + \varepsilon_{3t} \\
X_t &= C_t + I_t + G_t \\
P_t &= X_t - T_t - W_t^p \\
K_t &= K_{t-1} + I_t
\end{aligned}$$

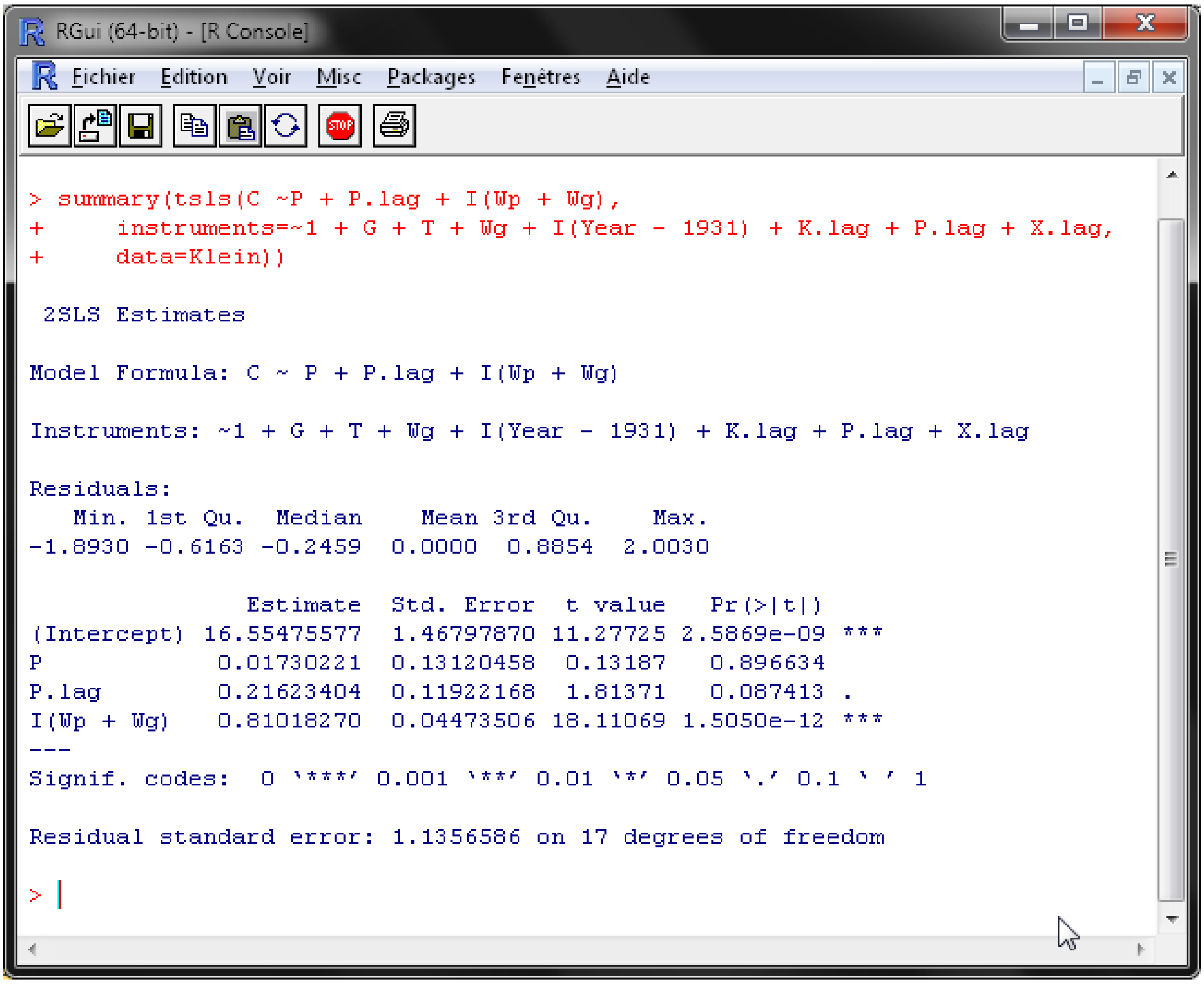
D'abord nous chargeons les données où la majorité des variables sont présentes:



Comme certaines variables décalées (lag) sont absentes, nous allons les créer:

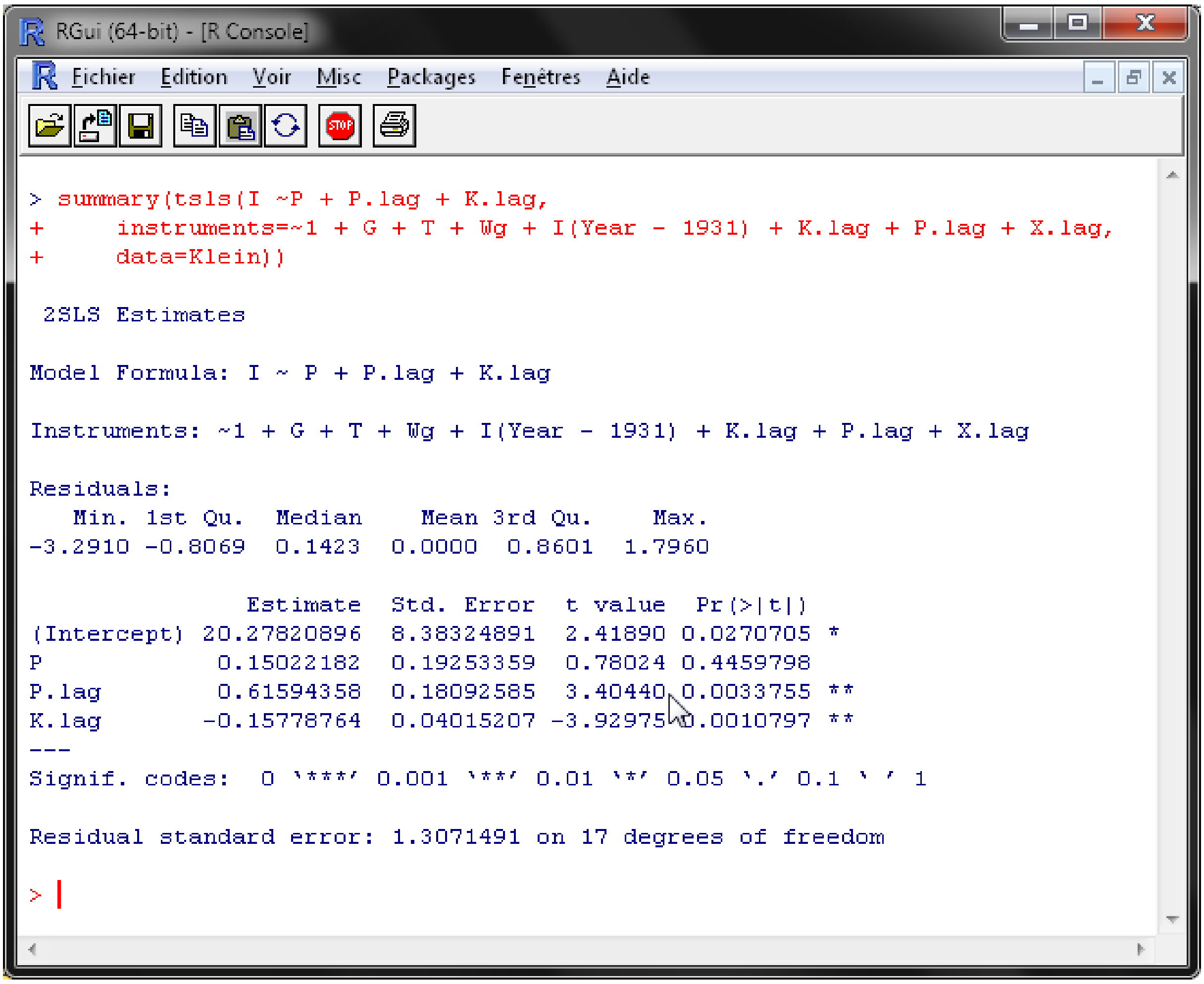


Nous résolvons maintenant la première équation structurelle en utilisant la fonction **tsls( )** du package **sem**:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> summary(tsls(C ~P + P.lag + I(Wp + Wg),
+ instruments=~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag,
+ data=Klein))
2SLS Estimates
Model Formula: C ~ P + P.lag + I(Wp + Wg)
Instruments: ~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag
Residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.8930 -0.6163 -0.2459 0.0000 0.8854 2.0030
Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.55475577 1.46797870 11.27725 2.5869e-09 ***
P 0.01730221 0.13120458 0.13187 0.896634
P.lag 0.21623404 0.11922168 1.81371 0.087413 .
I(Wp + Wg) 0.81018270 0.04473506 18.11069 1.5050e-12 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.1356586 on 17 degrees of freedom
>

La deuxième:



et enfin la troisième:

```
> summary(tsls(Wp ~X + X.lag + I(Year - 1931),
+     instruments=~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag,
+     data=Klein))

 2SLS Estimates

Model Formula: Wp ~ X + X.lag + I(Year - 1931)

Instruments: ~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag

Residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.2940 -0.4734  0.0145  0.0000  0.4487  1.1960

                 Estimate Std. Error  t value   Pr(>|t|)
(Intercept)    1.50029689 1.27568637  1.17607 0.25577411
X              0.43885907 0.03960266 11.08155 3.3679e-09 ***
X.lag          0.14667382 0.04316395  3.39806 0.00342209 **
I(Year - 1931) 0.13039569 0.03238839  4.02600 0.00087642 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

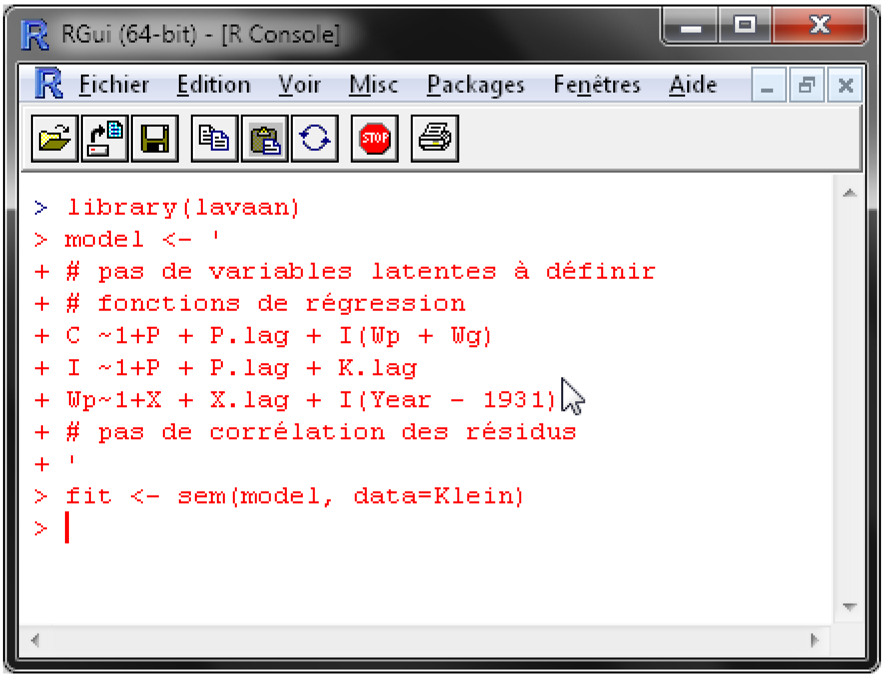
Residual standard error: 0.7671553 on 17 degrees of freedom

> 
```

Nous mettons les valeurs d'intérêt les unes à côté des autres:

```
               Estimate                   Estimate                      Estimate
(Intercept) 16.55475577  (Intercept) 20.27820896  (Intercept)          1.50029689
P            0.01730221  P            0.15022182  X                    0.43885907
P.lag        0.21623404  P.lag        0.61594358  X.lag                0.14667382
I(Wp + Wg)   0.81018270  K.lag       -0.15778764  I(Year - 1931)       0.13039569
```

Maintenant procédons de même avec la fonction **sem( )** du package **lavaan** qui est bien plus complet en paramètres que le package **sem**:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(lavaan)
> model <- '
+ # pas de variables latentes à définir
+ # fonctions de régression
+ C ~1+P + P.lag + I(Wp + Wg)
+ I ~1+P + P.lag + K.lag
+ Wp~1+X + X.lag + I(Year - 1931)
+ # pas de corrélation des résidus
+ '
> fit <- sem(model, data=Klein)
>

Ce qui donne:

```
> summary(fit, fit.measures=TRUE)
lavaan (0.5-18) converged normally after  85 iterations

                                                  Used       Total
  Number of observations                            21          22

  Estimator                                         ML
  Minimum Function Test Statistic               63.272
  Degrees of freedom                                12
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              297.300
  Degrees of freedom                                24
  P-value                                        0.000

User model versus baseline model:

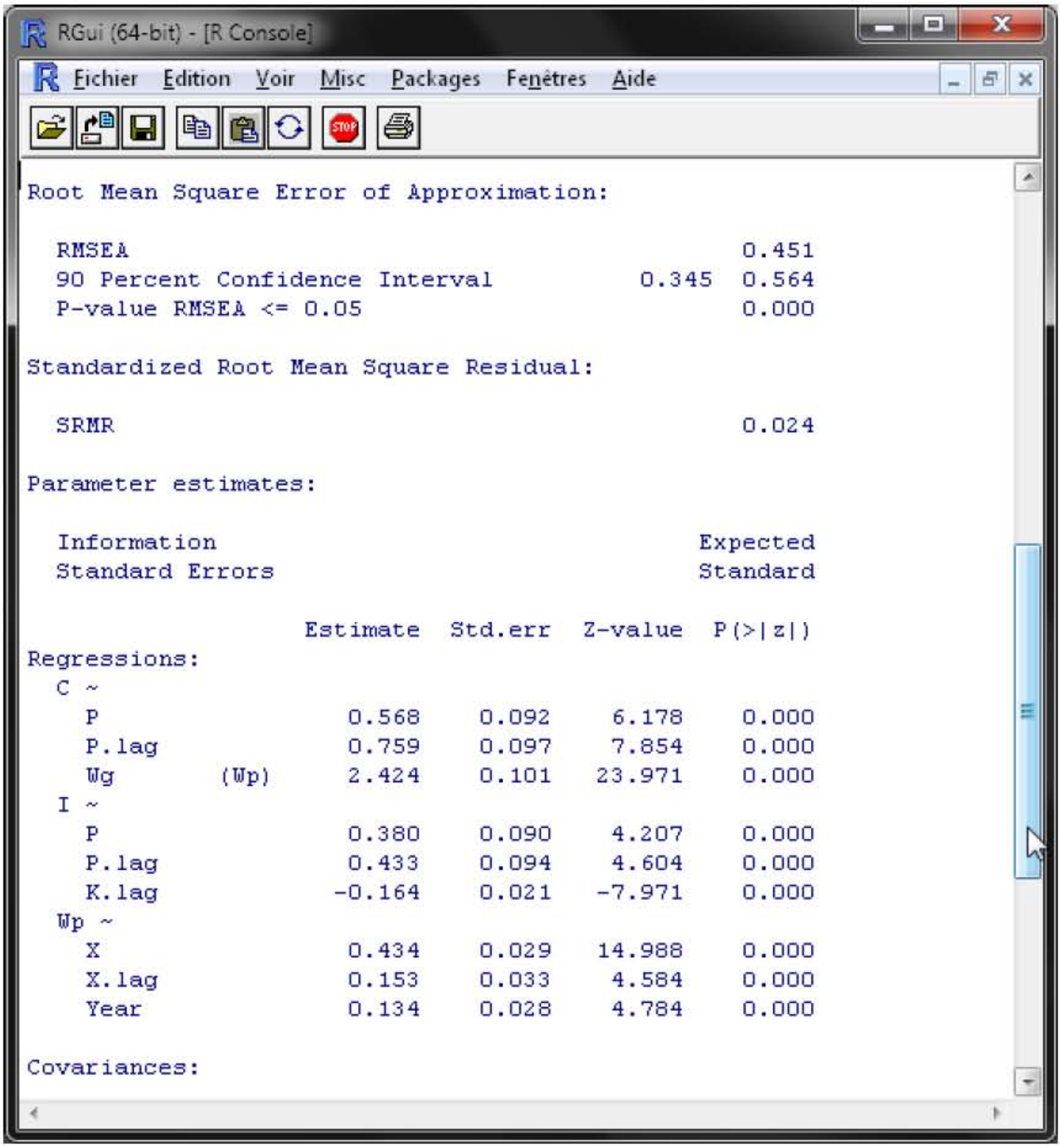
  Comparative Fit Index (CFI)                    0.812
  Tucker-Lewis Index (TLI)                       0.625

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)               -406.551
  Loglikelihood unrestricted model (H1)       -374.915

  Number of free parameters                         18
  Akaike (AIC)                                 849.103
  Bayesian (BIC)                               867.904
  Sample-size adjusted Bayesian (BIC)          812.337
```

et pour la suite:



```
Root Mean Square Error of Approximation:

  RMSEA                                         0.451
  90 Percent Confidence Interval         0.345  0.564
  P-value RMSEA <= 0.05                         0.000

Standardized Root Mean Square Residual:

  SRMR                                          0.024

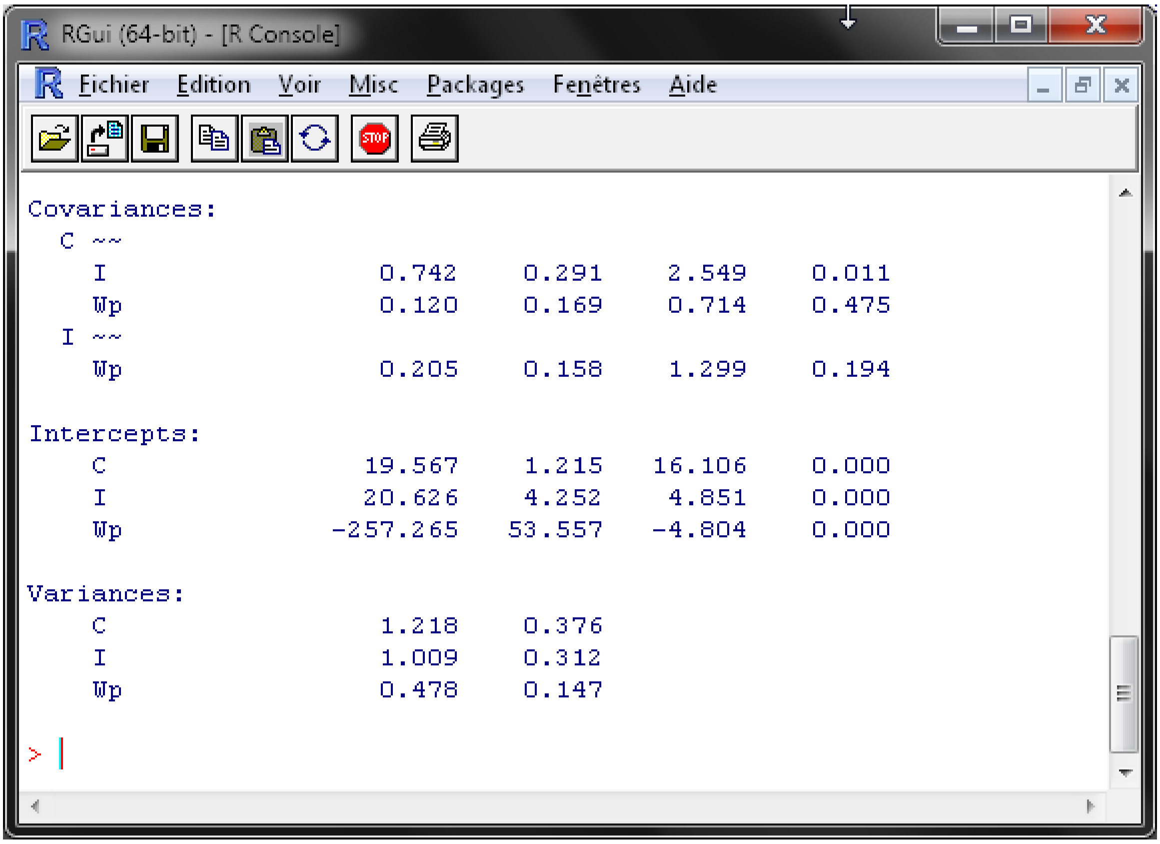
Parameter estimates:

  Information                                Expected
  Standard Errors                            Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  C ~
    P                 0.568    0.092    6.178    0.000
    P.lag             0.759    0.097    7.854    0.000
    Wg        (Wp)    2.424    0.101   23.971    0.000
  I ~
    P                 0.380    0.090    4.207    0.000
    P.lag             0.433    0.094    4.604    0.000
    K.lag            -0.164    0.021   -7.971    0.000
  Wp ~
    X                 0.434    0.029   14.988    0.000
    X.lag             0.153    0.033    4.584    0.000
    Year              0.134    0.028    4.784    0.000

Covariances:
```

et enfin:



Soit pour résumer les paramètres qui nous intéressent:

```
                   Estimate
Regressions:
  C ~
    P                 0.568
    P.lag             0.759
    Wg        (Wp)    2.424
  I ~
    P                 0.380
    P.lag             0.433
    K.lag            -0.164
  Wp ~                       Intercepts:
    X                 0.434      C                19.567
    X.lag             0.153      I                20.626
    Year              0.134      Wp             -257.265
```

Nous pouvons donc constater des différences considérables entre les deux méthodes dont voici un résumé et une comparaison avec MS Excel:

Coefficient	**sem**	**lavaan**	**Solveur XL**
γ_{10}	16.55475577	19.567	15.1386
γ_{11}	0.01730221	0.568	0.052837
γ_{12}	0.21623404	0.759	0.258205
β_{11}	0.08108270	2.424	0.811929
γ_{20}	20.27820896	20.626	-4.10789
γ_{21}	0.15022183	0.380	0.580019
γ_{22}	0.61593458	0.433	0.222549
β_{12}	-0.15778764	-0.164	-0.0406
γ_{30}	1.50029689	-257.265	4.801374
γ_{31}	0.43885907	0.434	0.140841
β_{31}	0.14557382	0.153	0.531118
β_{32}	0.13039569	0.134	-0.00441
SCE somme des carrées des erreurs	30827.9981	9208909.048	63.2049

Je pense que cela se passe de tout commentaire. Il serait intéressant de comparer avec un logiciel payant (je le ferai quand j'en aurai un sous la main).

Exemple 279.: Régression Bootstrap

R 3.5.1

Voir la page 1975.

Exemple 280.: Régression Jacknife

R 3.5.1

Voir la page 1981.

Exemple 281.: Modélisation Dose-Réponse

R 3.6.3

Ce qui suit est entièrement inspiré du blog de Claire Della Vedova:

https://statistique-et-logiciel-r.com/dose-reponse/

Il s'agit d'appliquer les techniques habituelles de régressions à des modèles particuliers connus dans la recherche médicale et biologique.

Nous allons ici utiliser le package **drc** (Dose-Response Curves) auquel un livre entier est dédié (nous mentionnerons le livre à la fin de cet exemple) et un jeu de donnés fournit avec, nommé *ryegrass*, qui sont des données issues d'une étude visant à étudier l'action conjointe des acides phénoliques sur l'inhibition de la croissance des racines de la plante Lolium perenne. La variable *conc* est la concentration d'acide férulique en [mM] (millimolar), 6 niveaux sont testés. La variable *rootl* est la longueur des racines de la plante, mesurée en [cm]:

```
> library("drc")
Loading required package: MASS

'drc' has been loaded.

Please cite R and 'drc' if used for a publication,
for references type 'citation()' and 'citation('drc')'.

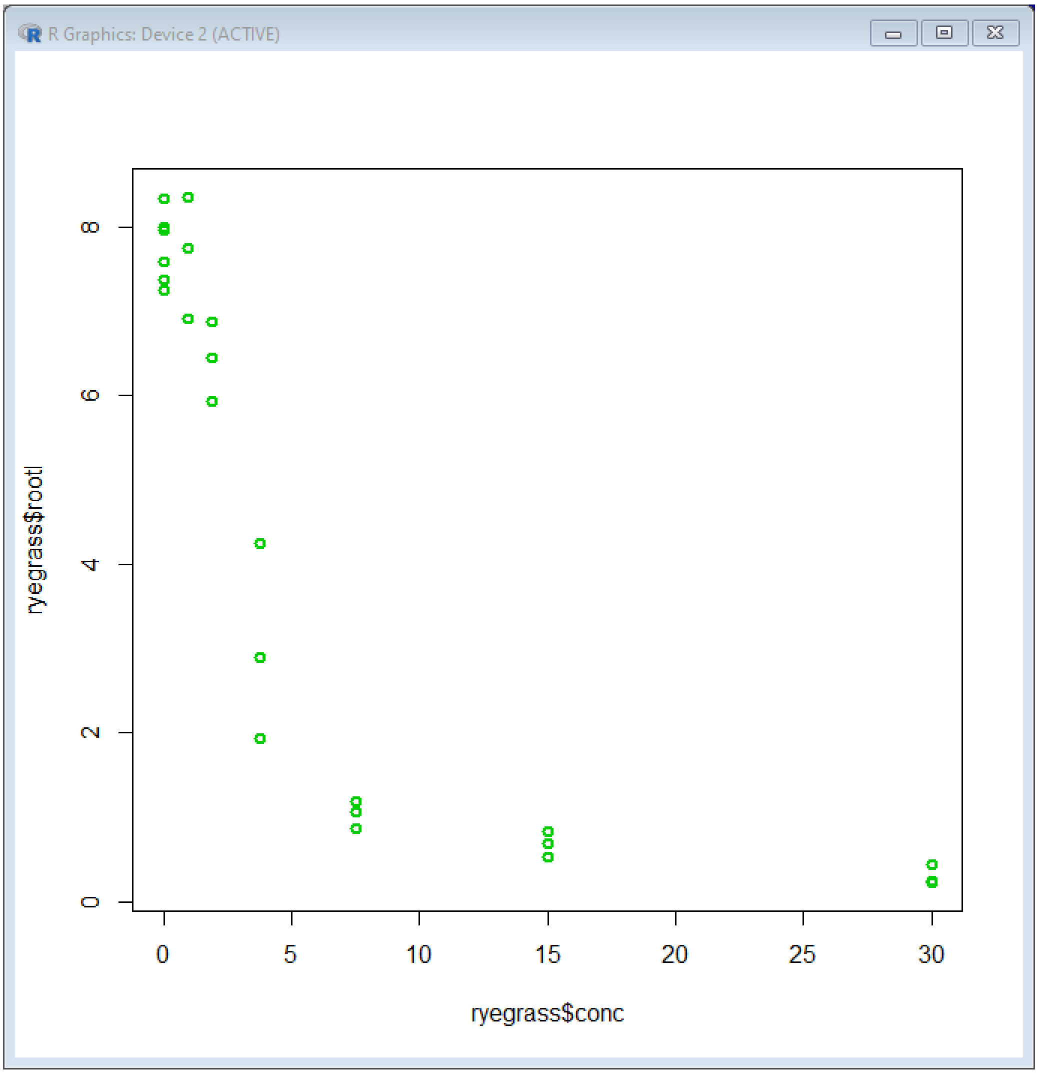

Attaching package: 'drc'

The following objects are masked from 'package:stats':

    gaussian, getInitial

> ryegrass
       rootl  conc
1  7.5800000  0.00
2  8.0000000  0.00
3  8.3285714  0.00
4  7.2500000  0.00
5  7.3750000  0.00
6  7.9625000  0.00
7  8.3555556  0.94
8  6.9142857  0.94
9  7.7500000  0.94
10 6.8714286  1.88
11 6.4500000  1.88
12 5.9222222  1.88
13 1.9250000  3.75
14 2.8857143  3.75
15 4.2333333  3.75
16 1.1875000  7.50
17 0.8571429  7.50
18 1.0571429  7.50
19 0.6875000 15.00
20 0.5250000 15.00
21 0.8250000 15.00
22 0.2500000 30.00
23 0.2200000 30.00
24 0.4400000 30.00
> plot(ryegrass$conc,ryegrass$rootl,lwd=2,col=3)
> 
```

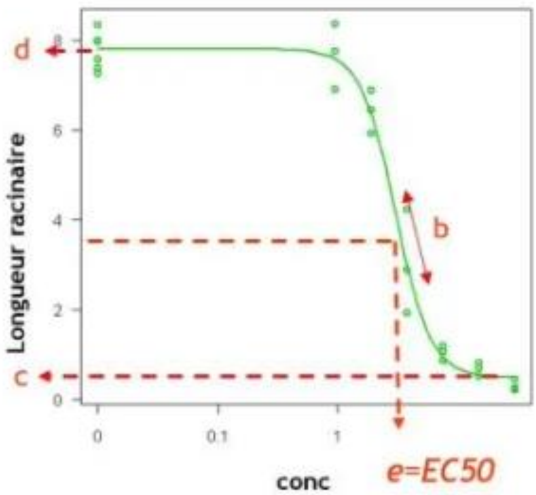
Avec le graphique correspondant:



Ensuite rappelons les modèles courants:

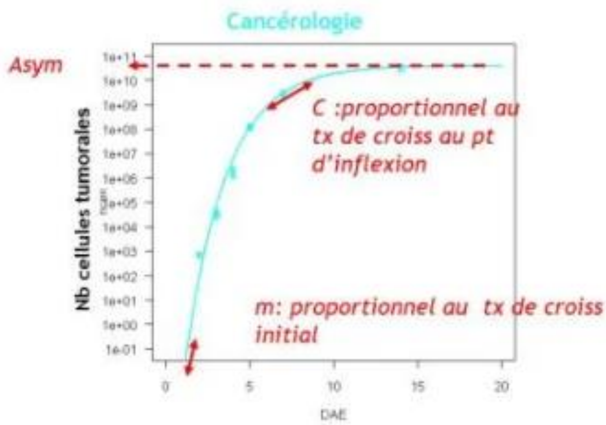
Log-logistique model:

$$f(x) = c + \frac{(d-c)}{[1+\exp(b(\log(x)-\log(e))]}$$



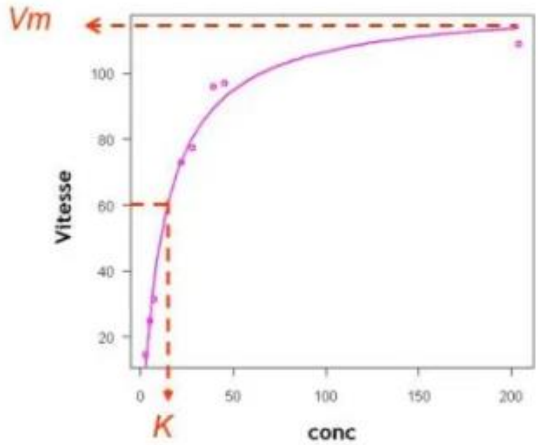
Gompertz model

$$f(x) = Asym \times \exp(-(m/c) \times \exp(-c \times x))$$



Mikaélis-Menten model

$$f(x) = \frac{Vm \times x}{K + x}$$



Allons-y pour l'ajustement du modèle log-logistique à 4 paramètres:

```
> ryegrass.ll4 <- drm(rootl~conc, data=ryegrass, fct=LL.4())
> summary(ryegrass.ll4)

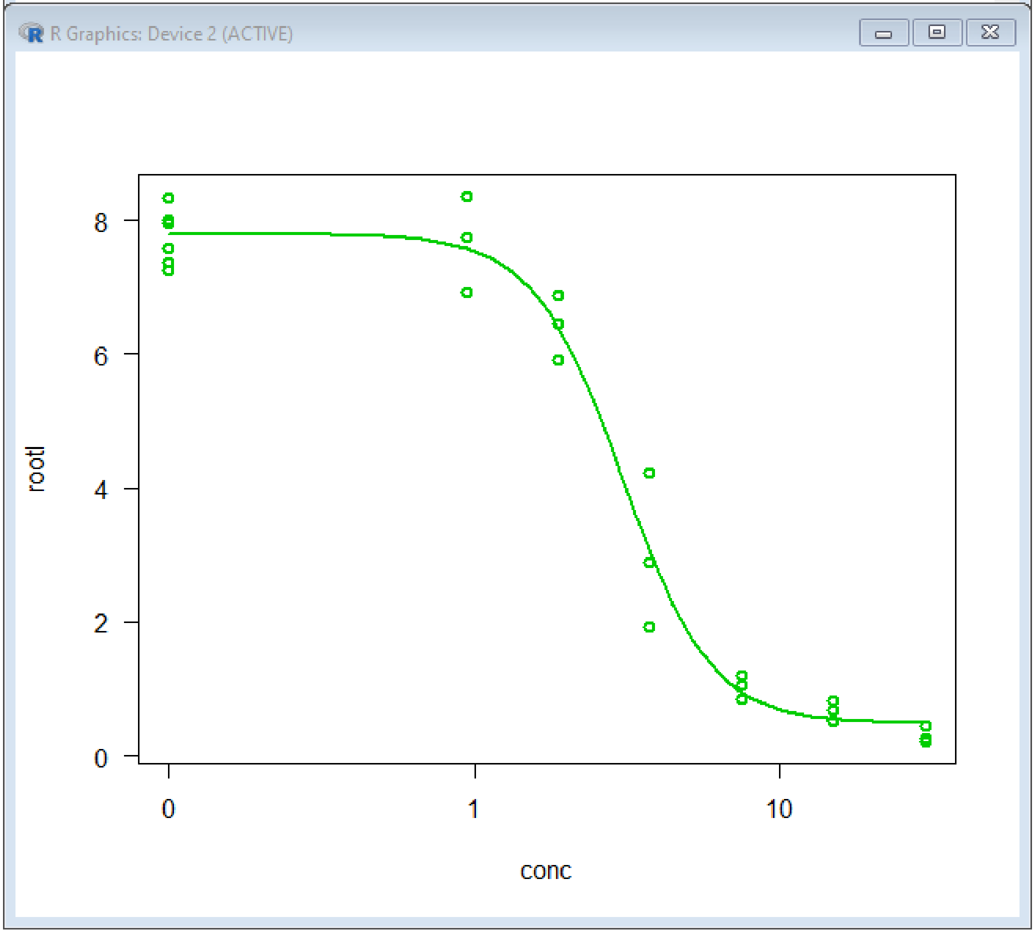
Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

              Estimate Std. Error t-value   p-value
b:(Intercept)  2.98222    0.46506  6.4125 2.960e-06
c:(Intercept)  0.48141    0.21219  2.2688   0.03451
d:(Intercept)  7.79296    0.18857 41.3272 < 2.2e-16
e:(Intercept)  3.05795    0.18573 16.4644 4.268e-13

Residual standard error:

 0.5196256 (20 degrees of freedom)
> plot(ryegrass.ll4,type="all",col=3,lwd=2)
> 
```



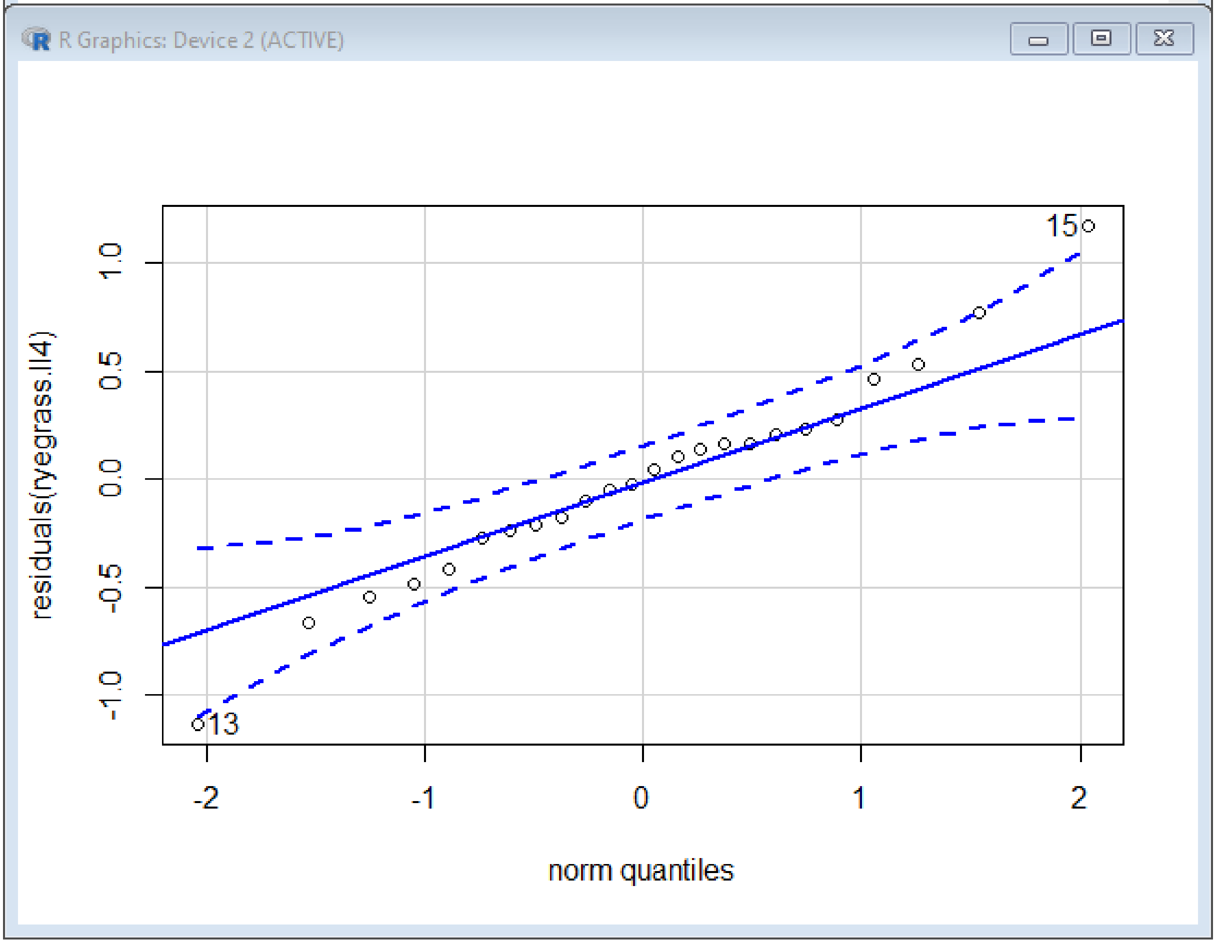
Évaluatons la normalité des résidus:

```
> library("car")
Loading required package: carData
> car::qqPlot(residuals(ryegrass.l14))
[1] 15 13
> shapiro.test(residuals(ryegrass.l14))

        Shapiro-Wilk normality test

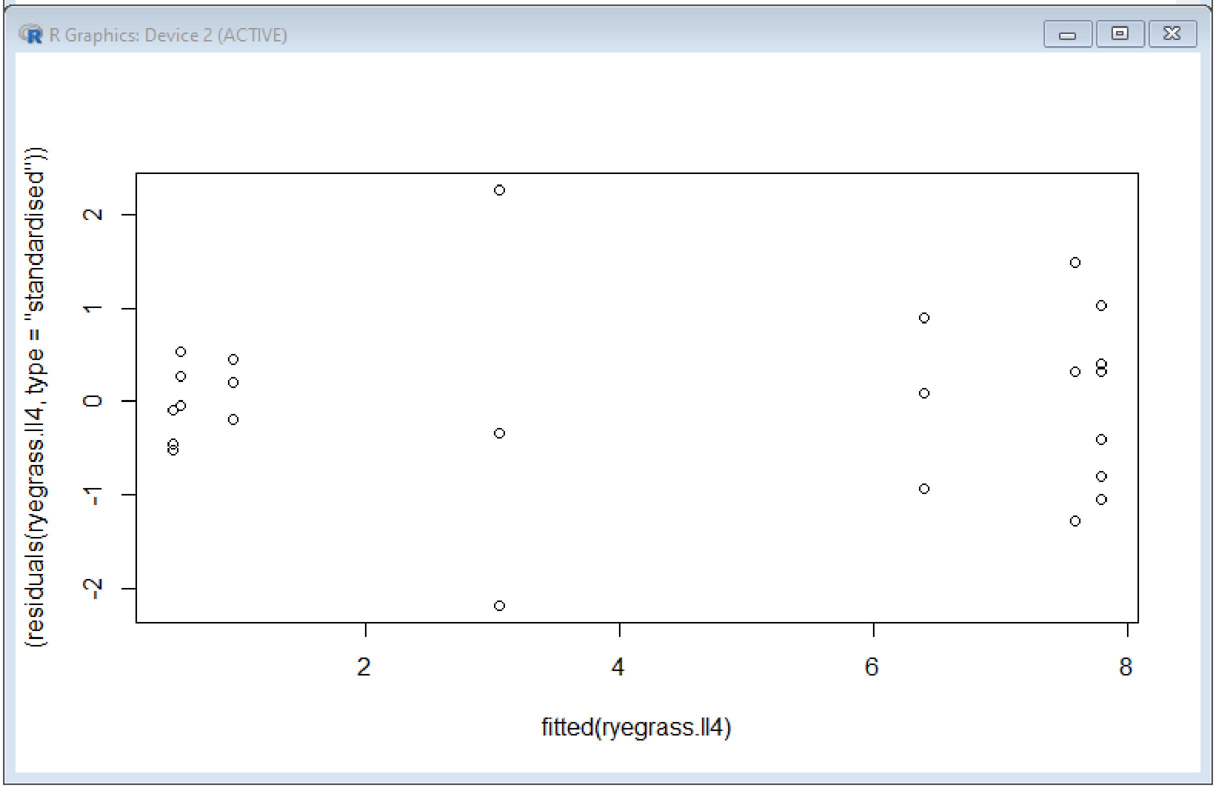
data:  residuals(ryegrass.l14)
W = 0.98232, p-value = 0.9345

>
```



C'est a posteriori bon signe. Maintenant évaluatons l'homogénéité des résidus:

```
> plot(fitted(ryegrass.ll4),(residuals(ryegrass.ll4,type="standardised")))
>
```



L'homogénéité semble plutôt rejetée.

Les tests de Levene et de Bartlett peuvent également être employés. L'hypothèse nulle de ces tests spécifie l'homogénéité des résidus, alors que l'hypothèse alternative spécifie la non-homogénéité:

```
> car::leveneTest(residuals(ryegrass.ll4),as.factor(ryegrass$conc))
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  6  1.9266 0.1344
      17
> bartlett.test(residuals(ryegrass.ll4),as.factor(ryegrass$conc))

        Bartlett test of homogeneity of variances

data:  residuals(ryegrass.ll4) and as.factor(ryegrass$conc)
Bartlett's K-squared = 13.216, df = 6, p-value = 0.03973

>
```

Les résultats de ces tests sont contradictoires. Au final, nous pouvons retenir la présence d'un défaut d'homogénéité des résidus.

Utilisons alors la transformation de Box-Cox:

```
> ryegrass.ll4BC<-boxcox(ryegrass.ll4)
Warning messages:
1: In sqrt(diag(varMat)) : NaNs produced
2: In sqrt(diag(varMat)) : NaNs produced
3: In sqrt(diag(varMat)) : NaNs produced
4: In sqrt(diag(varMat)) : NaNs produced
> summary(ryegrass.ll4BC)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

              Estimate Std. Error t-value   p-value
b:(Intercept)  2.61839    0.39151  6.6880 1.649e-06
c:(Intercept)  0.39083    0.10429  3.7474  0.001269
d:(Intercept)  7.86633    0.29558 26.6136 < 2.2e-16
e:(Intercept)  3.01662    0.21005 14.3612 5.354e-12

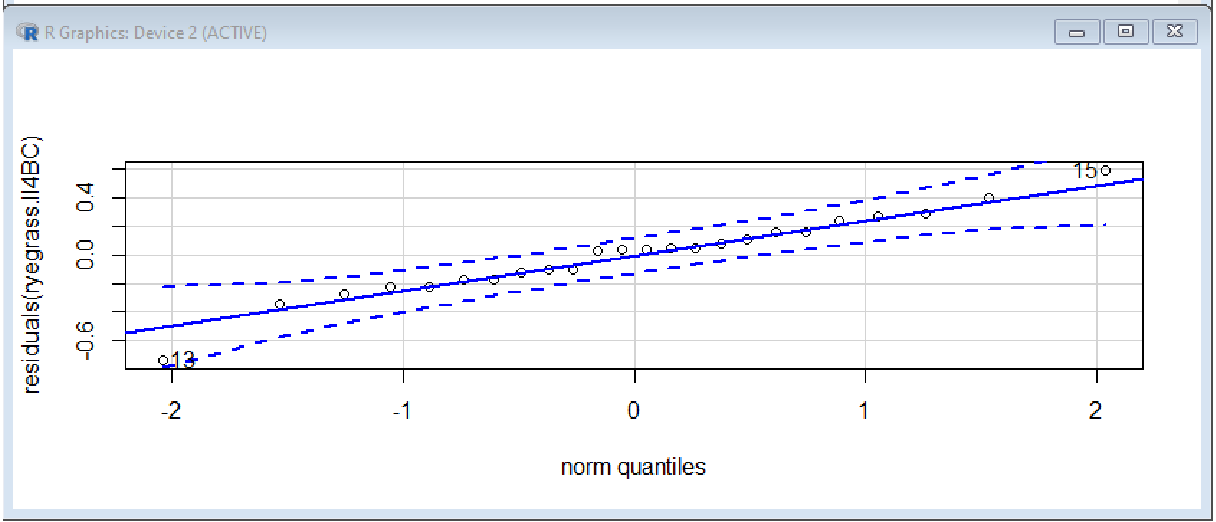
Residual standard error:

 0.2962958 (20 degrees of freedom)

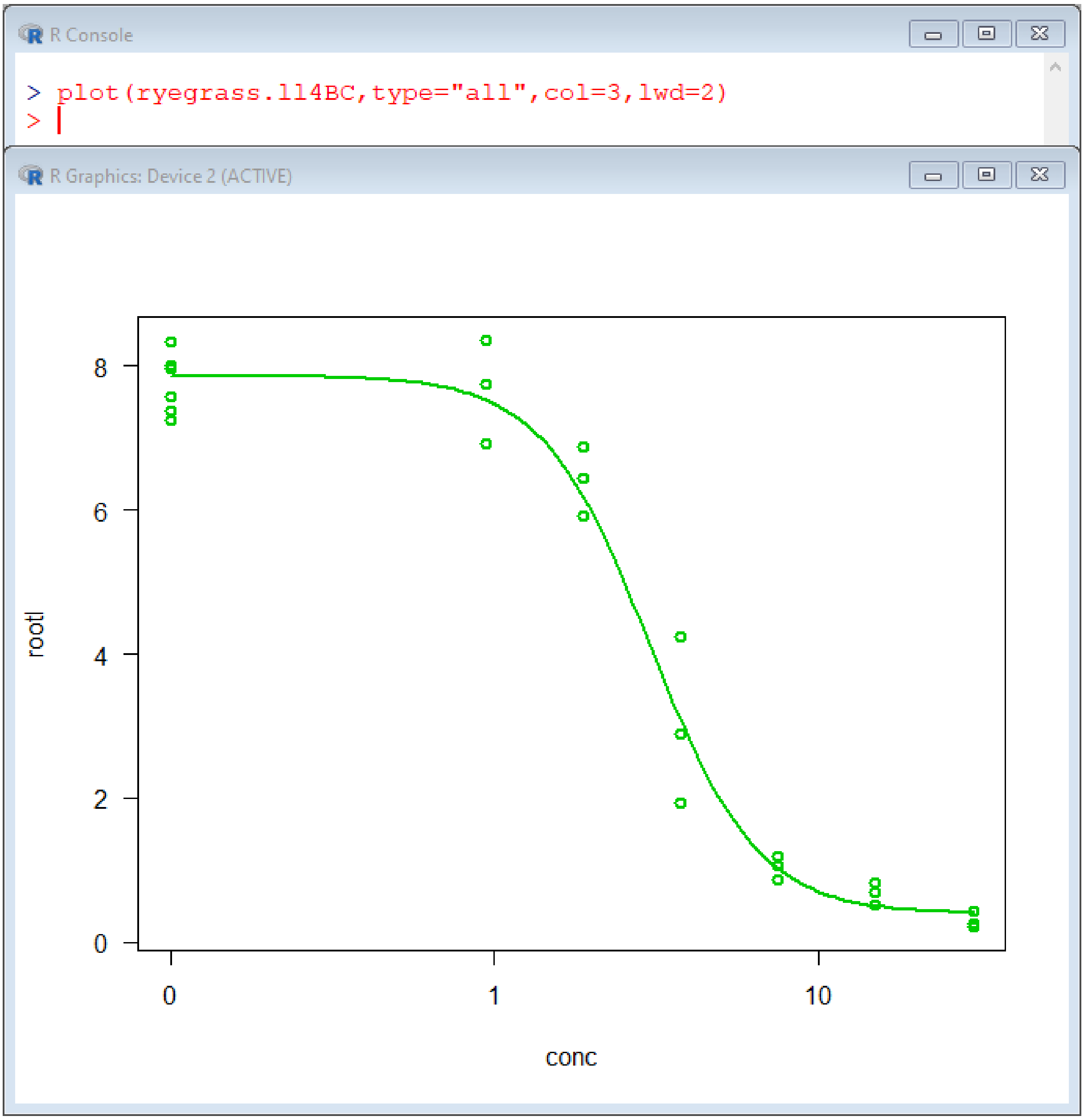
Non-normality/heterogeneity adjustment through optimal Box-Cox transformation

Estimated lambda: 0.5
Confidence interval for lambda: [0.269,0.949]

> qqPlot(residuals(ryegrass.ll4BC))
[1] 13 15
>
```



Et le modèle donne:



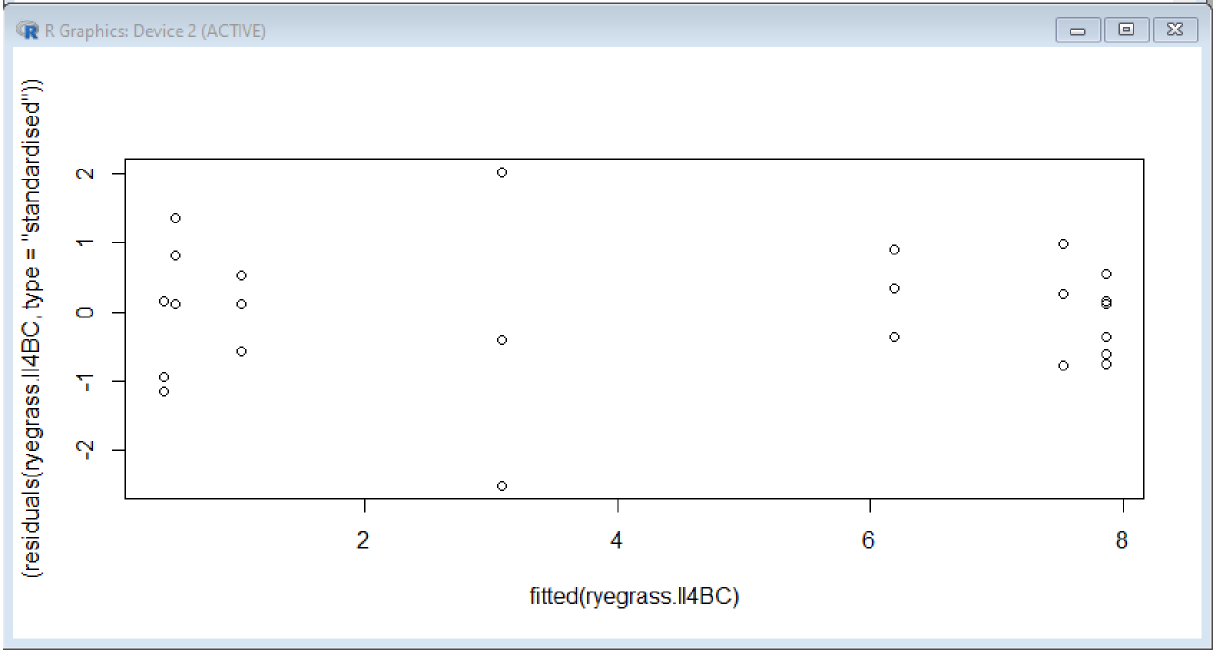
Donc la normalit'e des résides semble toujours ok. Maintenant vérifions leur homogénéité:

```
> plot(fitted(ryegrass.ll4BC),(residuals(ryegrass.ll4BC,type="standardised")))
> bartlett.test(residuals(ryegrass.ll4BC),as.factor(ryegrass$conc))

        Bartlett test of homogeneity of variances

data:  residuals(ryegrass.ll4BC) and as.factor(ryegrass$conc)
Bartlett's K-squared = 9.1898, df = 6, p-value = 0.1632

> leveneTest(residuals(ryegrass.ll4),as.factor(ryegrass$conc))
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  6  1.9266 0.1344
      17
>
```



Maintenant c'est Ok!

Faisons maintenant un test d'ajustement de Fisher (lack-of-fit) :

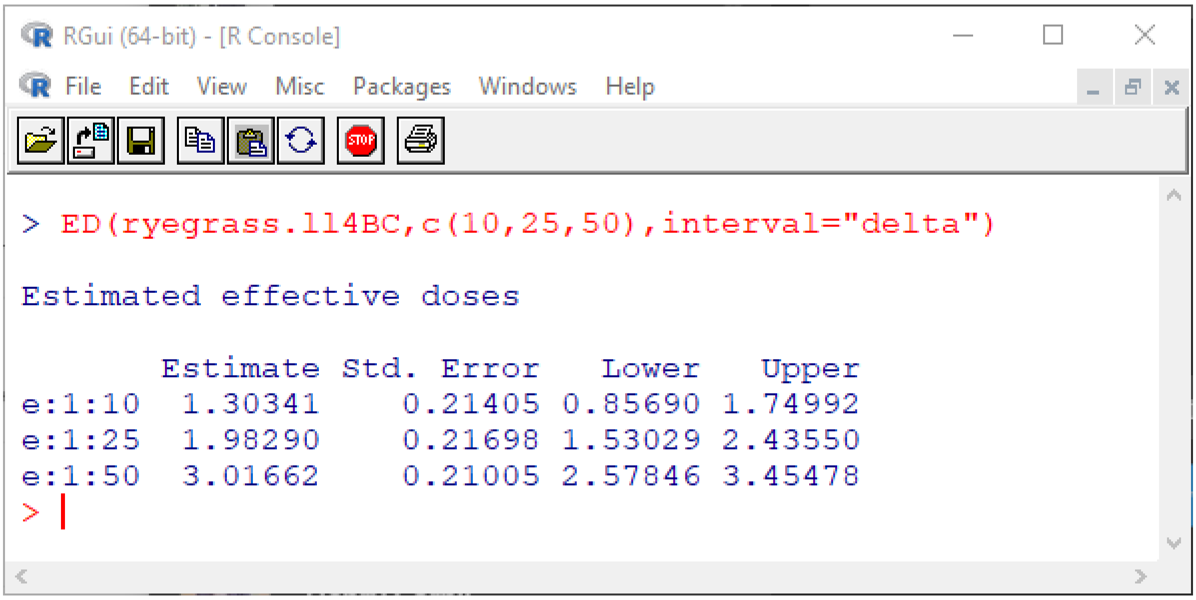
```
> modelFit(ryegrass.ll4BC)
Lack-of-fit test

          ModelDf    RSS Df F value p value
ANOVA          17 1.4292
DRC model      20 1.7558  3  1.2949  0.3084
>
```

C'est bon signe!

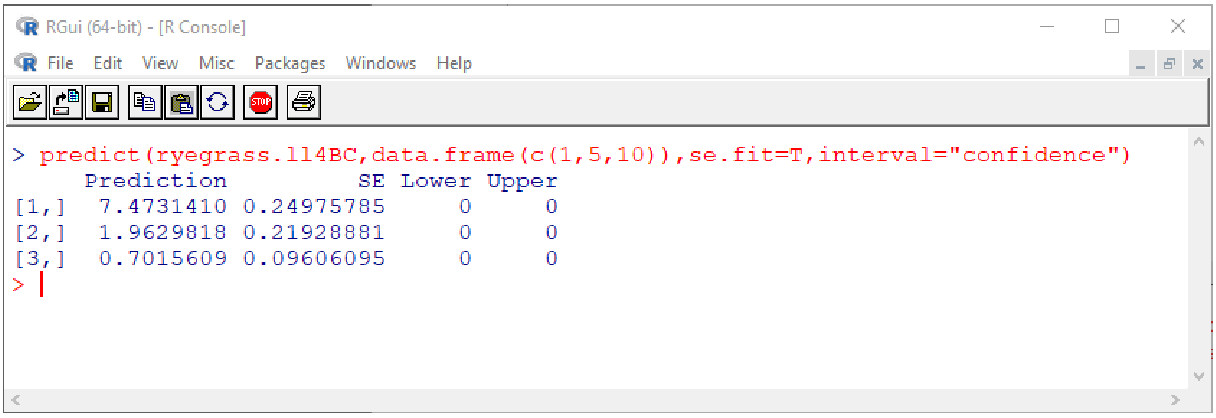
Pour conclure faisons un peu de prédictions.

D'abord faisons des estimations de la dose correspondant à une réponse (longueur de racine) donnée:



Objectivement je suis surpris par la lecture de ce résultat. Car plus la dose est grande, plus la longueur des racines est petite normalement. Il faut que je creuse le sujet pour voir si quelque chose m'a échappé ou pas.

Et pour terminer l'estimation de la réponse correspondant à une dose donnée:



16. Interpolations

Les techniques d'interpolation sont contrairement aux techniques de régression des outils mathématiques permettant d'ajuster au mois un courbe selon plusieurs critères empiriques (comme pour les régressions) mais dont le but n'est pas ensuite de pouvoir faire des projections (extrapolations) en avant et en arrière.

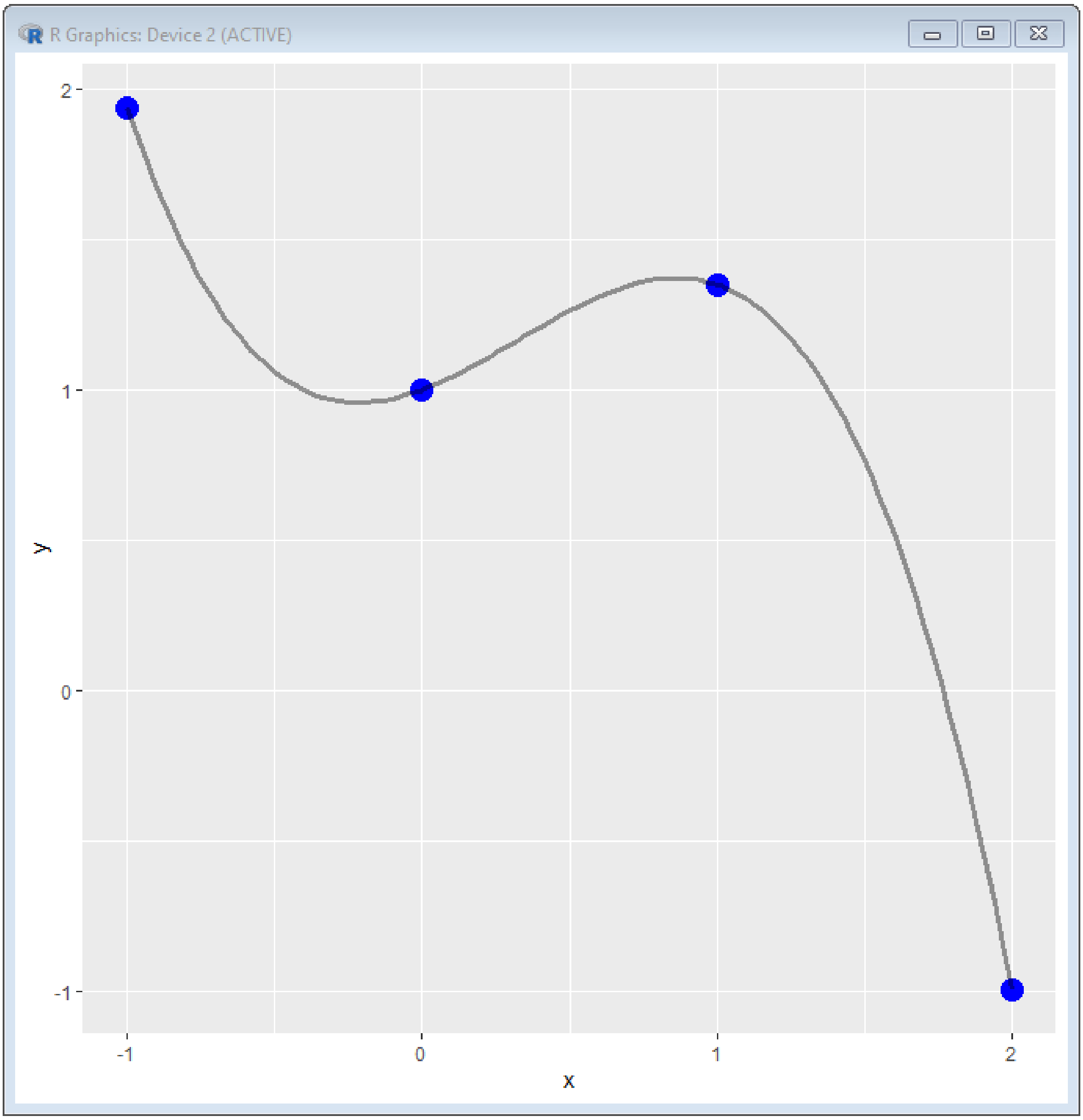
Exemple 282.: Interpolation par polynômes de Lagrange

R 3.6.2

Nous avons introduit et démontré dans le cours théorique, l'origine des polynômes de Taylor, Lagrange, Newton et Hermite. Voyons comment avec le package **polynom** et sa fonction **poly.calc( )** faire la même interpolation de Lagrange que celle calculée à la main dans le cours théorique:

```
> x <- c(-1, 0, 1, 2)
> y <- c(1.937, 1.000, 1.349, -0.995)
> dat <- data.frame(cbind(x, y))
>
> library("polynom")
> library("ggplot2")
> poly.calc(x, y)
1 + 0.3691667*x + 0.643*x^2 - 0.6631667*x^3
>
> f <- function(x) {
+   return(1 + 0.36916*x + 0.643*x^2 - 0.66316*x^3)
+ }
>
> ggplot(dat, aes(x=x, y=y)) +
+   geom_point(size=5, col='blue') +
+   stat_function(fun = f, size=1.25, alpha=0.4)
> 
```

Ce qui donne graphiquement:



Nous retrouvons les résultats identiques que ceux dans le cours théorique.

Exemple 283.: Interpolation par splines (basis + natural splines)

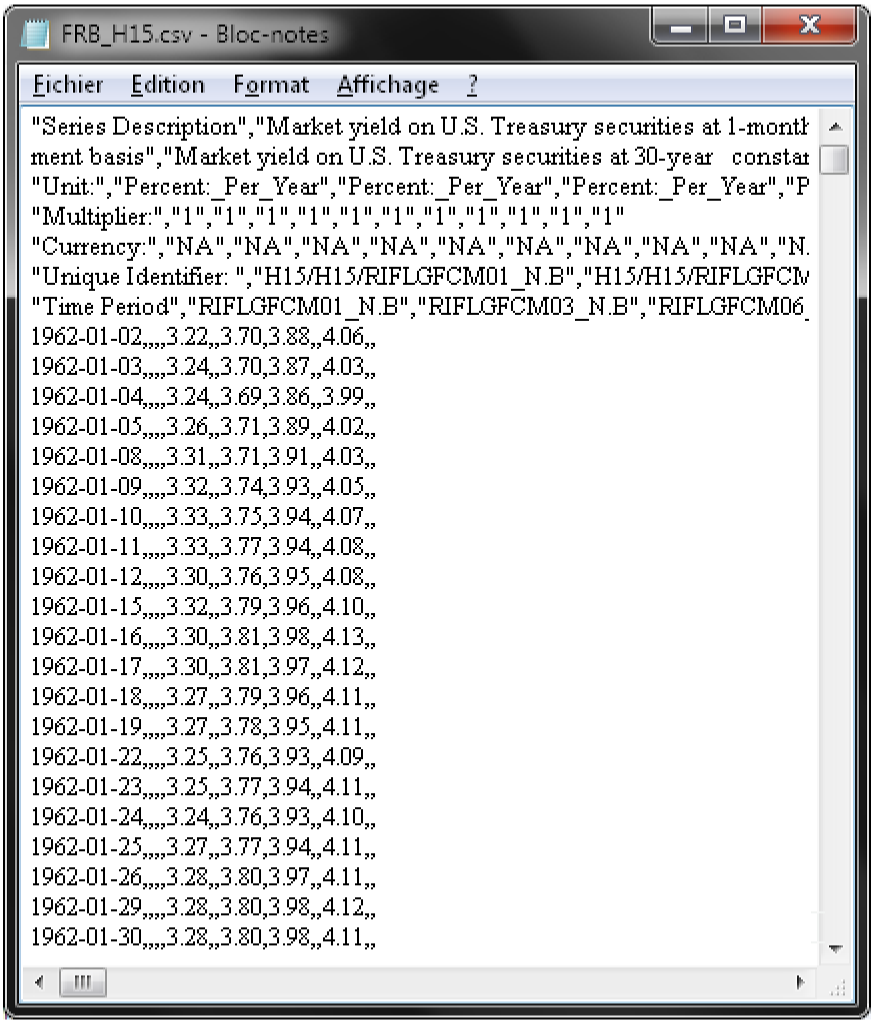
R 3.0.2

Les cas les plus courants d'interpolation sont les splines qui sont utilisées pour les courbes de taux en finance. Voyons donc un exemple de cela.

Pour l'exemple nous prenons les données de la banque fédérale américaine:

http://www.federalreserve.gov/datadownload/Choose.aspx?rel=H.15

téléchargeable en fichier *.csv:


FRB_H15.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
"Series Description","Market yield on U.S. Treasury securities at 1-montł
ment basis","Market yield on U.S. Treasury securities at 30-year   constaı
"Unit:","Percent:_Per_Year","Percent:_Per_Year","Percent:_Per_Year","P
"Multiplier:","1","1","1","1","1","1","1","1","1","1","1"
"Currency:","NA","NA","NA","NA","NA","NA","NA","NA","NA","N.
"Unique Identifier: ","H15/H15/RIFLGFCM01_N.B","H15/H15/RIFLGFCM
"Time Period","RIFLGFCM01_N.B","RIFLGFCM03_N.B","RIFLGFCM06_
1962-01-02,,,,3.22,,3.70,3.88,,4.06,,
1962-01-03,,,,3.24,,3.70,3.87,,4.03,,
1962-01-04,,,,3.24,,3.69,3.86,,3.99,,
1962-01-05,,,,3.26,,3.71,3.89,,4.02,,
1962-01-08,,,,3.31,,3.71,3.91,,4.03,,
1962-01-09,,,,3.32,,3.74,3.93,,4.05,,
1962-01-10,,,,3.33,,3.75,3.94,,4.07,,
1962-01-11,,,,3.33,,3.77,3.94,,4.08,,
1962-01-12,,,,3.30,,3.76,3.95,,4.08,,
1962-01-15,,,,3.32,,3.79,3.96,,4.10,,
1962-01-16,,,,3.30,,3.81,3.98,,4.13,,
1962-01-17,,,,3.30,,3.81,3.97,,4.12,,
1962-01-18,,,,3.27,,3.79,3.96,,4.11,,
1962-01-19,,,,3.27,,3.78,3.95,,4.11,,
1962-01-22,,,,3.25,,3.76,3.93,,4.09,,
1962-01-23,,,,3.25,,3.77,3.94,,4.11,,
1962-01-24,,,,3.24,,3.76,3.93,,4.10,,
1962-01-25,,,,3.27,,3.77,3.94,,4.11,,
1962-01-26,,,,3.28,,3.80,3.97,,4.11,,
1962-01-29,,,,3.28,,3.80,3.98,,4.12,,
1962-01-30,,,,3.28,,3.80,3.98,,4.11,,
```

Nous avons alors avec beaucoup d'astuce d'abord en prenant la dernière date:

```
R Console
> mydata<-read.csv("C:/FRB_H15.csv",header=T,sep=",")
> mydata<-tail(mydata,1)
> mydata
      Series.Description
13672         2014-05-21
      Market.yield.on.U.S..Treasury.securities.at.1.month...constant.maturity..quoted.on.investment.basis
13672                                                                                                0.04
      Market.yield.on.U.S..Treasury.securities.at.3.month...constant.maturity..quoted.on.investment.basis
13672                                                                                                0.04
      Market.yield.on.U.S..Treasury.securities.at.6.month...constant.maturity..quoted.on.investment.basis
13672                                                                                                0.06
      Market.yield.on.U.S..Treasury.securities.at.1.year...constant.maturity..quoted.on.investment.basis
13672                                                                                               0.09
      Market.yield.on.U.S..Treasury.securities.at.2.year...constant.maturity..quoted.on.investment.basis
13672                                                                                               0.37
      Market.yield.on.U.S..Treasury.securities.at.3.year...constant.maturity..quoted.on.investment.basis
13672                                                                                               0.79
      Market.yield.on.U.S..Treasury.securities.at.5.year...constant.maturity..quoted.on.investment.basis
13672                                                                                               1.55
      Market.yield.on.U.S..Treasury.securities.at.7.year...constant.maturity..quoted.on.investment.basis
13672                                                                                               2.08
      Market.yield.on.U.S..Treasury.securities.at.10.year...constant.maturity..quoted.on.investment.basis
13672                                                                                                2.54
      Market.yield.on.U.S..Treasury.securities.at.20.year...constant.maturity..quoted.on.investment.basis
13672                                                                                                3.13
```

On simplifie un peu l'affichage...:

```
R Console
> mydata<-mydata[,-1] #on élimine la ligne de titre
> label.term<-c("1M","3M","6M","1Y","2Y","3Y","5Y","7Y","10Y","20Y","30Y")
> colnames(mydata)<-label.term
> mydata
        1M   3M   6M   1Y   2Y   3Y   5Y   7Y  10Y  20Y  30Y
13672 0.04 0.04 0.06 0.09 0.37 0.79 1.55 2.08 2.54 3.13 3.42
> 
```

Et nous allons sur la dernière ligne droite en utilisant les commandes **bs( )** (b-splines) ou **ns( )** (natural splines) qui sont différents modèles de splines disponibles dans le package **splines**:

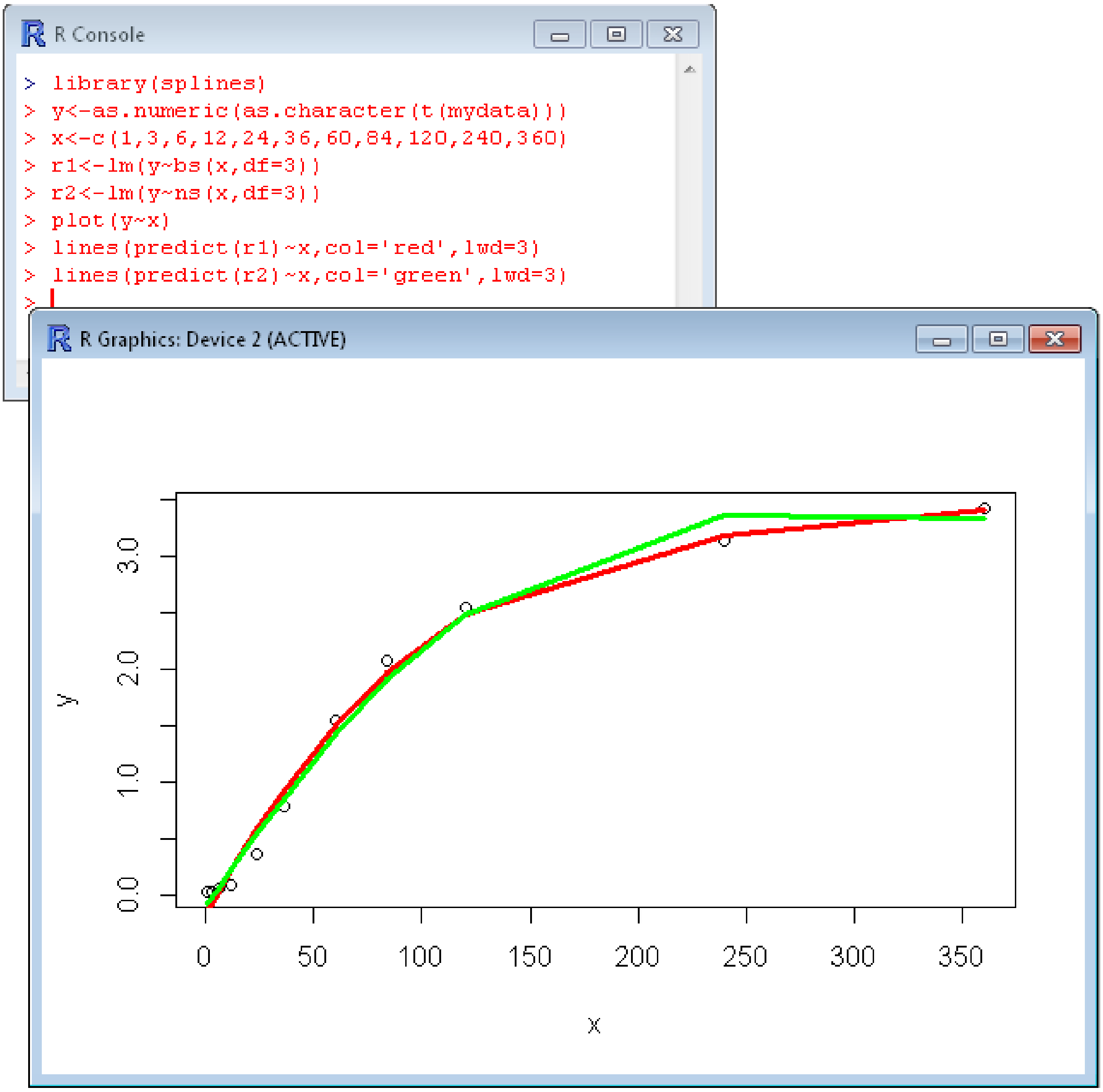


Figure 255 Graph avec interpolation par splines

Il n'est pas possible (et cela n'a pas de sens mathématiquement parlant à ma connaissance) de faire des projections (extrapolations) avec des interpolations par splines au-delà du domaine de définition d'origine.

N-splines (natural), B-splines (basis), RC-splines (restricted cubic)

Sinon voici un exemple plus court comparant trois types de splines différents (notez que les splines RC ne sont pas gérées par **lm( )** d'où le fait qu'on doive utiliser la fonction **ols( )** pour ces dernières:

```
> library("splines") #pour les n-spline et b-spline
> library("rms") #pour les rc-spline
> x   <- c(0.2,   0.23,   0.26,   0.29,   0.33,   0.46,    0.53 )
> y   <- c(0.211, 0.2026, 0.2034, 0.2167, 0.2177, 0.19225, 0.182)
> fit_lm <- lm(y ~ x)
> fit_ns <- lm(y ~ ns(x,3))
> fit_bs <- lm(y ~ bs(x,3))
> fit_rcs <- ols(y ~ rcs(x, 3))
>
> plot(y~x)
> lines(predict(fit_lm)~x,lwd=3)
> lines(predict(fit_ns)~x,col="green",lwd=3)
> lines(predict(fit_bs)~x,col="blue",lwd=3)
> lines(predict(fit_rcs)~x,col="red",lwd=3)
>
```

Ce qui donne graphiquement:

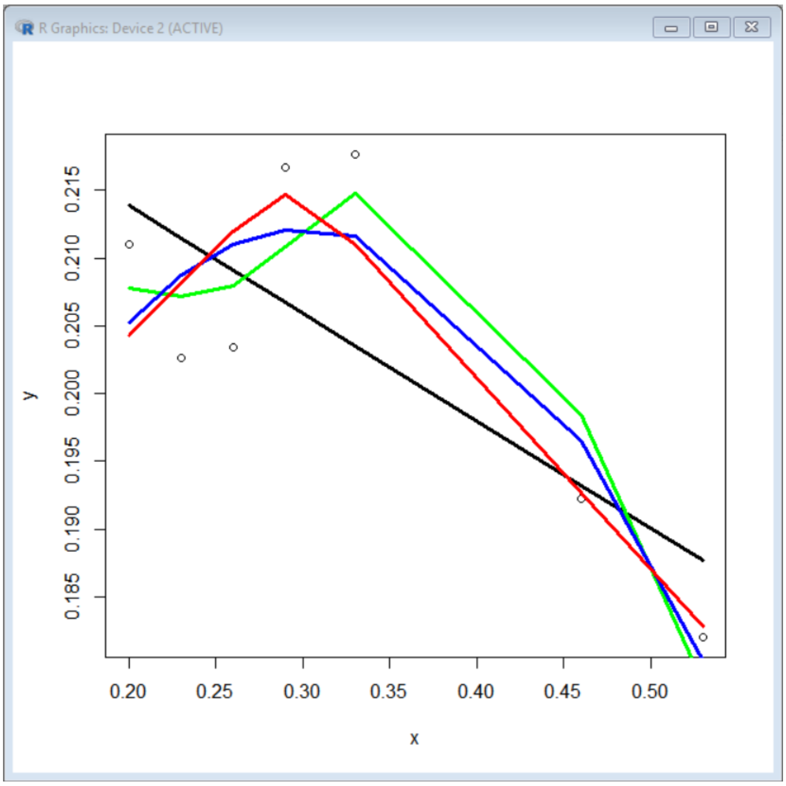
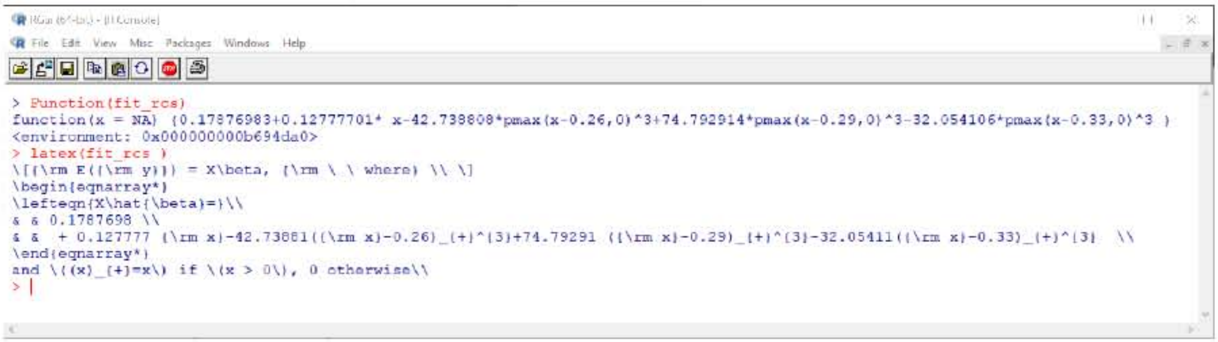


Figure 256 Graph interpolations n-splines, b-splines, rc-splines

et notons que le package **rms** a une fonction pour obtenir l'expression analytique de la spline de façon dactylographiée ou en LaTeX:



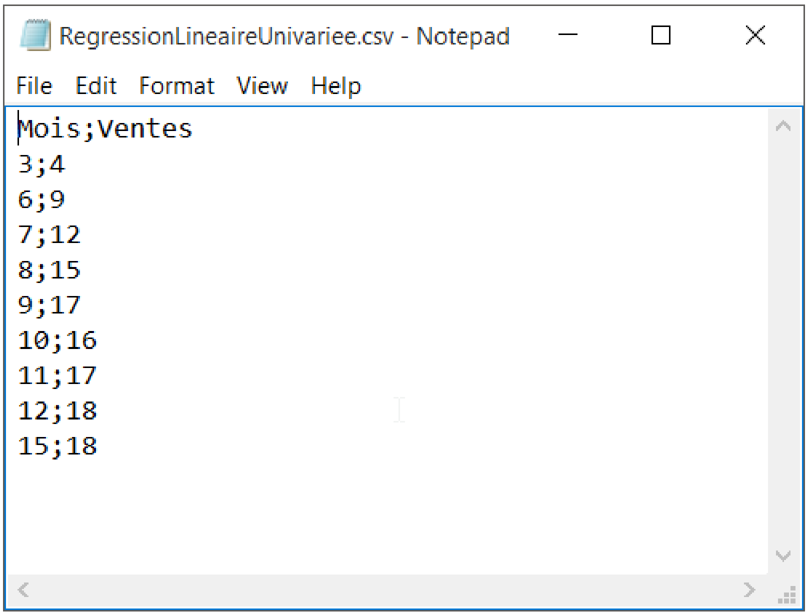
Exemple 284.: Interpolation locale par pondération (LOESS)

R 3.0.2

Bon ici nous n'allons pas trop insister car c'est de l'ingénierie numérique et comme nous l'avons vue dans le cours théorique de Méthodes Numériques où nous avons implémenté le code VBA du NIST dans Microsoft Excel il existe de nombreux algorithmes LOESS (LOcally Weighted regrESSion) différents qui donnent des résultats sensiblement différents.

Nous allons donc ici juste nous concentrer sur la mise en application de la méthode et le résultat visuel sans le commenter.

Les données unidimensionnelles seront:


RegressionLineaireUnivariee.csv - Notepad

File Edit Format View Help

```
Mois;Ventes
3;4
6;9
7;12
8;15
9;17
10;16
11;17
12;18
15;18
```

Ensuite viennent les commandes:

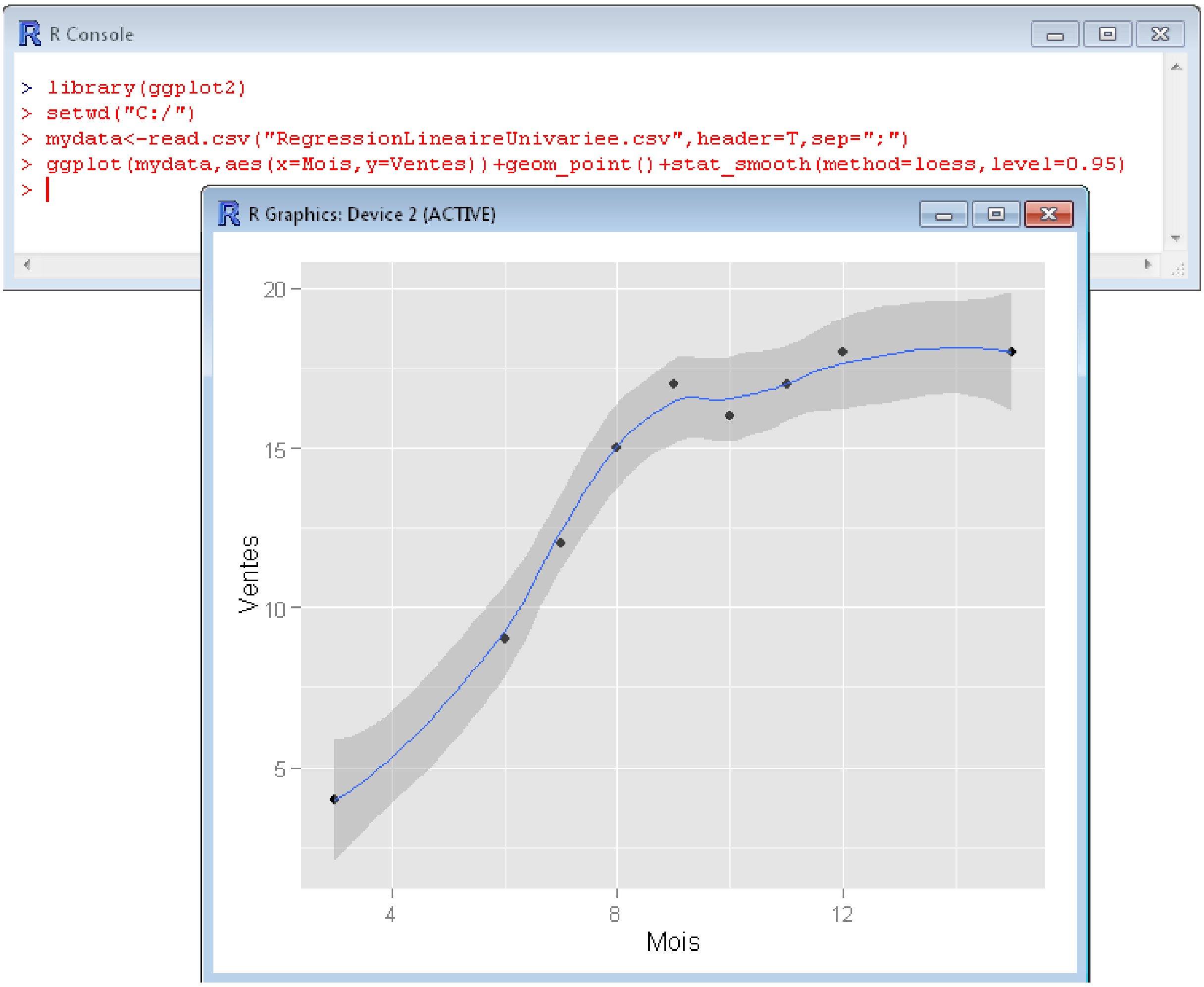


Figure 257 Graph interpolation LOESS with confidence interval (ggplot)

Comme nous allons le montrer maintenant, ce modèle ne permet (du moins à ma connaissance) pas d'extrapoler mais uniquement d'interpoler:

```
> ventes.lo<-loess(Ventes~Mois, mydata)
> ventes.lo$fitted
[1]  3.979455  9.288714 12.401458 15.077344 16.458592 16.541408 17.025920 17.656949 18.024158
> predict(ventes.lo, data.frame(Mois = seq(1, 20, 1)), se = TRUE)$fit
        1         2         3         4         5         6         7         8         9
       NA        NA  3.979455  5.323149  7.112823  9.288714 12.401458 15.077344 16.458592
       10        11        12        13        14        15        16        17        18
16.541408 17.025920 17.656949 18.002930 18.150027 18.024158        NA        NA        NA
       19        20
       NA        NA
>
```

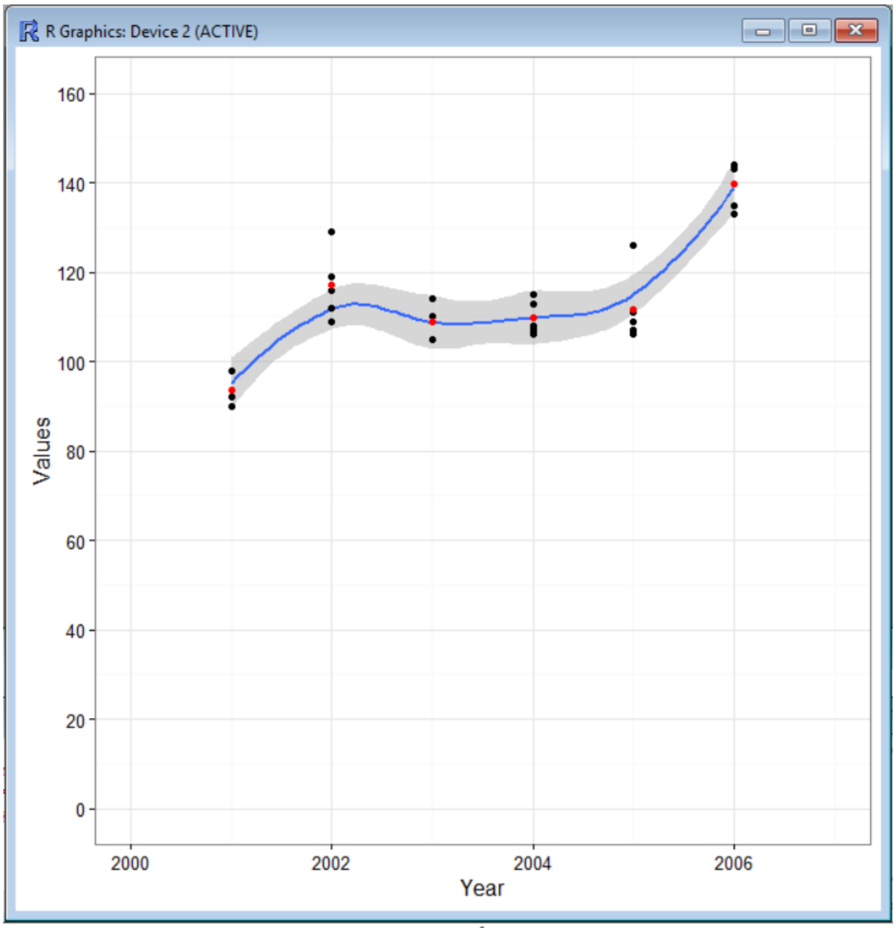
Ou un autre cas dont nous avons longuement discuté dans le cours théorique en considérant les données suivantes:

```
R Console
> library(ggplot2)
> mydata<-read.csv("c:/tmp/SmoothLinePlot.csv",header=T,sep=";")
> mydata
   Year Value
1  2001    90
2  2001    98
3  2001    98
4  2001    90
5  2001    92
6  2002   129
7  2002   112
8  2002   109
9  2002   119
10 2002   116
11 2003   110
12 2003   114
13 2003   110
14 2003   105
15 2003   105
16 2004   106
17 2004   115
18 2004   113
19 2004   107
20 2004   108
21 2005   126
22 2005   106
23 2005   111
24 2005   109
25 2005   107
26 2006   133
27 2006   143
28 2006   144
29 2006   135
30 2006   144
> 
```

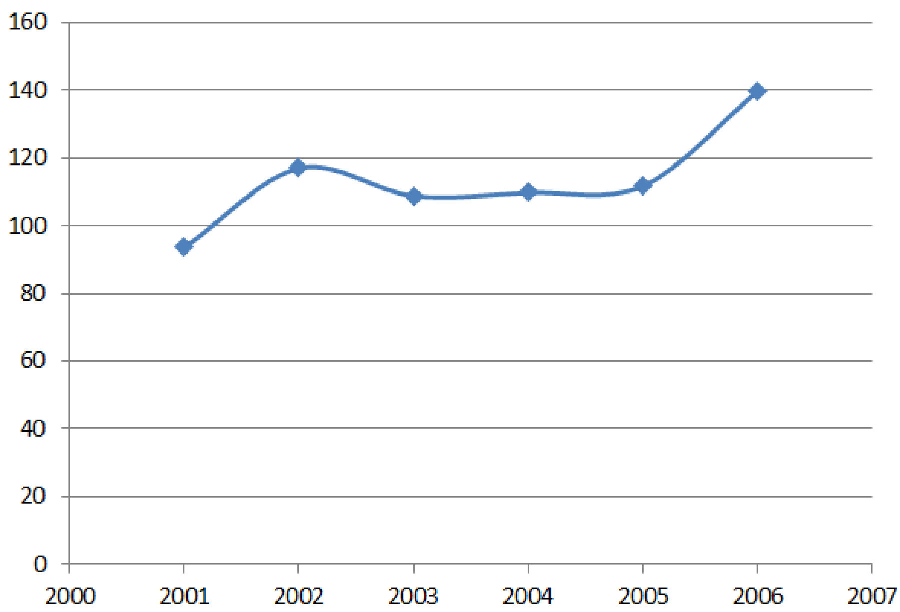
Et en utilisant la fonction **qplot( )** :

```
R Console
> qplot(Year, Value, data = mydata, geom = "smooth", method="loess", xlab = "Year", ylab = "Values", xlim = c(2000, 2007))+
+ scale_y_continuous(breaks = seq(0, 160, by = 20), limits = c(0, 160))+
+ theme_bw()+ geom_point()+ stat_summary(fun.y="mean", colour="red", geom="point")
> 
```

Ce qui donne:



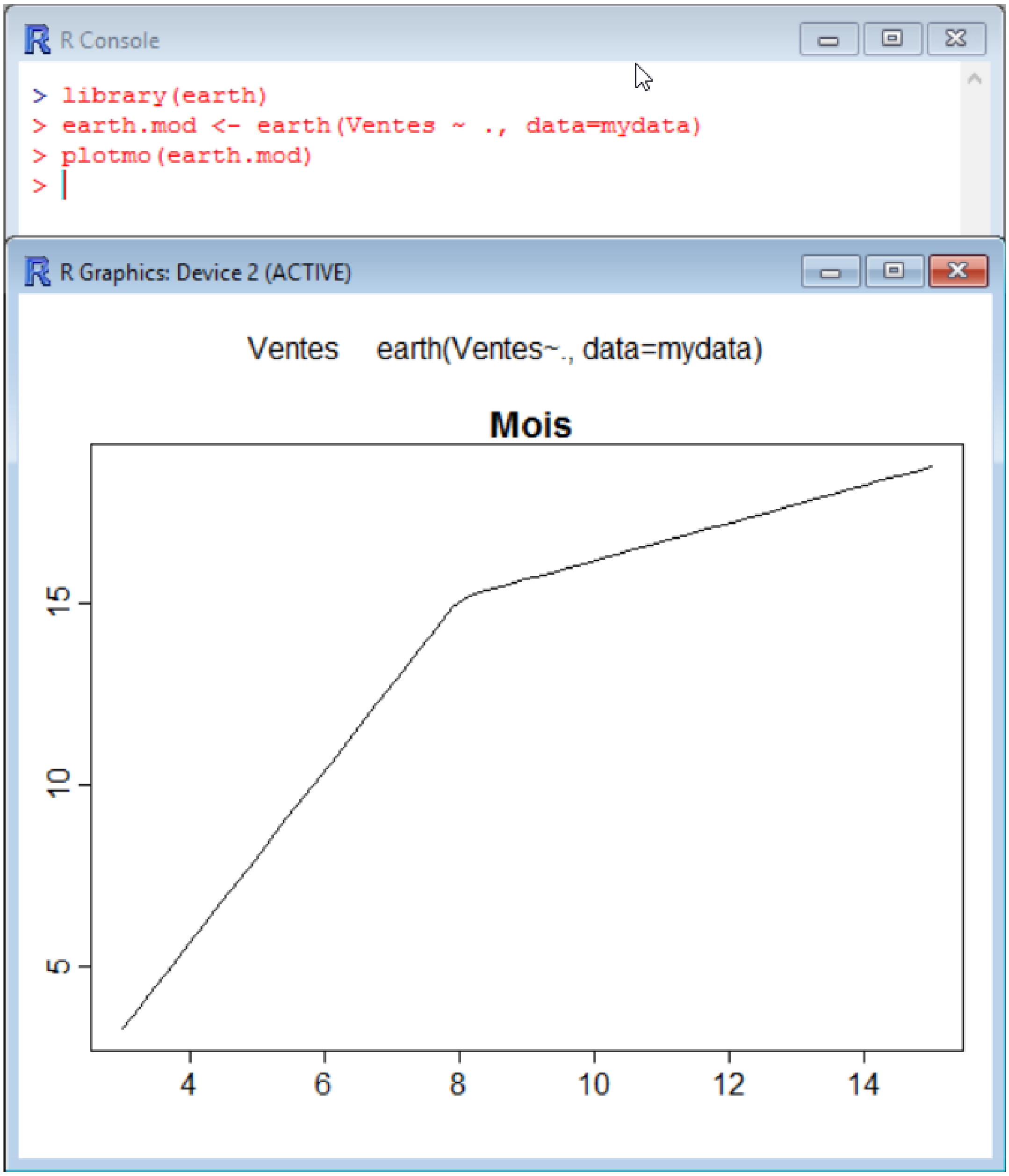
à comparer avec un tableur…:



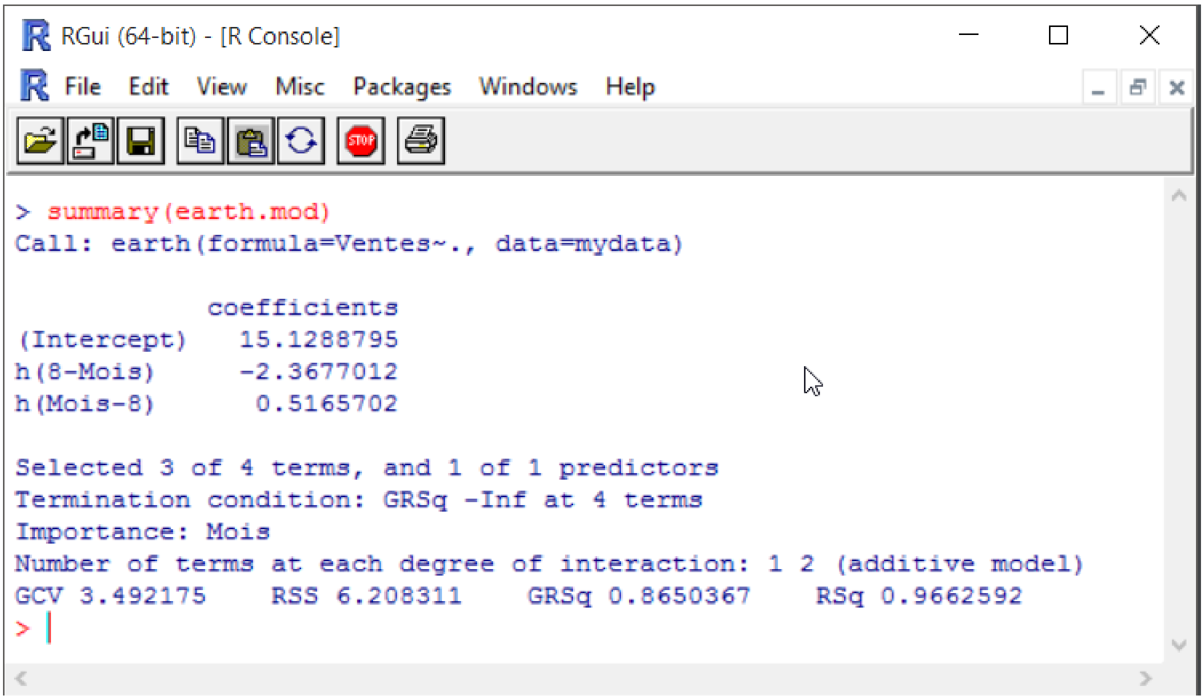
Exemple 285.: Interpolation Multivariate Adaptive Regression Splines (MARS)

R 3.1.2

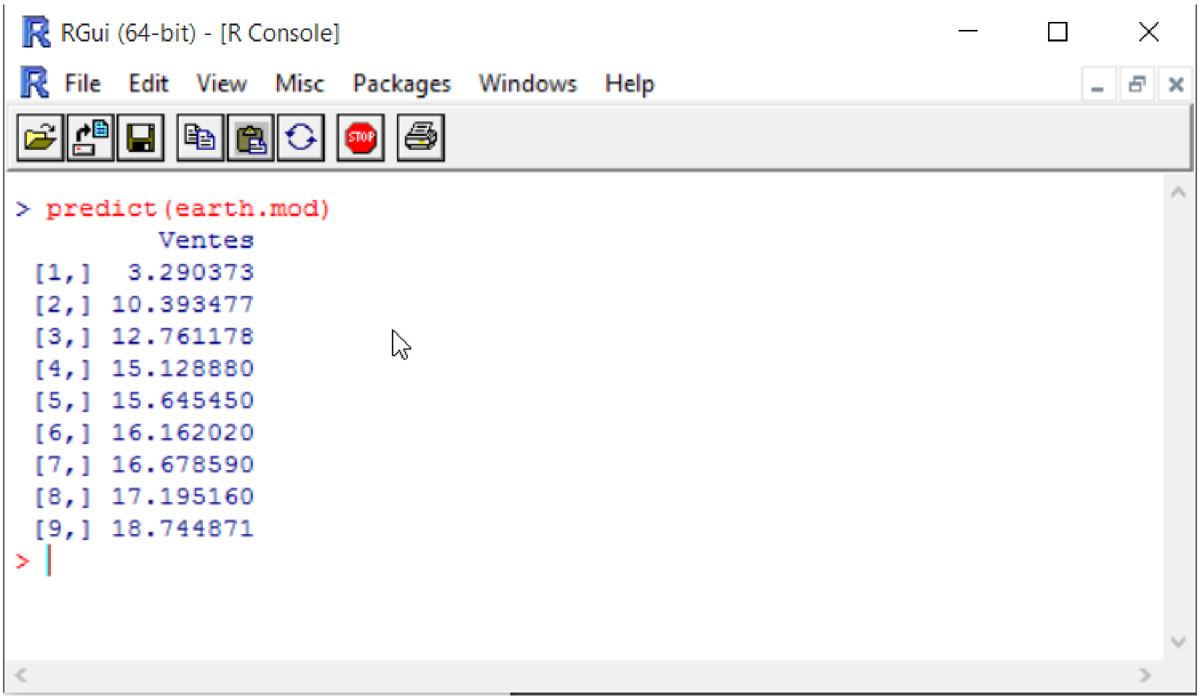
Voyons comment effectuer une interpolation MARS avec la fonction **earth( )** du package **earth**:



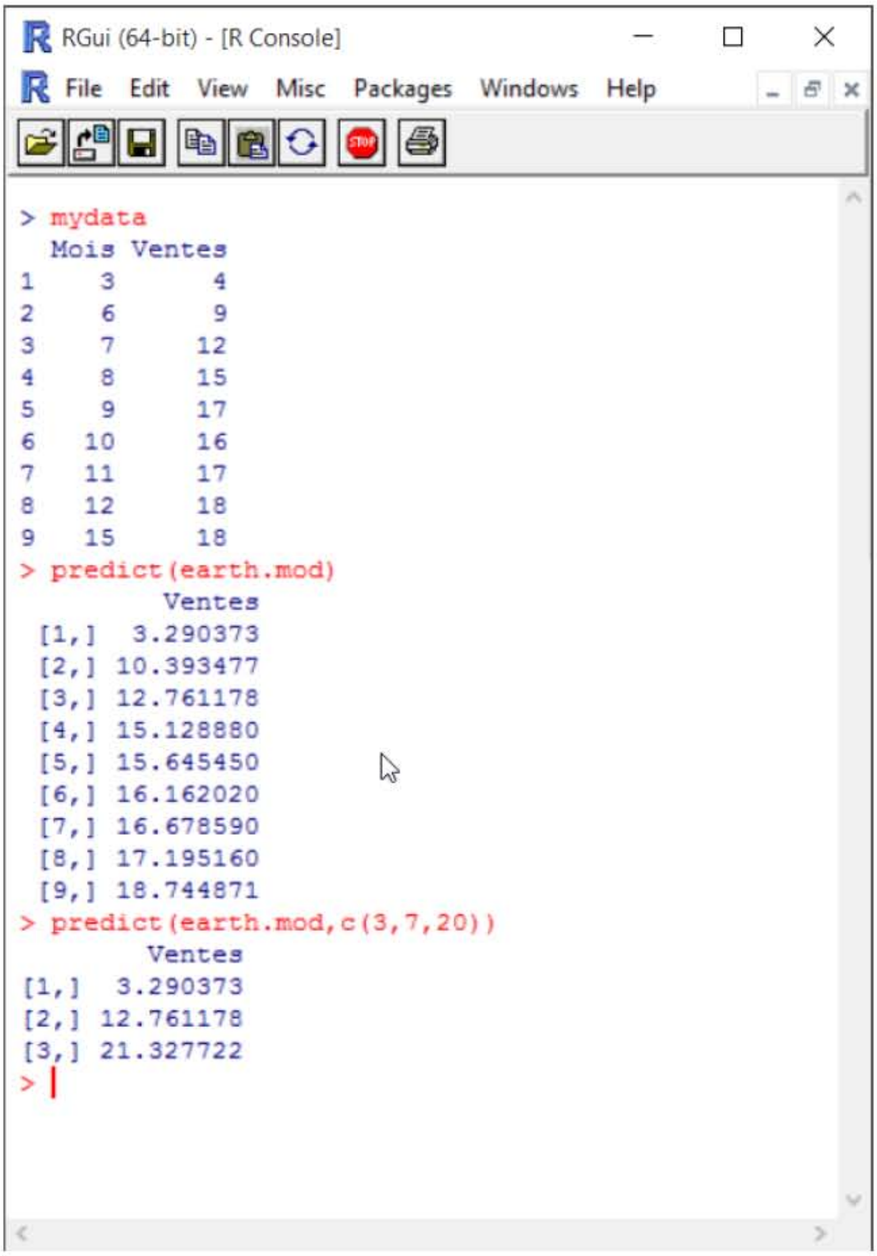
Nous avons alors pour résumé:



Si on veut les valeurs interpolées nous utilisons la fonction **predict( )** du package:

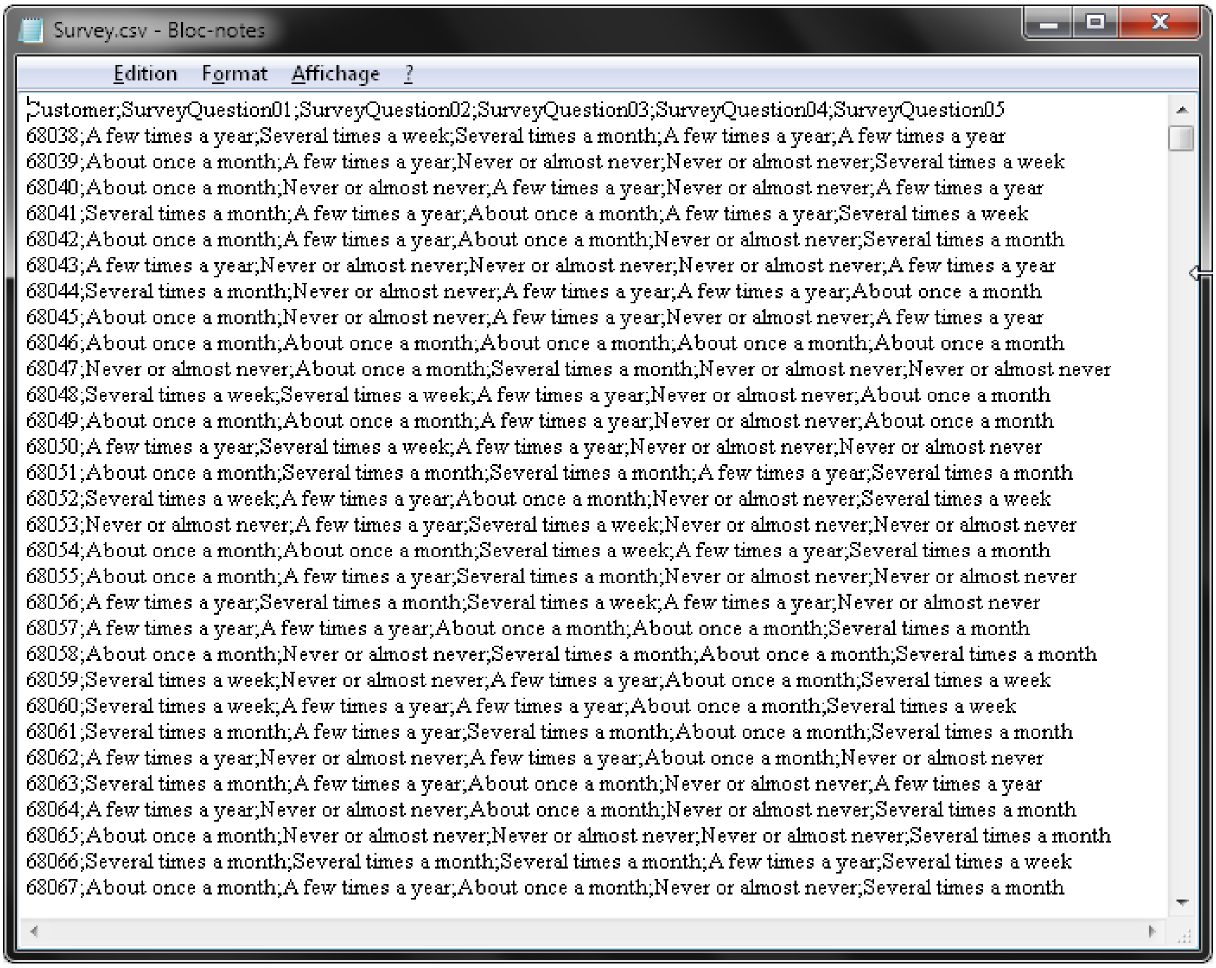


Nous pouvons aussi choisir les données interpolées voulues ou même extrapolées toujours avec la même fonction…:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> mydata
  Mois Ventes
1   3   4
2   6   9
3   7   12
4   8   15
5   9   17
6   10   16
7   11   17
8   12   18
9   15   18
> predict(earth.mod)
         Ventes
 [1,] 3.290373
 [2,] 10.393477
 [3,] 12.761178
 [4,] 15.128880
 [5,] 15.645450
 [6,] 16.162020
 [7,] 16.678590
 [8,] 17.195160
 [9,] 18.744871
> predict(earth.mod,c(3,7,20))
        Ventes
[1,] 3.290373
[2,] 12.761178
[3,] 21.327722
>

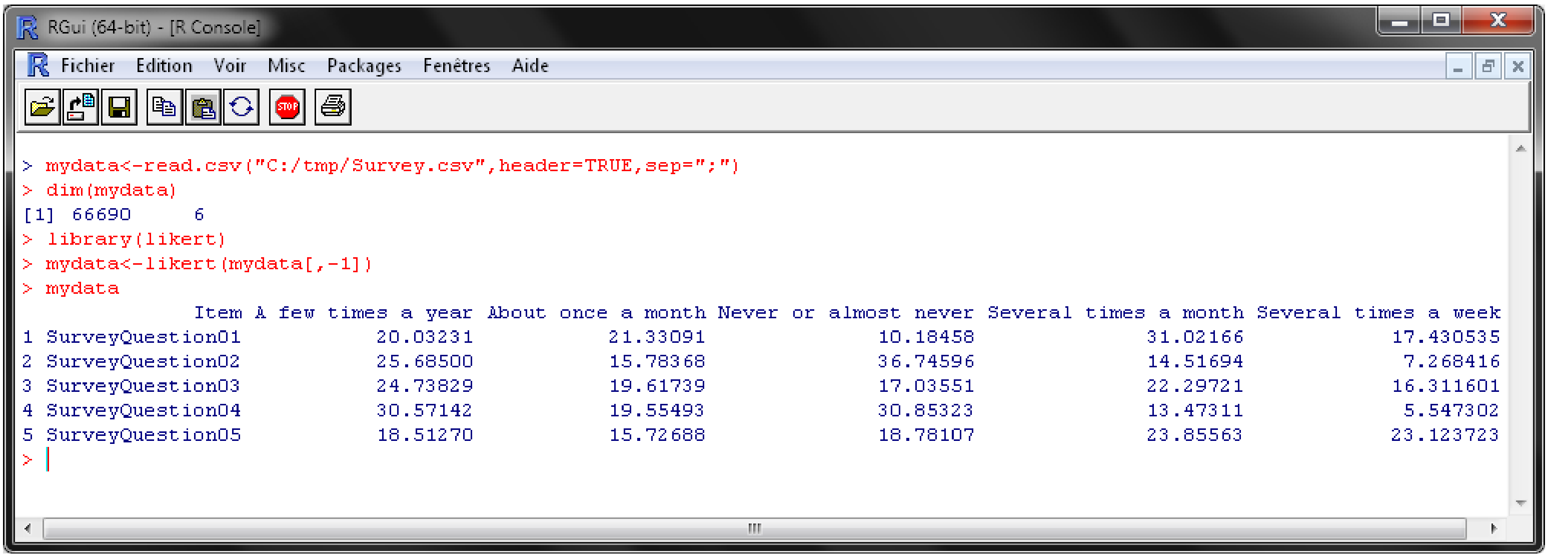
17. Sondages

Pour étudier les sondages, nous partons du fichier suivant ayant principalement des variables catégorielles:



```
Survey.csv - Bloc-notes
Edition  Format  Affichage  ?
Customer;SurveyQuestion01;SurveyQuestion02;SurveyQuestion03;SurveyQuestion04;SurveyQuestion05
68038;A few times a year;Several times a week;Several times a month;A few times a year;A few times a year
68039;About once a month;A few times a year;Never or almost never;Never or almost never;Several times a week
68040;About once a month;Never or almost never;A few times a year;Never or almost never;A few times a year
68041;Several times a month;A few times a year;About once a month;A few times a year;Several times a week
68042;About once a month;A few times a year;About once a month;Never or almost never;Several times a month
68043;A few times a year;Never or almost never;Never or almost never;Never or almost never;A few times a year
68044;Several times a month;Never or almost never;A few times a year;A few times a year;About once a month
68045;About once a month;Never or almost never;A few times a year;Never or almost never;A few times a year
68046;About once a month;About once a month;About once a month;About once a month;About once a month
68047;Never or almost never;About once a month;Several times a month;Never or almost never;Never or almost never
68048;Several times a week;Several times a week;A few times a year;Never or almost never;About once a month
68049;About once a month;About once a month;A few times a year;Never or almost never;About once a month
68050;A few times a year;Several times a week;A few times a year;Never or almost never;Never or almost never
68051;About once a month;Several times a month;Several times a month;A few times a year;Several times a month
68052;Several times a week;A few times a year;About once a month;Never or almost never;Several times a week
68053;Never or almost never;A few times a year;Several times a week;Never or almost never;Never or almost never
68054;About once a month;About once a month;Several times a week;A few times a year;Several times a month
68055;About once a month;A few times a year;Several times a month;Never or almost never;Never or almost never
68056;A few times a year;Several times a month;Several times a week;A few times a year;Never or almost never
68057;A few times a year;A few times a year;About once a month;About once a month;Several times a month
68058;About once a month;Never or almost never;Several times a month;About once a month;Several times a month
68059;Several times a week;Never or almost never;A few times a year;About once a month;Several times a week
68060;Several times a week;A few times a year;A few times a year;About once a month;Several times a week
68061;Several times a month;A few times a year;Several times a month;About once a month;Several times a month
68062;A few times a year;Never or almost never;A few times a year;About once a month;Never or almost never
68063;Several times a month;A few times a year;About once a month;Never or almost never;A few times a year
68064;A few times a year;Never or almost never;About once a month;Never or almost never;Several times a month
68065;About once a month;Never or almost never;Never or almost never;Never or almost never;Several times a month
68066;Several times a month;Several times a month;Several times a month;A few times a year;Several times a week
68067;About once a month;A few times a year;About once a month;Never or almost never;Several times a month
```

Avec le package **likert** et la fonction **likert( )** nous obtenons:



```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide
> mydata<-read.csv("C:/tmp/Survey.csv",header=TRUE,sep=";")
> dim(mydata)
[1] 66690     6
> library(likert)
> mydata<-likert(mydata[,-1])
> mydata
              Item A few times a year About once a month Never or almost never Several times a month Several times a week
1 SurveyQuestion01           20.03231           21.33091              10.18458              31.02166            17.430535
2 SurveyQuestion02           25.68500           15.78368              36.74596              14.51694             7.268416
3 SurveyQuestion03           24.73829           19.61739              17.03551              22.29721            16.311601
4 SurveyQuestion04           30.57142           19.55493              30.85323              13.47311             5.547302
5 SurveyQuestion05           18.51270           15.72688              18.78107              23.85563            23.123723
> |
```

Si on représente graphiquement les réponses par question dans un diagramme croisée en pourcentages, nous obtenons:

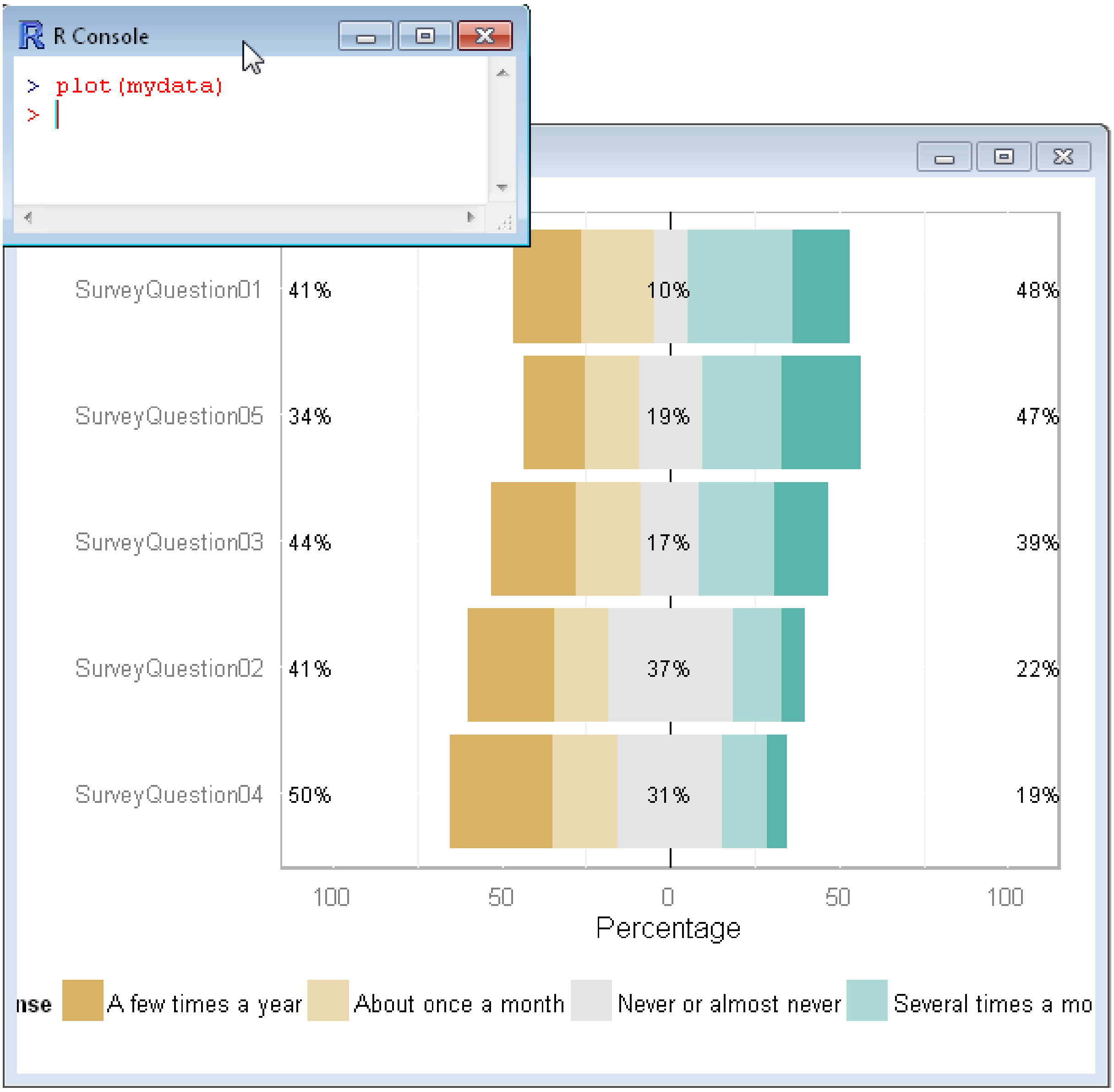


Figure 258 Graph de sondage Liekert à barre verticales groupées

Non-centré, c'est peut-être plus intuitif…, nous avons alors:

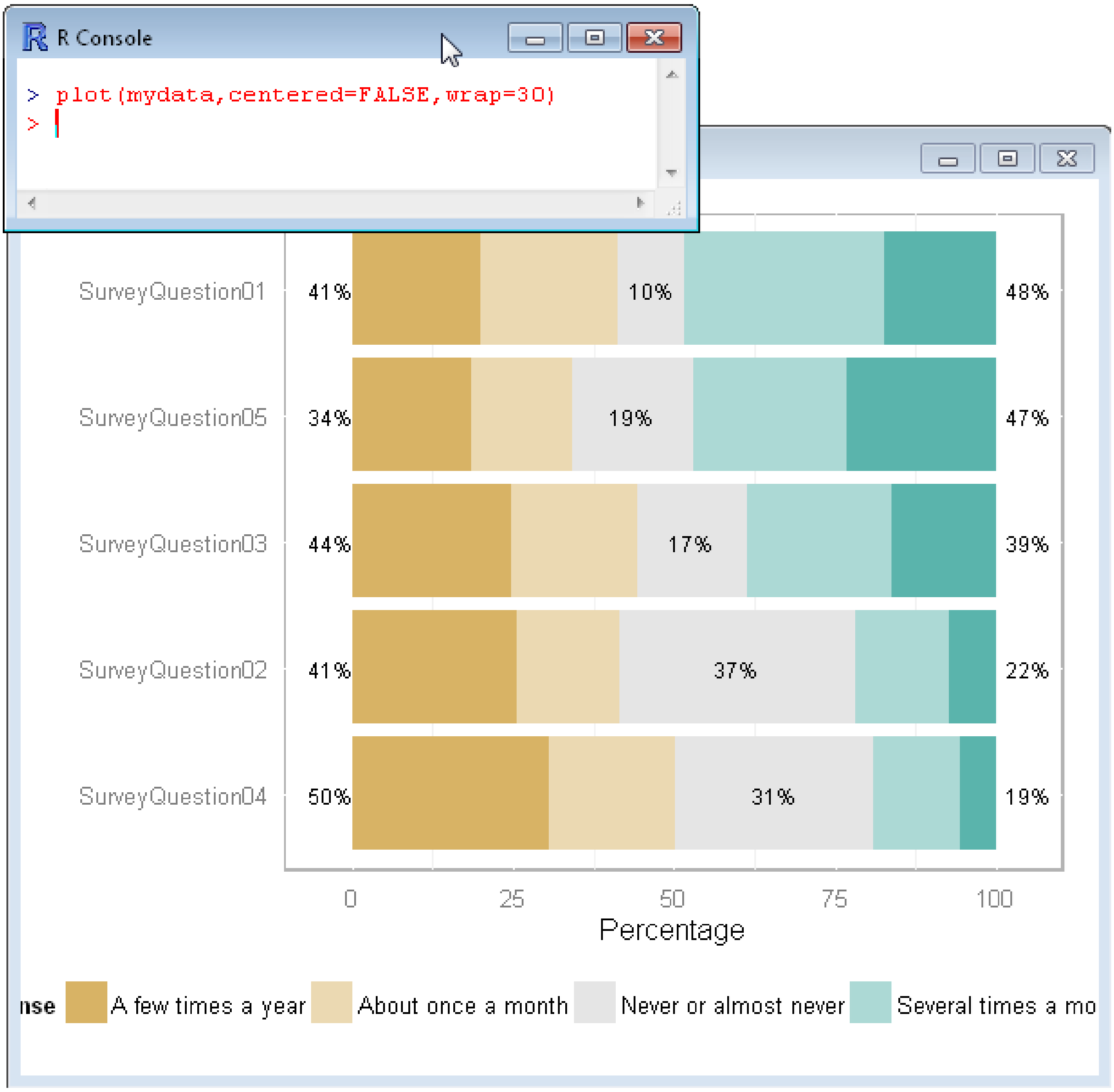


Figure 259 Graph de sondage Liekert à barre verticales groupées normalisées

Nous pouvons afficher les réponses sous formes de diagrammes densité:

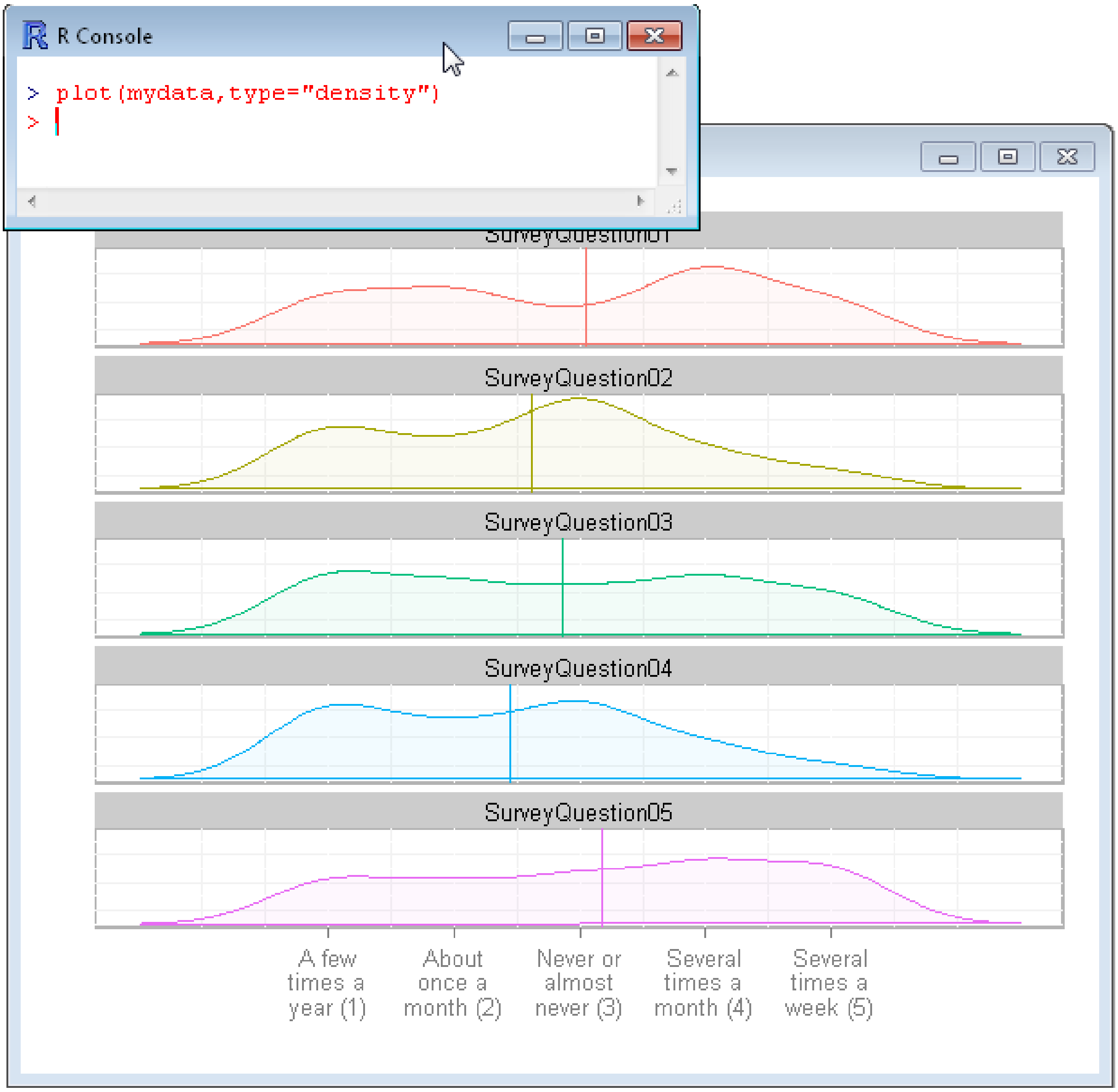
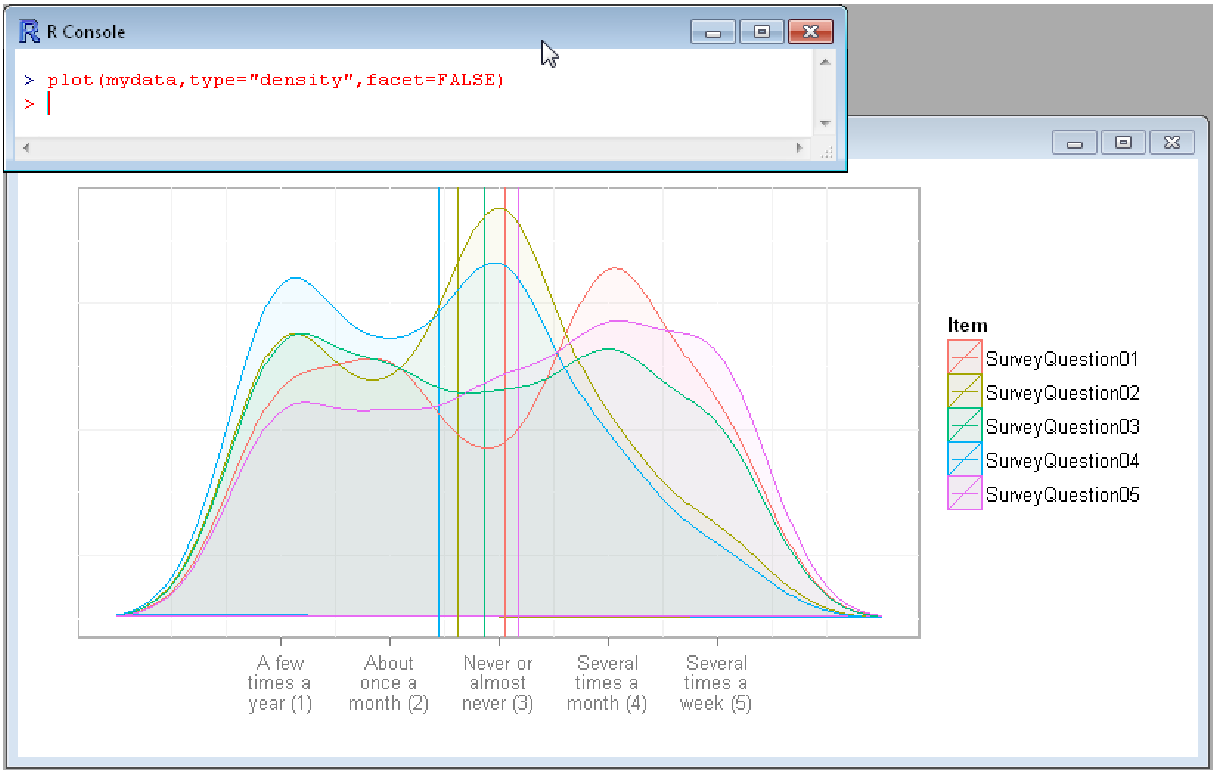


Figure 260 Graph de sondage Liekert avec fonction de densité

Ou sinon on peut afficher les réponses sous la forme d'une matrice de carte chaleur:


R Console
> plot(mydata,type="heat")
R Graphics: Device 2 (ACTIVE)
SurveyQuestion05
3.17 (1.4)
18.5%
15.7%
18.8%
23.9%
23.1%
SurveyQuestion04
2.44 (1.2)
30.6%
19.6%
30.9%
13.5%
5.5%
SurveyQuestion03
2.86 (1.4)
24.7%
19.6%
17.0%
22.3%
16.3%
SurveyQuestion02
2.62 (1.2)
25.7%
15.8%
36.7%
14.5%
7.3%
SurveyQuestion01
3.04 (1.4)
20.0%
21.3%
10.2%
31.0%
17.4%
Mean (SD)
A few times a year
About once a month
Never or almost never
Several times a month
Several times a week
Percent
100
75
50
25
0

Ou encore:


R Console
> plot(mydata,type="density",facet=FALSE)
A few times a year (1)
About once a month (2)
Never or almost never (3)
Several times a month (4)
Several times a week (5)
Item
SurveyQuestion01
SurveyQuestion02
SurveyQuestion03
SurveyQuestion04
SurveyQuestion05

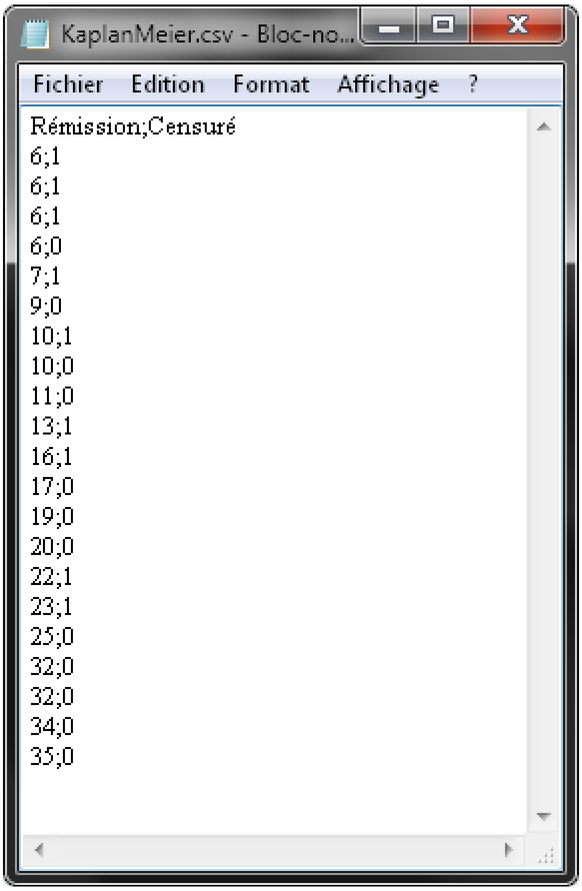
18. Fiabilité/Survie

Exemple 286.: Analyse de données censurées selon modèle non paramétrique de Kaplan-Meier

R 3.0.2

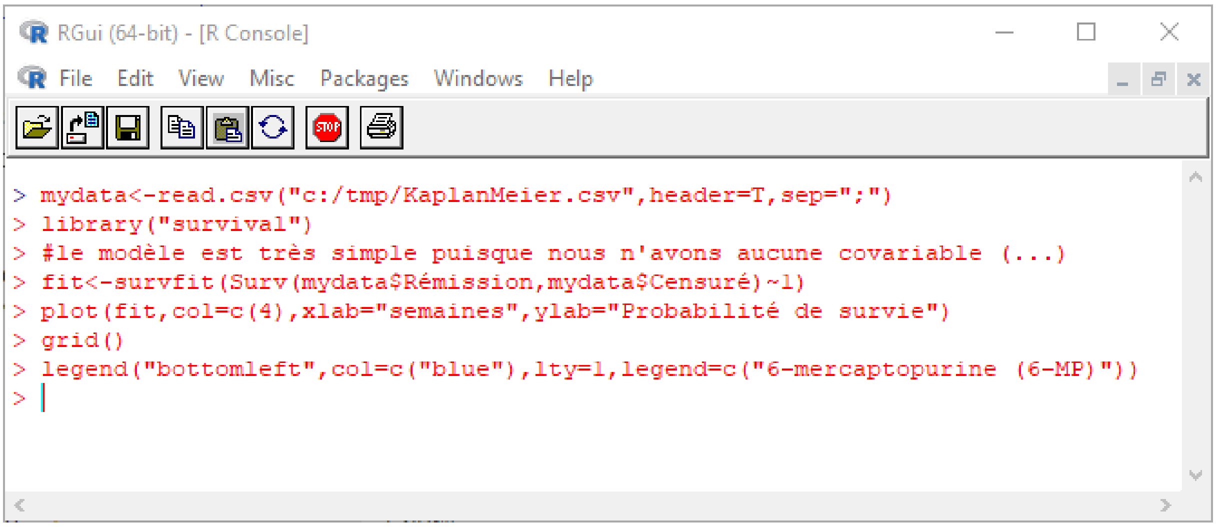
Le but ici va être à nouveau de vérifier si nous obtenons bien les mêmes résultats que ceux calculés à la main dans le cours théorique lorsque nous avons étudié la démonstration de l'estimateur de survie de Kaplan-Meier.

Nous partons du tableau vu en cours mais adapté aux exigences de Minitab et R (il n'est vraiment pas aisé de deviner que c'est sous cette forme que les choses doivent être représentées):




```
Rémission;Censuré
6;1
6;1
6;1
6;0
7;1
9;0
10;1
10;0
11;0
13;1
16;1
17;0
19;0
20;0
22;1
23;1
25;0
32;0
32;0
34;0
35;0
```

Ensuite, en utilisant le package **Survival**, nous utilisons la commande **survfit( )** avec **Surv( )** de la manière suivante:



le ~1 signifiant qu'il n'y a pas de facteur explicatif particulier supplémentaires (comme l'âge, le sexe ou autre).

Ce qui donne graphiquement (je n'ai pas trouvé comment désactiver les deux intervalles de confiance sachant que nous n'en avons pas fait la démonstration mathématique dans le cours théorique):

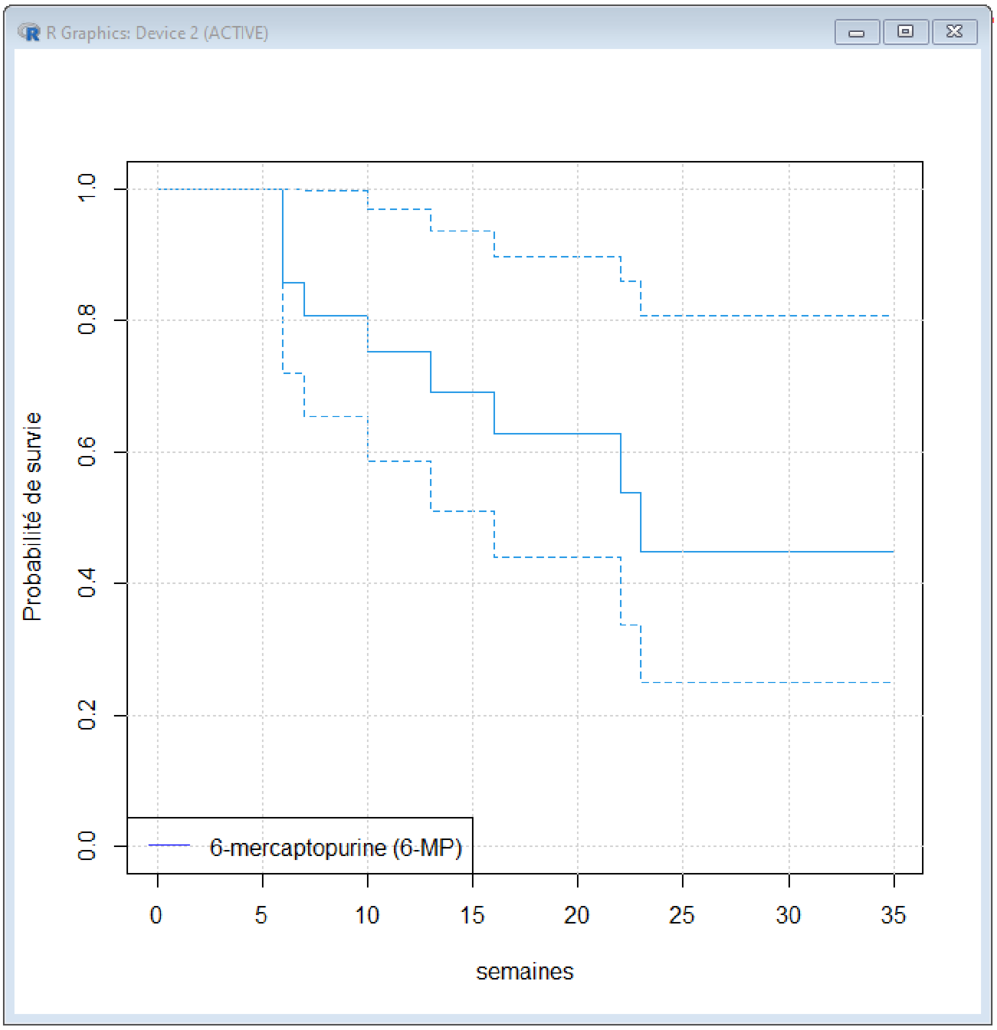
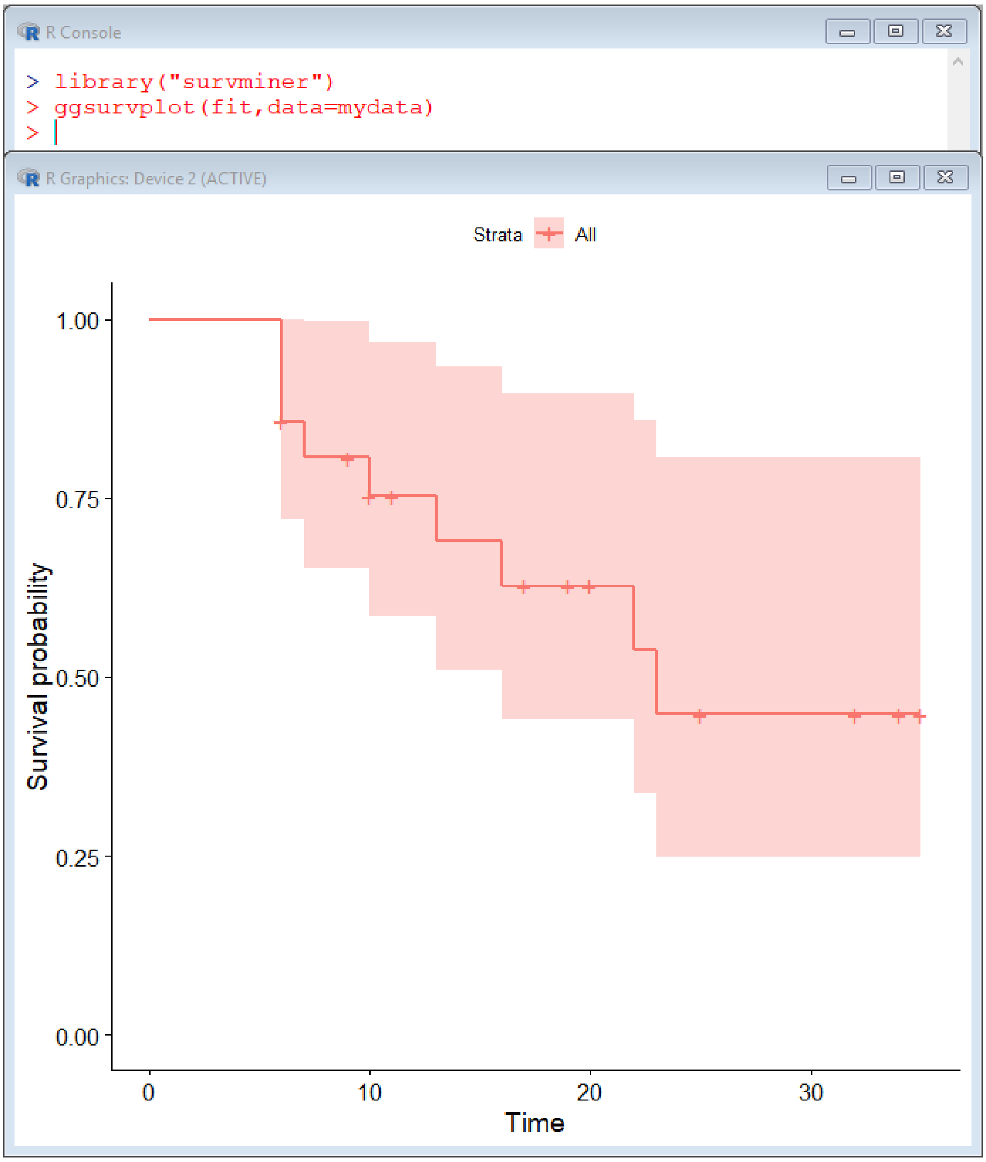
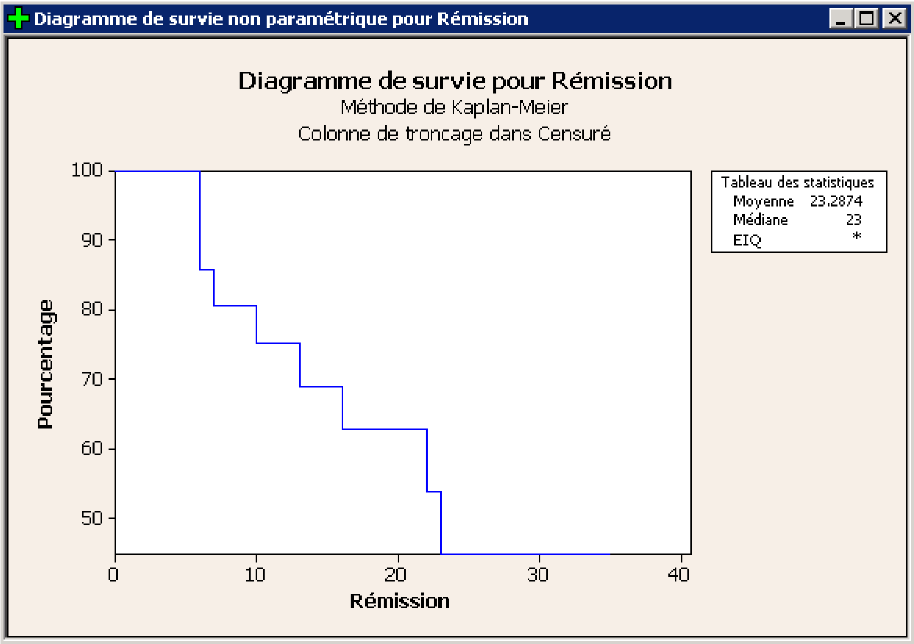


Figure 261 Graph de survie de Kaplan-Meier

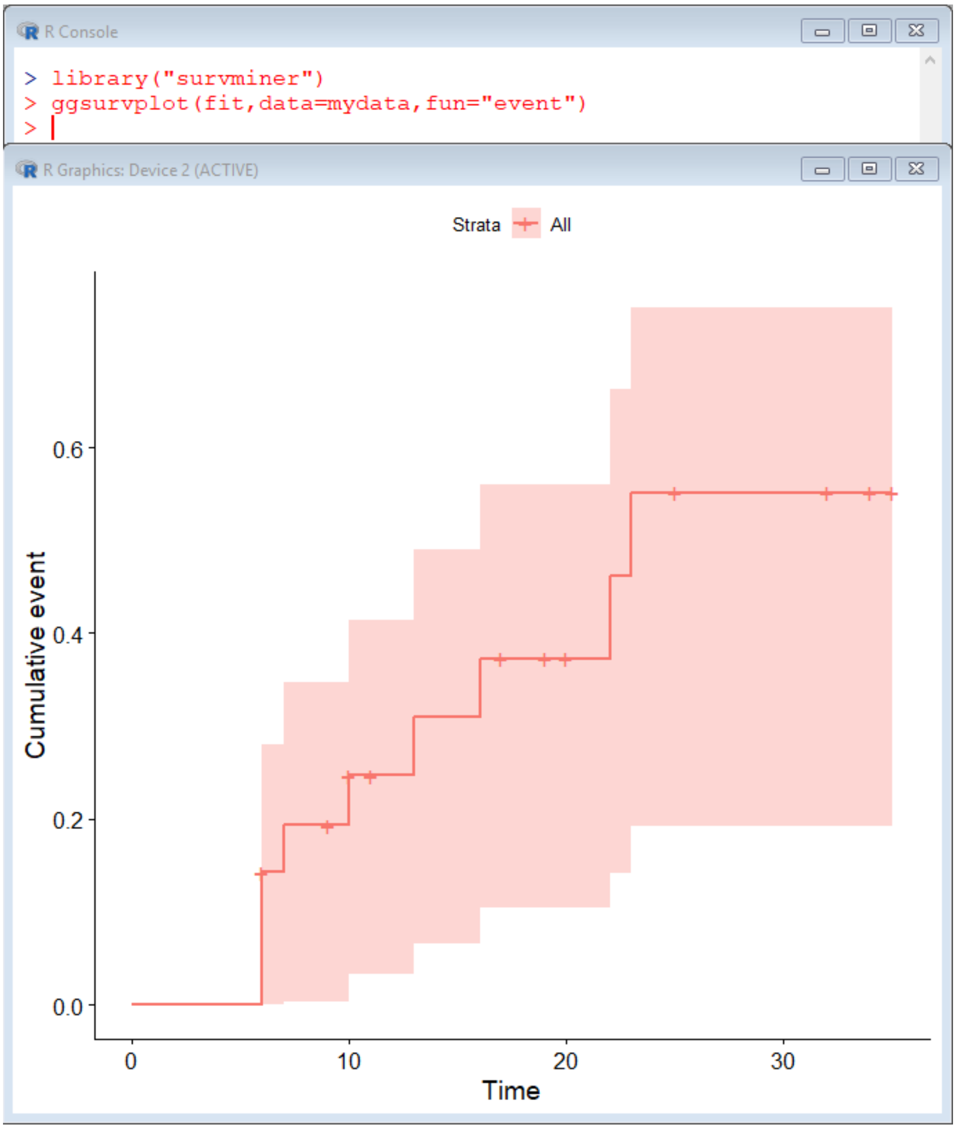
Ou avec la fonction **ggsurvplot( )** du package **survminer**:



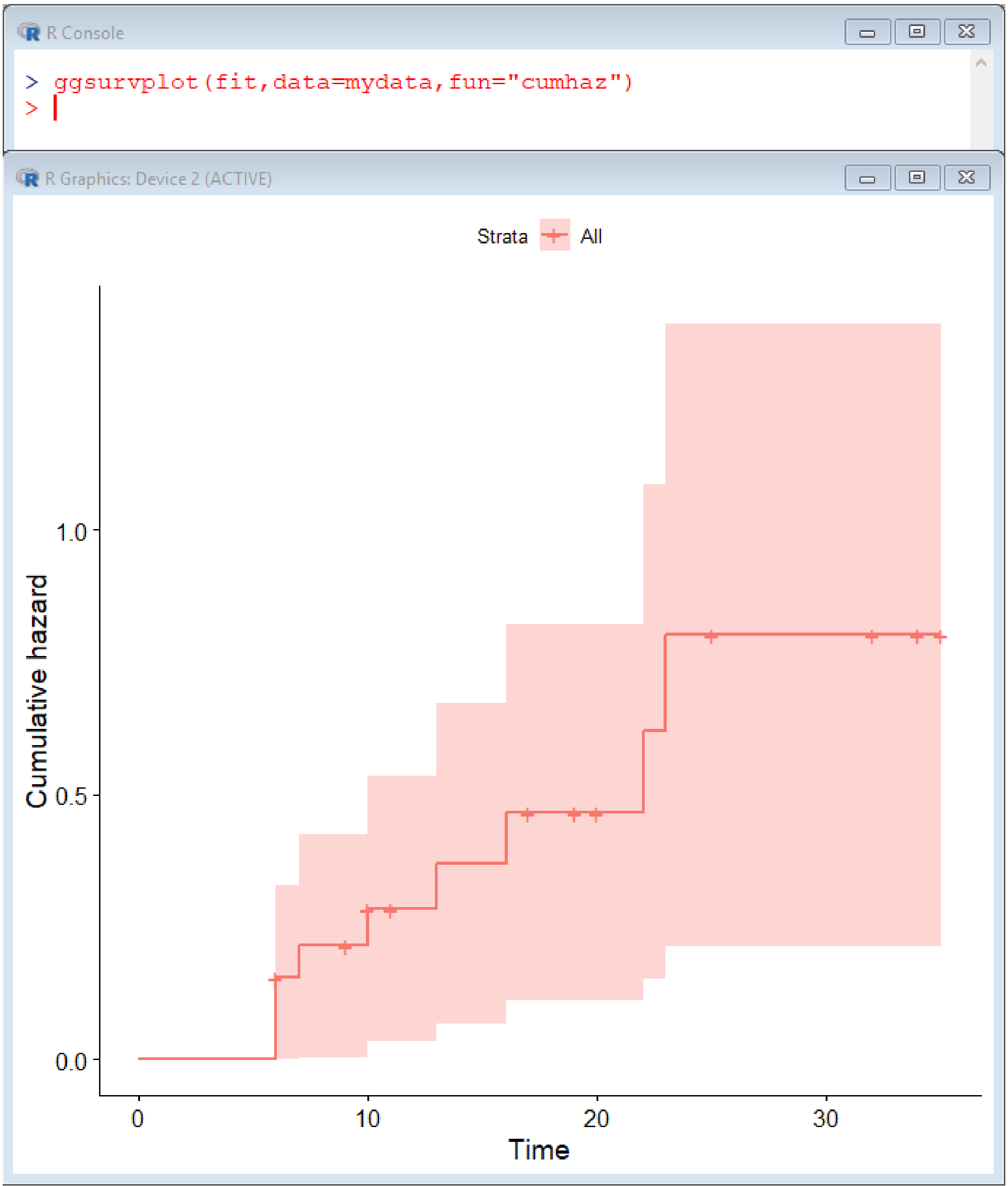
à comparer avec Minitab (on appréciera donc dans R l'indication des données censurées qui sont a priori absentes dans Minitab):



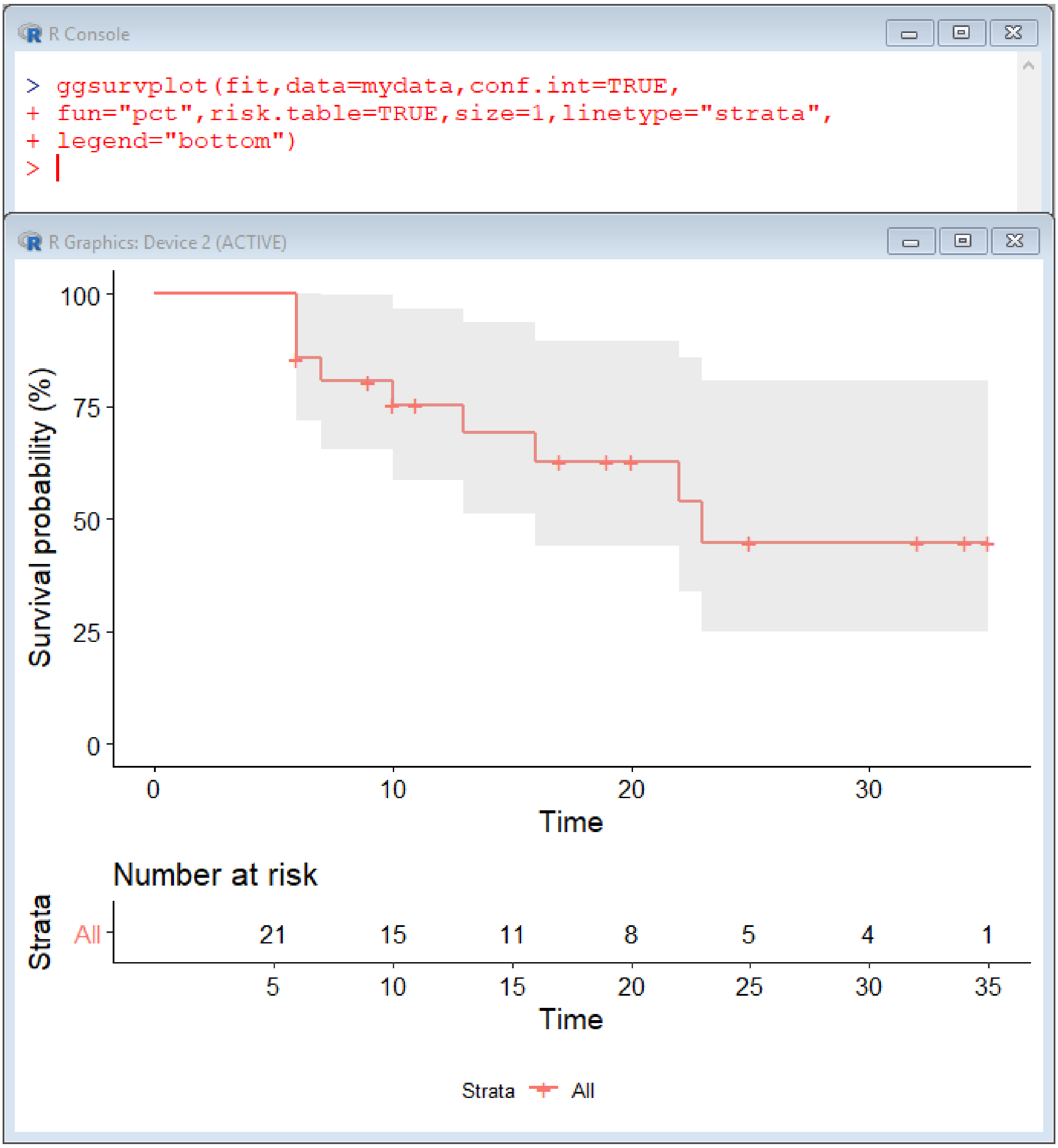
Sinon nous pouvons avoir aussi le graphique des évènements cumulés:



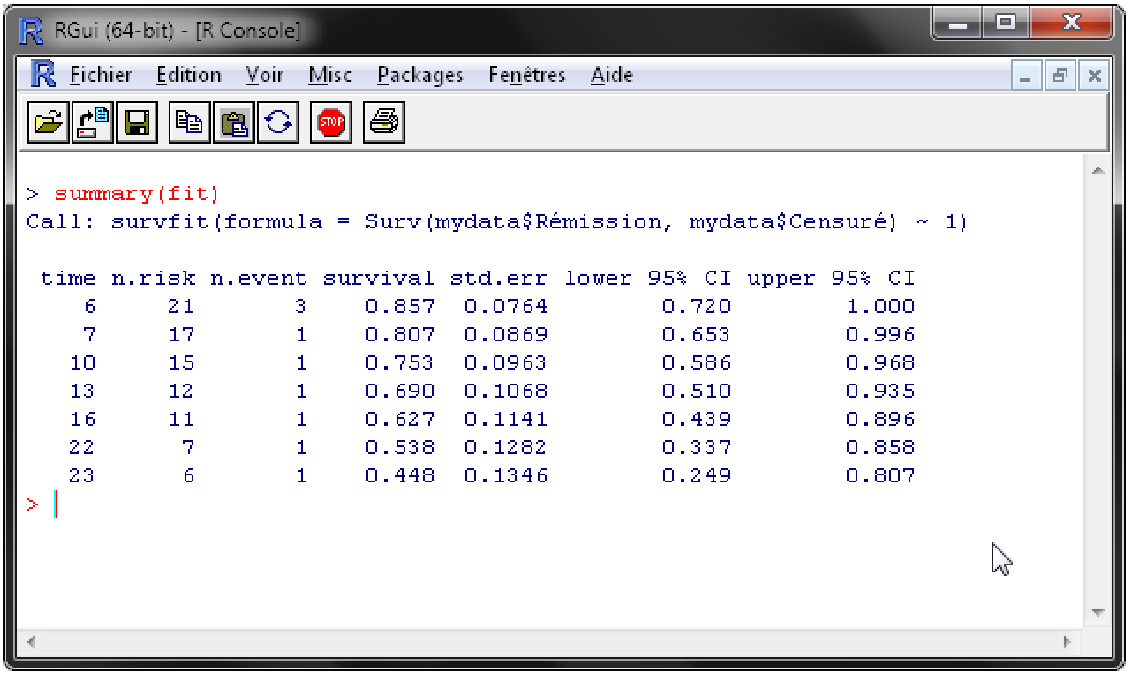
Ou le graphique du hasard cumulé:



Ou encore:

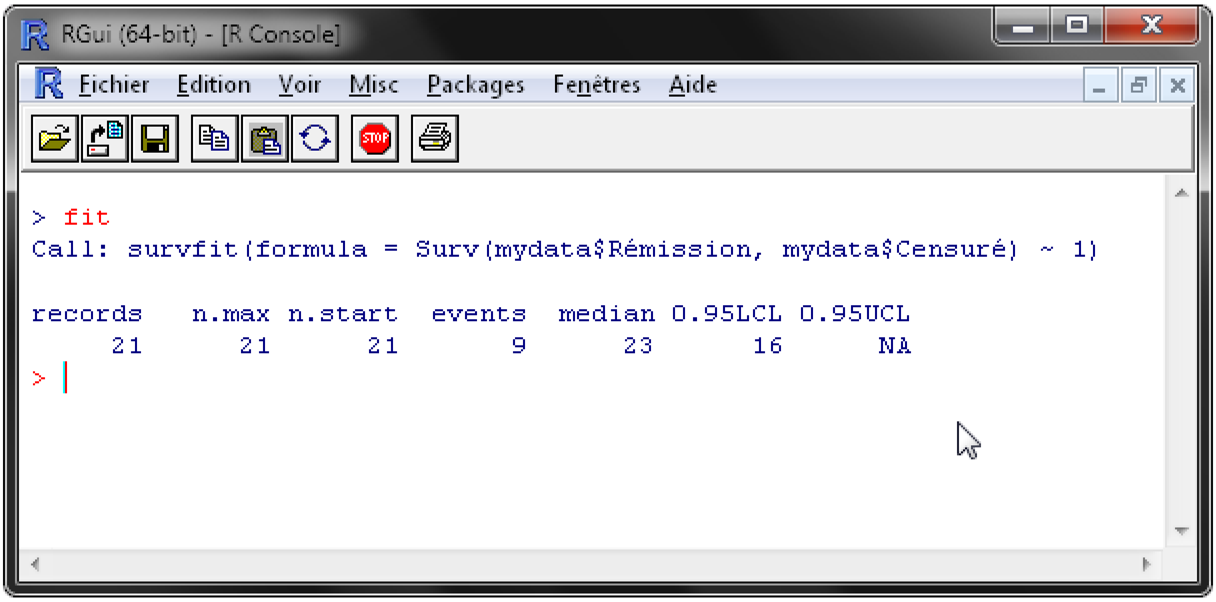


Bon les graphiques c'est bien joli mais nous voulons des chiffres! Nous écrivons alors:



et nous retrouvons tous les chiffres calculés à la main dans le cours théorique (excepté pour les deux dernières colonnes puisque nous n'avons pas démontré mathématiquement comme obtenir l'intervalle de confiance).

Et en écrivant simplement:

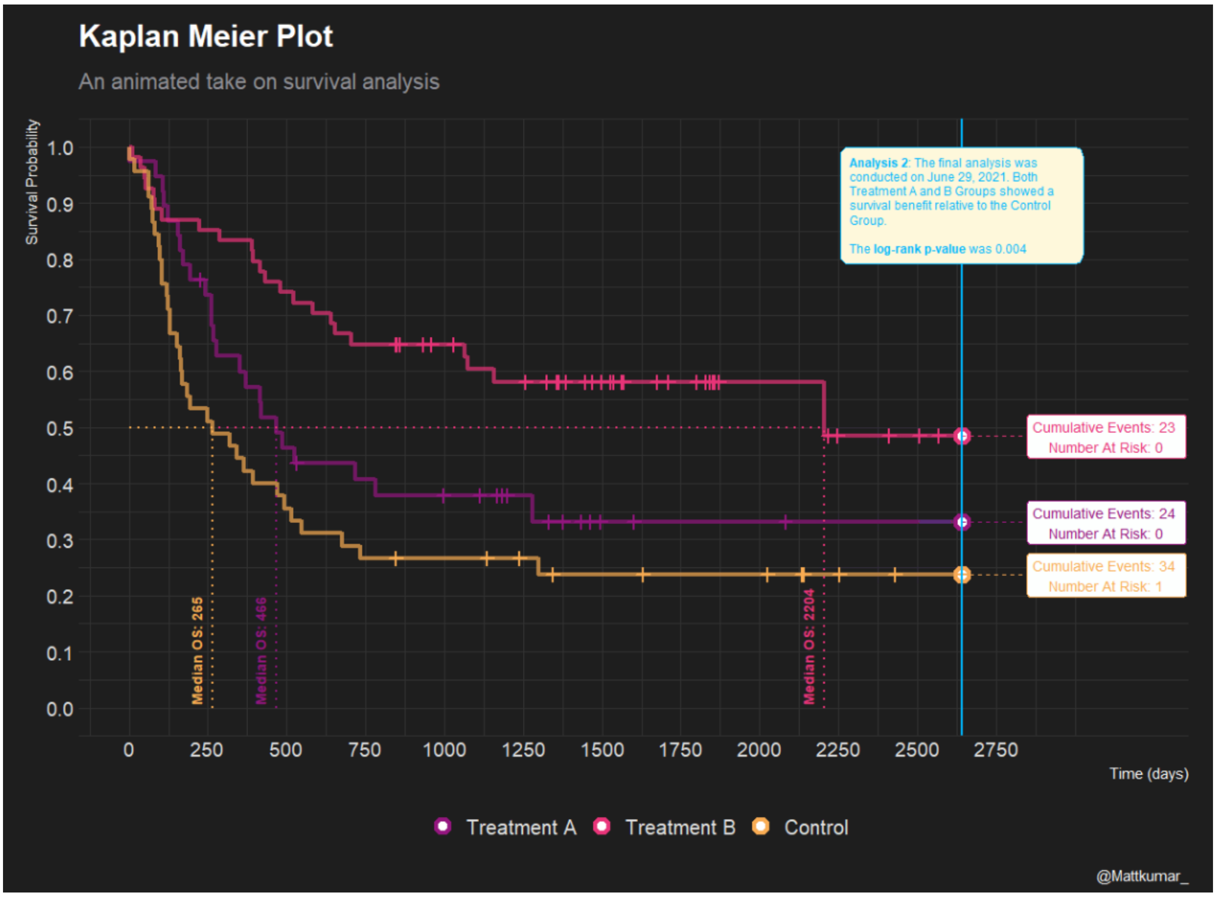


Nous retrouvons la médiane visible dans le coin du graphique de Minitab.

Pour ceux qui veulent apprendre à faire des des courbes de Kaplan-Meier magnifiques et animées avec annotations et pauses pendant l'animation, nous vous conseillons de consulter le code suivant:

https://github.com/mattkumar/tte_animations/blob/main/tte_animation.R

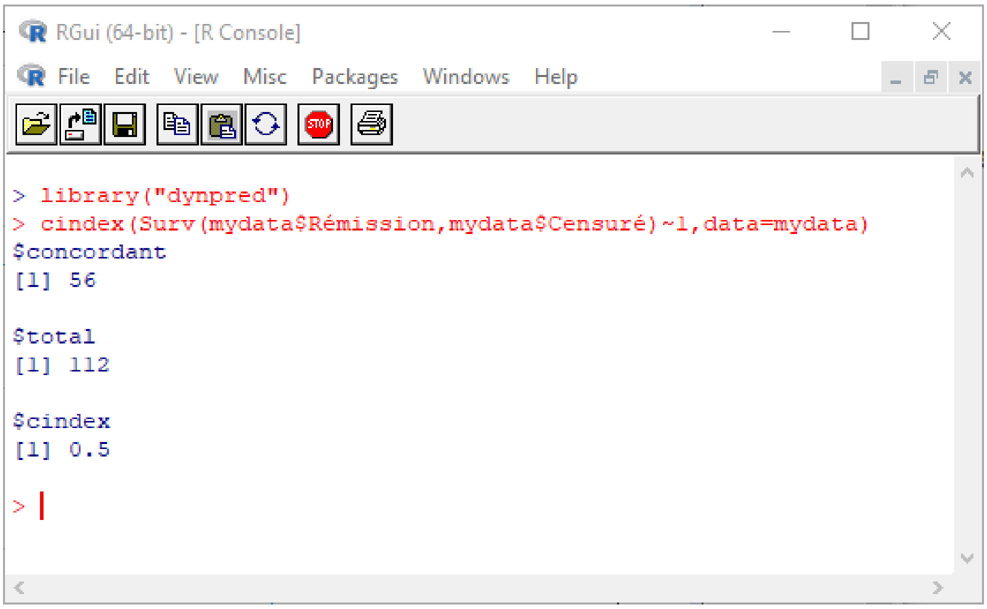
de Matthew Kumar (Senior Computational Scientist at Bayer Pharmaceuticals).



Index de concordance de Harrell

On ne peut pas reprendre ici l'exemple fait dans le cours théorique qui était exprès tout petit pour faire le calcul à la main. Nous allons donc reprendre les données de l'exemple précédent pour calculer l'index de concordance (exemple qui est malheureusement un peut trop gros pour faire les calculs à la main dans un temps raisonnable dans le cours théorique).

Pour cela nous allons utiliser la fonction **cindex( )** du package **dynpred**:



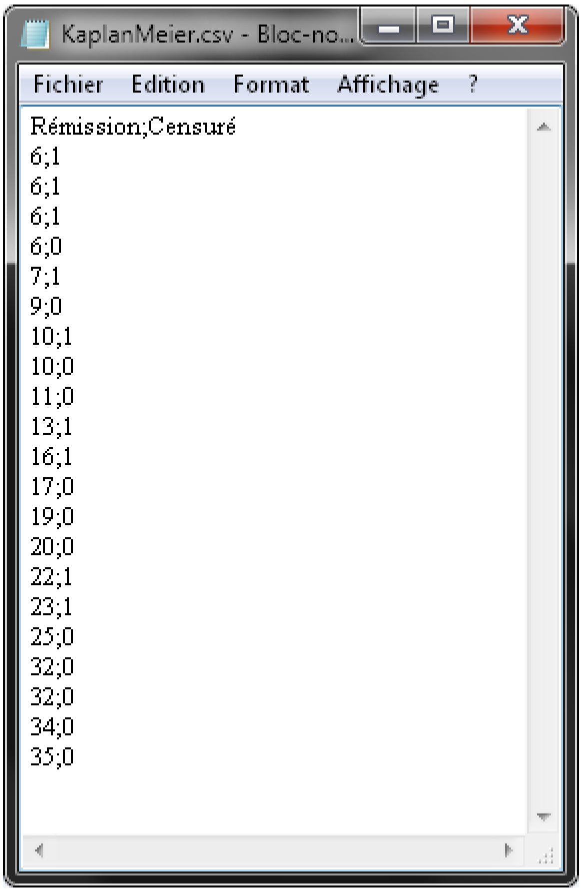
Vue la valeur du C index, notre modèle ne fait donc pas mieux que le hasard (pile ou face). Donc le modèle n'est pas utile dans son état actuel!

Exemple 287.: Ajustement de Weibull pour données censurées à droite

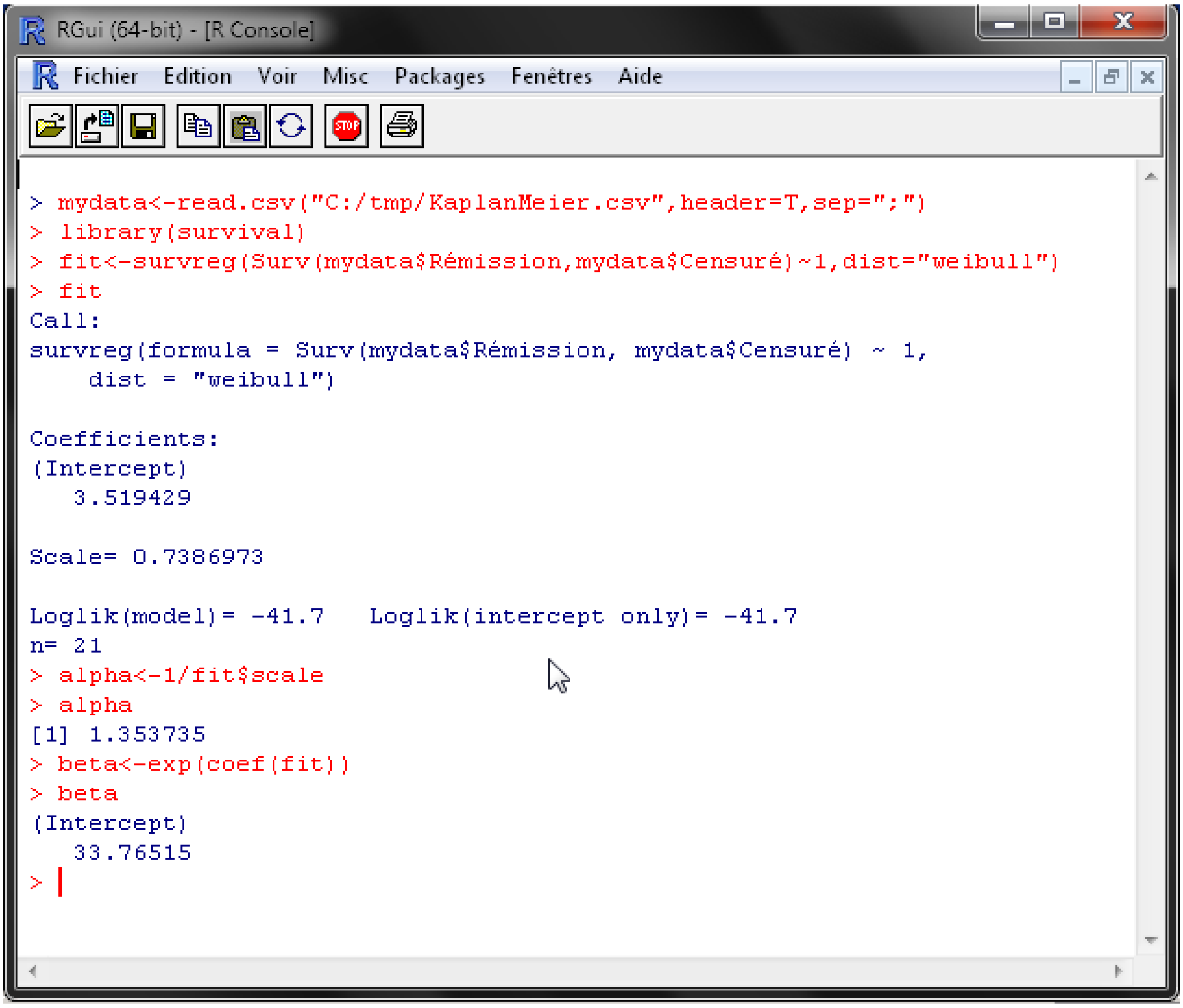
R 3.1.1 & **R 2.12.2**

Nous allons ici vérifier si nous retrouvons les résultats des calculs faits dans Microsoft Excel suit à la démonstration faite dans le cours de Statistique théorique.

Nous partons donc des mêmes données que celles pour l'analyse de Kaplan-Meier puisque ce sont celles que nous avons aussi utilisées pour le calcul dans Microsoft Excel après avoir fait le calcul à la main de l'estimateur de maximum de vraisemblance de la loi de Weibull à deux paramètres avec données censurées à droite:



Nous utilisons à nouveau le package **survival** avec la fonction **survreg( )** . Puisque cette dernière renvoie une paramétrisation peu connue dans le cadre de la survie nous allons devoir les transformer:



Donc pour le paramètre de forme nous obtenons 1.35 (à comparer avec Microsoft Excel où nous avions obtenu 1.18 et Minitab où nous avions obtenu 2.50) et pour le paramètre d'échelle 33.76 (à comparer avec Microsoft Excel où nous avions obtenu 27.16 et Minitab où nous avions obtenu 26.02).

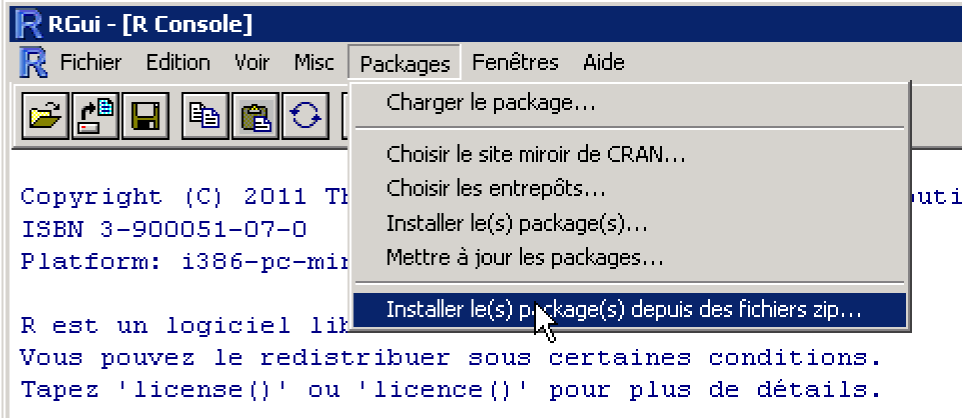
Maintenant pour obtenir au moins le graphique d'ajustement comme le font au minimal les logiciels commerciaux et sans passer son temps à recréer la roue avec un script il va falloir d'abord télécharger la version **2.12.2** de R disponible sur cette page:

http://cran.r-project.org/bin/windows/base/old/

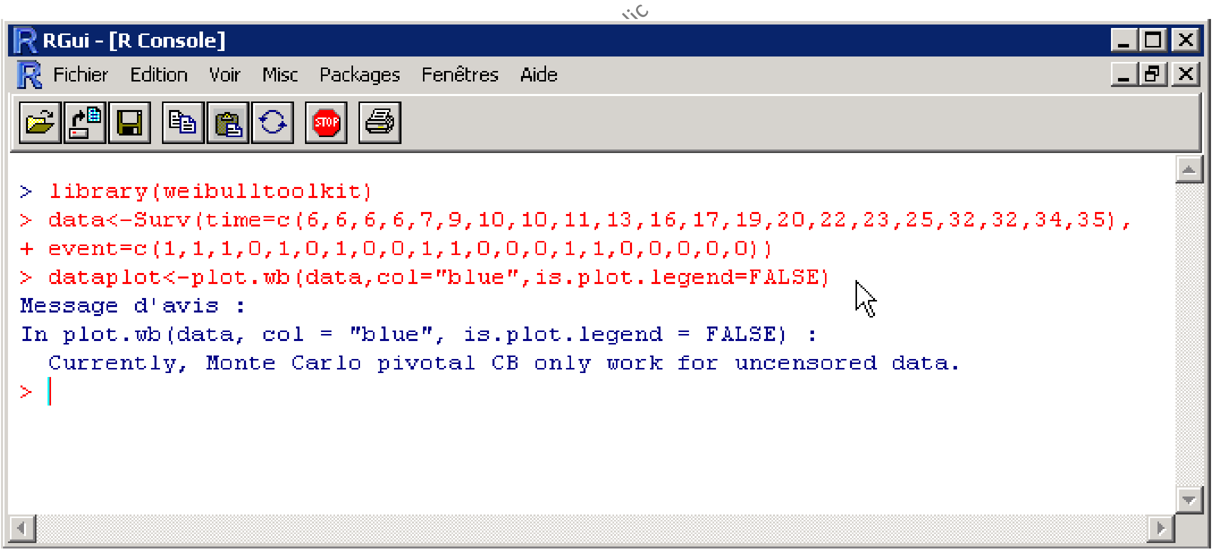
et ensuite télécharger en local le package **weibulltoolkit** disponible ici:

http://www.foxgo.net/10/post/2011/10/weibull-toolkit-for-r.html

Une fois l'installation de R 2.12.2 effectuée ainsi que celle du fichier Zip:



Une fois l'installation faite, nous pouvons utiliser sans autre le package:



Ce qui donnera:

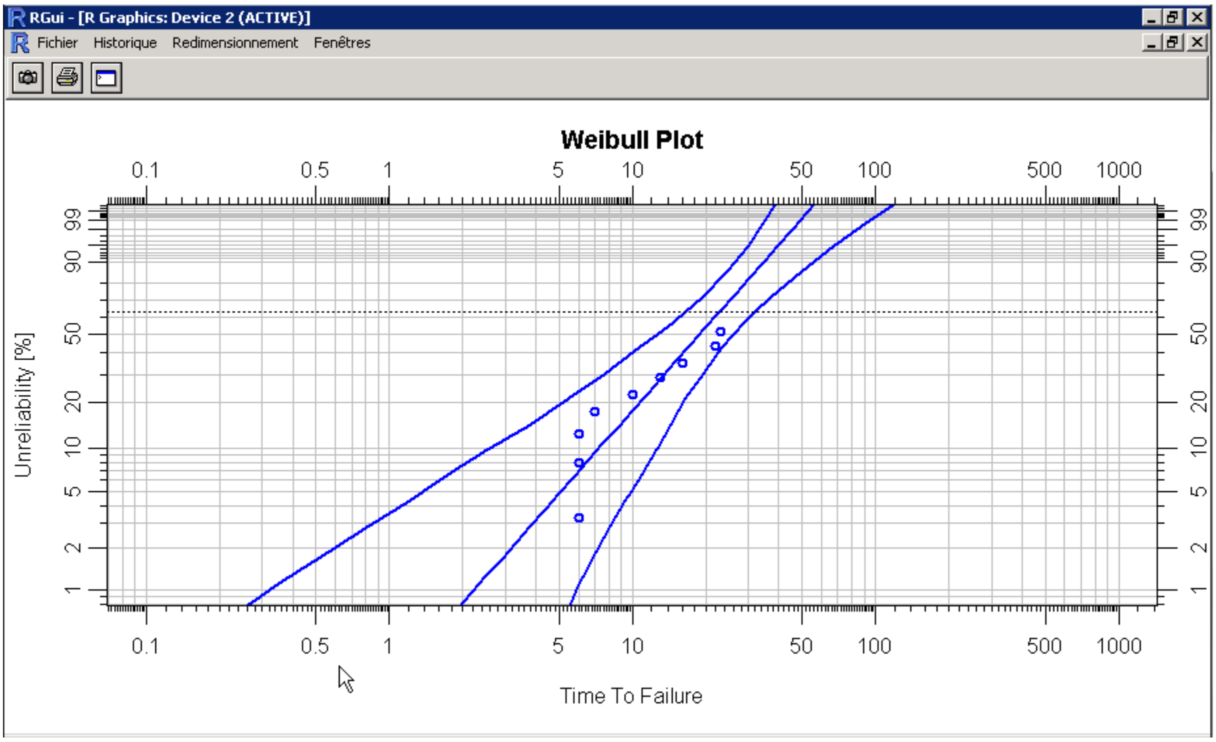
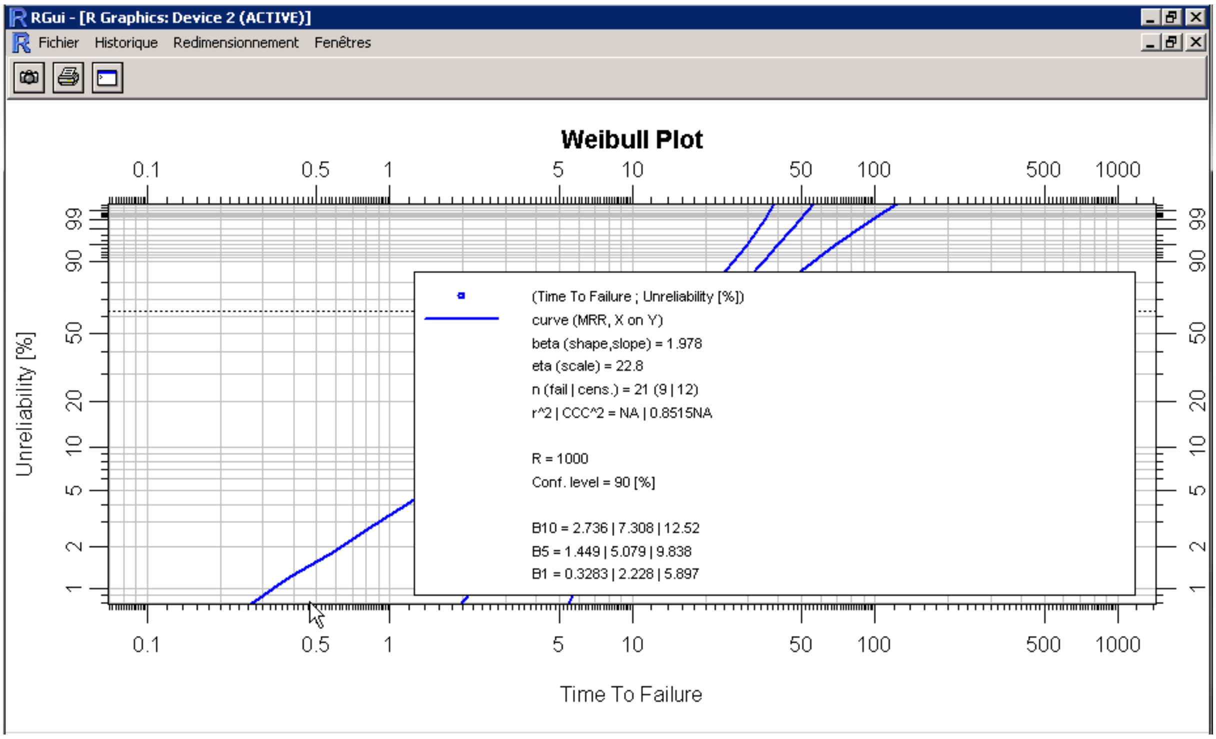


Figure 262 Graph de Weibull d'analyse de survie pour données censurées

En enlevant le paramètre *is.plot.legend=FALSE* nous obtenons une légende un peu envahissante:



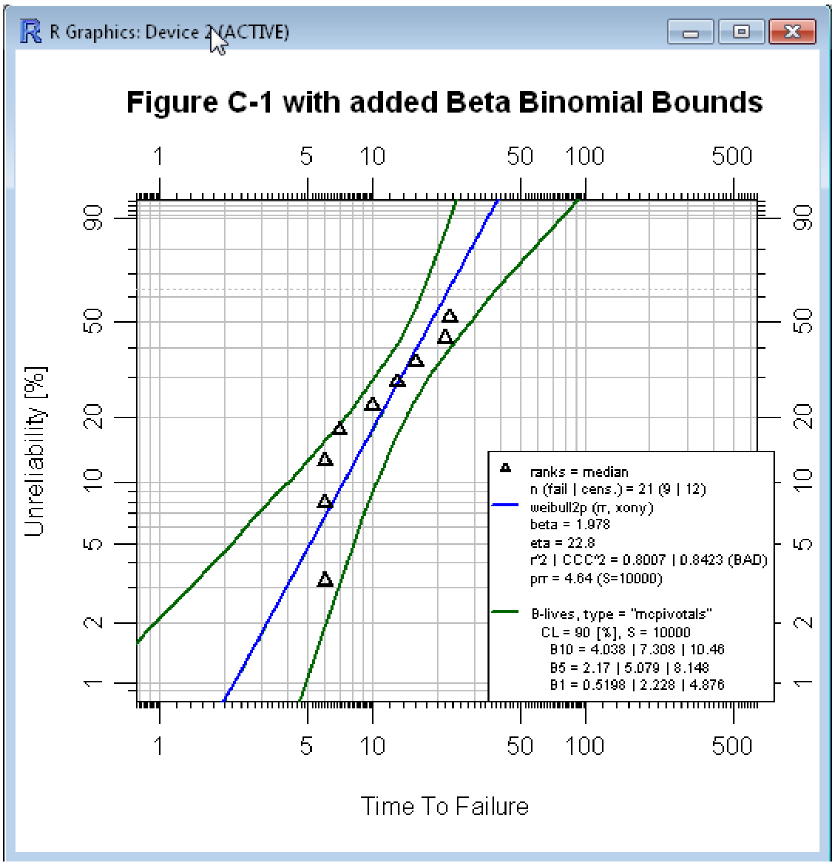
Ou sinon avec l'ensemble des packages **abrem** suivants qui ne nécessitent pas de télécharger une vieille version de R:

- abremPivotals_0.2.8.zip
- debias_0.1.9.zip
- pivotals_0.1.10.zip
- abrem_0.1.17.zip

Et en les installant à ce jour (début 2015) aussi manuellement, nous avons:

```
R Console
> library(abrem)
> AppBCData<-data.frame( time=c(6,6,6,6,7,9,10,10,11,13,16,17,19,20,22,23,25,32,32,34,35),
+ event=c(1,1,1,0,1,0,1,0,0,1,1,0,0,0,1,1,0,0,0,0,0))
> FigC.1<-Abrem(AppBCData, pch=2)
> FigC.1<-abrem.fit(FigC.1, col="blue")
> FigC.1<-abrem.conf(FigC.1, col="darkgreen")
Message d'avis :
In FUN(X[[1L]], ...) :
  calculateSingleConf: Currently, MC Pivotal bounds for (heavily) censored data
are still experimental and probably too optimistic.
> plot(FigC.1, xlim=c(1,500), ylim=c(.01,.90),
+ main="Figure C-1 with added Beta Binomial Bounds",legend.text.size=0.6)
>
```

Ce qui donne (très intéressant de comparer avec la précédente approche):



Exemple 288.: Régression à modèle de hasard proportionnel de Cox

R 3.0.2

Nous allons reprendre ici exactement le même exemple à hasard proportionnel non stratifié dépendant du temps que celui fait dans le cours théorique après la démonstration mathématique (pénible) des notions fondamentales de ce modèle....

Nous allons pour cela utiliser la commande **coxph( )** du package **survival**:

```
> library(survival)
> test<-list(start=c(0,0,0,0),
+ stop=c(6,98,189,374),
+ event=c(1,1,1,1),
+ BMI=c(31.4,21.5,27.1,22.7))
> summary(coxph(Surv(start,stop,event)~BMI,test))
Call:
coxph(formula = Surv(start, stop, event) ~ BMI, data = test)

  n= 4, number of events= 4

      coef exp(coef) se(coef)     z Pr(>|z|)
BMI 0.2227    1.2495   0.2161 1.031    0.303

    exp(coef) exp(-coef) lower .95 upper .95
BMI     1.249     0.8003    0.8181     1.908

Concordance= 0.667  (se = 0.245 )
Rsquare= 0.263   (max possible= 0.796 )
Likelihood ratio test= 1.22  on 1 df,   p=0.2693
Wald test            = 1.06  on 1 df,   p=0.3026
Score (logrank) test = 1.24  on 1 df,   p=0.2663

> 
```

Nous retrouvons donc le même résultat qu'à la main (et qu'avec MedCalc!) pour le coefficient. Pour le reste, cela sort à ce jour du contenu du cours théorique!

Les statistiques du rapport de vraisemblance, de Wald et du score chi carré au bas de la sortie sont des tests asymptotiquement équivalents de l'hypothèse nulle omnibus que tous les β sont nuls. Nous voyons ici que nous ne pouvons pas rejeter cette hypothèse…

Et au niveau graphique:

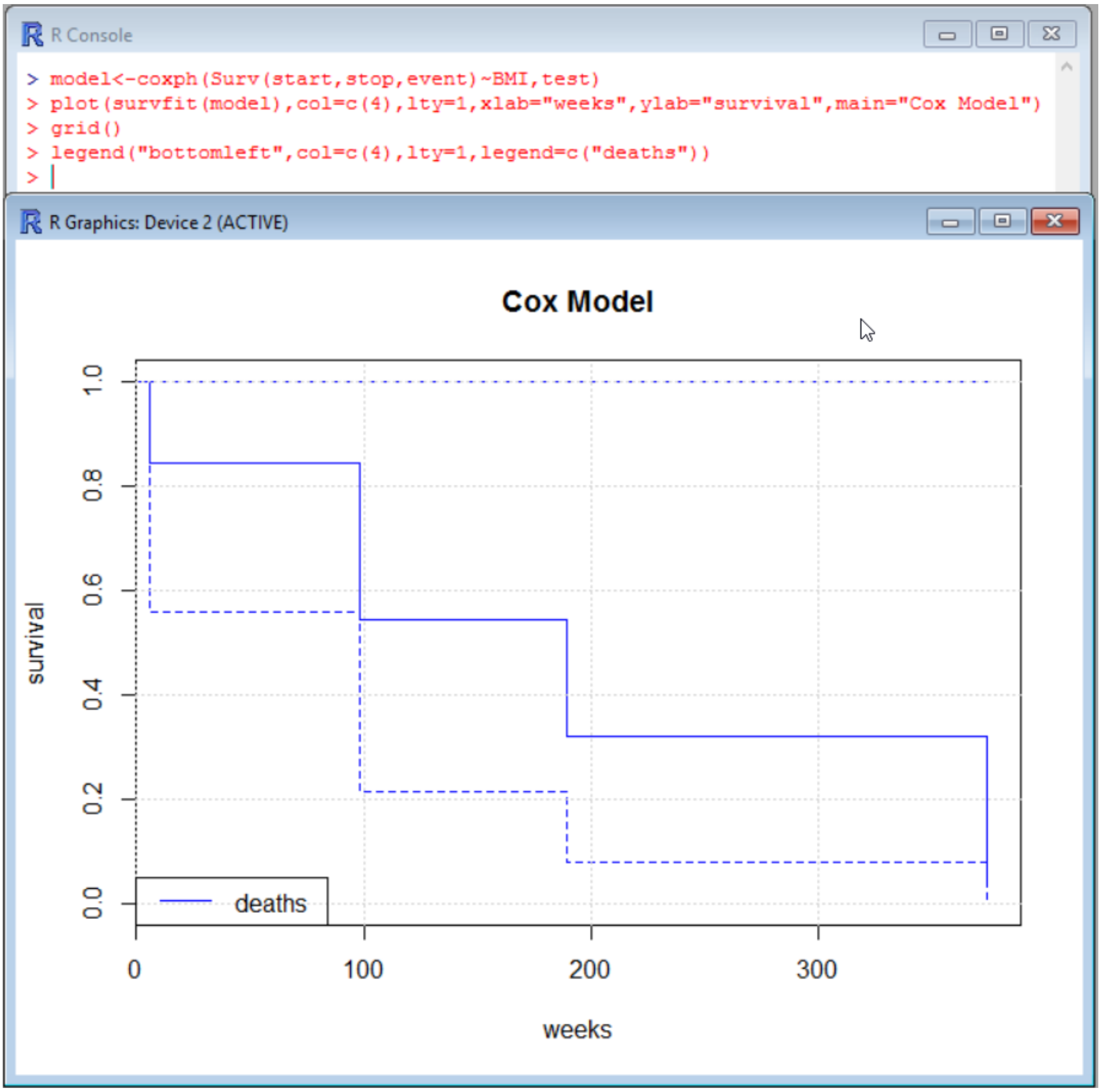


Figure 263 Graphique de survie de Cox

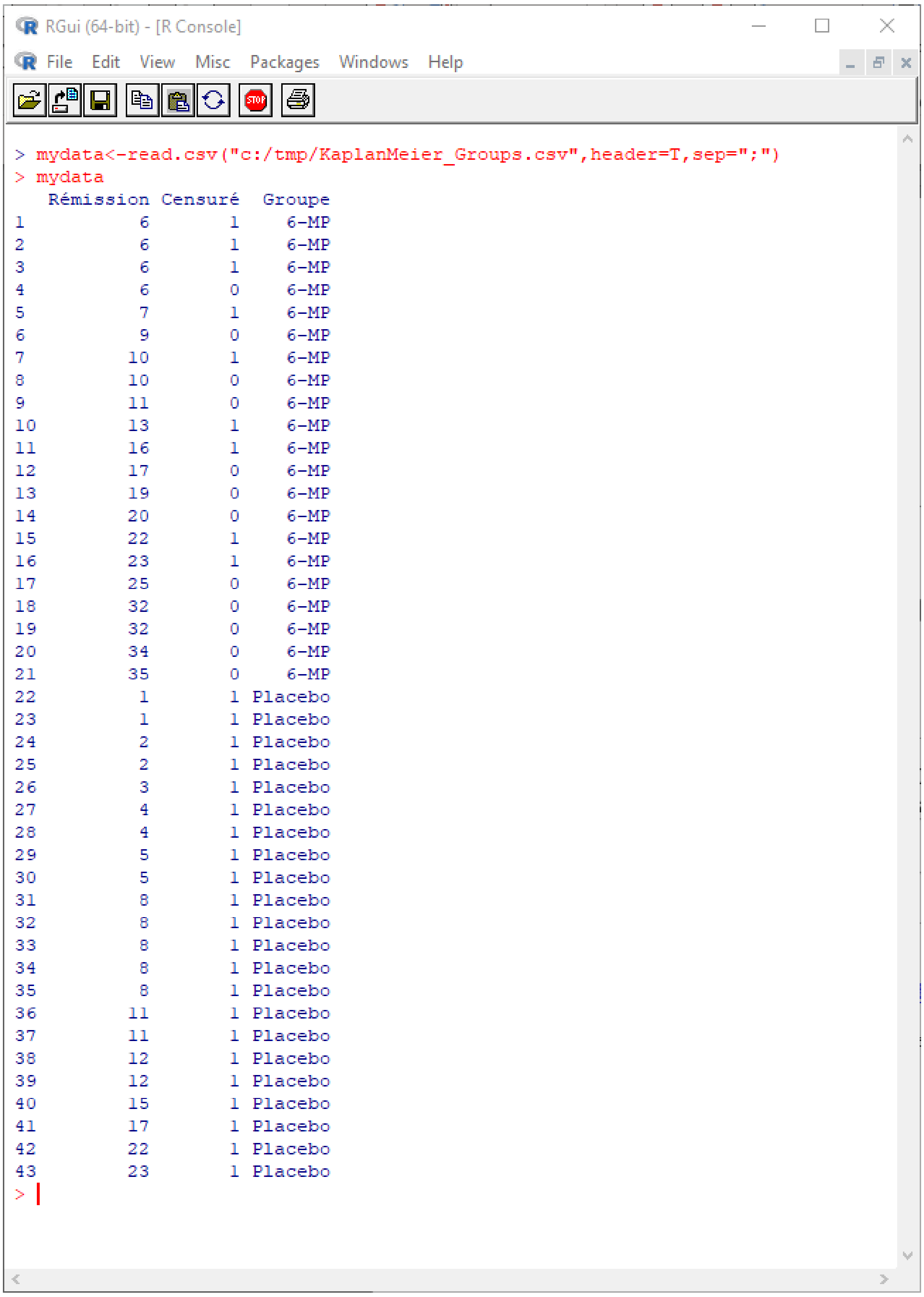
Exemple 289.: Comparaisons des courbes de Survie de Kaplan-Meier, Cox, Aalens, paramètriques (exponentiel et Weibull) et par forêts aléatoires

R 3.0.2

L'exemple qui suit est repris et traduit de l'article suivant de Anri Smith (2020-04-03):

https://rstudio-pubs-static.s3.amazonaws.com/611640_351fb7c7d8fd4ae6bf6dc27dbb2f1fb7.html

Nous allons reprendre les mêmes données que dans le cours théorique:

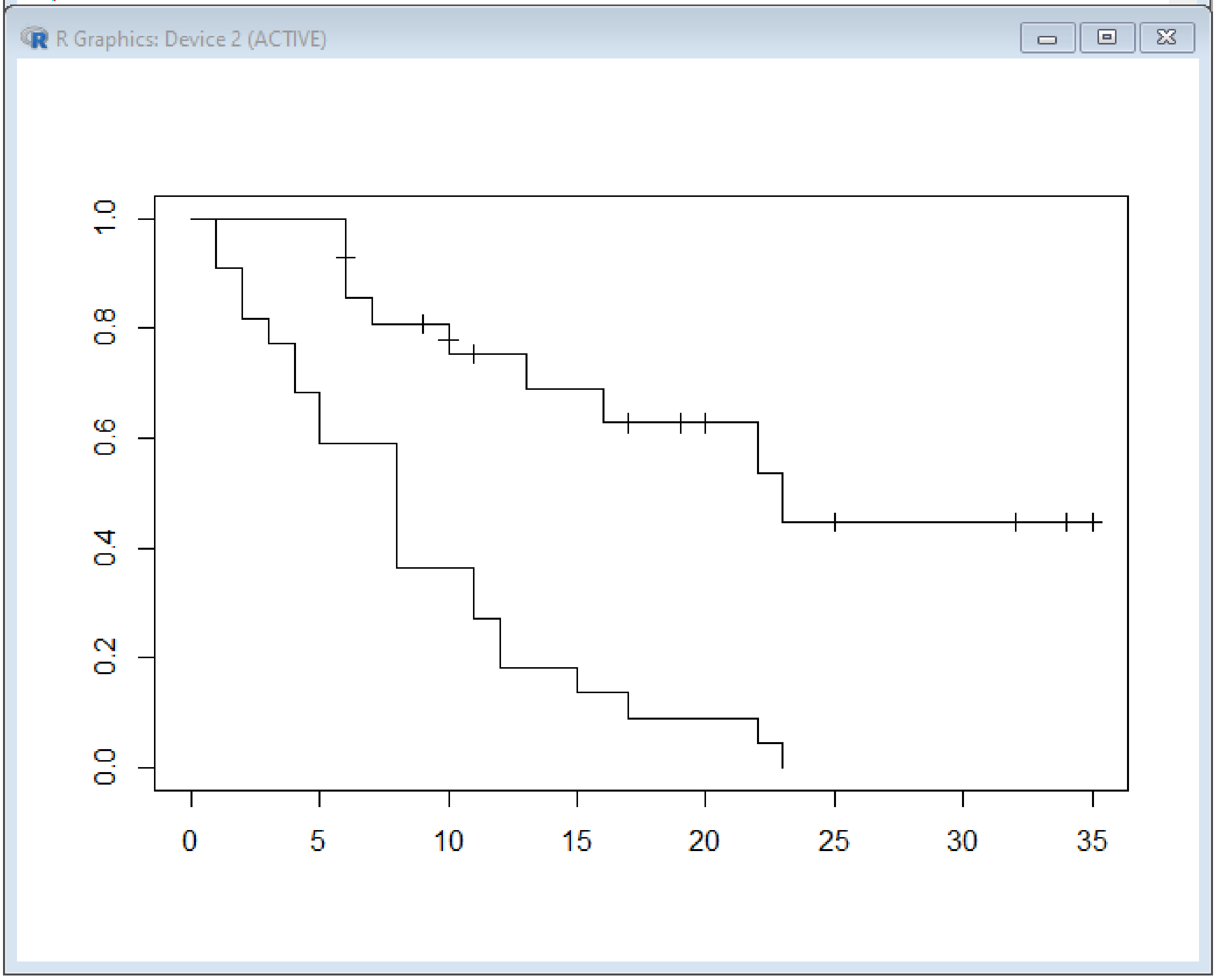



```
> mydata<-read.csv("c:/tmp/KaplanMeier_Groups.csv",header=T,sep=";")
> mydata
   Rémission Censuré  Groupe
1          6       1    6-MP
2          6       1    6-MP
3          6       1    6-MP
4          6       0    6-MP
5          7       1    6-MP
6          9       0    6-MP
7         10       1    6-MP
8         10       0    6-MP
9         11       0    6-MP
10        13       1    6-MP
11        16       1    6-MP
12        17       0    6-MP
13        19       0    6-MP
14        20       0    6-MP
15        22       1    6-MP
16        23       1    6-MP
17        25       0    6-MP
18        32       0    6-MP
19        32       0    6-MP
20        34       0    6-MP
21        35       0    6-MP
22         1       1 Placebo
23         1       1 Placebo
24         2       1 Placebo
25         2       1 Placebo
26         3       1 Placebo
27         4       1 Placebo
28         4       1 Placebo
29         5       1 Placebo
30         5       1 Placebo
31         8       1 Placebo
32         8       1 Placebo
33         8       1 Placebo
34         8       1 Placebo
35         8       1 Placebo
36        11       1 Placebo
37        11       1 Placebo
38        12       1 Placebo
39        12       1 Placebo
40        15       1 Placebo
41        17       1 Placebo
42        22       1 Placebo
43        23       1 Placebo
> 
```

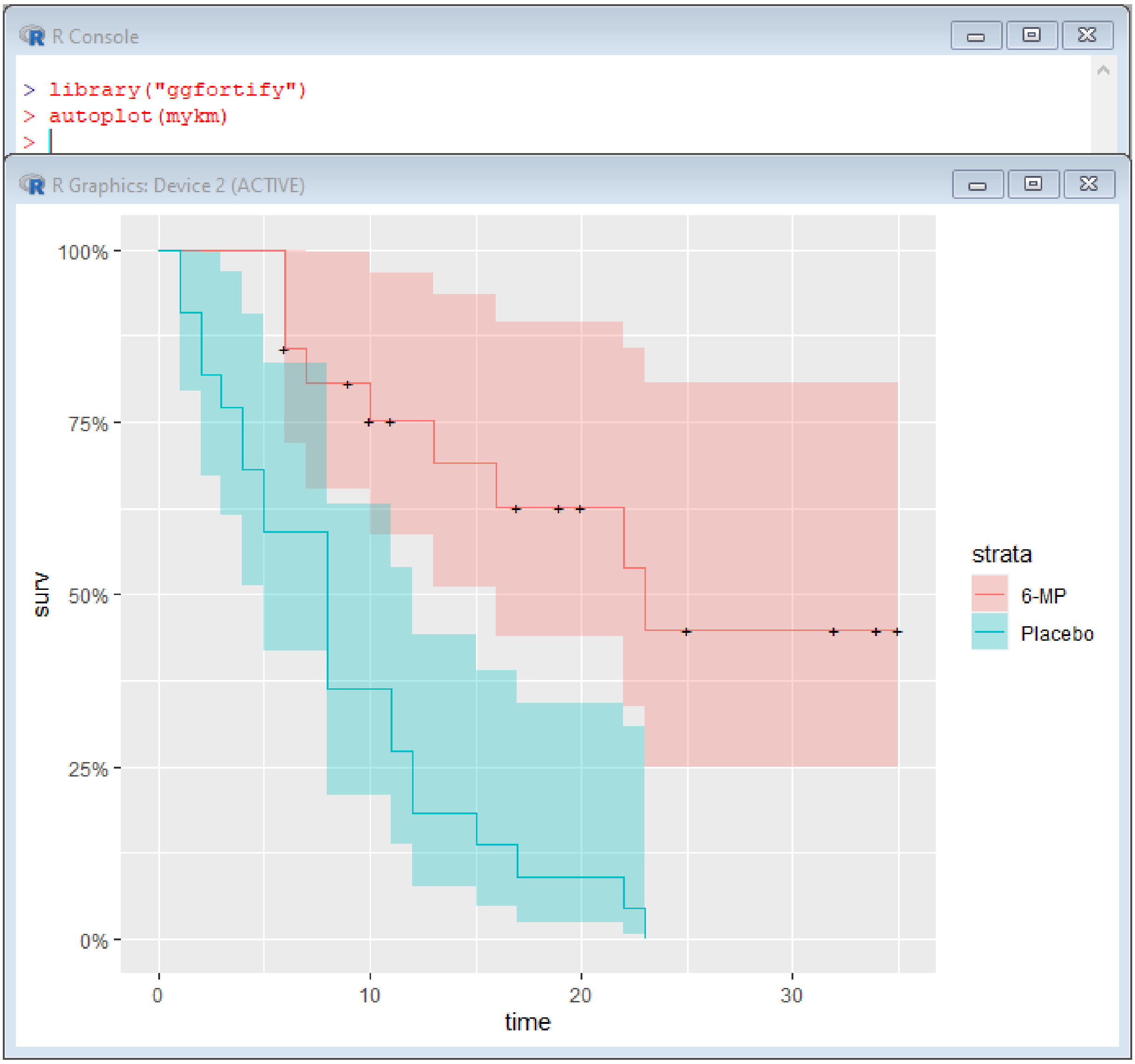
Modèle de Kaplan-Meier

Faisons d'abord une analyse de Kaplan-Meier traditionnelle:

```
> library("survival")
> Surv(mydata$Rémission, mydata$Censuré)
 [1]  6   6   6   6+  7   9+ 10  10+ 11+ 13  16  17+ 19+ 20+ 22  23  25+ 32+ 32+
[20] 34+ 35+  1   1   2   2   3   4   4   5   5   8   8   8   8   8  11  11  12
[39] 12  15  17  22  23
>
> mykm <- survfit(Surv(Rémission, Censuré) ~ Groupe, data = mydata)
> plot(mykm, mark.time = TRUE)
>
```



Ou avec la fonction **autoplot( )** du package **ggfortify**:



Exécutons un test de log-rank:

$$Z = \frac{\sum_{j=1}^{J}(O_{1j}-E_{1j})}{\sqrt{\sum_{j=1}^{J} V_j}} \to N(0,1)$$

avec la fonction **survdiff( )**:

```
> library("ggfortify")
> autoplot(mykm)
> survdiff(Surv(mydata$Rémission, mydata$Censuré) ~ Groupe, data = mydata)
Call:
survdiff(formula = Surv(mydata$Rémission, mydata$Censuré) ~ Groupe,
    data = mydata)

                N Observed Expected (O-E)^2/E (O-E)^2/V
Groupe=6-MP    21        9     19.5      5.67      17.1
Groupe=Placebo 22       22     11.5      9.64      17.1

 Chisq= 17.1  on 1 degrees of freedom, p= 4e-05
>
```

Modèle à hasards proportionnels de Cox

Faisons maintenant l'analyse sur les mêmes données mais avec le modèle à hasard proportionnel de Cox:

```
> (cox<-coxph(Surv(Rémission, Censuré) ~ Groupe, data = mydata))
Call:
coxph(formula = Surv(Rémission, Censuré) ~ Groupe, data = mydata)

                 coef exp(coef) se(coef)     z        p
GroupePlacebo 1.5821    4.8649   0.4103 3.856 0.000115

Likelihood ratio test=16.9  on 1 df, p=3.933e-05
n= 43, number of events= 31
> summary(cox)
Call:
coxph(formula = Surv(Rémission, Censuré) ~ Groupe, data = mydata)

  n= 43, number of events= 31

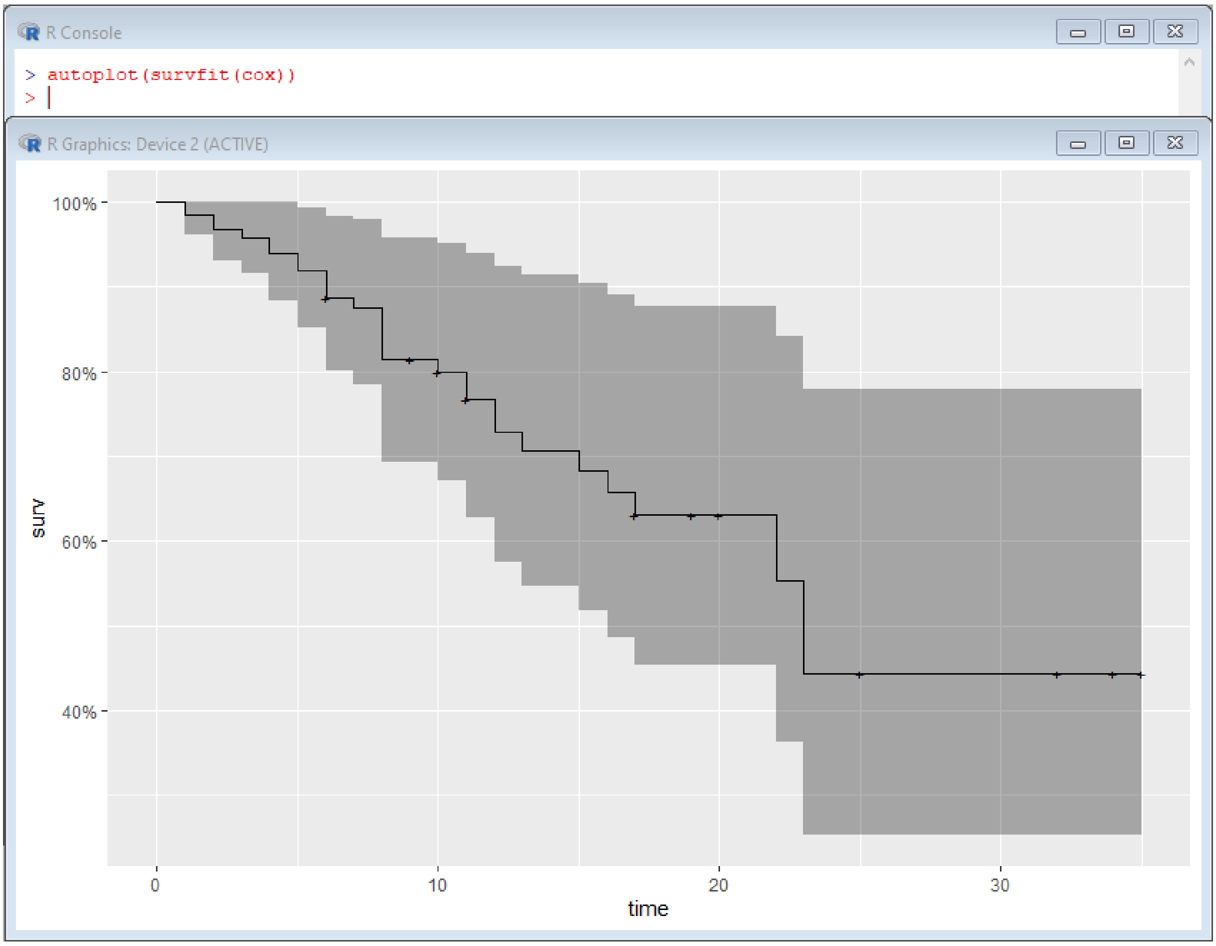
                 coef exp(coef) se(coef)     z Pr(>|z|)
GroupePlacebo 1.5821    4.8649   0.4103 3.856 0.000115

              exp(coef) exp(-coef) lower .95 upper .95
GroupePlacebo     4.865     0.2056     2.177     10.87

Concordance= 0.689  (se = 0.041 )
Likelihood ratio test= 16.9  on 1 df,   p=4e-05
Wald test            = 14.87  on 1 df,   p=1e-04
Score (logrank) test = 17.64  on 1 df,   p=3e-05

>
```

Là aussi on peut faire un graphique:



Modèle de régressaion de Aalen

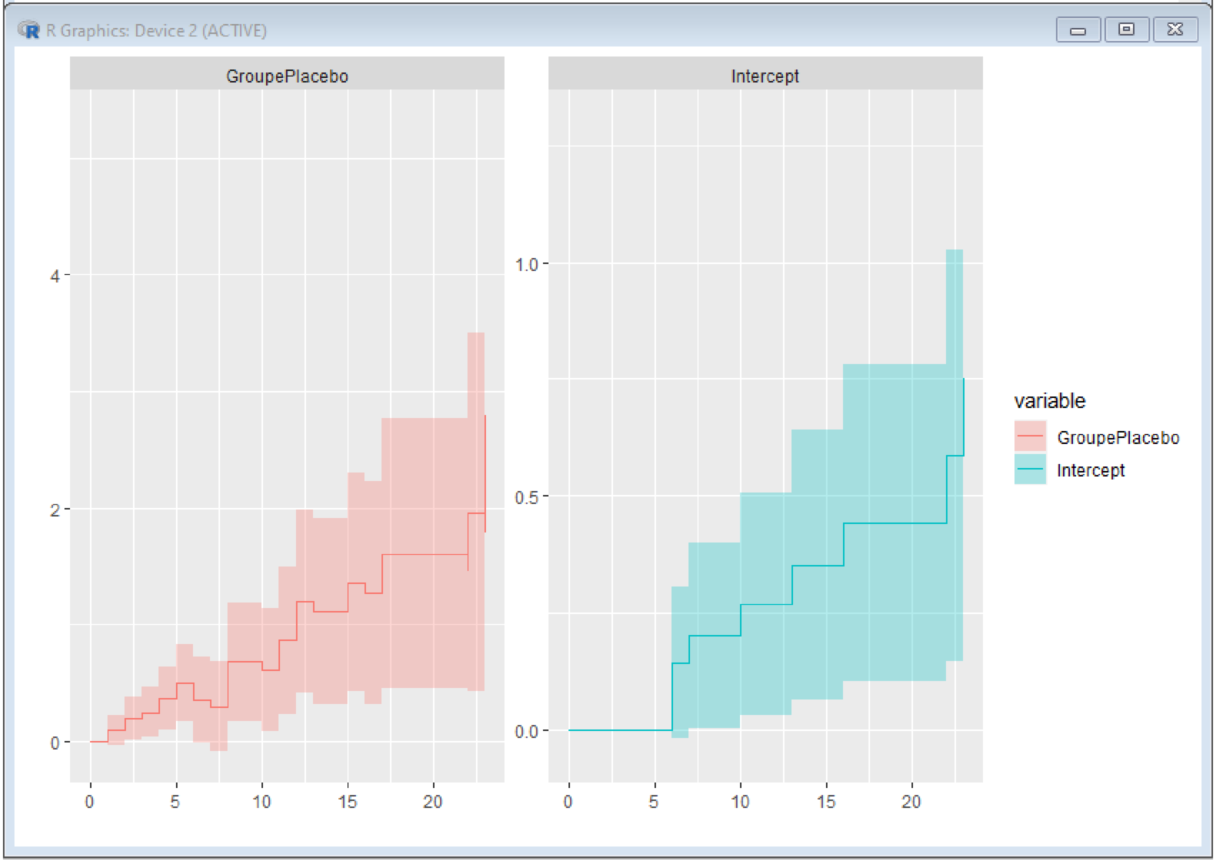
Maintenant voyons avec la méthode de régression de Aalen que nous n'avons pas étudié dans le cours théorique avec la fonction **aareg( )**:

```
> (aalen <- aareg(Surv(Rémission, Censuré) ~ Groupe, data = mydata))
Call:
aareg(formula = Surv(Rémission, Censuré) ~ Groupe, data = mydata)

  n= 43 
    17 out of 17 unique event times used

              slope   coef se(coef)    z        p
Intercept     0.040 0.0186   0.0062 3.00 0.002700
GroupePlacebo 0.179 0.0543   0.0156 3.48 0.000506

Chisq=12.09 on 1 df, p=0.000506; test weights=aalen
> autoplot(aalen)
> 
```



Modèle paramétrique exponentiels et Weibull

Toujours sur les mêmes données faisonss une analyse avec les modèles paramétriques vus dans le cours théorique avec la fonction **flexsurvreg( )** du package **flexsurv**:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("flexsurv")
> (weib_mod<- flexsurvreg(Surv(Rémission, Censuré) ~ Groupe, data = mydata, dist = "weibull"))
Call:
flexsurvreg(formula = Surv(Rémission, Censuré) ~ Groupe, data = mydata,
    dist = "weibull")

Estimates:
               data mean  est     L95%    U95%    se      exp(est)  L95%    U95%
shape              NA      1.391   1.048   1.847   0.201      NA        NA      NA
scale              NA     33.384  20.585  54.141   8.236      NA        NA      NA
GroupePlacebo    0.512    -1.261  -1.853  -0.669   0.302   0.283     0.157   0.512

N = 43,  Events: 31,  Censored: 12
Total time at risk: 549
Log-likelihood = -109.3629, df = 3
AIC = 224.7257

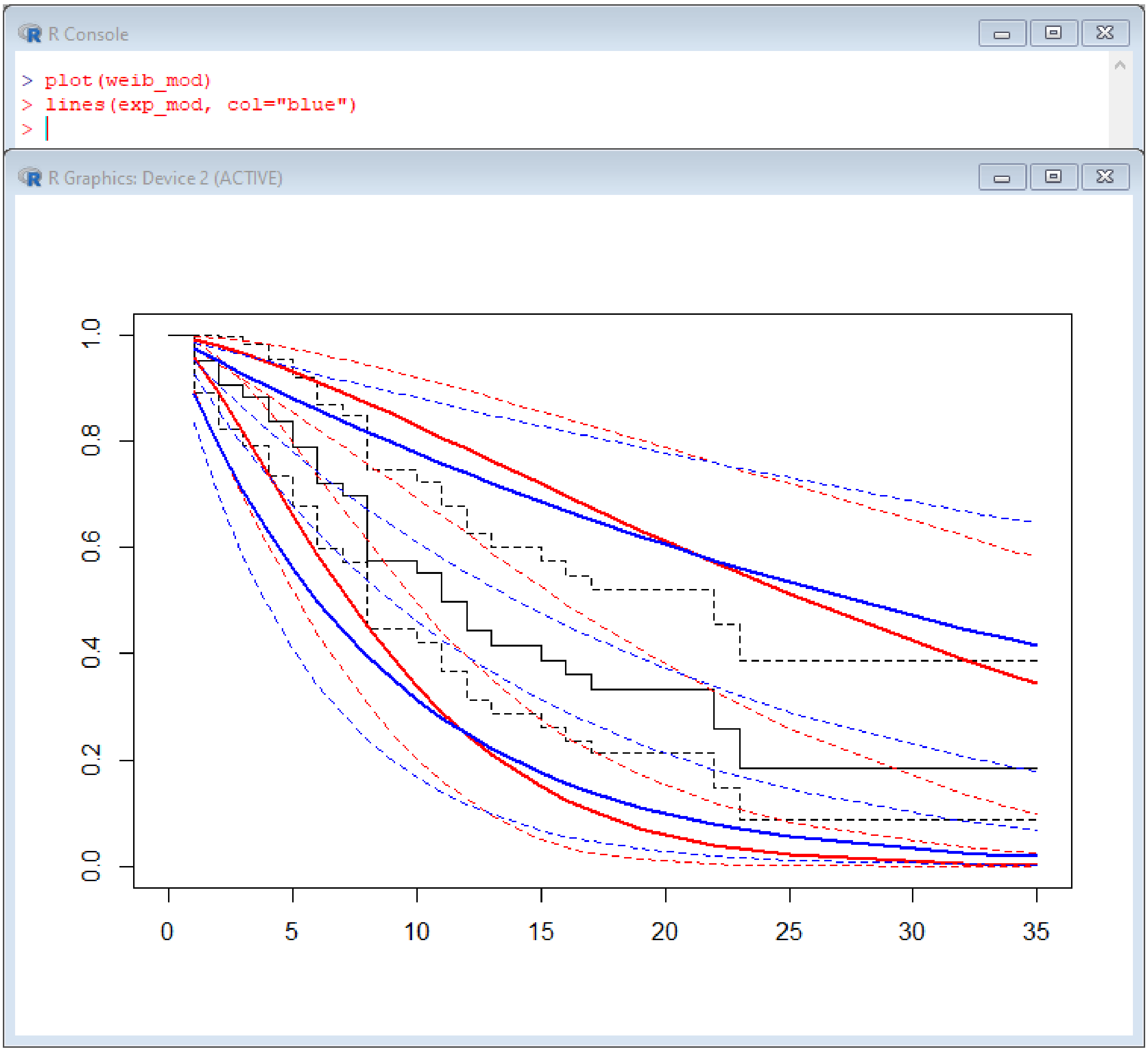
> (exp_mod <- flexsurvreg(Surv(Rémission, Censuré) ~ Groupe, data = mydata, dist = "exp"))
Call:
flexsurvreg(formula = Surv(Rémission, Censuré) ~ Groupe, data = mydata,
    dist = "exp")

Estimates:
               data mean  est      L95%     U95%     se       exp(est)  L95%     U95%
rate               NA     0.02507  0.01304  0.04818  0.00836       NA        NA        NA
GroupePlacebo    0.51163  1.53012  0.75459  2.30564  0.39568  4.61871   2.12674  10.03062

N = 43,  Events: 31,  Censored: 12
Total time at risk: 549
Log-likelihood = -111.6065, df = 2
AIC = 227.213

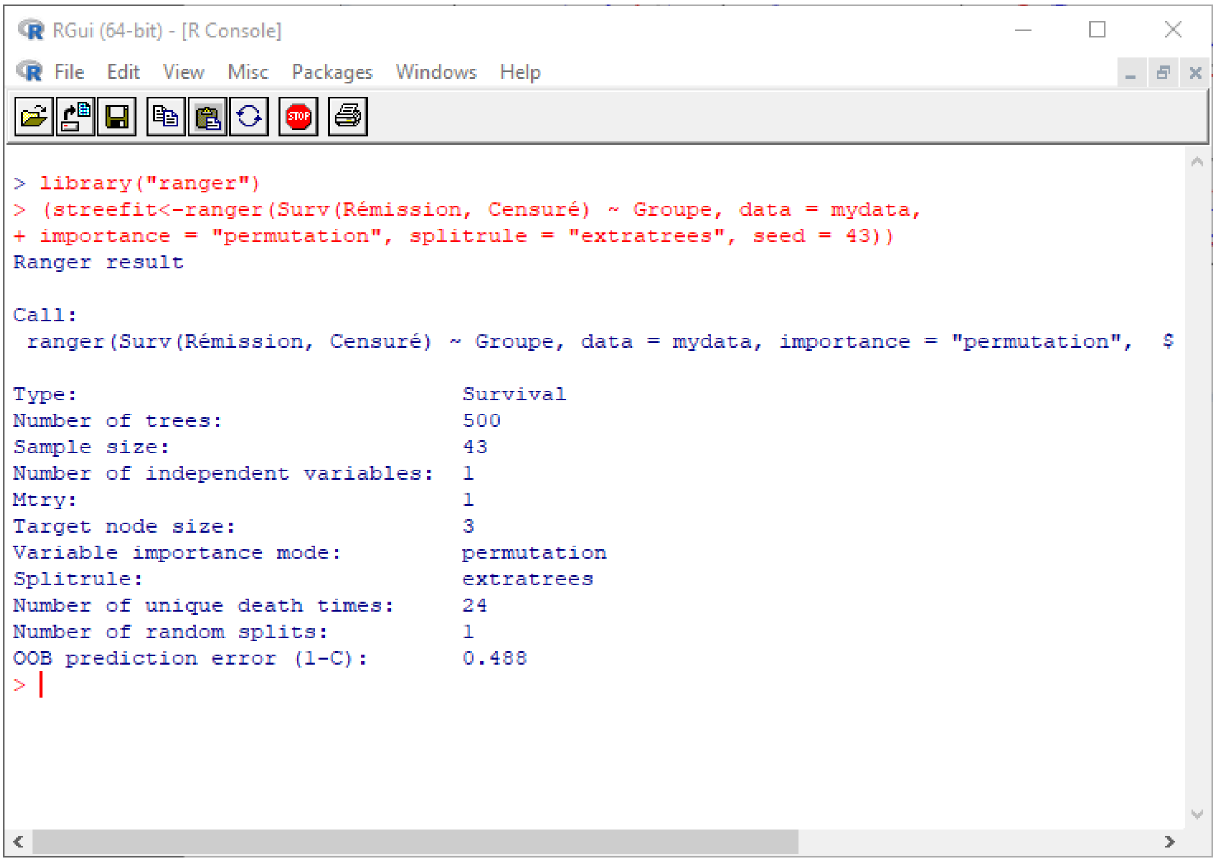
> |
```

Et graphiquement en superposant les deux:



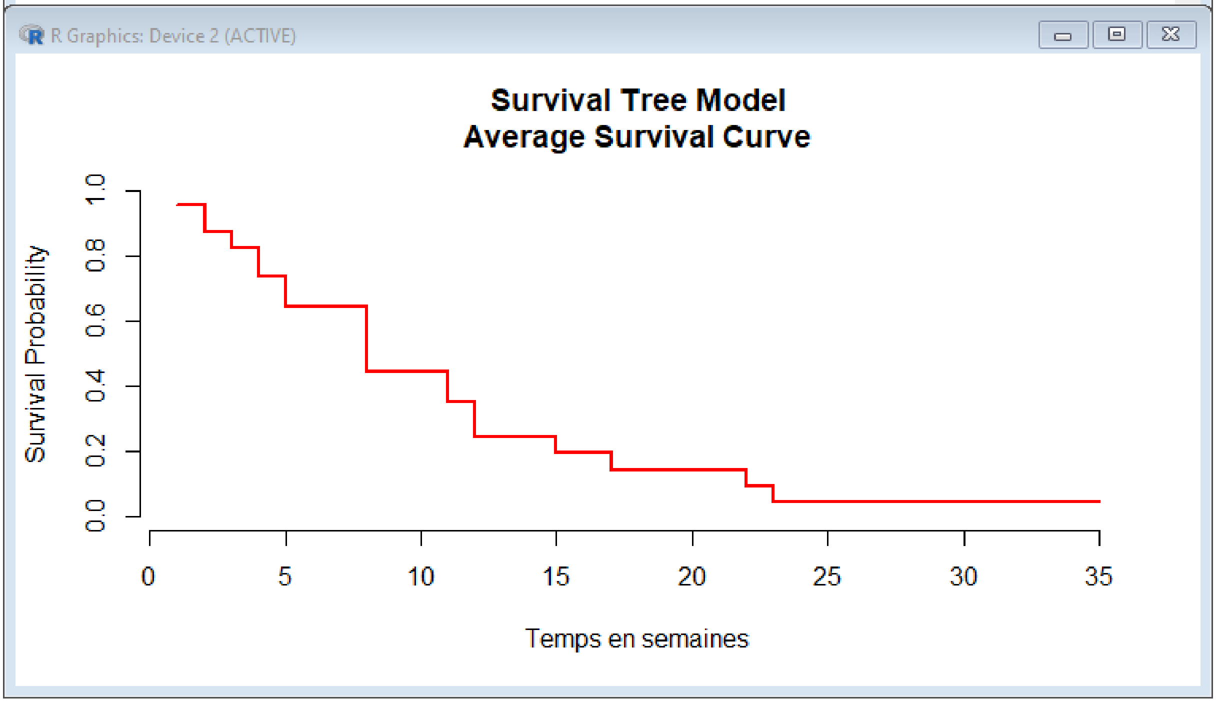
Arbres de Survies

Nous entraînons d'abord le modèle avec la fonction **ranger( )** du package **ranger**:



Et nous pouvons faire un graphique des temps de décès médians:

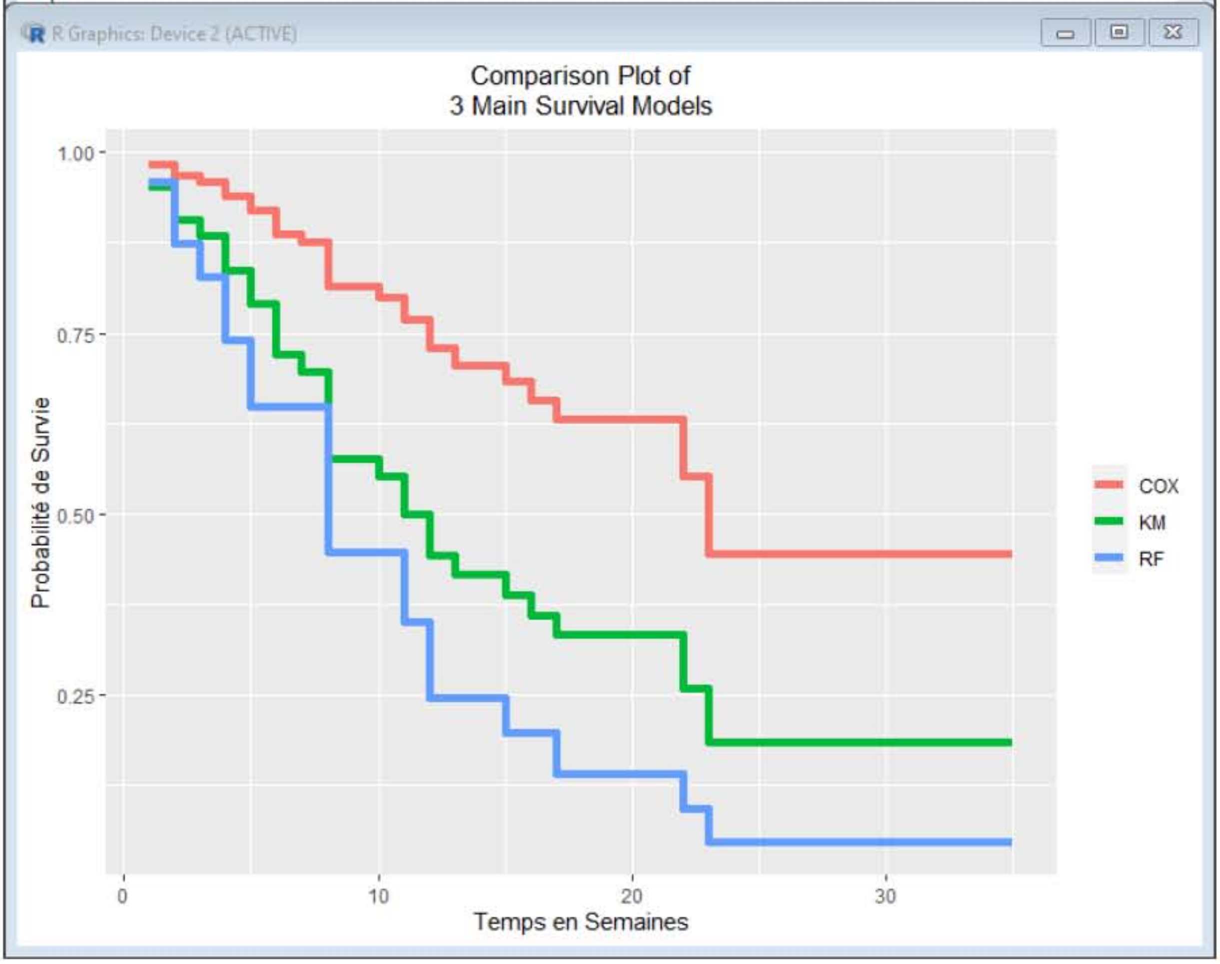
```
> #calcul du temps de survie médian
> death_times <- streefit$unique.death.times
> surv_prob <- data.frame(streefit$survival)
> med_prob <-sapply(surv_prob,median)
>
> #plot de l'arbre de régression de survie médian
> plot(death_times, med_prob,type = "s", ylim = c(0,1), col = "red", lwd = 2, bty = "n",
+      ylab = "Survival Probability", xlab = "Temps en semaines",
+ main = "Survival Tree Model\nAverage Survival Curve")
>
```



Comparaisons KM, Cox et RF

Tout est dans le titre!

```
> mykml <- survfit(Surv(Rémission, Censuré) ~ 1, data = mydata)
> mykml <- survfit(Surv(Rémission, Censuré) ~ 1, data = mydata)
> kmdf <- data.frame(mykml$time, mykml$surv, KM = rep("KM", 24))
> names(kmdf) <- c("Temps en Semaines","Probabilité de Survie","Modèle")
>
> coxfit <- survfit(cox)
> coxdf <- data.frame(coxfit$time,coxfit$surv,COX = rep("COX", 24))
> names(coxdf) <- c("Temps en Semaines","Probabilité de Survie","Modèle")
>
> rfdf <- data.frame(death_times,med_prob,RF = rep("RF", 24))
> names(rfdf) <- c("Temps en Semaines","Probabilité de Survie","Modèle")
>
> plotdf <- rbind(kmdf,coxdf,rfdf)
>
> p <- ggplot(plotdf, aes(x = plotdf[,1] , y = plotdf[,2] , color = plotdf[,3]))
> p + geom_step(size = 2) + labs(x = "Temps en Semaines", y = "Probabilité de Survie",
+   color = "", title = "Comparison Plot of\n3 Main Survival Models") +
+   theme(plot.title = element_text(hjust = 0.5))
>
```



19. Plans d'expérience

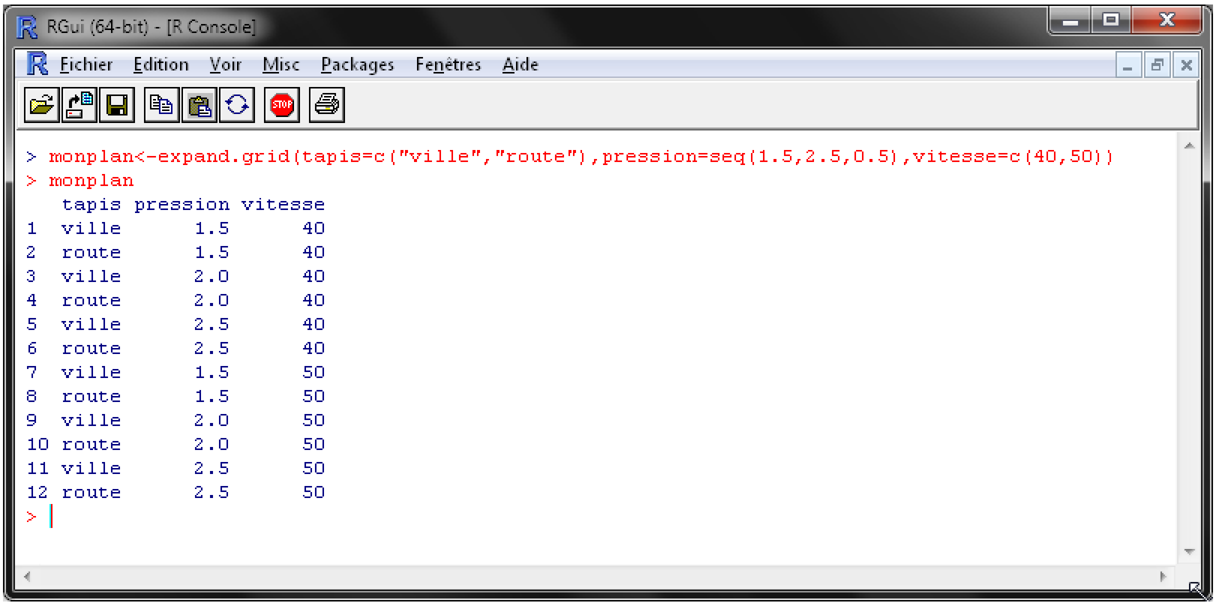
Le but ici va être de voir dans quelle mesure nous pouvons reproduire dans R les plans d'expérience vu dans le cours théorique et comparer les résultats au logiciel Minitab que nous avons utilisé dans le cours de Génie Industriel.

Remarque: À ce jour (début 2014), R est très loin des possibilités des logiciels comme JMP ou Minitab pour les plans d'expérience mais rien ne dit que dans quelques années...

Exemples 290.: Générer un plan factoriel complet (sans package)

R 3.0.2

Nous allons ici générer un plan d'expérience complet pour pouvoir prendre nos mesures avec la fonction **expand.grid( )** :



Ensuite, n'oubliez pas que vous pouvez éditer directement les données dans R avec la commande **edit(monplan)**. Ce sera tout pour le besoin le plus élémentaire imaginable...

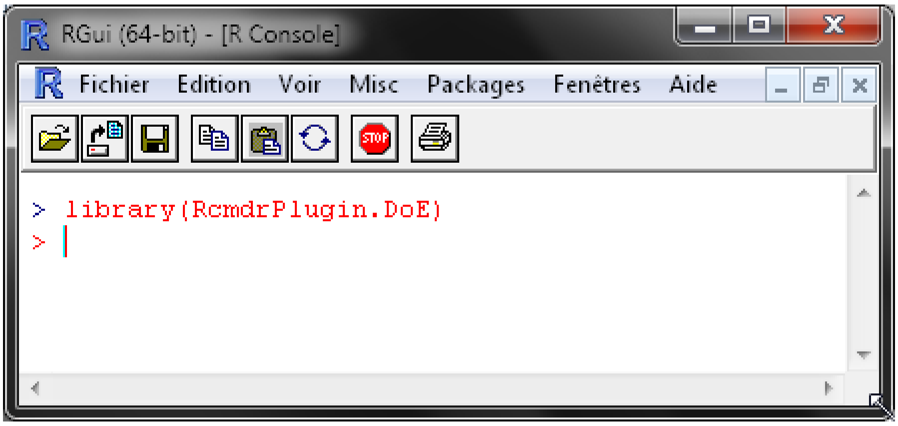
Exemples 291.: Constructions de plans d'expériences

R 3.0.2

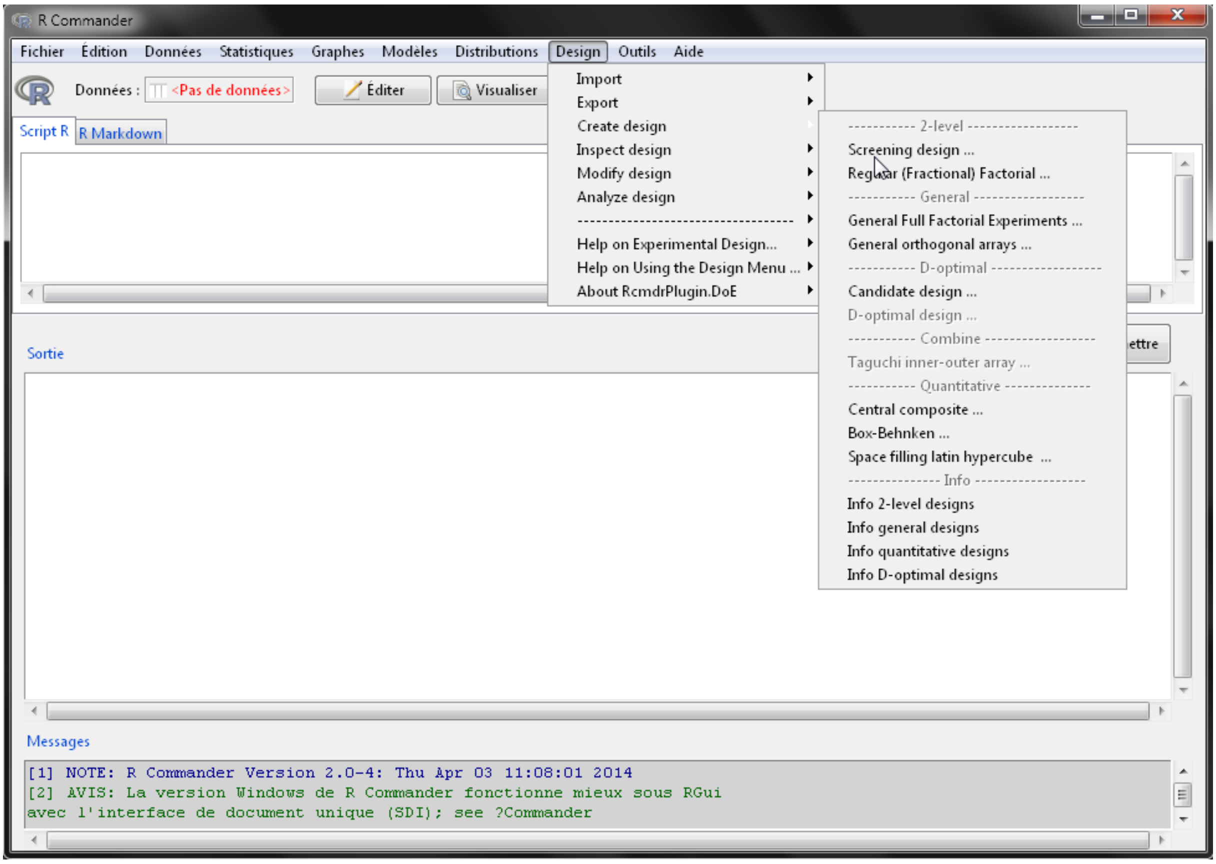
Dans cet exercice, le lecteur n'aura pas besoin d'ouvrir de fichiers car le seul objectif ici est de vérifier que R propose les plans d'expérience de base que nous avons étudiés dans le cours théorique de Génie Industriel sur le même sujet avant de passer à des exemples concrets.

Concernant le fait d'aller plus loin avec les plans d'expérience, vous avez en gros deux alternatives:

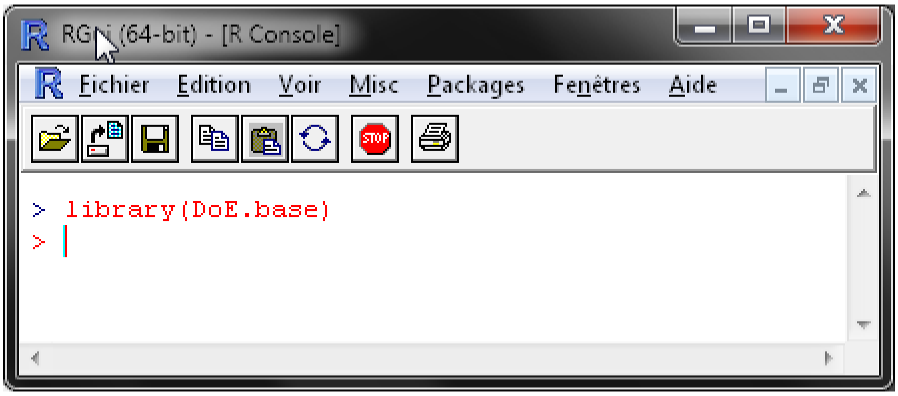
1. Vous installez et chargez le package **RcmdrPlugin.DoE**:



Ce qui vous donnera une interface WYSIWYG:



2. Installer et utiliser le package **DoE.base** et toute faire en ligne de commande:

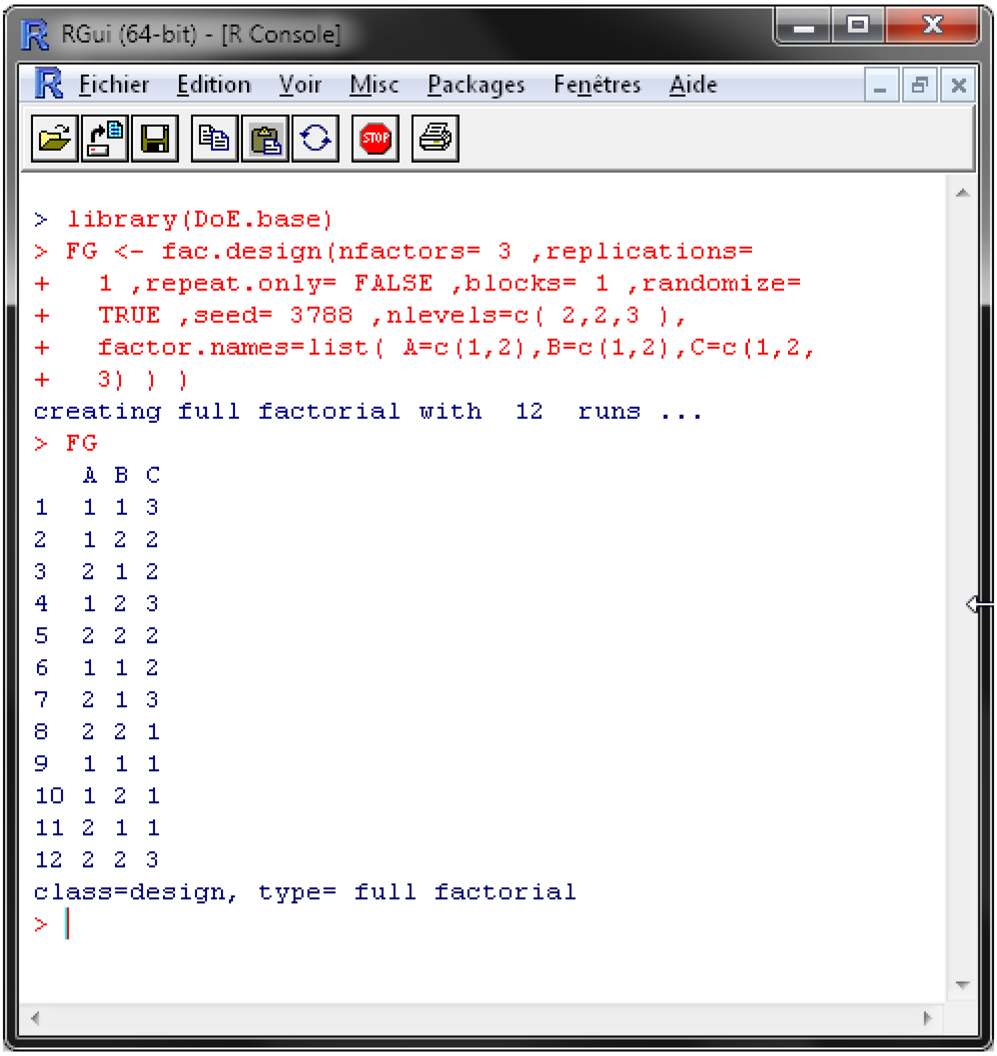


Pour les raisons déjà citées au début de cet e-book, je vais privilégier la deuxième méthode.

Plans factoriels

Plan factoriel complet général

Nous allons commencer par générer le même plan factoriel complet général que dans le cours théorique de Génie Industriel en utilisant donc la commande **fac.design( )** du package **DoE.base**:



C'est juste mais la sortie est moins pertinente qu'avec Minitab.

Analyse conjointe

Nous avons vu dans le cours théorique que l'analyse conjointe consiste simplement dans le cas linéaire à faire un plan d'expérience factoriel complet mais appliqué au domaine du Marketing afin de trouver les combinaisons optimales des propriétés d'un produit qui maximise la préférence des consommateurs.

Donc d'abord à partir du package **conjoint** nous créons rapidement un plan factoriel complet des trois propriétés (*Marque*, *Couleur*, *Prix*) qui nous intéressent dans cette étude à l'aide de la fonction **caFactorialDesign( )** :

```
> library(conjoint)
> experiment<-expand.grid(
+ Marque<-c("A","B","C"),
+ Couleur<-c("Rouge","Bleu"),
+ Prix<-c("50","100","150"))
> design<-caFactorialDesign(data=experiment,type="full")
> colnames(design) <- c("Marque","Couleur","Prix")
> print(design)
   Marque Couleur Prix
1       A   Rouge   50
2       B   Rouge   50
3       C   Rouge   50
4       A    Bleu   50
5       B    Bleu   50
6       C    Bleu   50
7       A   Rouge  100
8       B   Rouge  100
9       C   Rouge  100
10      A    Bleu  100
11      B    Bleu  100
12      C    Bleu  100
13      A   Rouge  150
14      B   Rouge  150
15      C   Rouge  150
16      A    Bleu  150
17      B    Bleu  150
18      C    Bleu  150
> 
```

Dés lors en associant les préférences à chacune des ces combinaisons nous avons:

R Console

```
> Preferences<-c(5,7,10,8,9,9,5,5,7,5,6,7,0,3,5,2,5,6)
> model<-caModel(Preferences, design)
> print(model)

Call:
lm(formula = frml)

Residuals:
     Min       1Q   Median       3Q      Max
-1.33333 -0.48611  0.08333  0.47222  1.33333

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.77778    0.22222  26.000  6.4e-12 ***
factor(x$Marque)1  -1.61111    0.31427  -5.127 0.000251 ***
factor(x$Marque)2   0.05556    0.31427   0.177 0.862632
factor(x$Couleur)1 -0.55556    0.22222  -2.500 0.027915 *
factor(x$Prix)1     2.22222    0.31427   7.071  1.3e-05 ***
factor(x$Prix)2     0.05556    0.31427   0.177 0.862632
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9428 on 12 degrees of freedom
Multiple R-squared:  0.9004,    Adjusted R-squared:  0.8589
F-statistic:  21.7 on 5 and 12 DF,  p-value: 1.251e-05

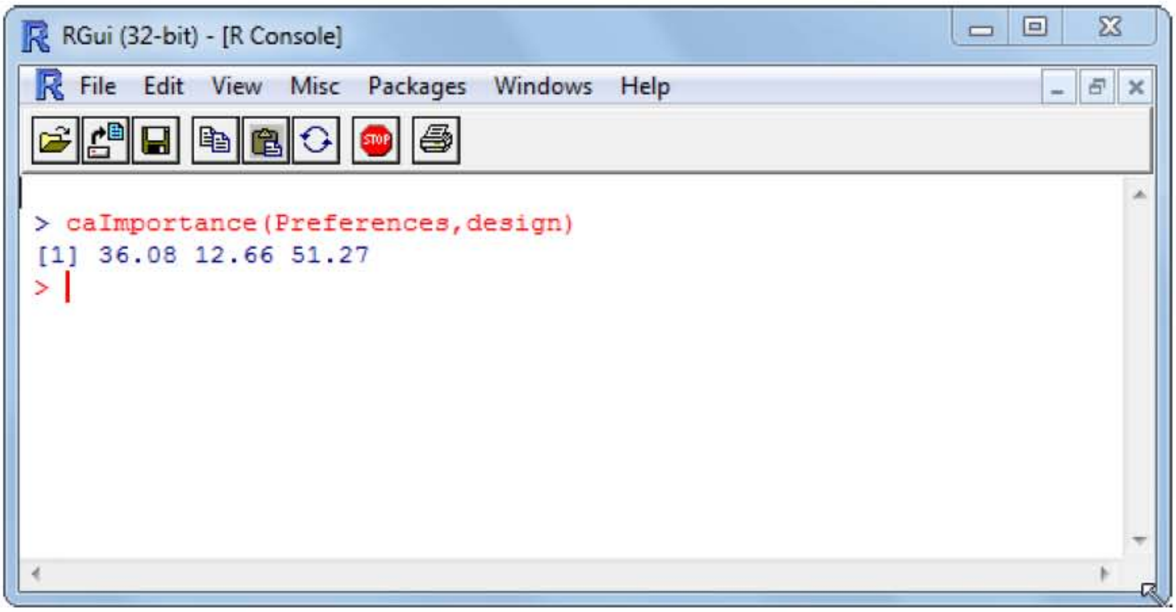
> |
```

Il s'agit donc d'une simple régression linéaire (logique) mais dont la forme communiquée peut perturber plus d'un non spécialiste:

1. Certains niveaux de facteurs sont absents et ceux qui ont étudié la théorie savent pourquoi et les autres non…

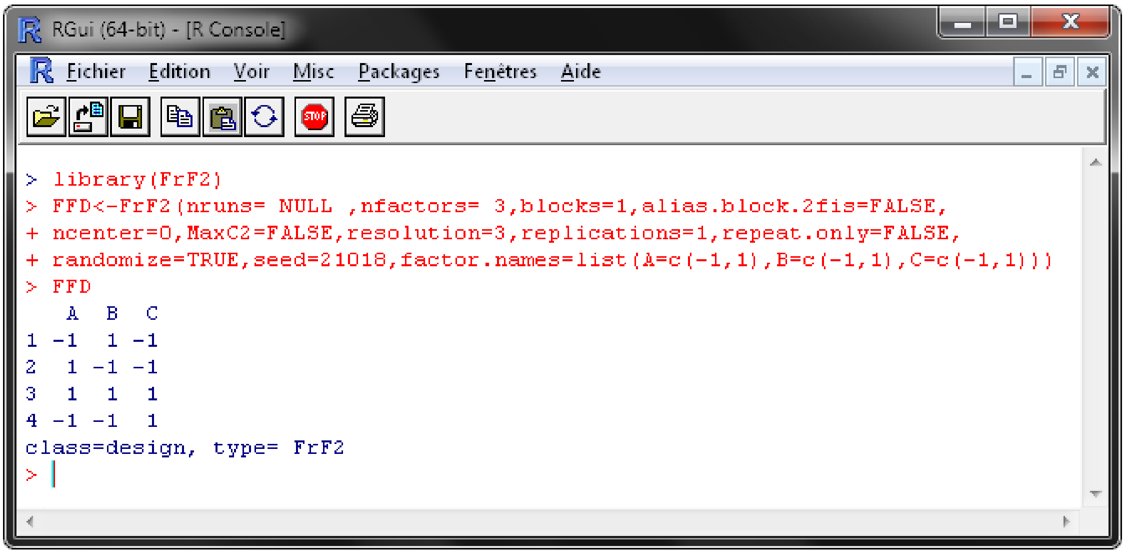
2. Les valeurs optimales des variables ne sont pas données automatiques

Nous avons aussi avec la fonction **caImportance( )** du même package les importances relatives des trois facteurs dans l'ordre d'entrée (même si en amplitude nous somme très loin des valeurs obtenues dans le cours théorique 63.02%, 5.76%, 31.22% l'ordre d'importance au moins… correspond…):


RGui (32-bit) - [R Console]
File Edit View Misc Packages Windows Help
> caImportance(Preferences,design)
[1] 36.08 12.66 51.27
>

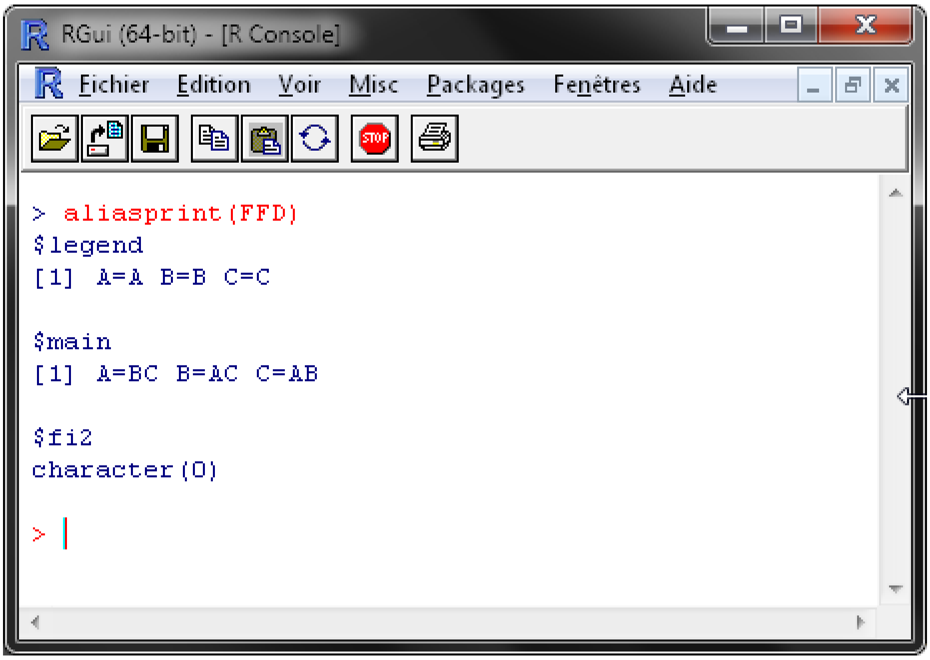
Plan factoriel (à deux niveaux) fractionnaire

Maintenant voyons comment générer un plan fractionnaire (dans le cas présent ce sera un plan factoriel fractionnaire comme celui vu en cours) où nous laissons le soin à R de choisir les meilleurs générateurs (alias). Nous allons utiliser la commande **FrF2( )** du package **FrF2**:

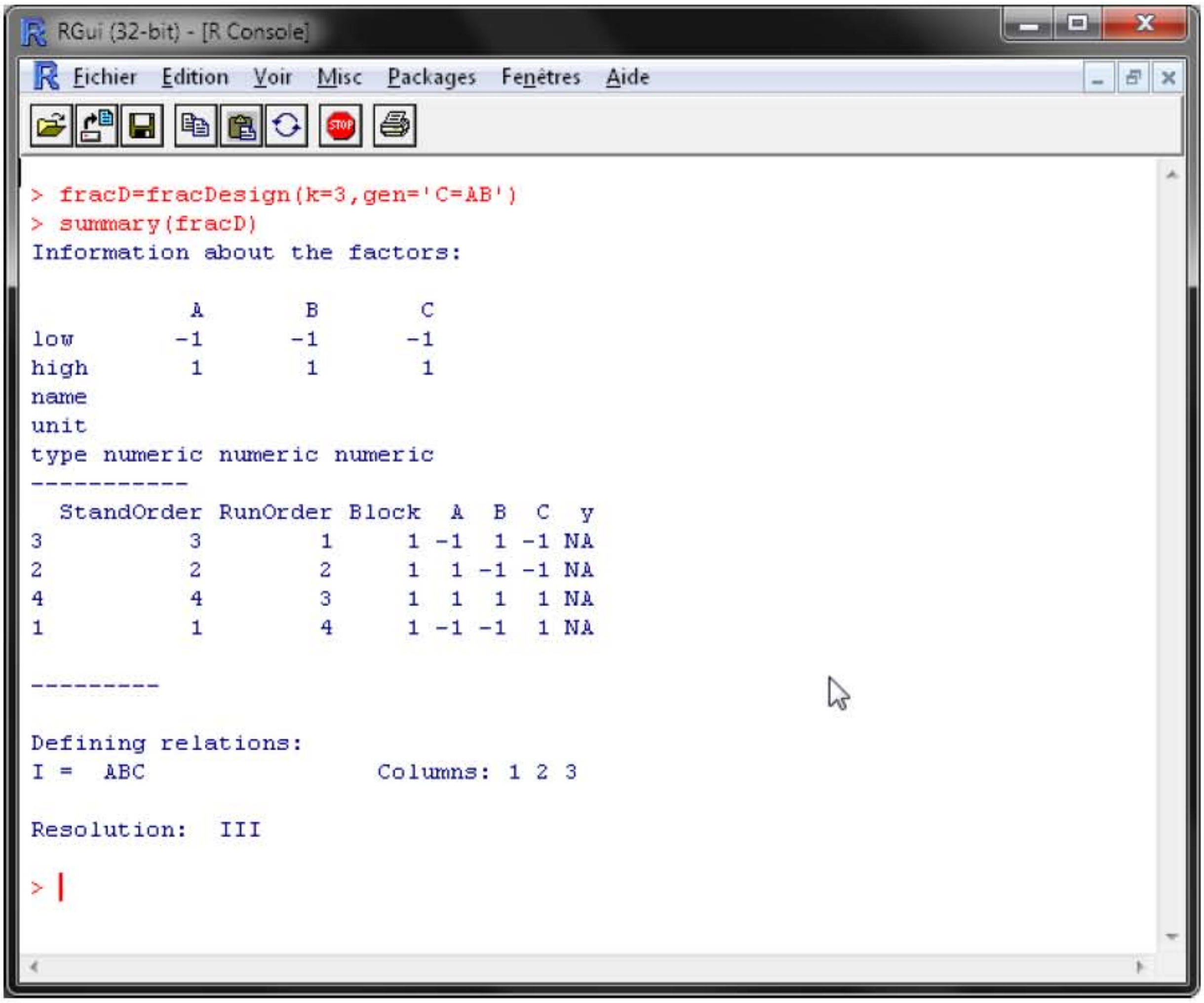


Nous voyons déjà dans la fenêtre de sessions que nous avons le même générateur et les mêmes alias que ceux vus en cours.

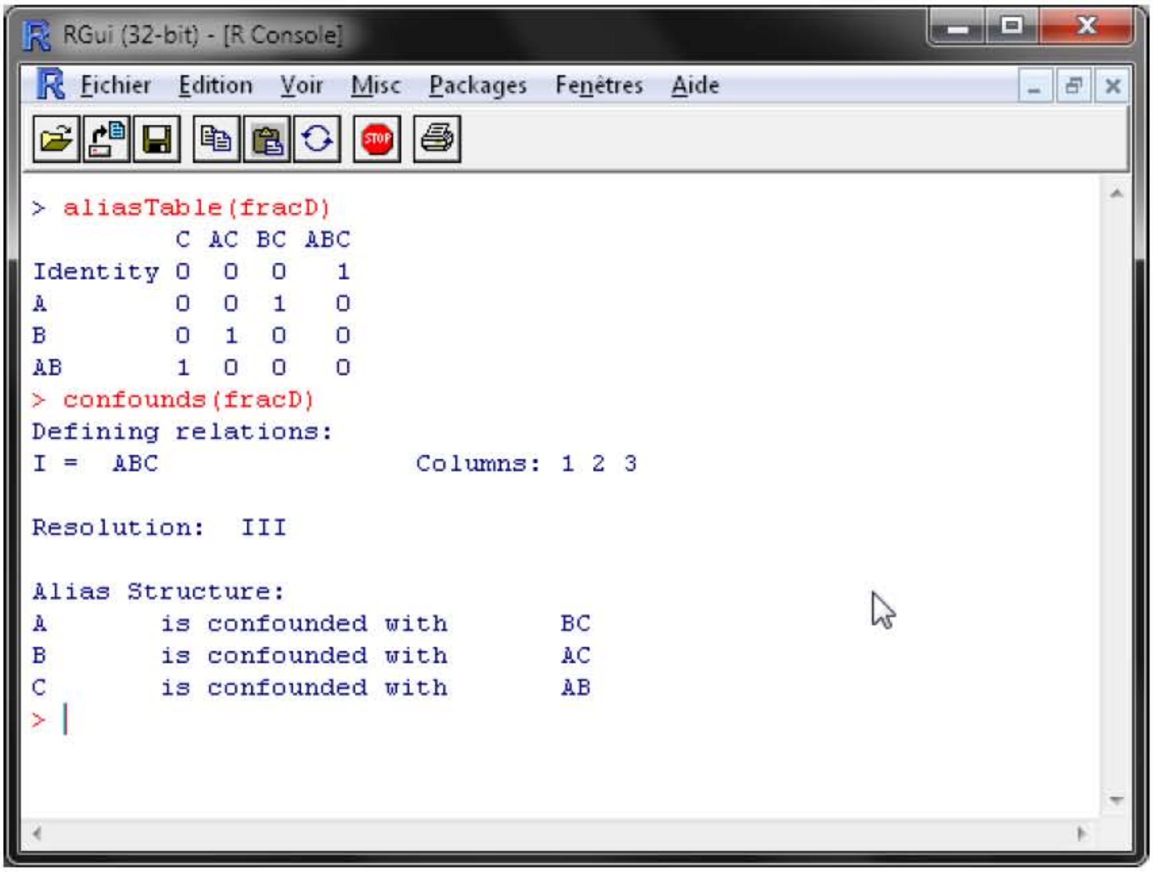
Avec la commande **aliasprint( )** on peut avoir les alias (qui sont donc les mêmes que Minitab et que dans le cours théorique):



Le package **qualityTools** permet selon mon opinion d'obtenir un résultat plus propre d'abord avec la fonction **fdo.frac( )** :



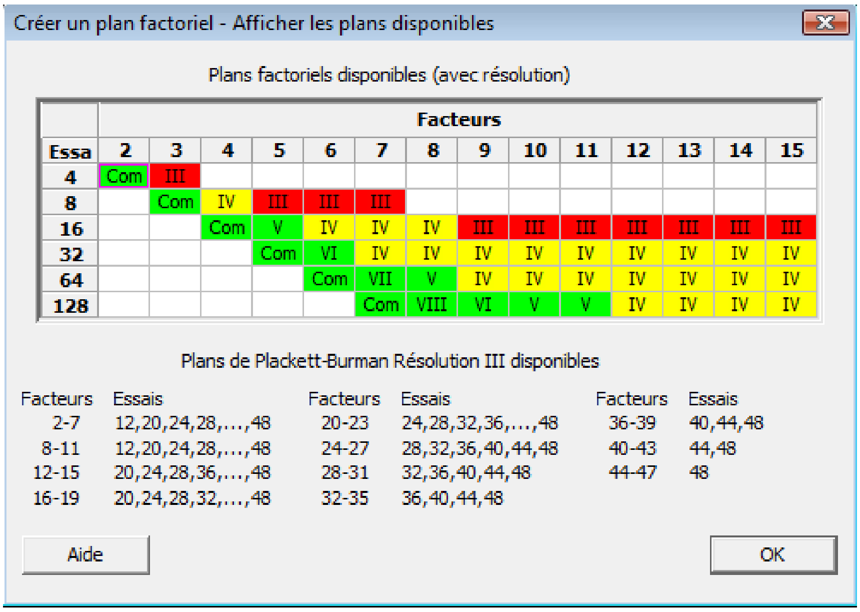
et avec la fonction **aliasTable( )** nous pouvons obtenir les alias et avec **confounds( )** les confusions et nous voyons bien que nous retombons sur les mêmes que dans le cours théorique:



Plan factoriel (à deux niveaux) de screening (résolution III)

Maintenant voyons le vrai intérêt d'un logiciel pour la base des plans d'expérience. Comme nous l'avons mentionné dans le cours de statistique théorique, au-delà de 4 facteurs il devient pénible de déterminer la matrice d'Hadamard pour les plans fractionnaires et les alias optimaux. Alors voyons comment R nous facilite la tâche.

Nous ferons comme dans le cours théorique et avec Minitab, nous allons prendre 5 facteurs et choisir le complet, le résolution V et le résolution III (comme le rappelle le schéma suivant de Minitab):



ou le schéma équivalent donné par l'interface DOE WYSIWYG:

Requested information

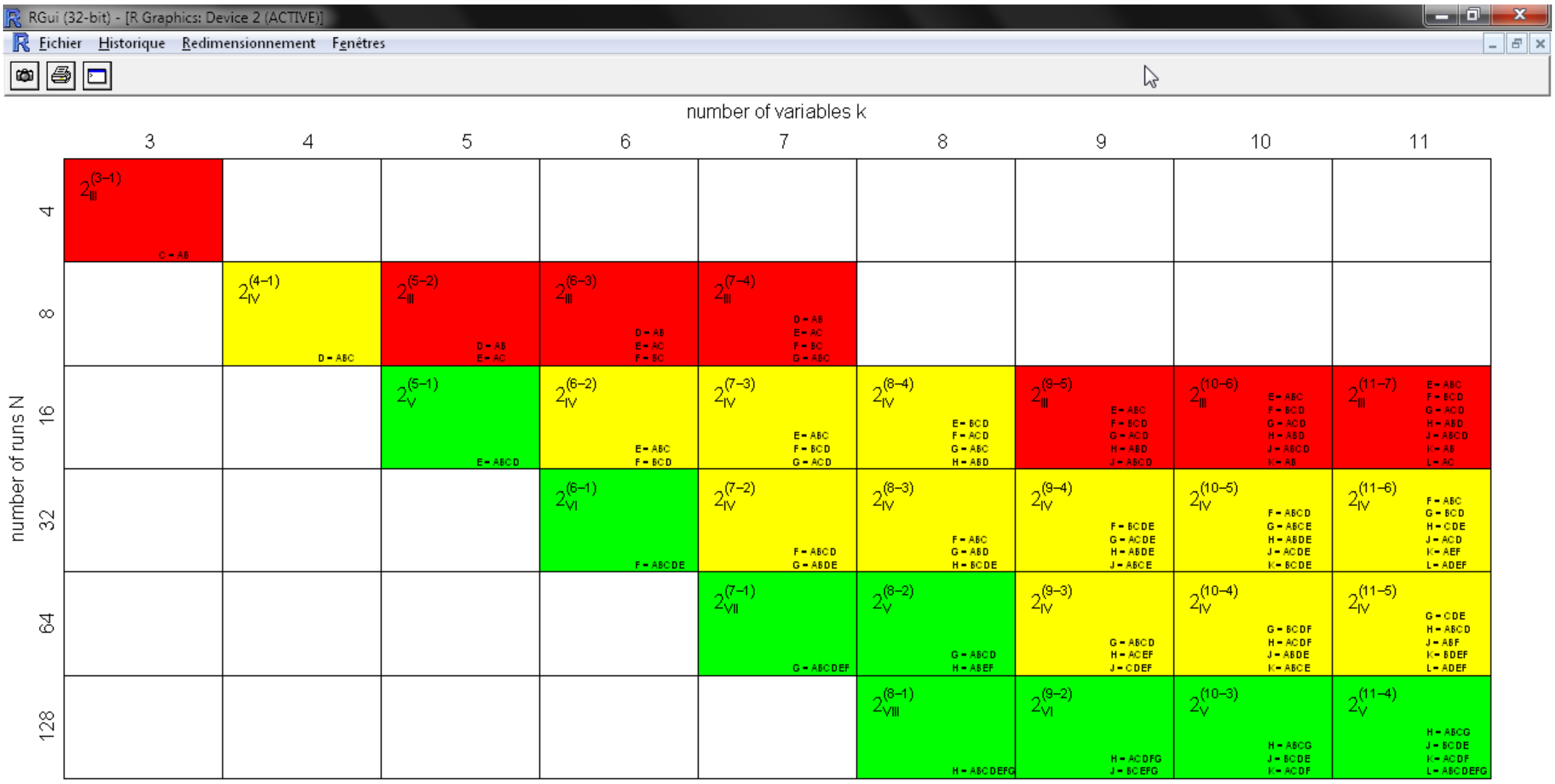
You must CLOSE this window for continuing work with the design dialogue.

number of runs

number of factors	8	16	32	64	128	256	512	1024	2048	4096
						only the MA design				
3	full									
4	IV	full								
5	III	V	full							
6	III	IV	VI	full						
7	III	IV	IV	VII	full					
8		IV	IV	V	VIII	full				
9		III	IV	IV	VI	IX	full			
10		III	IV	IV	V	VI	X	full		
11		III	IV	IV	V	VI	VII	XI	full	
12		III	IV	IV	IV	VI	VI	VIII	XII	full
13		III	IV	IV	IV	V	VI	VII	VIII	XIII
14		III	IV	IV	IV	V	VI	VII	VIII	IX
15		III	IV	IV	IV	V	VI	VII	VIII	VIII
16			IV	IV	IV	V	VI	VI	VIII	VIII
17			III	IV	IV	V	VI	VI	VII	VIII
18			III	IV	IV	IV	VI	VI	VII	VIII
19			III	IV	IV	IV	V	VI	VII	VIII
20			III	IV	IV	IV	V	VI	VII	VIII
21			III	IV	IV	IV	V	VI	VII	VIII
22			III	IV	IV	IV	V	VI	VII	VIII
23			III	IV	IV	IV	V	VI	VII	VIII
24			III	IV	IV	IV	IV	VI	VI	VIII
Resolution III up to			31	63	127			factors.		
Resolution IV up to				32	64	80	160	factors.		
Resolution V up to number of factors:								33	47	65
Resolution VI up to number of factors:								24	34	48
First design is MA up to number of factors:			31	63	127	36	29	28	32	26

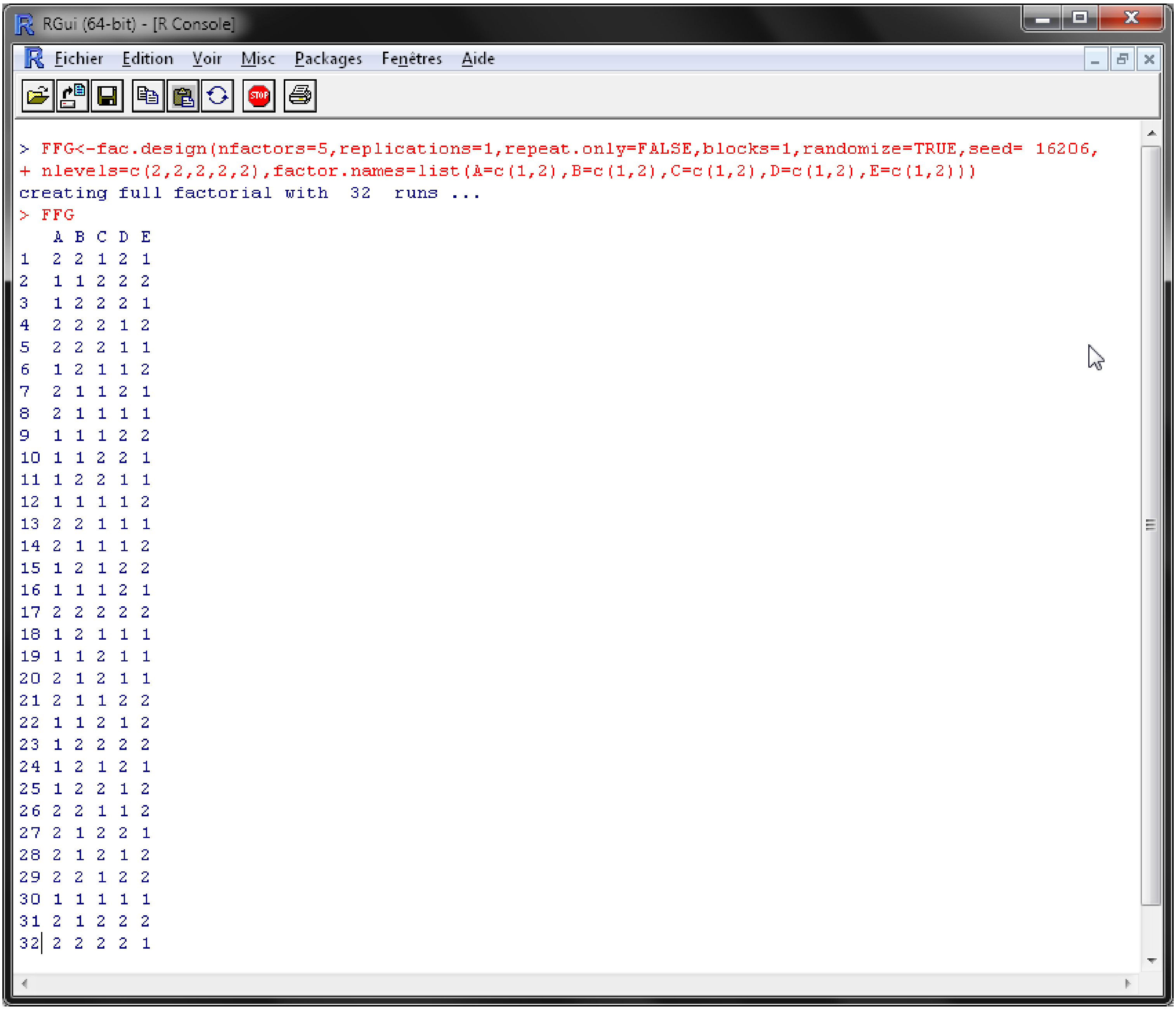
Additional irregular resolution V designs from general orthogonal arrays menu:
128 runs (up to 15 factors), 256 runs (up to 19 factors), 2048 runs (up to 63 factors)

ou encore par la fonction **fracChoose( )** du package **qualityTools**:

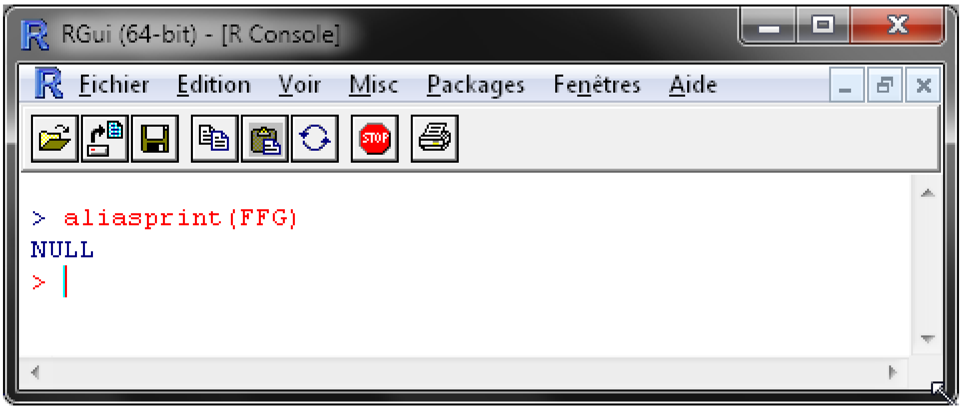


qui lors du clic sur un des cas permet de récupérer le plan dans l'espace de travail R.

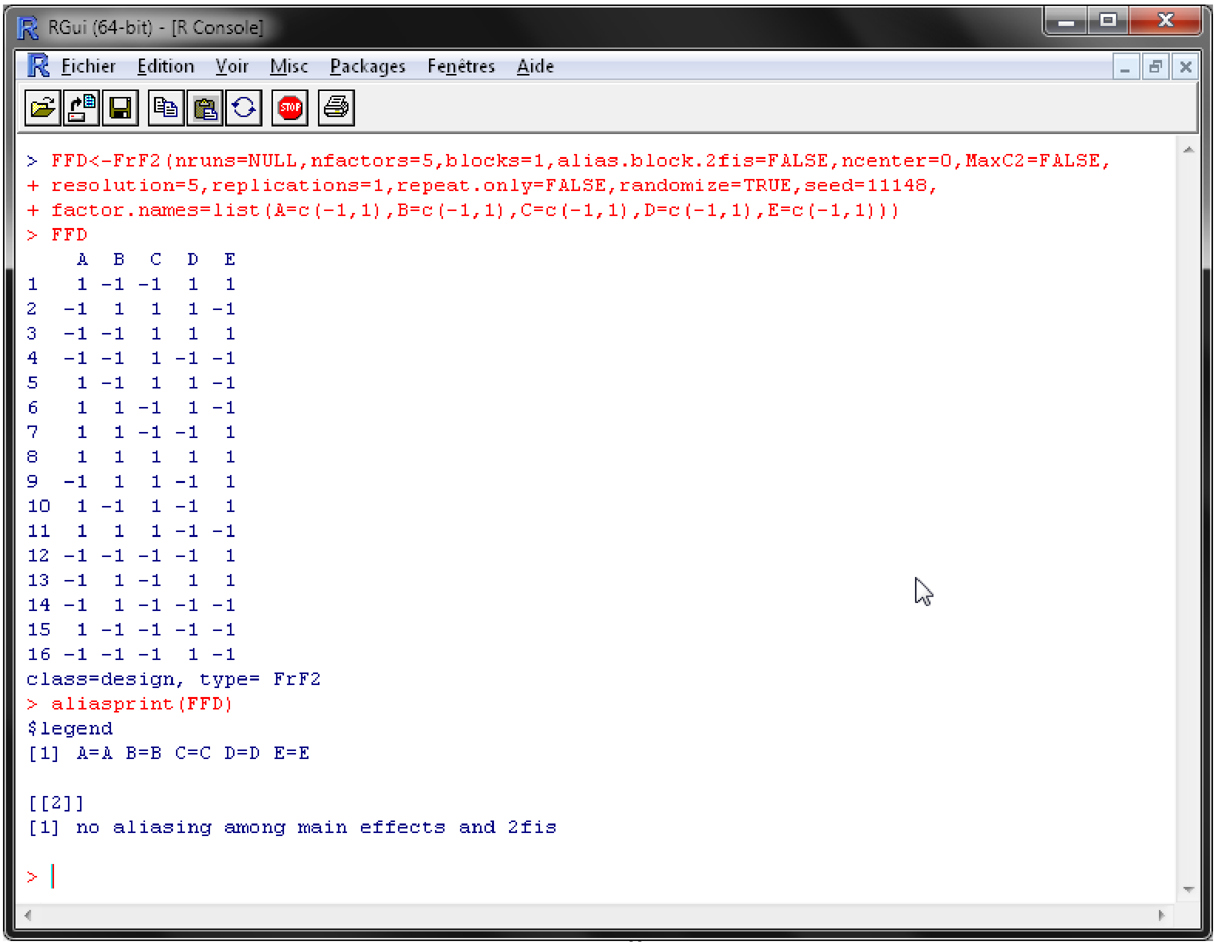
Commençons donc par le plan factoriel complet à 5 facteurs de 2 niveaux:



Il n'y a donc ici évidemment aucun alias:

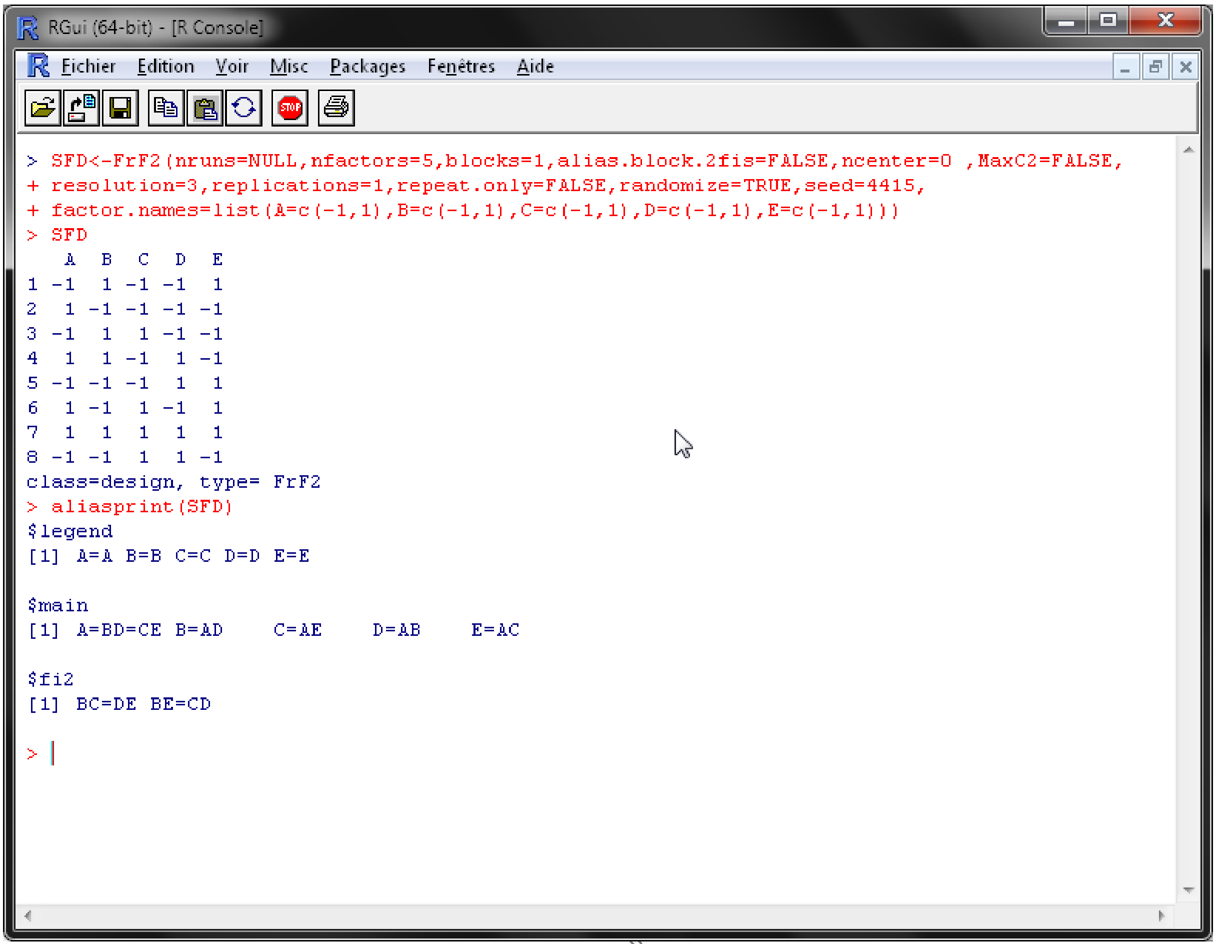


Ensuite, nous créons un plan factoriel fractionnaire de résolution V:



Nous voyons bien les 16 essais comme dans le cours théorique et Minitab par contre les alias ne son pas communiqués.

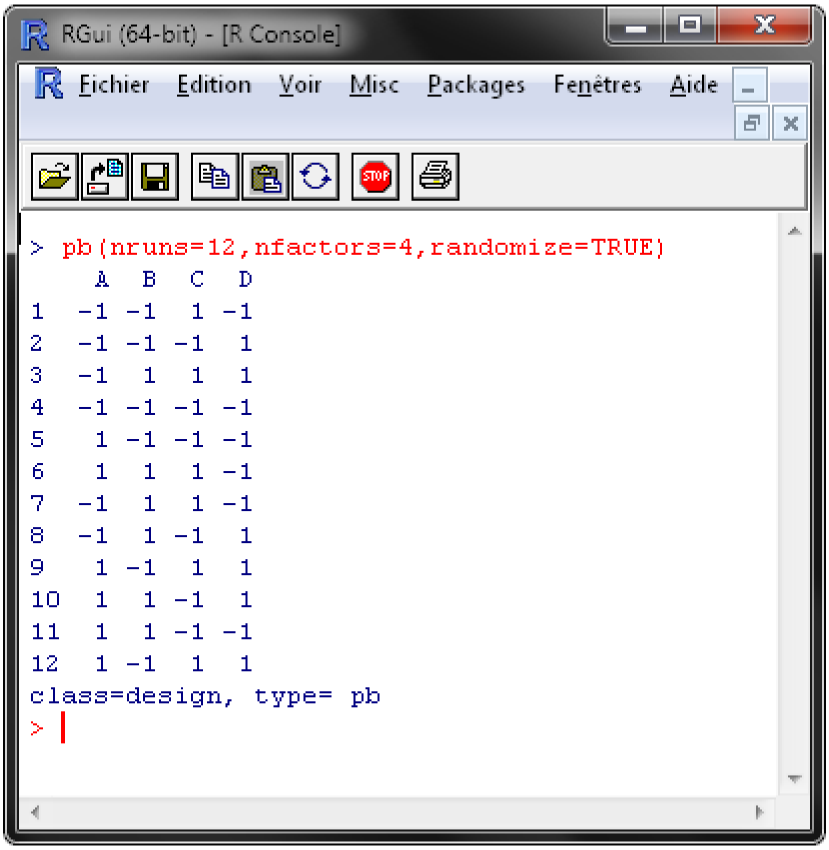
Maintenant voyons le plan factoriel fractionnaire ¼ de résolution III (plan de screening) Il est très intéressant de voir comment les alias vont être choisis:



Le tableau et les alias sont conformes à Minitab mais la manière dont ils sont indiqués est très vague (contrairement à Minitab) et peut porter à confusion.

Plan factoriel de Plackett-Burman

Maintenant, voyons les plans de Plackett-Burman que nous avons aussi étudié dans le cours de statistique théorique. Faisons l'exemple avec le plus petit nombre de facteurs où ces plans commencent à être vraiment intéressants comme nous l'avons vu dans le cours. Nous avons alors en utilisant la commande **pb( )** du package **FrF2**:



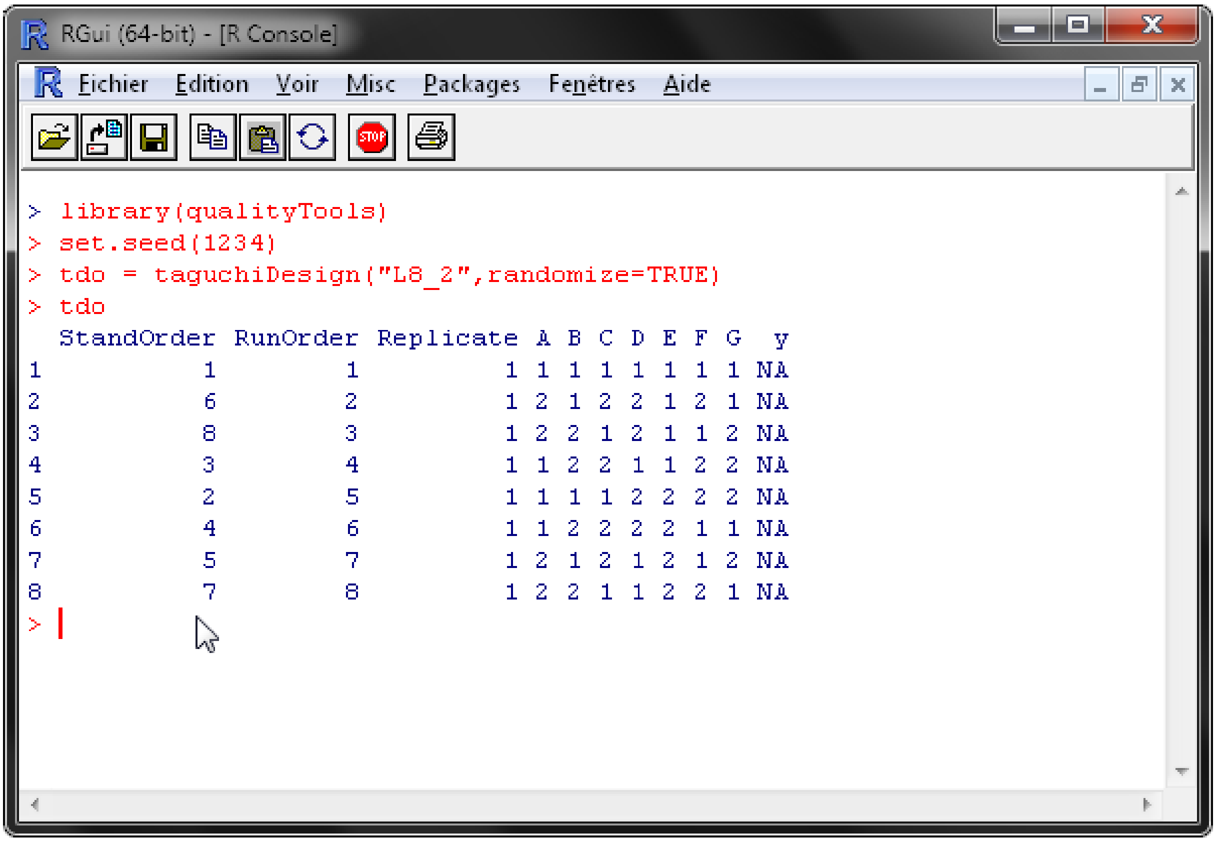
Nous observons donc que la fenêtre de session n'affiche pas les alias (ce qui est dommage puisque les plans de Plackett-Burman sont des plans à résolution III comme nous l'avons dit en cours). Il n'est pas non plus possible de les obtenir avec la command **aliasprint( )** .

Le plan obtenu ci-dessus n'est pas le même qu'avec Minitab et ne correspond donc pas à l'algorithme de Plackett-Burman que nous avons vu dans le cours théorique. Donc à creuser...

Le plan obtenu est lui plutôt surprenant. Nous n'avons que 4 colonnes pour les facteurs de base. Certes il n'est pas utile d'en avoir plus mais cela aurait été sympathique de pouvoir choisir...

Plan factoriel de Taguchi

Maintenant, pour en finir avec la base des plans d'expérience, voyons encore les plans de Taguchi. Pour cet exemple, nous allons utiliser le magnifique package **qualityTools** et la commande **taguchiDesign( )** qui y est intégrée. Nous avons alors pour construire le même plan de Taguchi que dans le cours théorique et Minitab:

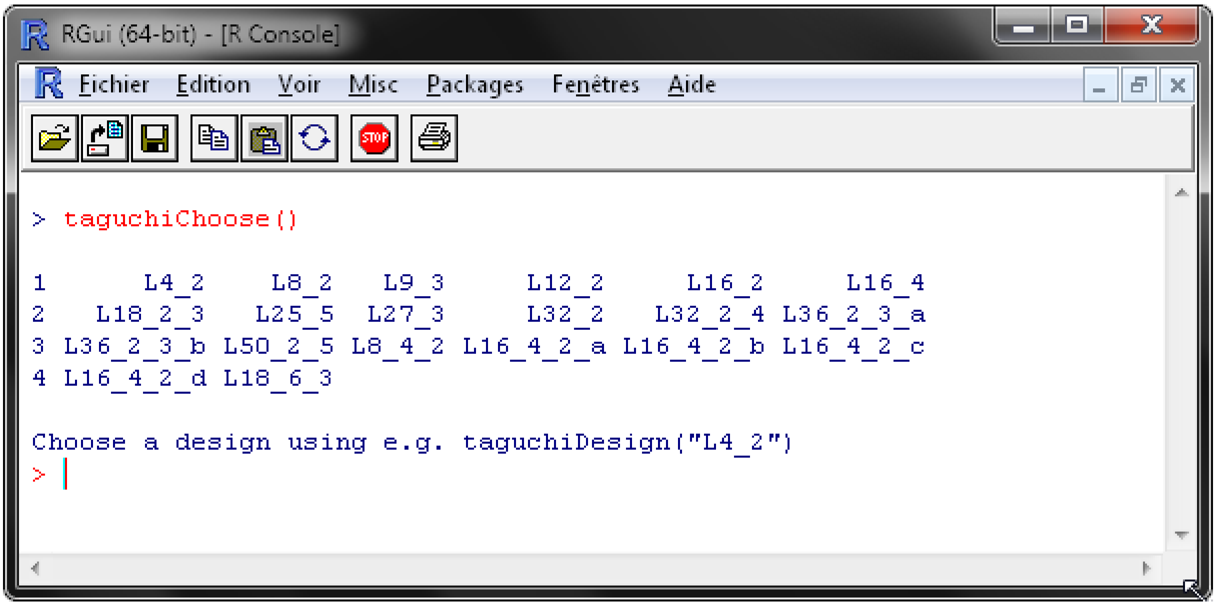


Voici si jamais la liste des plans de Taguchi tels qu'indiqués dans l'aide de la commande (juste pour info et en comparaison à Minitab):

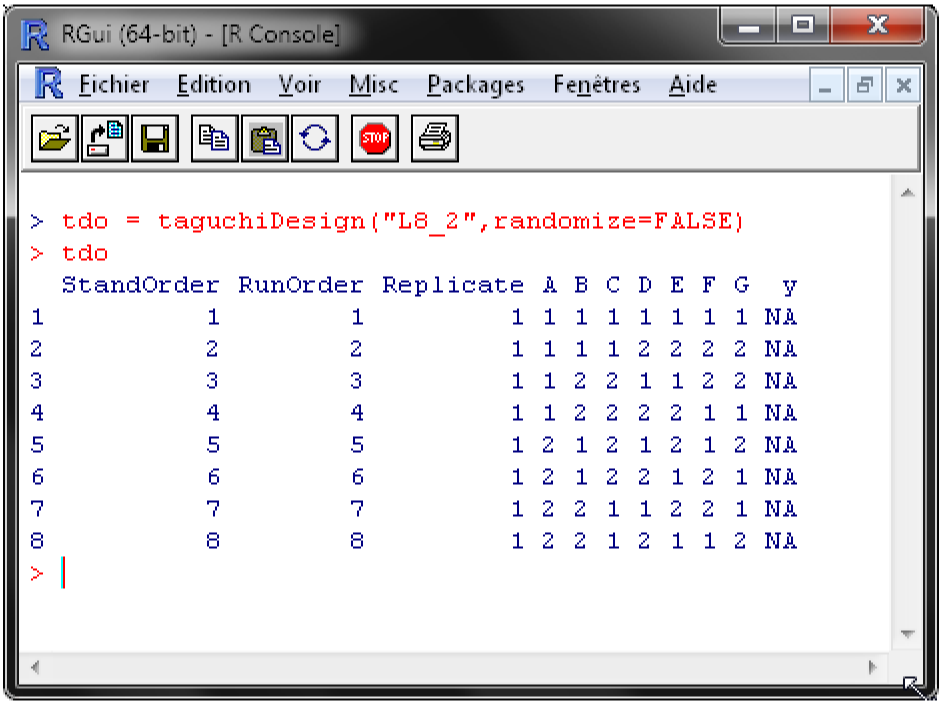
- “L4_2” for three two-level factors
- “L8_2” for seven two-level factors
- “L9_3” for four three-level factors
- “L12_2” for 11 two-level factors
- “L16_2” for 16 two-level factors
- “L16_4” for 16 four-level factors
- “L18_2_3” for one two-level and seven three-level factors
- “L25_5” for six five-level factors
- “L27_3” for 13 three-level factors
- “L32_2” for 32 two-level factors
- “L32_2_4” for one two-level factor and nine four-level factors
- “L36_2_3_a” for 11 two-level factors and 12 three-level factors
- “L36_2_3_b” for three two-level factors and 13 three-level factors
- “L50_2_5” for one two-level factor and eleven five-level factors
- “L8_4_2” for one four-level factor and four two-level factors
- “L16_4_2_a” for one four-level factor and 12 two-level factors

- “L16_4_2_b” for two four-level factors and nine two-level factors
- “L16_4_2_c” for three four-level factors and six two-level factors
- “L16_4_2_d” for five four-level factors and two two-level factors
- “L18_6_3” for one six-level factors and six three-level factors

Ce que nous pouvons directement obtenir dans R:

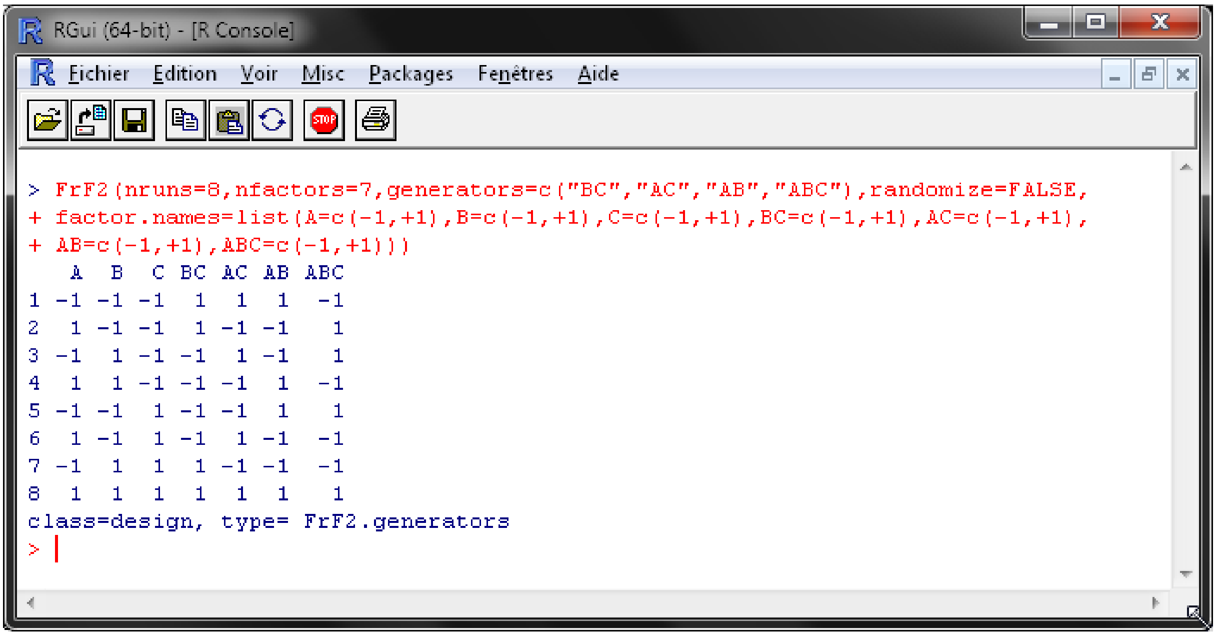


Donc contrairement à Minitab ce package donne la table de Taguchi complète ce qui est plus confortable intellectuellement. Profitons donc pour comparer par rapport à ce que nous avions dans le cours théorique en désactivant la randomization:



C'est donc parfaitement conforme!

Maintenant, voyons si nous retrouvons ce plan de Taguchi avec le plan factoriel "classique" comme nous l'avons fait dans le cours théorique et Minitab en spécifiant les générateurs. Nous écrivons alors avec le package **FrF2**:



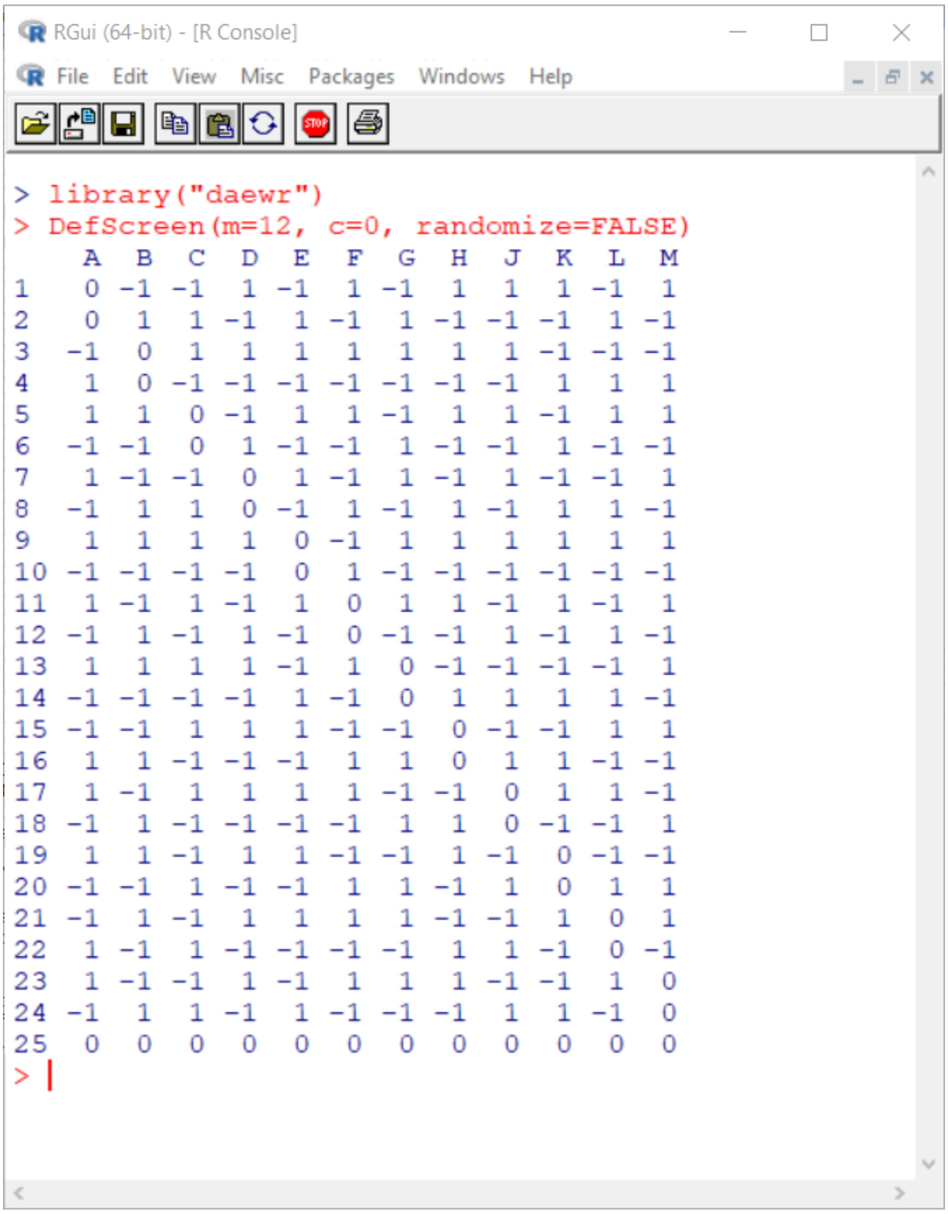
Ce qui à part l'ordre, est parfaitement conforme au tableau obtenu dans le cours et où l'écriture est beaucoup plus logique et intuitive à mon goût que dans Minitab:

Essais	F3	F2	F23	F1	F13	F12	F123	Repos
1	+	+	+	+	+	+	+	+
2	+	+	+	-	-	-	-	+
3	+	-	-	+	+	-	-	+
4	+	-	-	-	-	+	+	+
5	-	+	-	+	-	+	-	+
6	-	+	-	-	+	-	+	+
7	-	-	+	+	-	-	+	+
8	-	-	+	-	+	+	-	+

Plan de criblage définitif (definitive screening design)

Comme pour les autres plans, nous allons d'abord ici juste vérifier que nous arrivons à retomber sur plan d'expérience vu dans le cours théorique. Ici le but va être de vérifier si on peut effectivement reproduire le plan *C*(m=12,*n*=2) vu dans le cours théorique (on va pas tous les faire non plus…).

Donc toujours avec le package **dawr** (relatif au livre *Design and Analysis of Experiments with R*) avec la fonction **DefScreen( )**:



Ce qui est bien conforme à ce que nous avons vu dans le cours théorique (à part que le format dans le cours théorique était plus… "lisible")!

Analyse et comparaison d'un plan d'expérience identique en factoriel complet général et factoriel à 2 niveaux

Avant de passer au gros exercice fait à la main dans le cours théorique et dans Minitab il va être utile de commencer par un cas hyper simpliste qui avait consisté à résoudre à la main (et aussi dans Minitab) le système suivant:

$$\begin{aligned}
a_0 \cdot 1 + a_1(40) + a_2(1.5) + a_{12}(60)(1.5) &\cong 32'700 \\
a_0 \cdot 1 + a_1(50) + a_2(1.5) + a_{12}(50)(1.5) &\cong 32'680 \\
a_0 \cdot 1 + a_1(40) + a_2(3) + a_{12}(40)(3) &\cong 31'710 \\
a_0 \cdot 1 + a_1(50) + a_2(3) + a_{12}(50)(3) &\cong 33'220
\end{aligned}$$

Ce qui nous avait donné sous forme matricielle:

$$\begin{bmatrix} 1 & 40 & 1.5 & 60 \\ 1 & 50 & 1.5 & 75 \\ 1 & 40 & 3 & 120 \\ 1 & 50 & 3 & 150 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_{12} \end{bmatrix} = \begin{bmatrix} 32'700 \\ 32'680 \\ 31'710 \\ 33'220 \end{bmatrix}$$

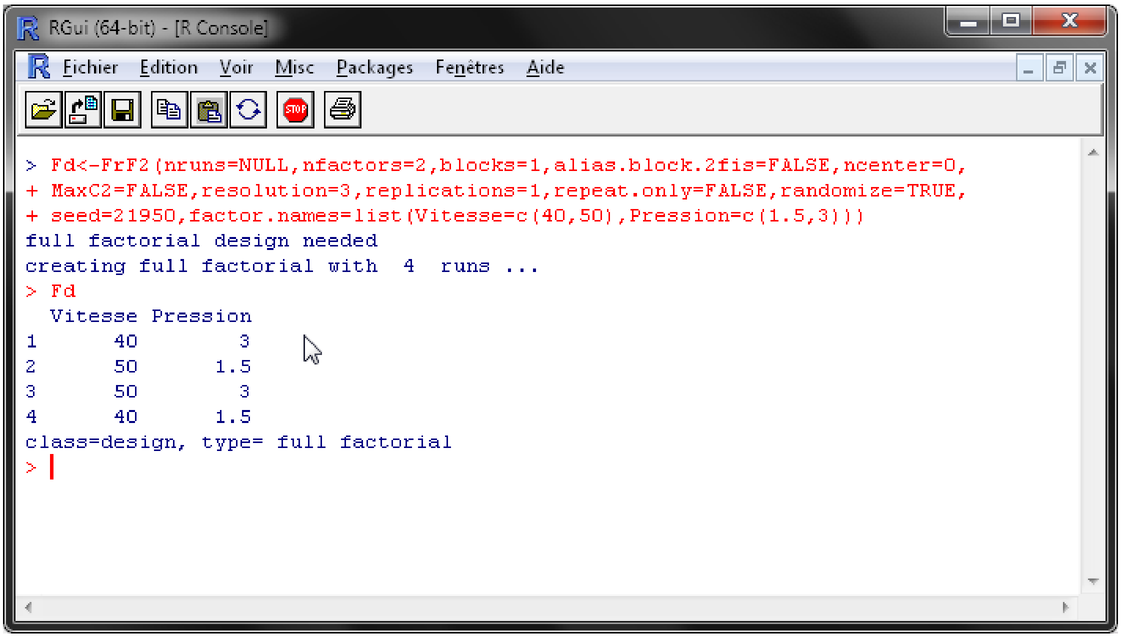
La solution déterminée avec la vitesse et pression seule avait donc été de:

$$y \cong 39'890 - 155x_1 - 4740x_2 + 102x_{12}$$

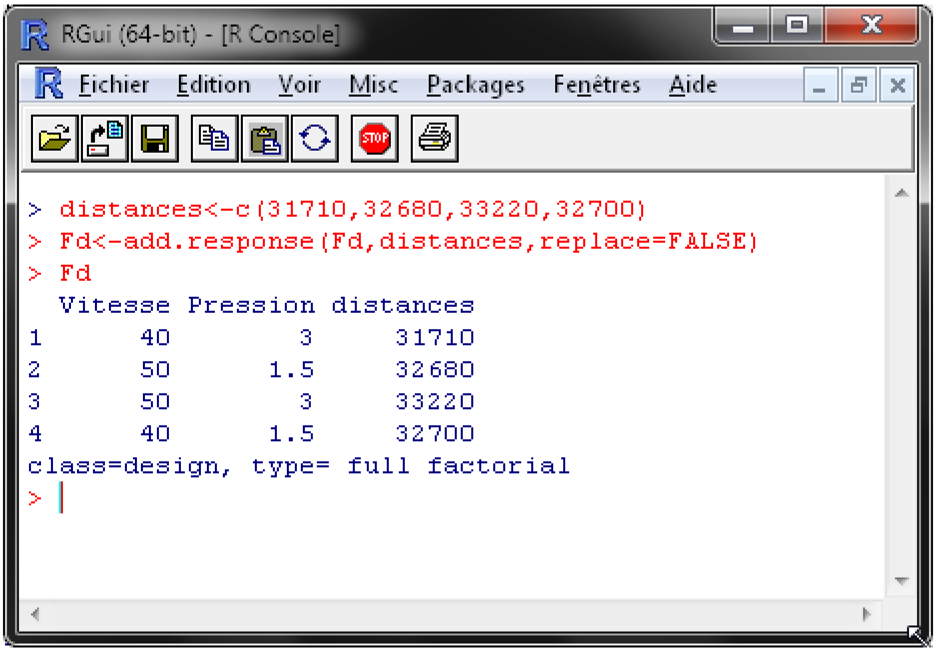
Vérifions cela avec R et comparons deux packages.

Package FrF2

D'abord nous construisons le plan factoriel (complet):



Nous rajoutons les distances (mesures sans répétitions):



Et nous lançons l'analyse:

```
> ModeleLineaire<-lm(distances~Pression+Vitesse+Pression:Vitesse,data=Fd)
> ModeleLineaire

Call:
lm.default(formula = distances ~ Pression + Vitesse + Pression:Vitesse,
    data = Fd)

Coefficients:
       (Intercept)            Pression1              Vitesse1   Pression1:Vitesse1
           32577.5               -112.5                 372.5                382.5

> summary(ModeleLineaire)

Call:
lm.default(formula = distances ~ Pression + Vitesse + Pression:Vitesse,
    data = Fd)

Residuals:
ALL 4 residuals are 0: no residual degrees of freedom!

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         32577.5         NA      NA       NA
Pression1            -112.5         NA      NA       NA
Vitesse1              372.5         NA      NA       NA
Pression1:Vitesse1    382.5         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:    NaN
F-statistic:   NaN on 3 and 0 DF,  p-value: NA

>
```

Et nous voyons que bien qu'ayant défini un plan factoriel complet général, les coefficients:

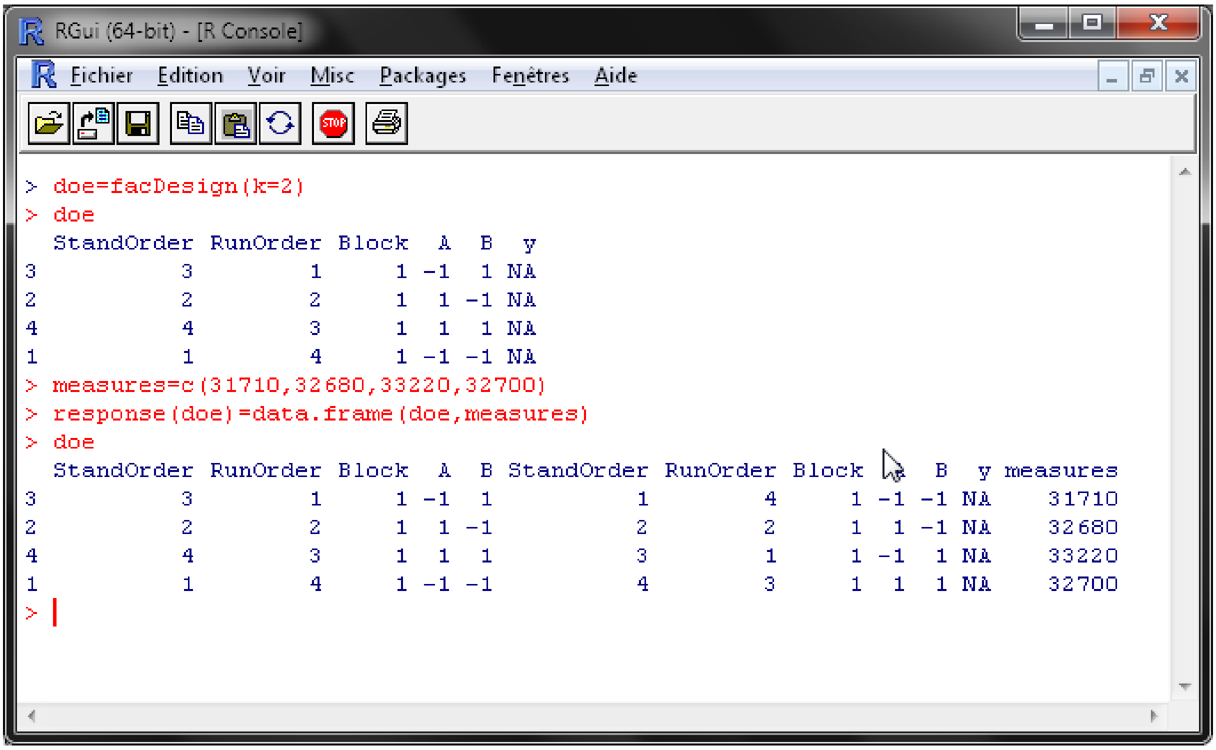
$$y \cong 32577.5 - 372.5x_1 + 112.5x_2 + 382.5x_{12}$$

sont donnés par rapport à un modèle aux variables codées (-1,1). Je n'ai pas trouvé la possibilité comme sur Minitab de pouvoir choisir si le modèle final devait être donné en variables codées ou non.

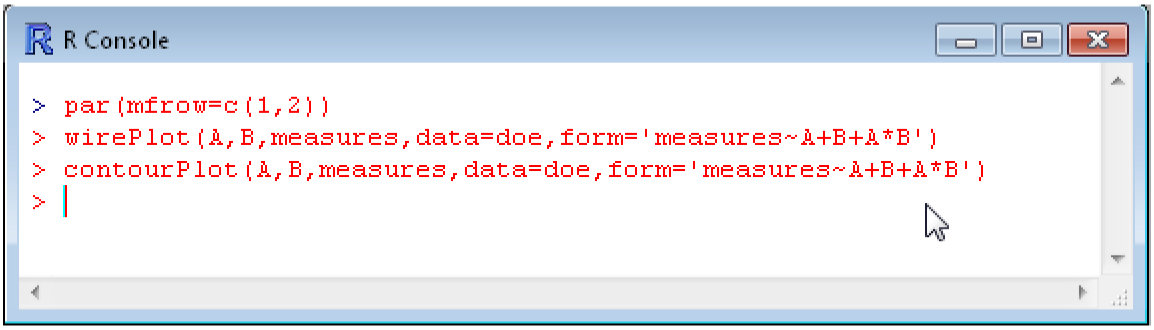
Donc on ne pourra pas comparer le plan factoriel général complet au plan factoriel général au plan factoriel complet à 2 niveaux puisque ce deuxième nous est imposé!

Package qualityTools

Nous créons le plan avec la fonction **facDesign( )** :



Ensuite, nous passons d'abord à une analyse qualitative avec les fonctions **wirePlot( )** et **contourPlot( )** :



et nous voyons que le fait qu'on ne puisse pas bouger les légendes, changer les axes, tourner les graphes est une désavantage majeur par rapport à Minitab:

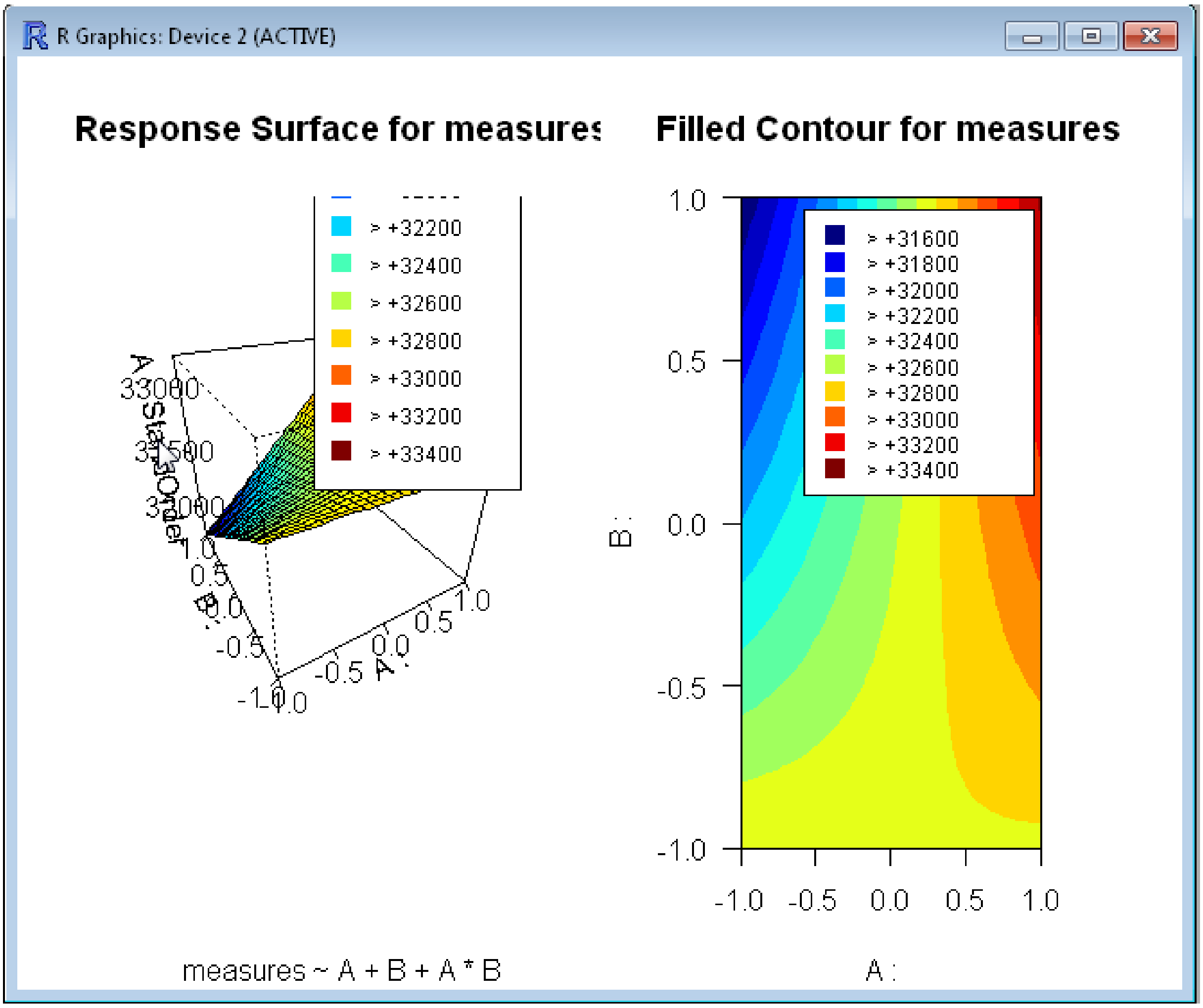
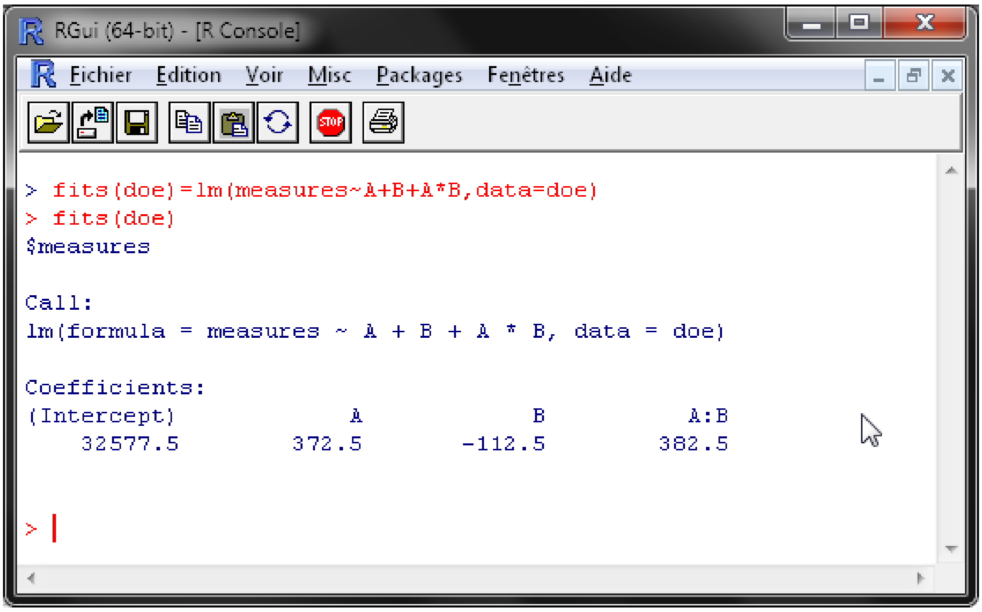


Figure 264 Graph de réponse de surface pour plan d'expérience (DoE)

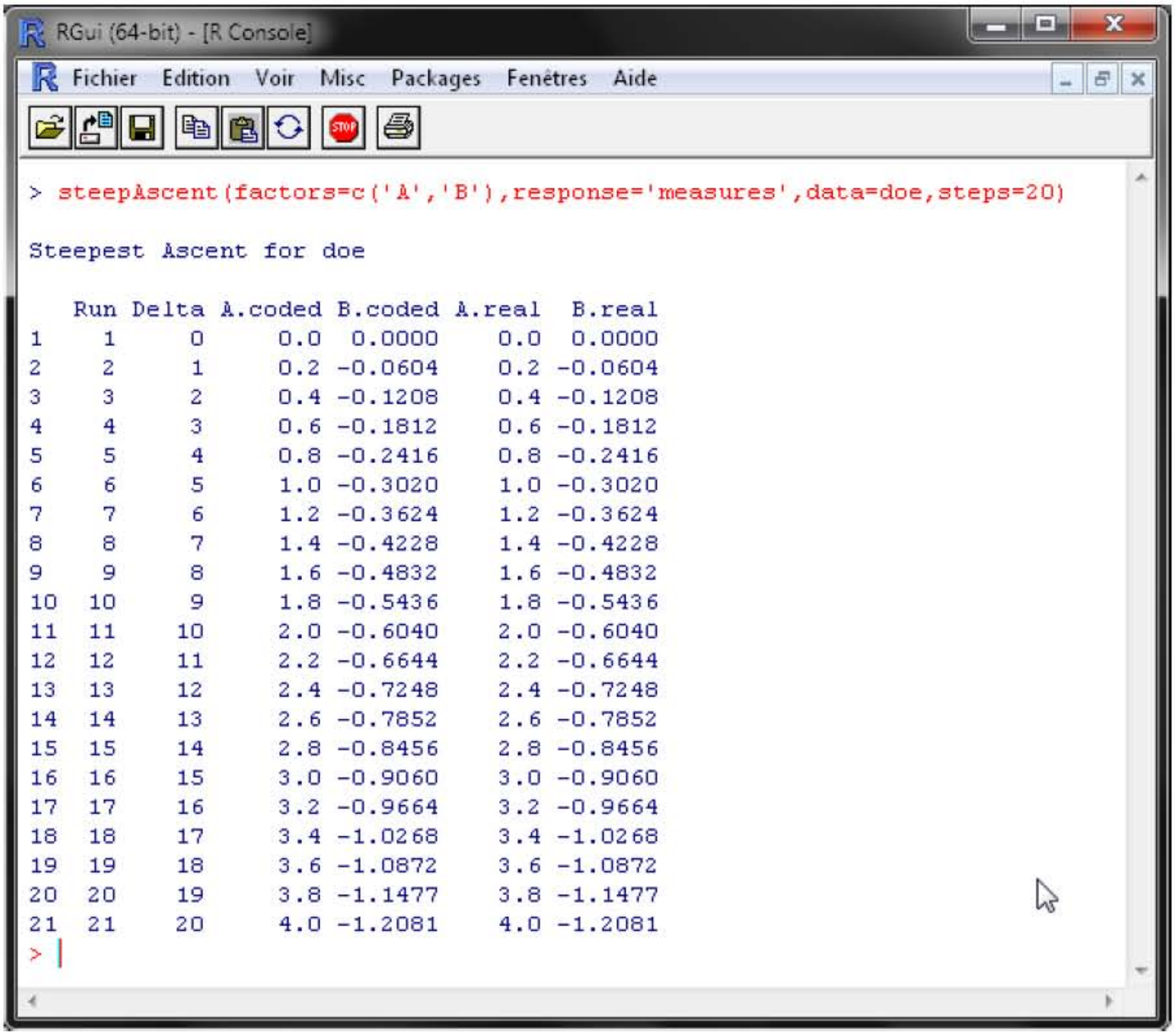
L'analyse qualitative est bien évidemment conforme à l'intuition.

Maintenant voyons ce que nous pouvons obtenir au niveau du modèle linéaire:



Nous obtenons le même résultat qu'avant ce qui est plutôt rassurant!

Nous pouvons demander à la fonction **steepAscent( )** du package **qualityTools** de chercher le maxima même en dehors des limites codées mais évidemment au vu du modèle ci-dessus il est évident que le maxima est à l'infini en vitesse et à une pression qui tend vers zéro... comme quoi il faut toujours se méfier des modèles en dehors de leur domaine de définition:



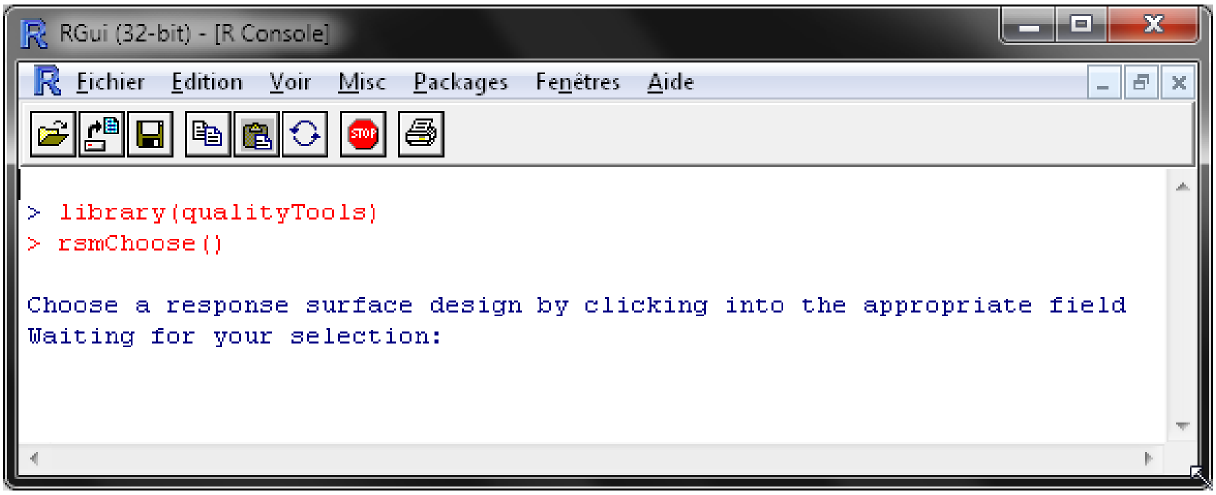
Plans de surfaces (quadratiques)

Plans composites centrés

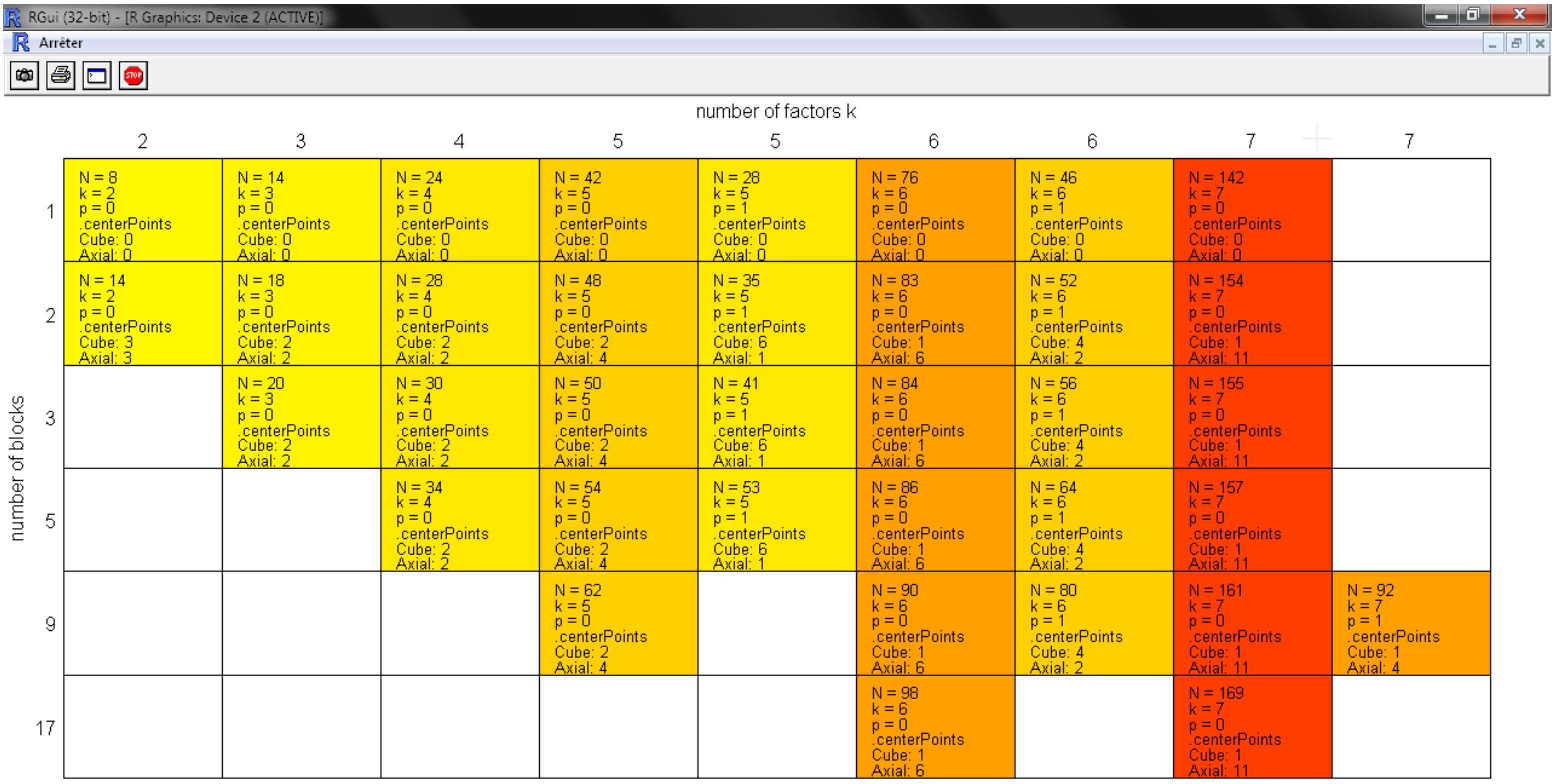
Bon nous n'avons rien démontré dans le cours théorique par rapport à ce type de plans dont la construction est relativement très beaucoup empirique et faut aussi d'avoir une démonstration mathématique pédagogique et élégante sous la main.

Nous allons comme dans le cours Minitab considérer les plans CC comme des black box avec des valeurs en entrées et en sortie dont nous connaissons les hypothèses d'utilisation.

Voyons d'abord ce que nous pouvons obtenir avec la fonction **rsmChoose( )** du package **qualityTools**:

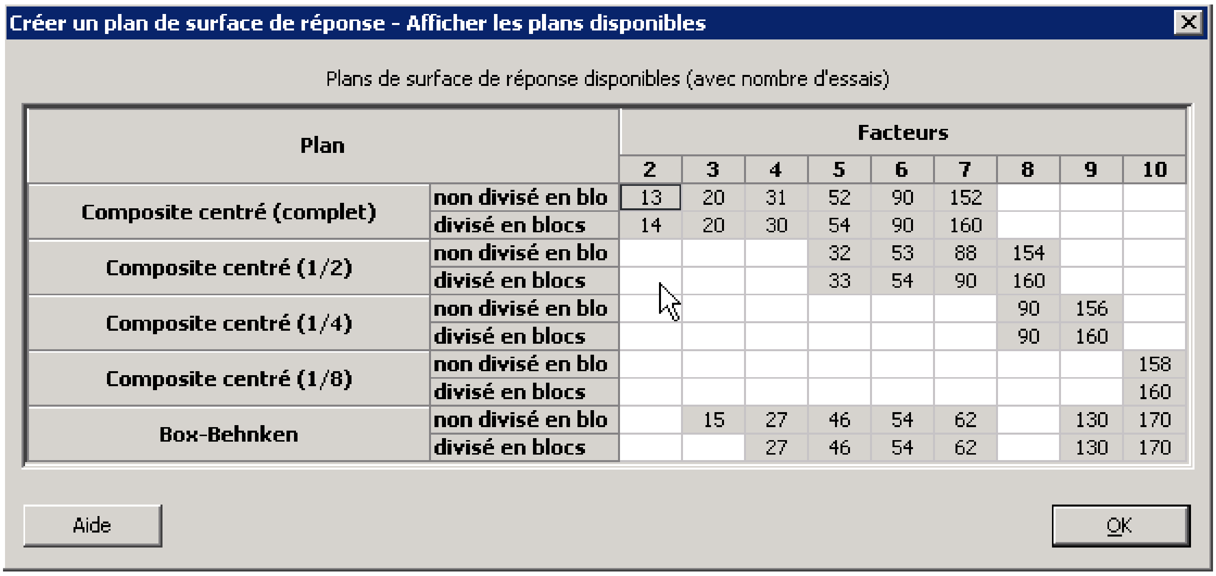


Nous avons alors:



number of blocks \ number of factors k	2	3	4	5	5	6	6	7	7
1	N = 8 k = 2 p = 0 .centerPoints Cube: 0 Axial: 0	N = 14 k = 3 p = 0 .centerPoints Cube: 0 Axial: 0	N = 24 k = 4 p = 0 .centerPoints Cube: 0 Axial: 0	N = 42 k = 5 p = 0 .centerPoints Cube: 0 Axial: 0	N = 28 k = 5 p = 1 .centerPoints Cube: 0 Axial: 0	N = 76 k = 6 p = 0 .centerPoints Cube: 0 Axial: 0	N = 46 k = 6 p = 1 .centerPoints Cube: 0 Axial: 0	N = 142 k = 7 p = 0 .centerPoints Cube: 0 Axial: 0	
2	N = 14 k = 2 p = 0 .centerPoints Cube: 3 Axial: 3	N = 18 k = 3 p = 0 .centerPoints Cube: 2 Axial: 2	N = 28 k = 4 p = 0 .centerPoints Cube: 2 Axial: 2	N = 48 k = 5 p = 0 .centerPoints Cube: 2 Axial: 4	N = 35 k = 5 p = 1 .centerPoints Cube: 6 Axial: 1	N = 83 k = 6 p = 0 .centerPoints Cube: 1 Axial: 6	N = 52 k = 6 p = 1 .centerPoints Cube: 4 Axial: 2	N = 154 k = 7 p = 0 .centerPoints Cube: 1 Axial: 11	
3		N = 20 k = 3 p = 0 .centerPoints Cube: 2 Axial: 2	N = 30 k = 4 p = 0 .centerPoints Cube: 2 Axial: 2	N = 50 k = 5 p = 0 .centerPoints Cube: 2 Axial: 4	N = 41 k = 5 p = 1 .centerPoints Cube: 6 Axial: 1	N = 84 k = 6 p = 0 .centerPoints Cube: 1 Axial: 6	N = 56 k = 6 p = 1 .centerPoints Cube: 4 Axial: 2	N = 155 k = 7 p = 0 .centerPoints Cube: 1 Axial: 11	
5			N = 34 k = 4 p = 0 .centerPoints Cube: 2 Axial: 2	N = 54 k = 5 p = 0 .centerPoints Cube: 2 Axial: 4	N = 53 k = 5 p = 1 .centerPoints Cube: 6 Axial: 1	N = 86 k = 6 p = 0 .centerPoints Cube: 1 Axial: 6	N = 64 k = 6 p = 1 .centerPoints Cube: 4 Axial: 2	N = 157 k = 7 p = 0 .centerPoints Cube: 1 Axial: 11	
9				N = 62 k = 5 p = 0 .centerPoints Cube: 2 Axial: 4		N = 90 k = 6 p = 0 .centerPoints Cube: 1 Axial: 6	N = 80 k = 6 p = 1 .centerPoints Cube: 4 Axial: 2	N = 161 k = 7 p = 0 .centerPoints Cube: 1 Axial: 11	N = 92 k = 7 p = 1 .centerPoints Cube: 1 Axial: 4
17						N = 98 k = 6 p = 0 .centerPoints Cube: 1 Axial: 6		N = 169 k = 7 p = 0 .centerPoints Cube: 1 Axial: 11	

À comparer avec Minitab...:



Créer un plan de surface de réponse - Afficher les plans disponibles

Plans de surface de réponse disponibles (avec nombre d'essais)

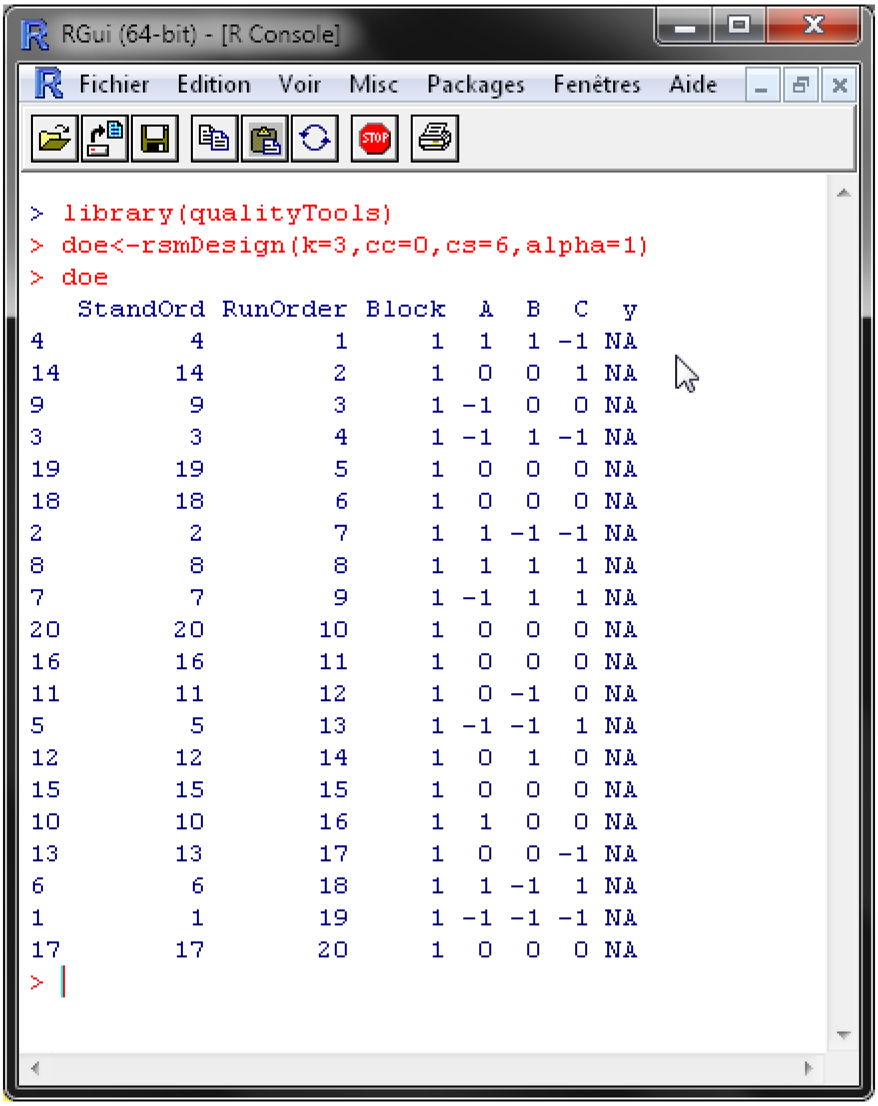
Plan		Facteurs								
		2	3	4	5	6	7	8	9	10
Composite centré (complet)	non divisé en blo	13	20	31	52	90	152			
	divisé en blocs	14	20	30	54	90	160			
Composite centré (1/2)	non divisé en blo				32	53	88	154		
	divisé en blocs				33	54	90	160		
Composite centré (1/4)	non divisé en blo							90	156	
	divisé en blocs							90	160	
Composite centré (1/8)	non divisé en blo									158
	divisé en blocs									160
Box-Behnken	non divisé en blo		15	27	46	54	62		130	170
	divisé en blocs			27	46	54	62		130	170

Aide OK

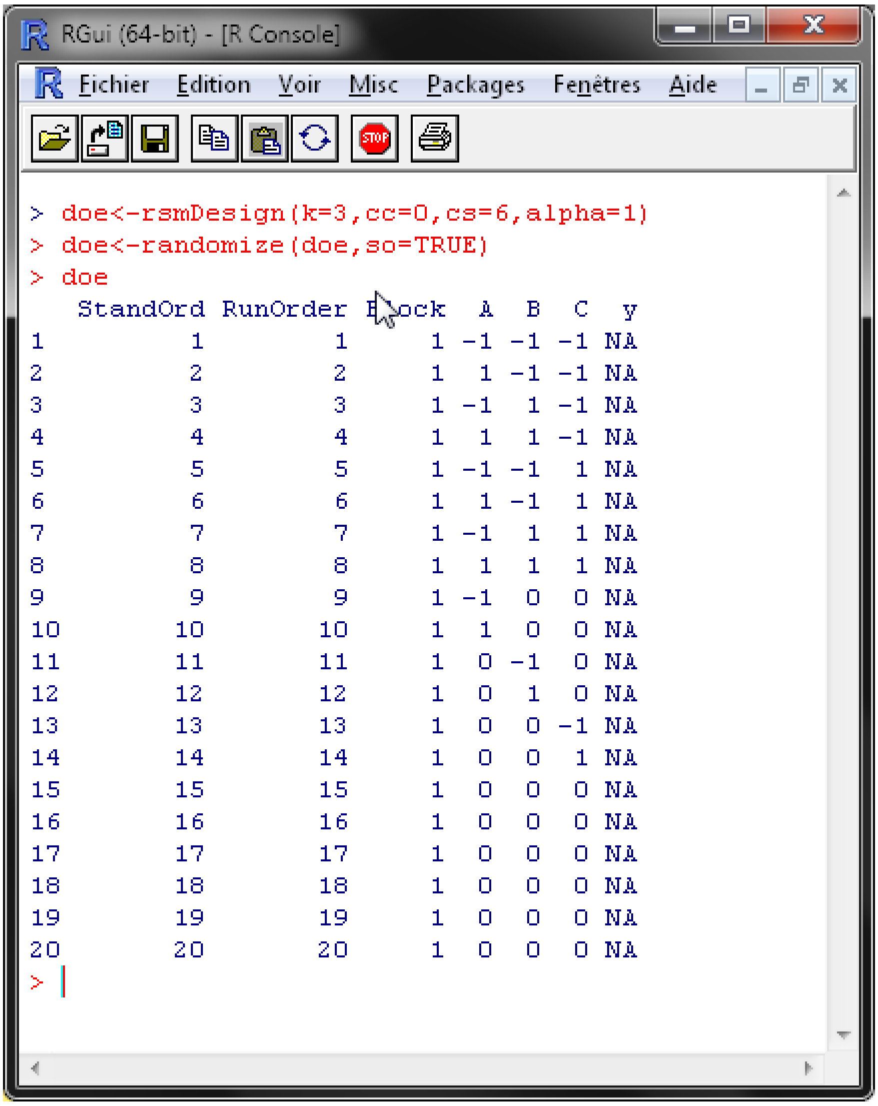
Analyse d'un plan composite centré

Le but va être ici de construire et d'analyse le même plan composite face centré qu'avec Minitab.

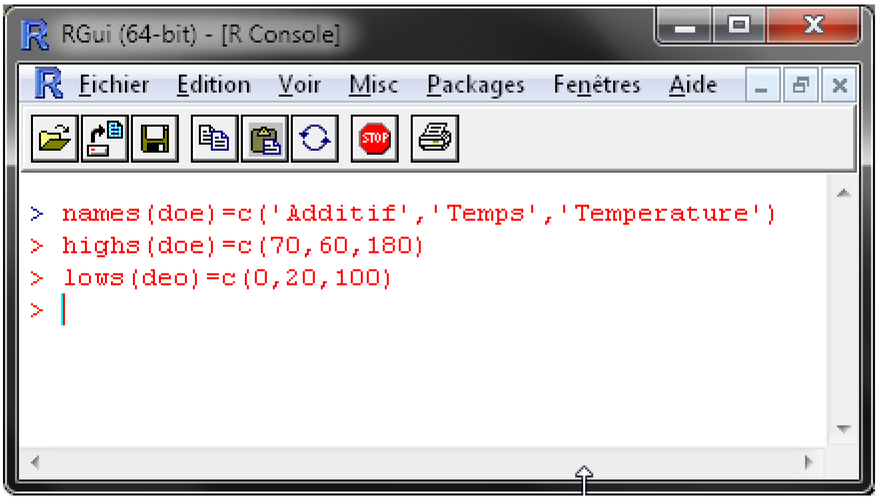
Pour cela nous commençons par utiliser la fonction **rsmDesign( )** avec les paramètres ad hoc pour retomber sur le même plan que dans le cours Minitab:



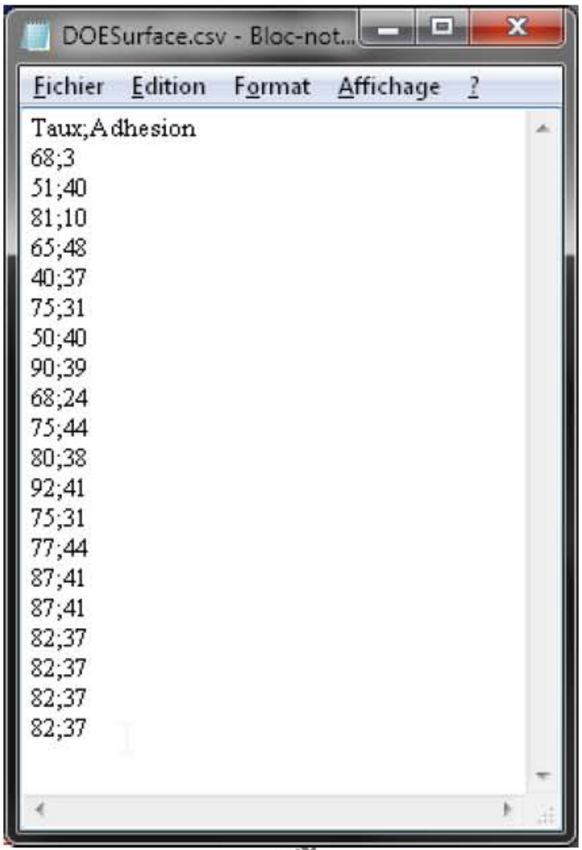
ou dans l'ordre en utilisant la commande **randomize( )** :



Nous pouvons déjà mettre maintenant les noms des facteurs et leurs niveaux réels en utilisant les commandes **names( )** , **highs( )** et **lows( )** :



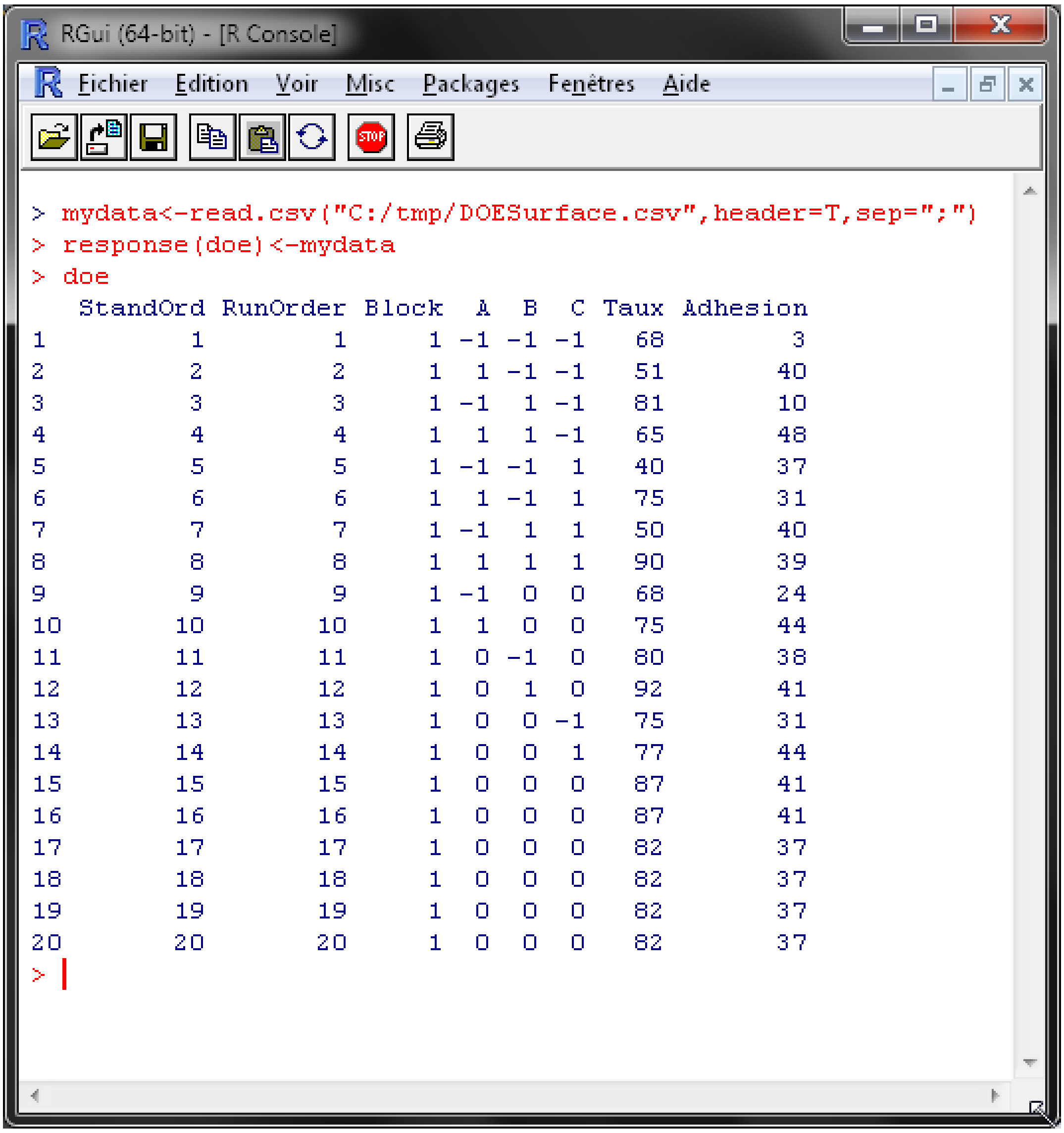
Maintenant, nous importons les valeurs mesurées des deux réponses d'intérêt qui sont les mêmes valeurs que dans le cours Minitab.


DOESurface.csv - Bloc-not...

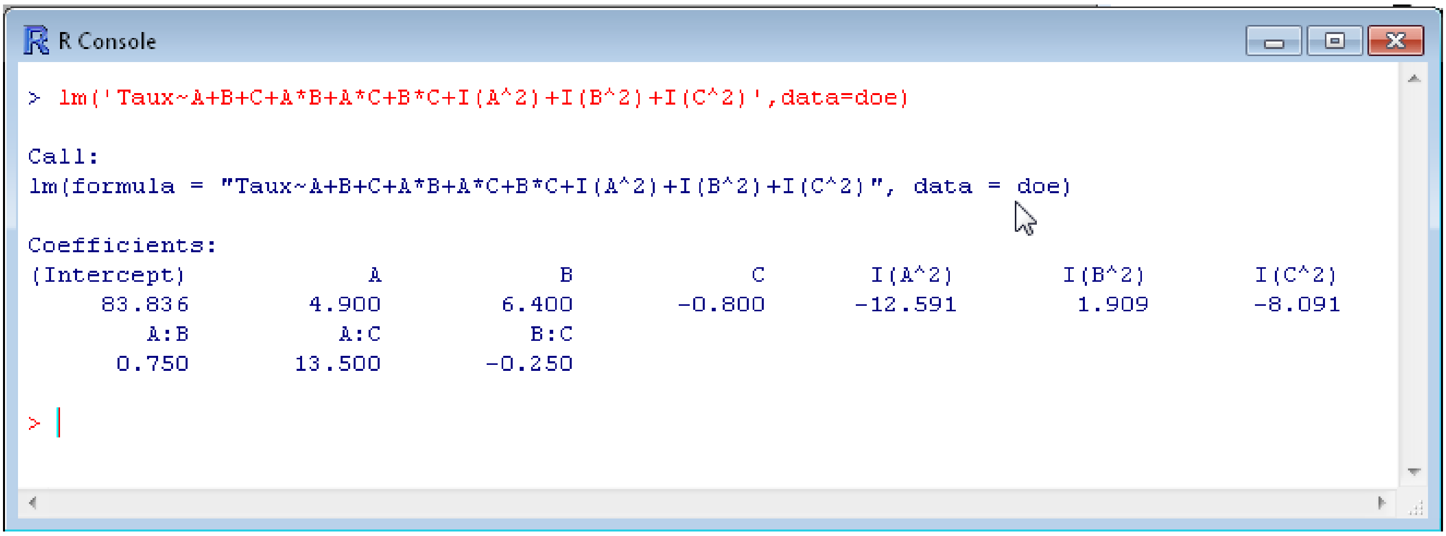
Fichier Edition Format Affichage ?

```
Taux;Adhesion
68;3
51;40
81;10
65;48
40;37
75;31
50;40
90;39
68;24
75;44
80;38
92;41
75;31
77;44
87;41
87;41
82;37
82;37
82;37
82;37
```

Pour cela, nous utilisons la fonction **read.csv( )** comme à l'habitude mais particulièrement la fonction **response( )** du package **qualityTools**:



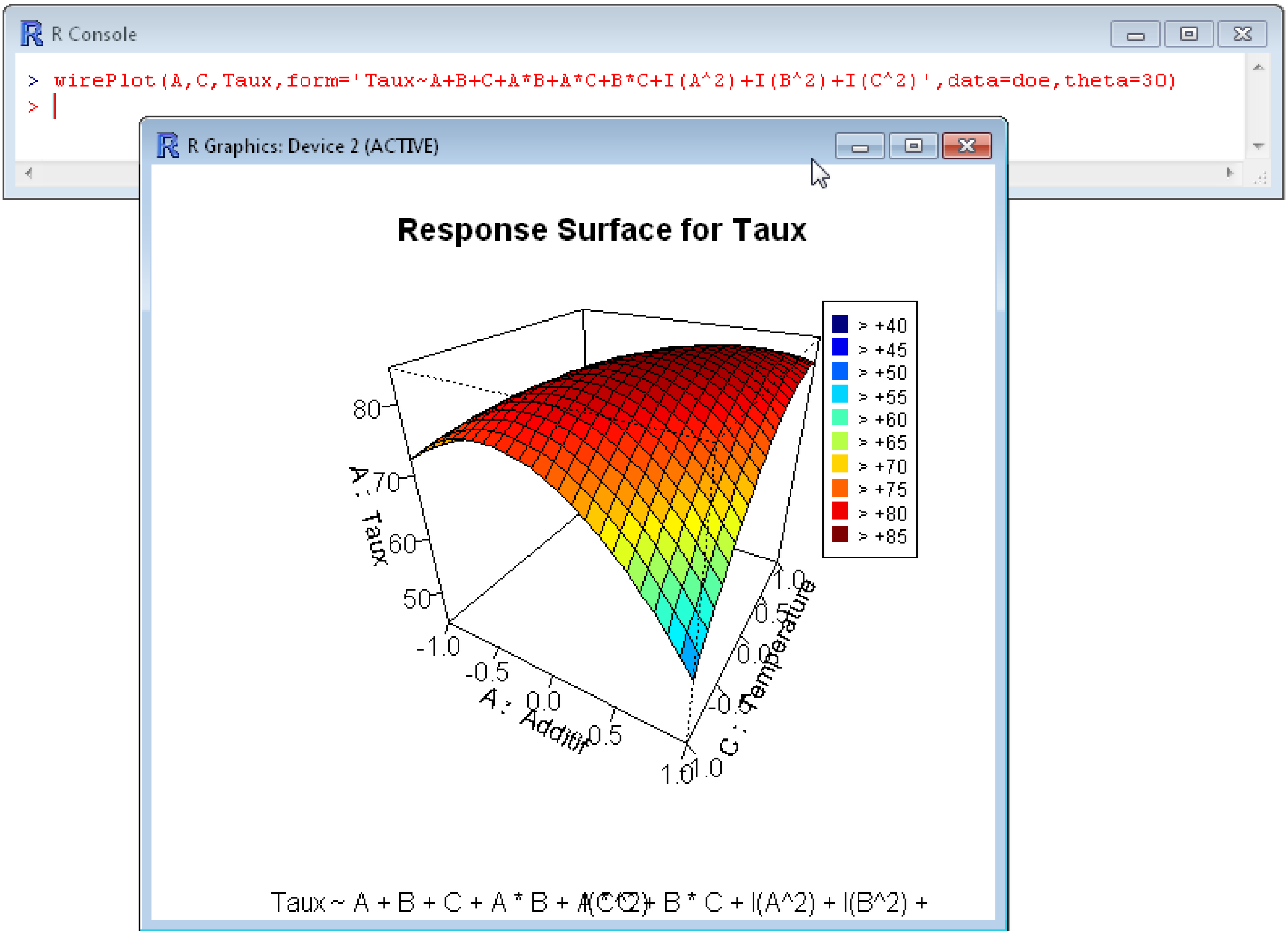
Ensuite, nous pouvons faire une régression quadratique classique:

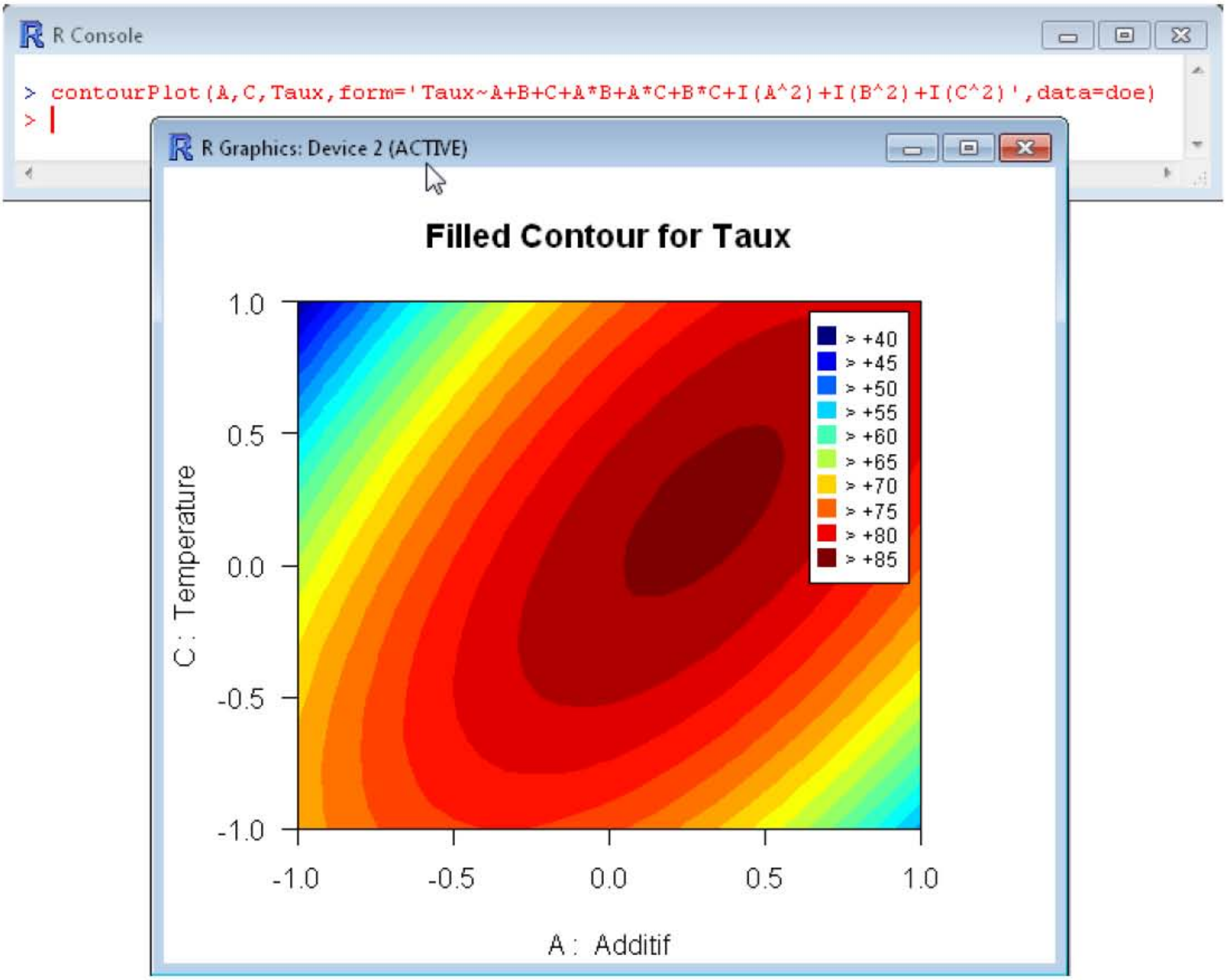


Là nous pouvons déjà constater qu'il est très dommageable que ce package utilise les unités codées pour faire la régression plutôt que les unités réelles... donc perte de temps en perspective par rapport à Minitab pour exploiter cette régression et aussi pour faire une comparaison directe avec Minitab.

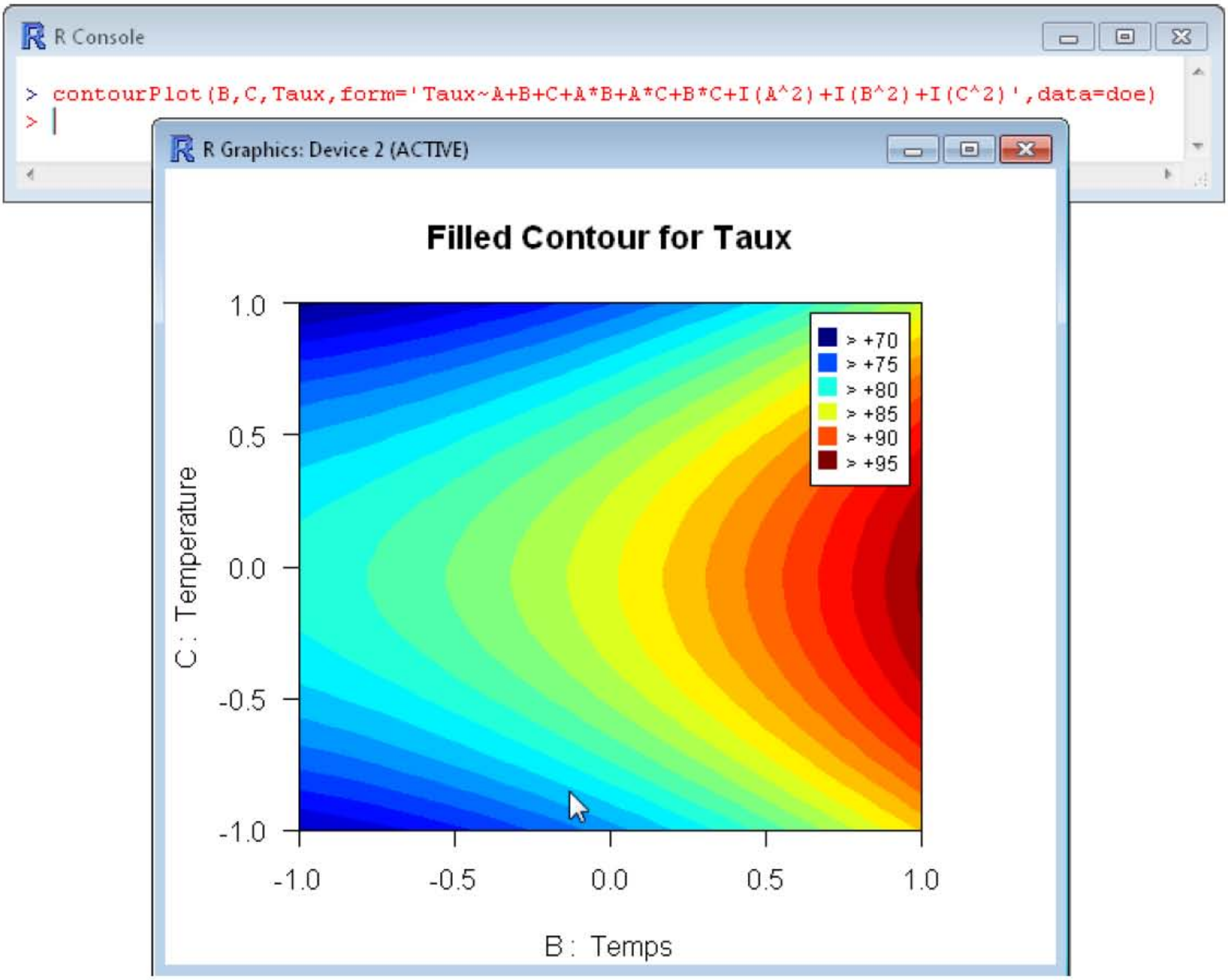
Ensuite vient la partie graphique où nous pouvons constater que les résultats sont très similaires à Minitab (difficile de dire s'ils sont parfaitement identiques).

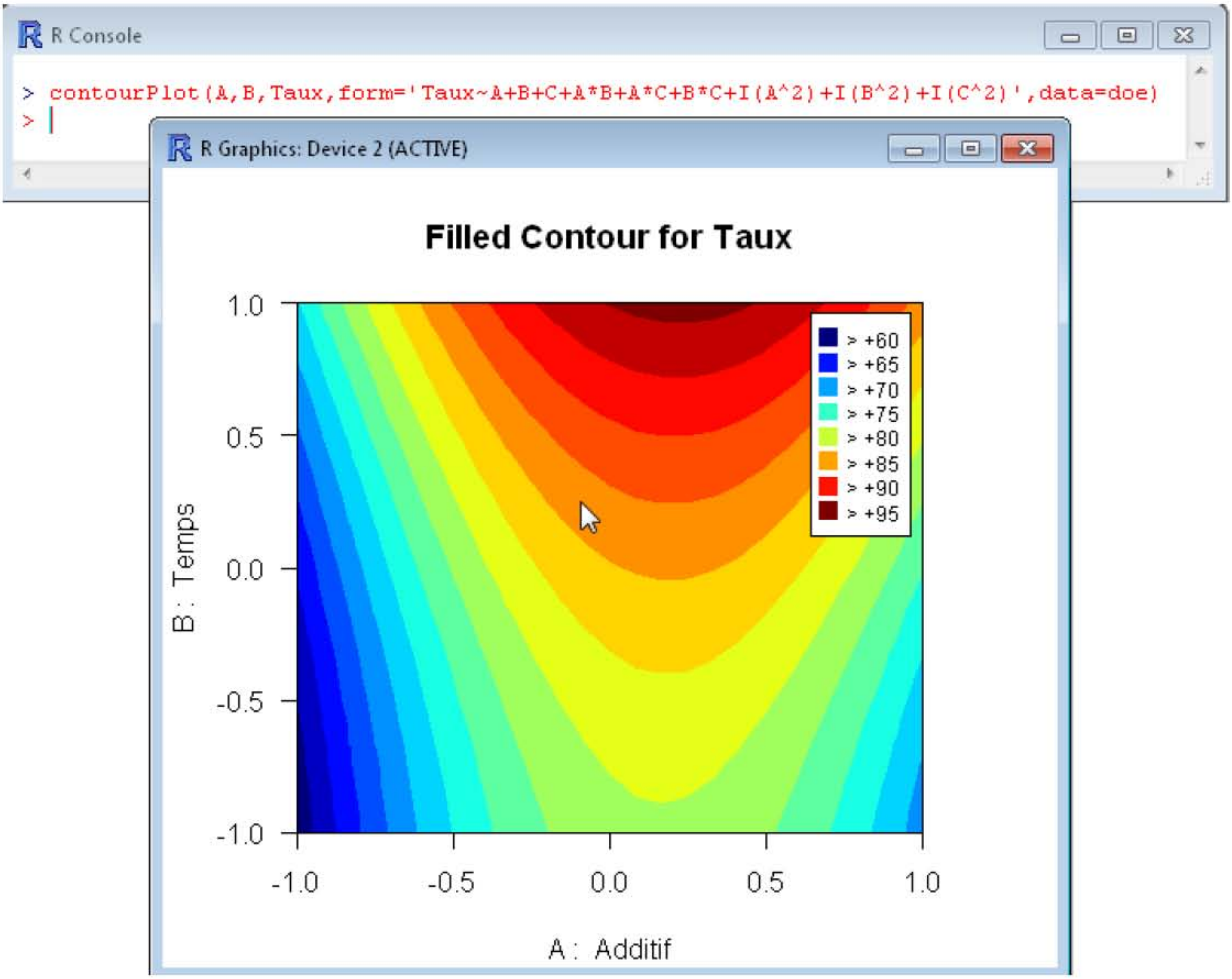
Nous commençons avec les facteurs *Taux* et la *Température*:



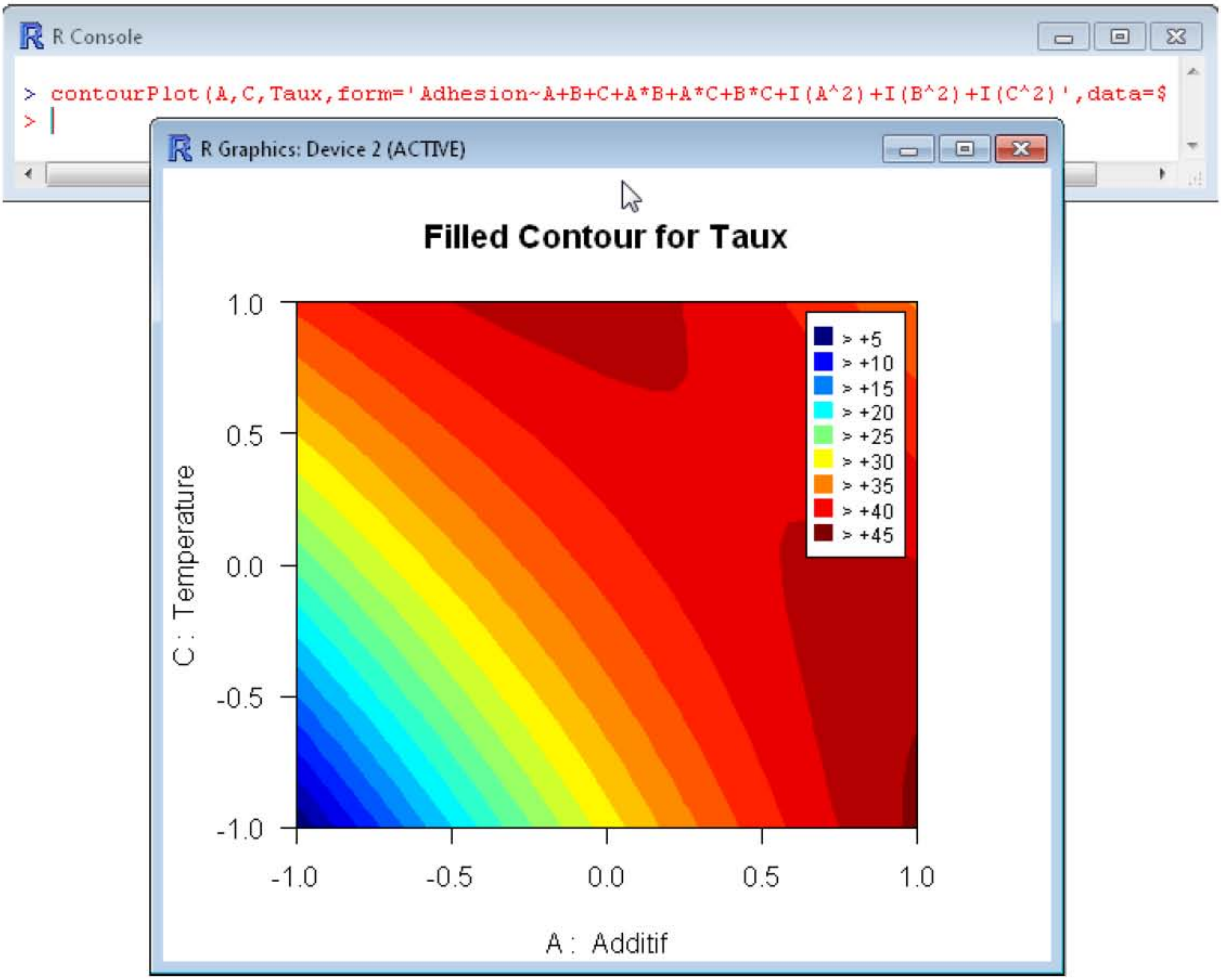

R Console
> contourPlot(A,C,Taux,form='Taux~A+B+C+A*B+A*C+B*C+I(A^2)+I(B^2)+I(C^2)',data=doe)
R Graphics: Device 2 (ACTIVE)
Filled Contour for Taux
> +40
> +45
> +50
> +55
> +60
> +65
> +70
> +75
> +80
> +85
C : Temperature
A : Additif
1.0
0.5
0.0
-0.5
-1.0

Nous poursuivons avec les facteurs T*emps* et *Température*:

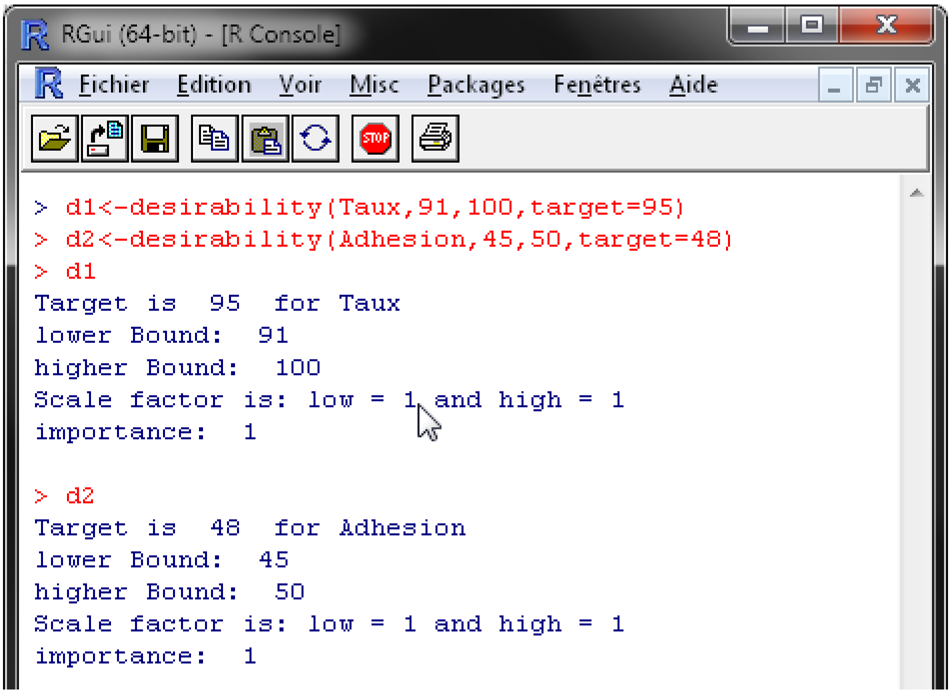



R Console
> contourPlot(A,B,Taux,form='Taux~A+B+C+A*B+A*C+B*C+I(A^2)+I(B^2)+I(C^2)',data=doe)
R Graphics: Device 2 (ACTIVE)
Filled Contour for Taux
> +60
> +65
> +70
> +75
> +80
> +85
> +90
> +95
B : Temps
A : Additif
1.0
0.5
0.0
-0.5
-1.0

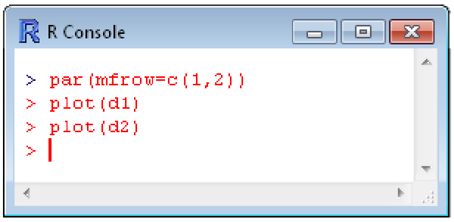
Et jetons un coup d'oeil aux courbes de nivaux avec les facteurs *Additif* et *Température*:



Maintenant définissons les contraintes de désirabilité pour chacune des réponses et prenons les options le plus proches de celles de Minitab à l'aide de la fonction **desirability( )** :



Nous pouvons représenter graphiquement la désirabilité:



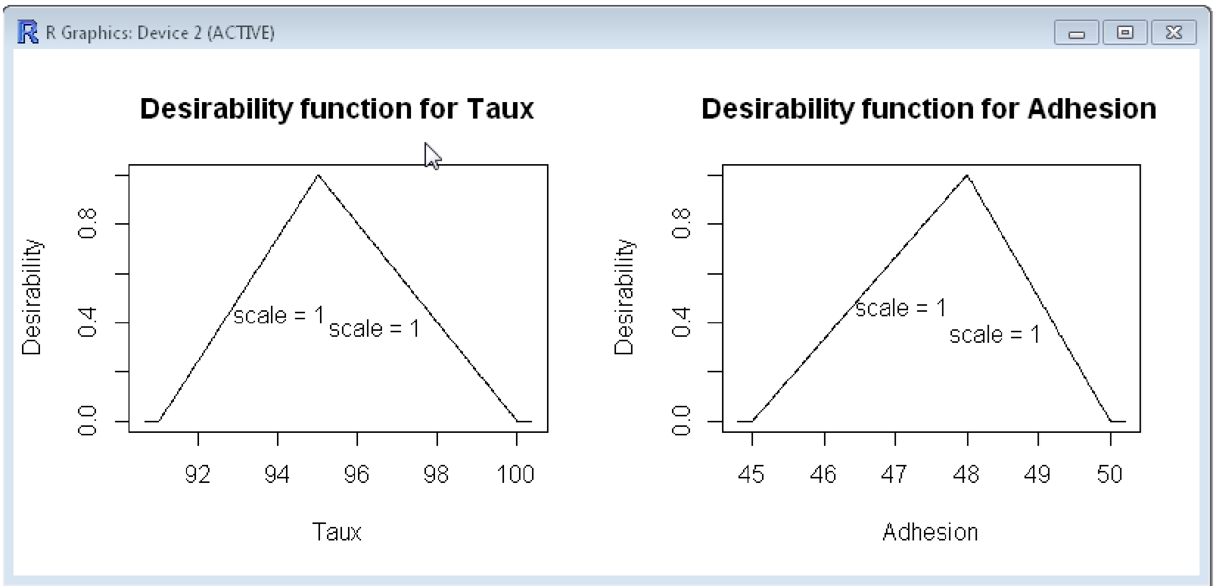
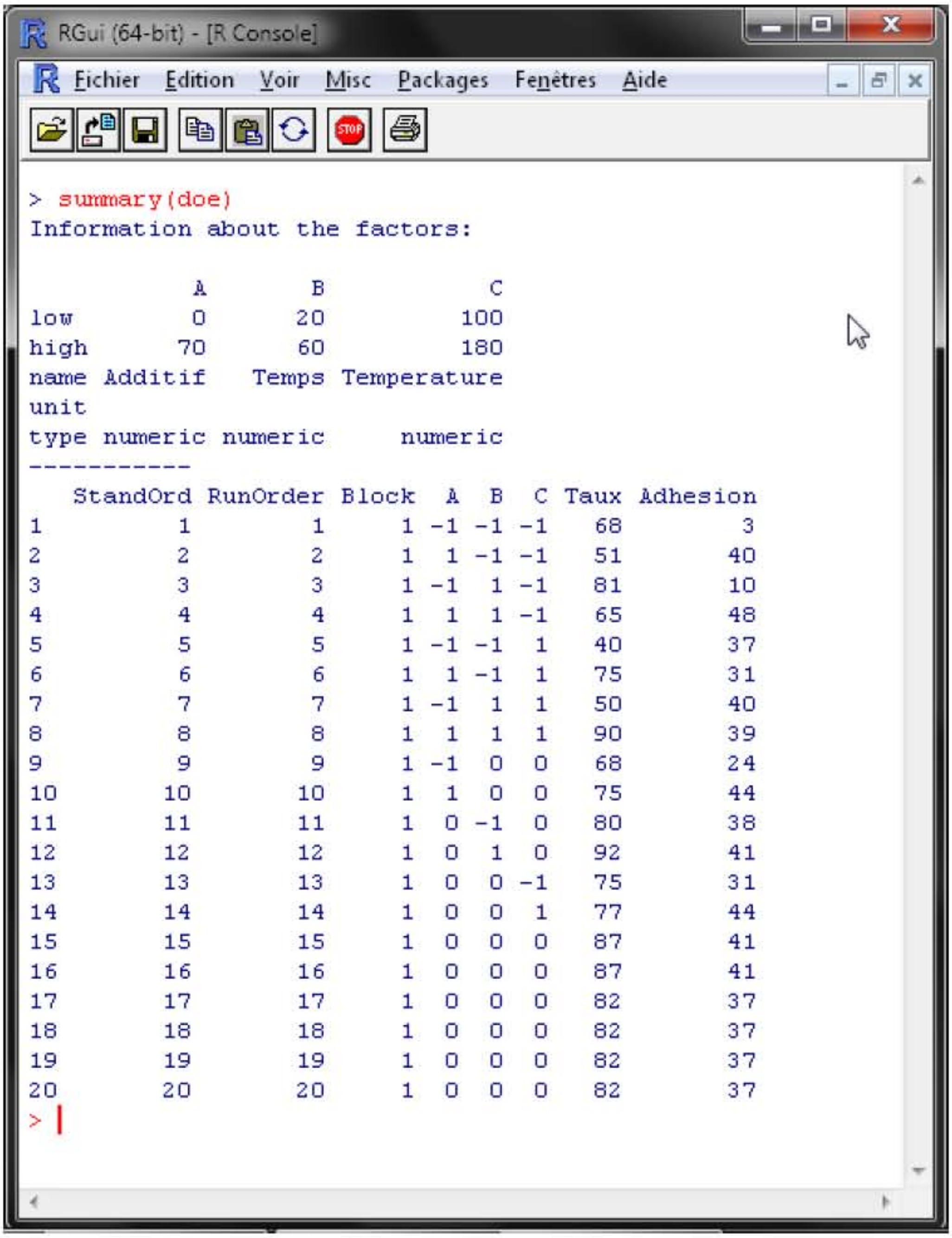
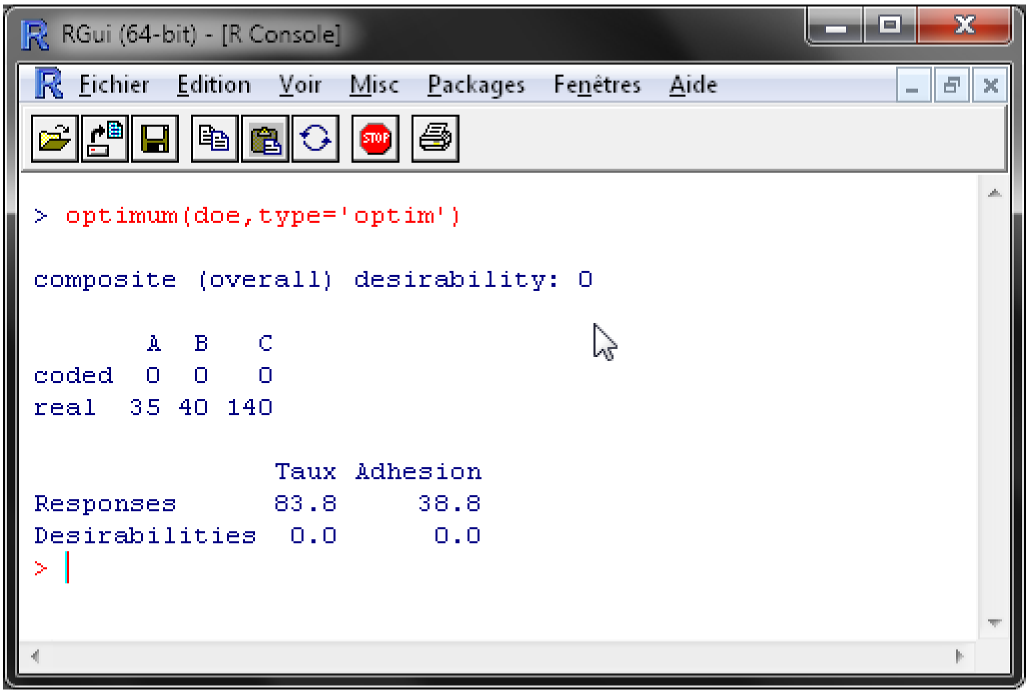


Figure 265 Graph de désirabilité pour plans d'expérience (DoE)

Faisons un petit résumé pour contrôler si jusqu'ici tout est bien défini:



Maintenant avec la fonction **optimum( )** du package **qualityTools** nous allons pouvoir vérifier si nous retombons sur la même réponse que Minitab:



et nous pouvons conster que nous en sommes très loin...

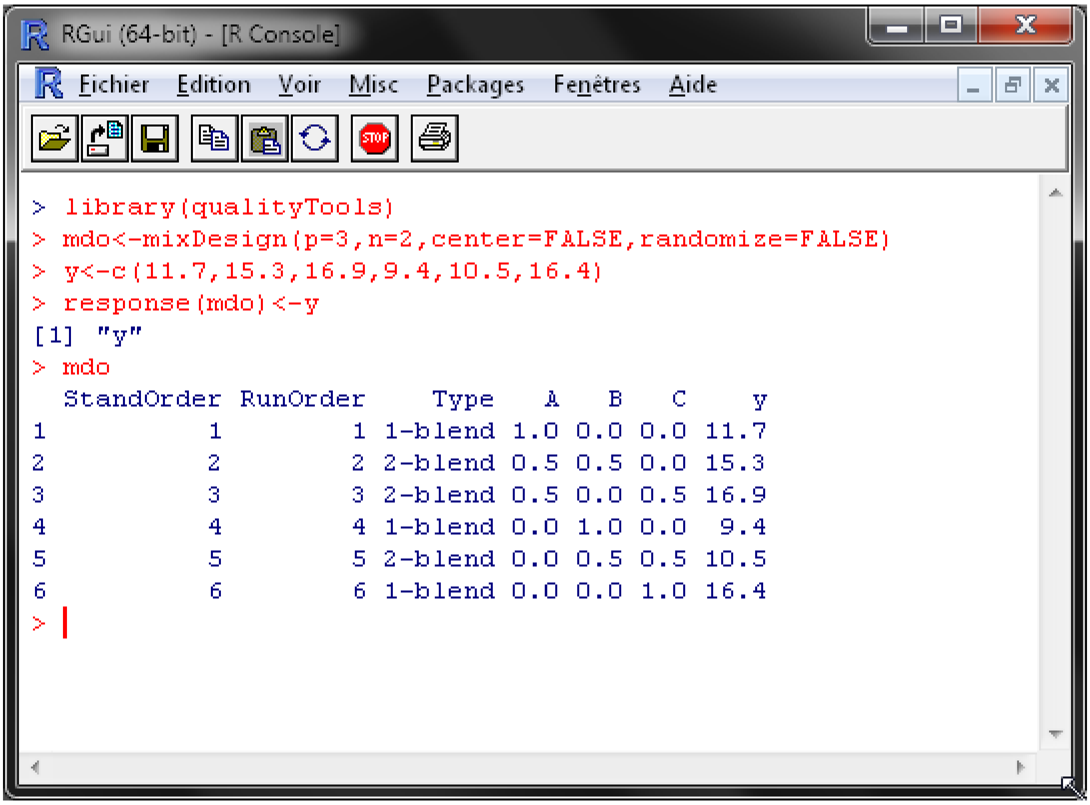
ToDo: Voir si c'est un bug ou une erreur de manipulation de ma part

Plans de mélanges

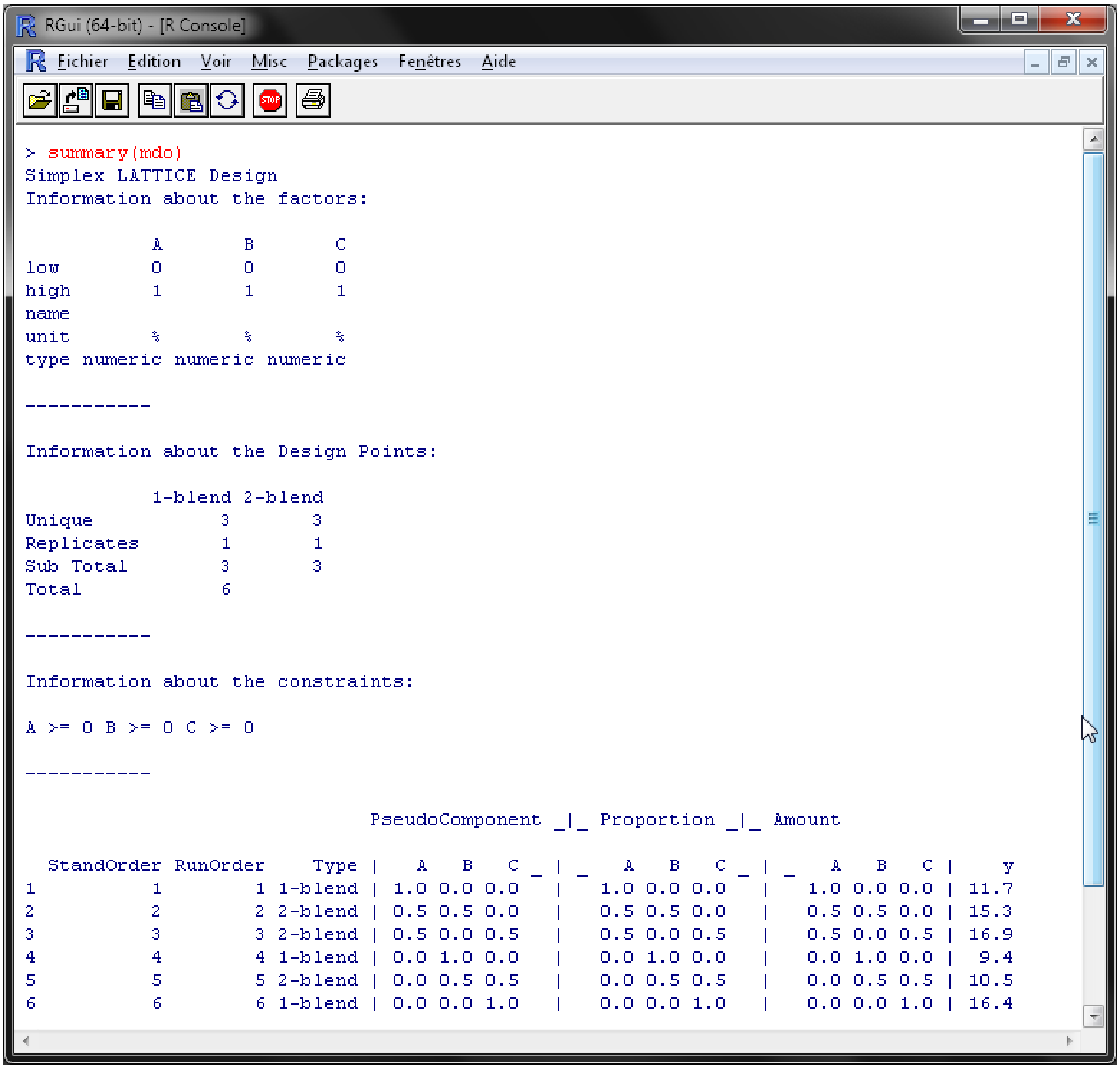
Plan de mélange centré du simplexe

Le but va être de vérifier ici (comme à l'habitude) les calculs appliqués relatifs aux démonstrations mathématiques faites dans le cours théorique relativement aux plans de mélanges avec points aux sommets et comparer aussi accessoirement les résultats obtenus avec Minitab.

Nous allons utiliser la commande **mixDesign( )** du package **qualityTools**:



La construction du plan de mélange étant faite, faisons-en quelques analyses avec d'abord un résumé:




```
> summary(mdo)
Simplex LATTICE Design
Information about the factors:

           A       B       C
low        0       0       0
high       1       1       1
name
unit       %       %       %
type numeric numeric numeric

-----------

Information about the Design Points:

           1-blend 2-blend
Unique           3       3
Replicates       1       1
Sub Total        3       3
Total            6

-----------

Information about the constraints:

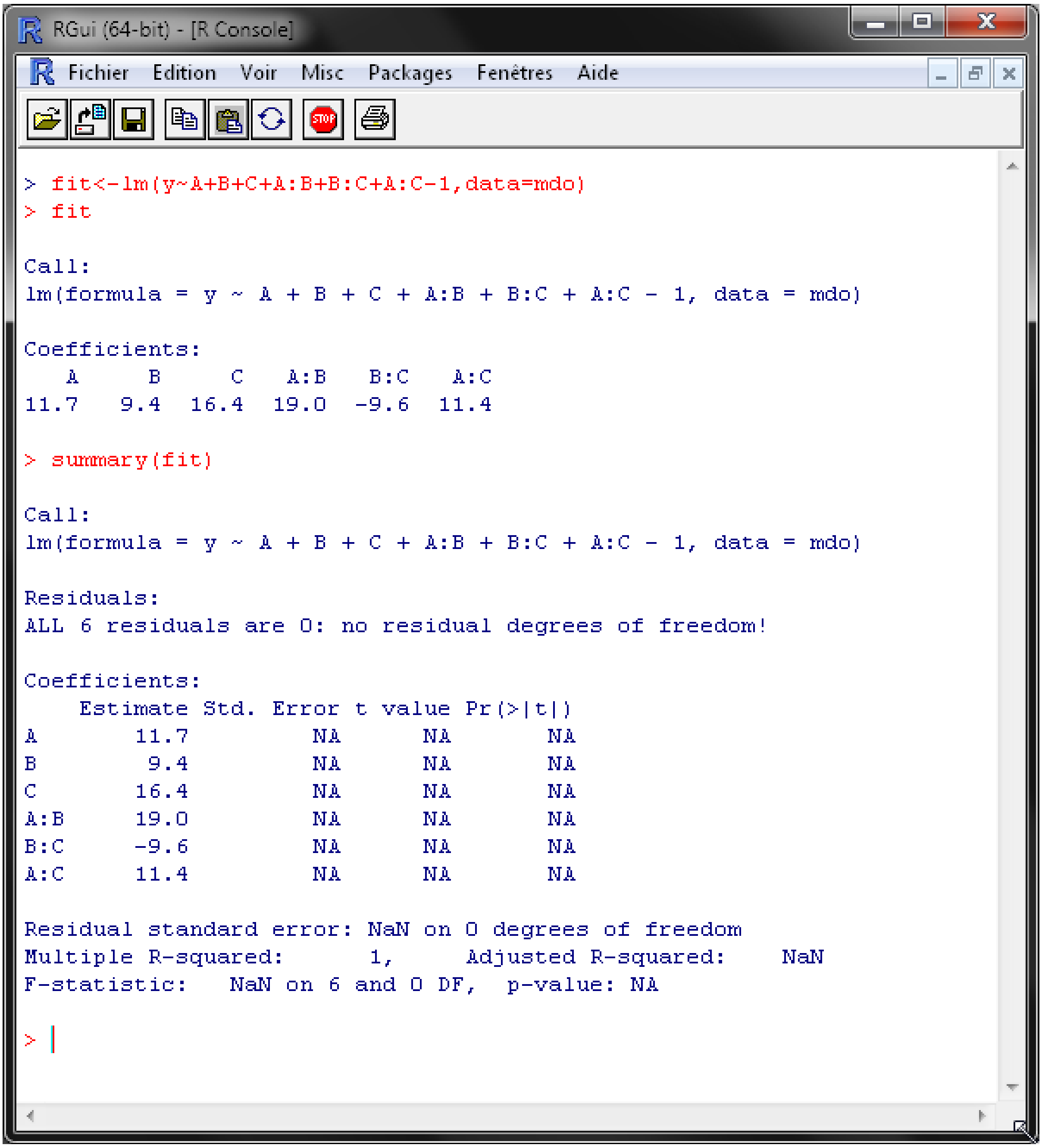
A >= 0 B >= 0 C >= 0

-----------

                              PseudoComponent _|_ Proportion _|_ Amount

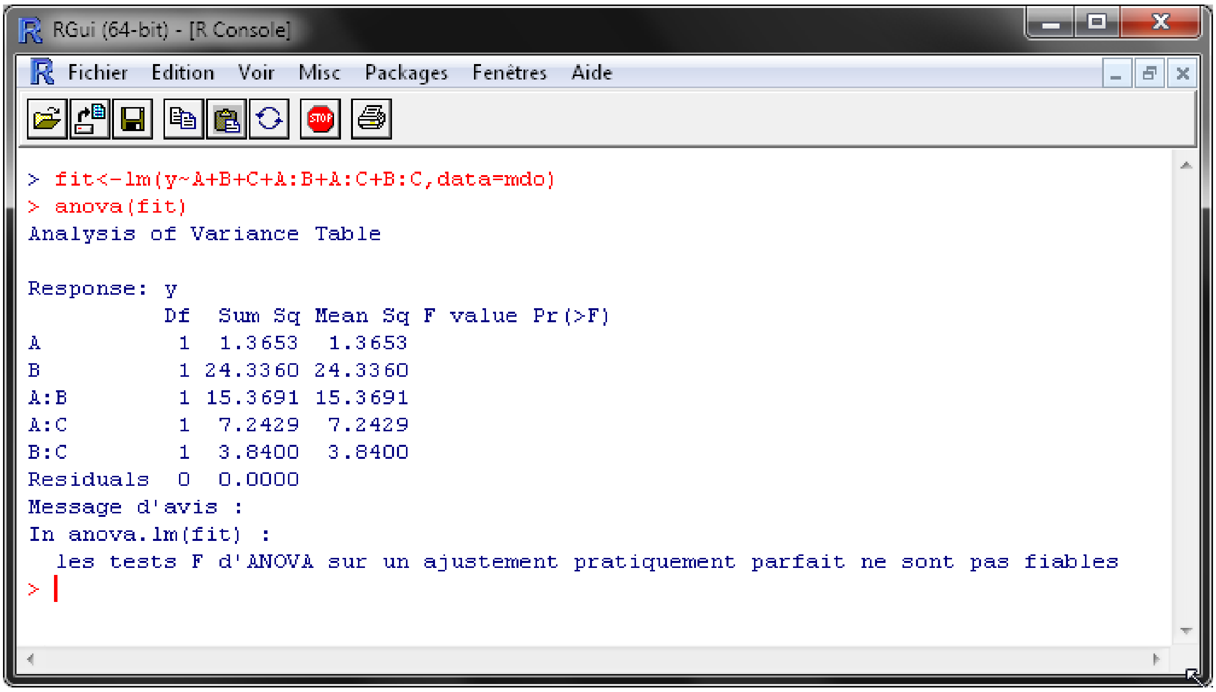
  StandOrder RunOrder    Type |   A   B   C _ | _   A   B   C _ | _   A   B   C |    y
1          1        1 1-blend | 1.0 0.0 0.0   |   1.0 0.0 0.0   |   1.0 0.0 0.0 | 11.7
2          2        2 2-blend | 0.5 0.5 0.0   |   0.5 0.5 0.0   |   0.5 0.5 0.0 | 15.3
3          3        3 2-blend | 0.5 0.0 0.5   |   0.5 0.0 0.5   |   0.5 0.0 0.5 | 16.9
4          4        4 1-blend | 0.0 1.0 0.0   |   0.0 1.0 0.0   |   0.0 1.0 0.0 |  9.4
5          5        5 2-blend | 0.0 0.5 0.5   |   0.0 0.5 0.5   |   0.0 0.5 0.5 | 10.5
6          6        6 1-blend | 0.0 0.0 1.0   |   0.0 0.0 1.0   |   0.0 0.0 1.0 | 16.4
```

Ensuite, nous appliquons le classique modèle linéaire généralisé gaussien avec interactions:



Alors nous retrouvons bien les valeurs des coefficients calculés à la main mais chose curieuse... R (comme Minitab) n'arrive pas à calculer l'erreur ni les intervalles de confiance et la *p*-value alors que nous avons démontré qu'un tel calcul était possible (et nous l'avons fait!). Raison pour laquelle... (encore une fois!) il est important de rappeler qu'il faut toujours plusieurs progiciels de statistiques pour faire des études sérieuses et un peu plus sûres.

Faisons maintenant une ANOVA du modèle quadratique complet:



Nous pouvons voir que celle-ci est plus intéressant qu'avec Minitab où nous n'avions qu'un résumé objectivement peu exploitable (Minitab faisant la somme des termes linéaires et la somme des termes quadratiques dans le tableau de l'ANOVA).

Finissons par un plot (le rendu est meilleur que Minitab mais les finitions prennent à ce jour plus de temps que dans Minitab) en utilisant respectivement les commandes **contourPlot3( )** et **wirePlot3( )** du package **qualityTools** (ce qui nous permet d'éviter d'expliciter les termes d'interactions car il les prendra en compte automatique)

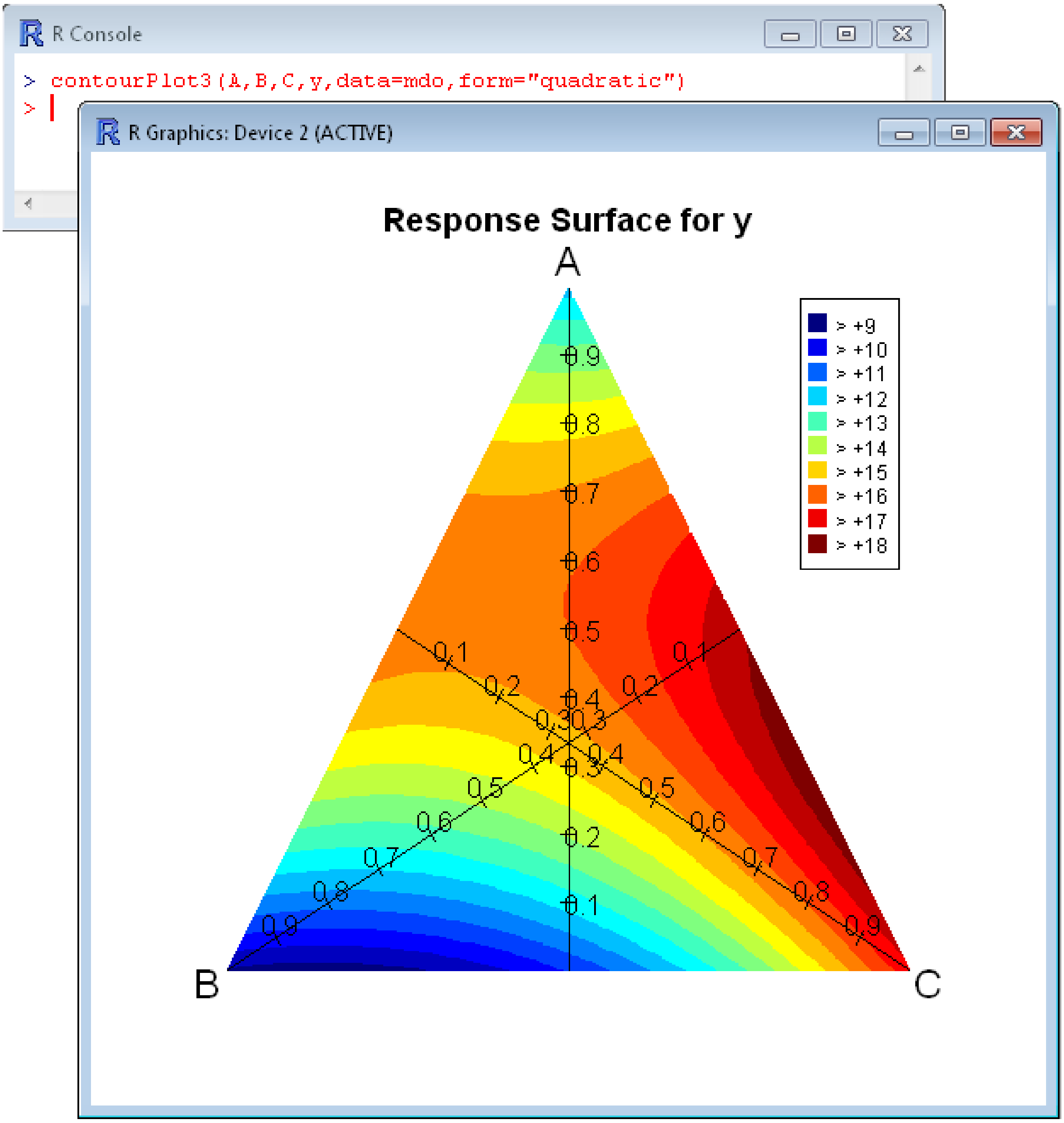


Figure 266 Graph de surface de réponse ternaire pour plans d'expérience (DoE)

et pour la surface de réponse:

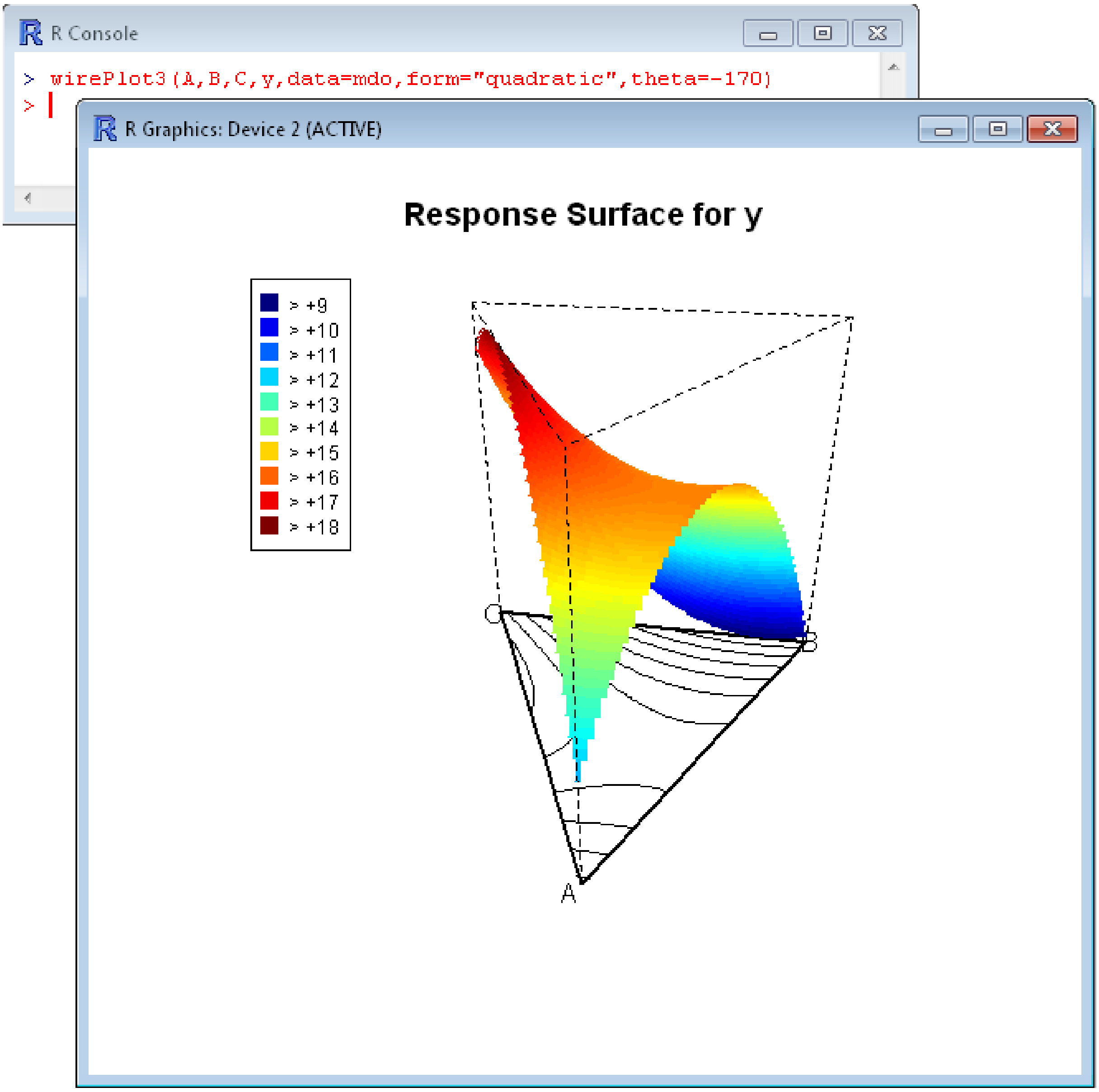


Figure 267 Graph de surface de réponse 3D ternaire pour plans d'expérience (DoE)

Nous avons donc beaucoup moins d'options que dans Minitab mais pour du gratuit c'est quand même extraordinaire.

20. Ingénierie de la qualité (Six Sigma/SPC)

Le but de ce chapitre va être de vérifier si nous pouvons nous débarrasser de Minitab en utilisant R pour le domaine de l'ingénierie de la qualité et l'application de la méthode Six Sigma dans la maîtrise statistique des processus (SPC: Statistical Process Control).

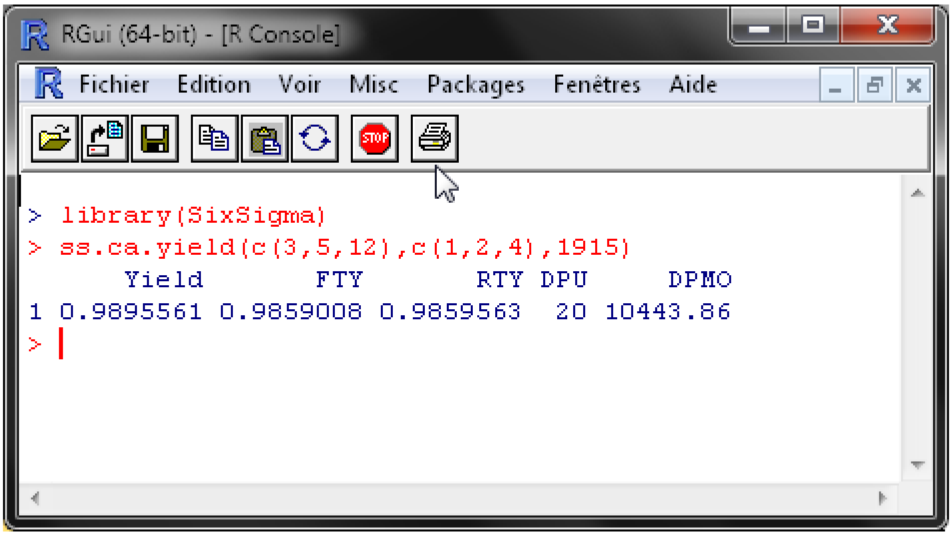
La majorité de ce chapitre (si ce n'est la totalité en réalité) se satellise autour du package **qcc**.

Exemple 292.: Rendement Global Combiné et DPU/DPMO

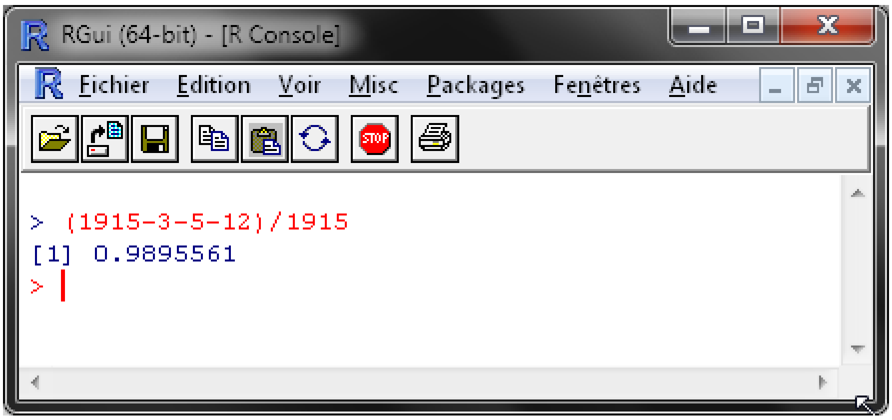
R 3.1.0

Nous avons étudié dans le cours de génie industriel la notion de rendement global combiné et de DPU/DPMO dans le cadre de processus auxquels nous appliquons l'approche Six Sigma.

Voyons ce que le package **SixSigma** nous propose avec la commande **ss.ca.yield( )** dans le cas d'un procédé de fabrication où sur un total de 1915 unités produites nous avons à la respectivement aux trois premières étapes du processus 3, 5 et 12 éléments défectueux et nous avons à chaque étape respectivement réinjecté 1, 2 et 4 éléments retouchés (corrigé):

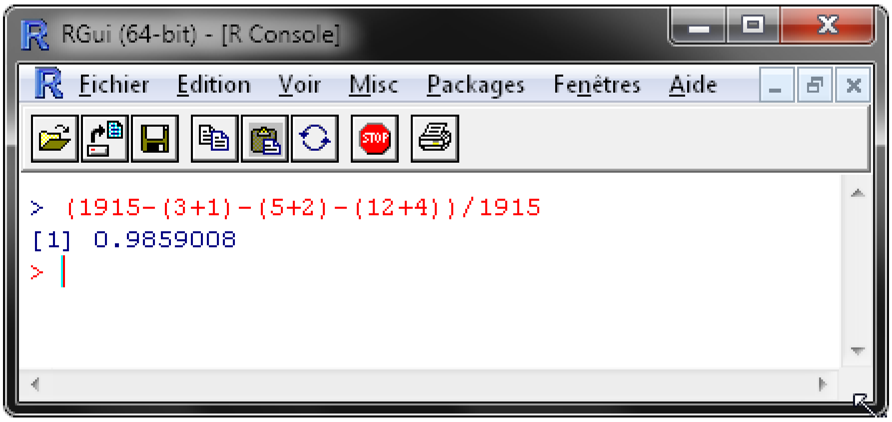


Contrôlons la première valeur le taux d'efficacité du processus (Yield):



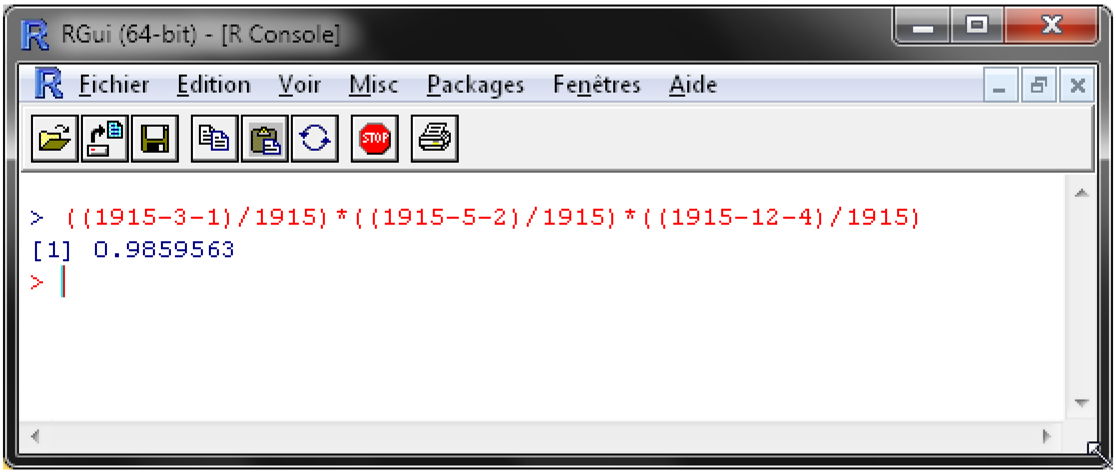
tout est ok, c'est logique... c'est trivial!

Contrôlons la deuxième valeur qui est le taux d'efficacité avec retouches (FTY: First Time Yield):



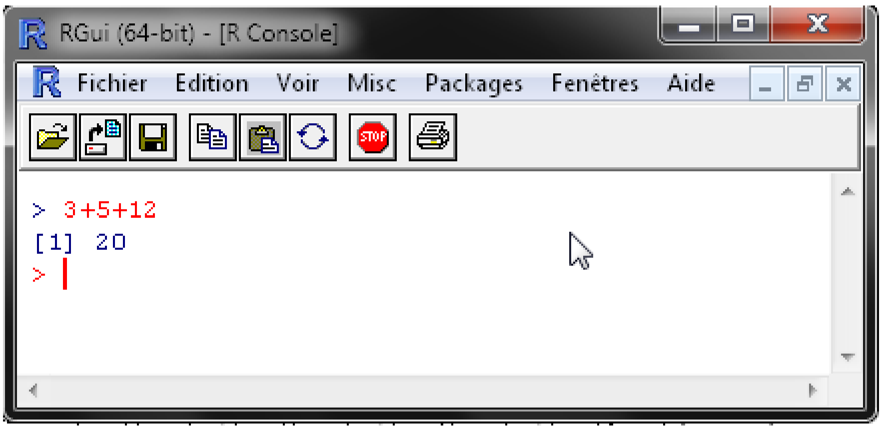
C'est conforme aussi, logique et trivial!

Voyons comment le package **SixSigma** calcule la troisième valeur qui est le rendement global combiné (RTY: Rolled Troughput Yield):



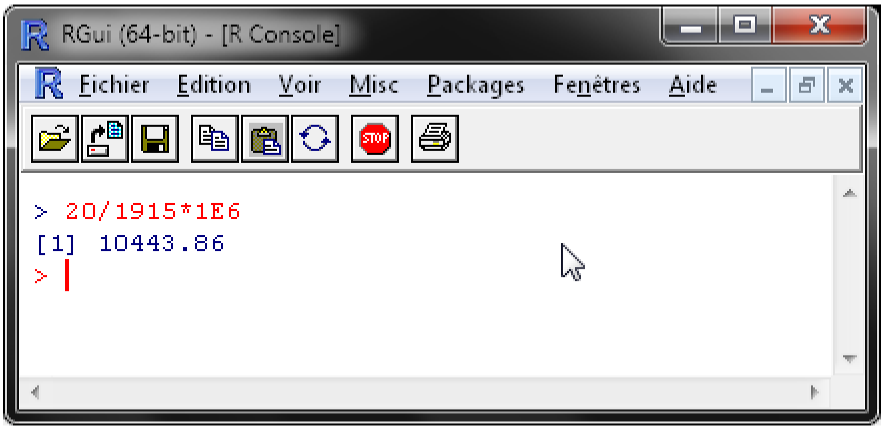
La définition utilisée n'est donc pas commune... puisqu'à chaque étape il reprend la quantité initiale. Dommage... (du moins à ma connaissance).

Bon pour le DPU c'est simple:



Mais là encore... la définition prise par le package **SixSigma** n'est pas forcément des plus judicieuses puisque normalement le DPU est le ratio du nombre total d'éléments défectueux sur le nombre total d'unité de production observées...

Et pour le DPMO:



Là aussi la définition utilisée est discutable.

Raison pour laquelle il est toujours important de rappeler dans les rapports la définition utilisée dans votre entreprise!

Voir par exemple (outre le support du cours théorique), le site suivant pour un définition assez courante:

http://www.isixsigma.com/dictionary/dpo/

Exemple 293.: Fonction de coût de perte de qualité de Taguchi

R 3.1.0

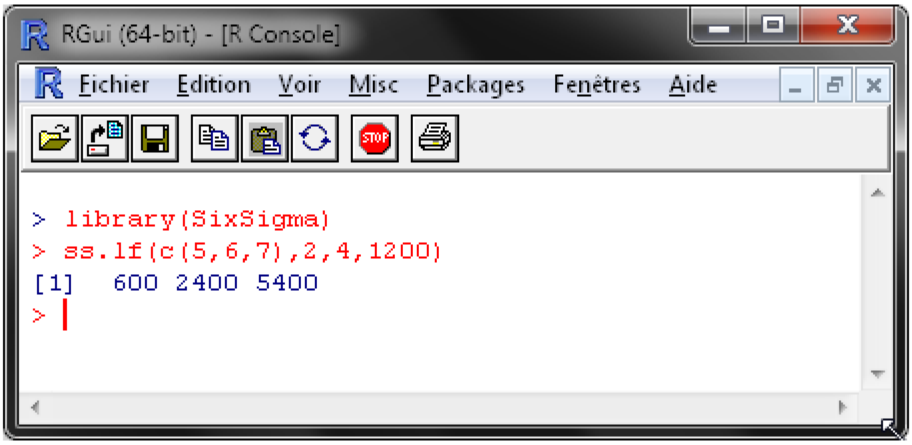
Nous allons vérifier ici que nous retrouvons donc les mêmes valeurs que celles calculées à la main dans le cours théorique de Génie Industriel.

Rappelons le contexte:

Dans un projet, une tâche de 4 jours a une marge totale de 2 jours. Si nous sortons de cette marge, le projet va prendre du retard pour un coût de 1200.-/jour.

Quel est le coût réel associé à 1, 2 et 3 jours de retard selon le modèle de Taguchi?

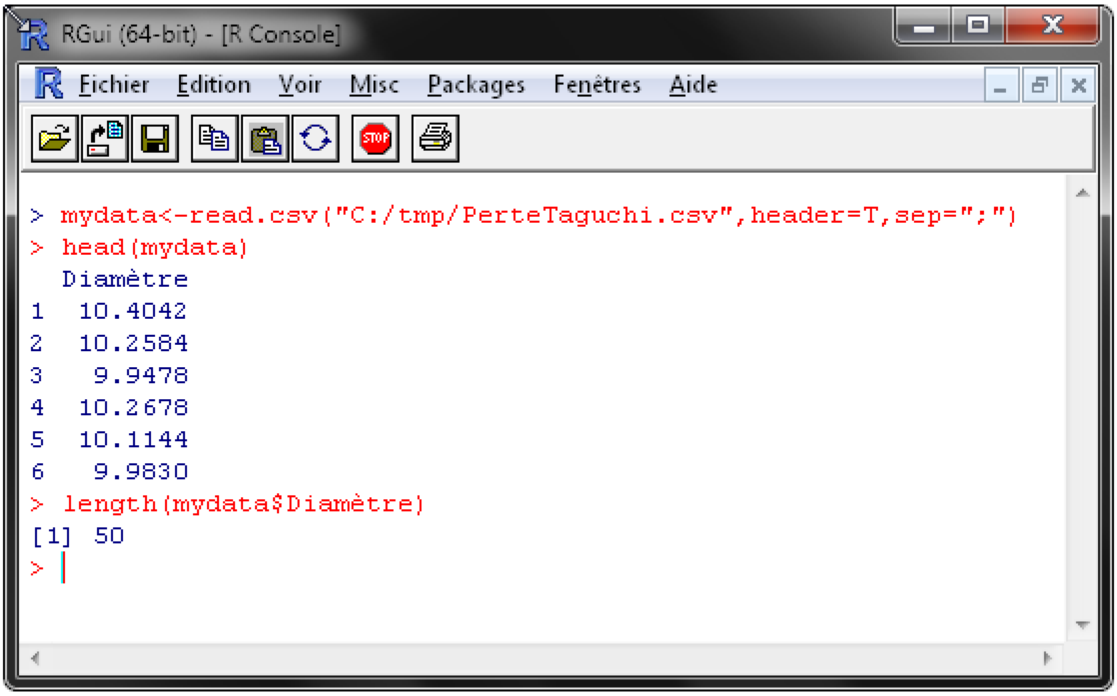
Dès lors, en utilisant la commande **ss.lf( )** du package SixSigma, nous obtenons:



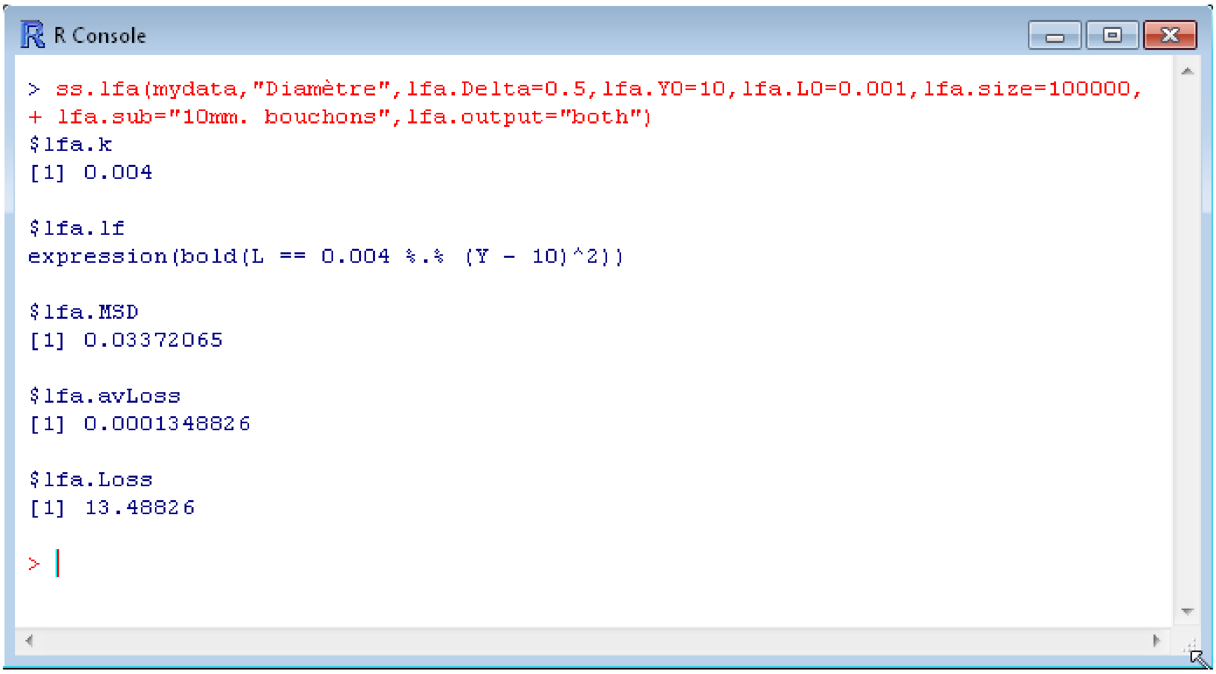
Soit exactement les mêmes valeurs que celles calculées à la main!

Maintenant voyons un cas de calcul avec la situation vue dans le cours théorique, appelée "le nominal est le meilleur.

Nous importons d'abord les données:



Ensuite, nous utilisons la commande **ss.lfa( )** du package **SixSigma** ce qui nous donne la sortie suivante dans un premier temps:



Ce qui correspond bien aux calculs faits à la main dans le cours théorique. Nous avons aussi automatiquement le graphique suivant donné par R qui correspond aussi à ce qui a été vu dans le cours théorique:

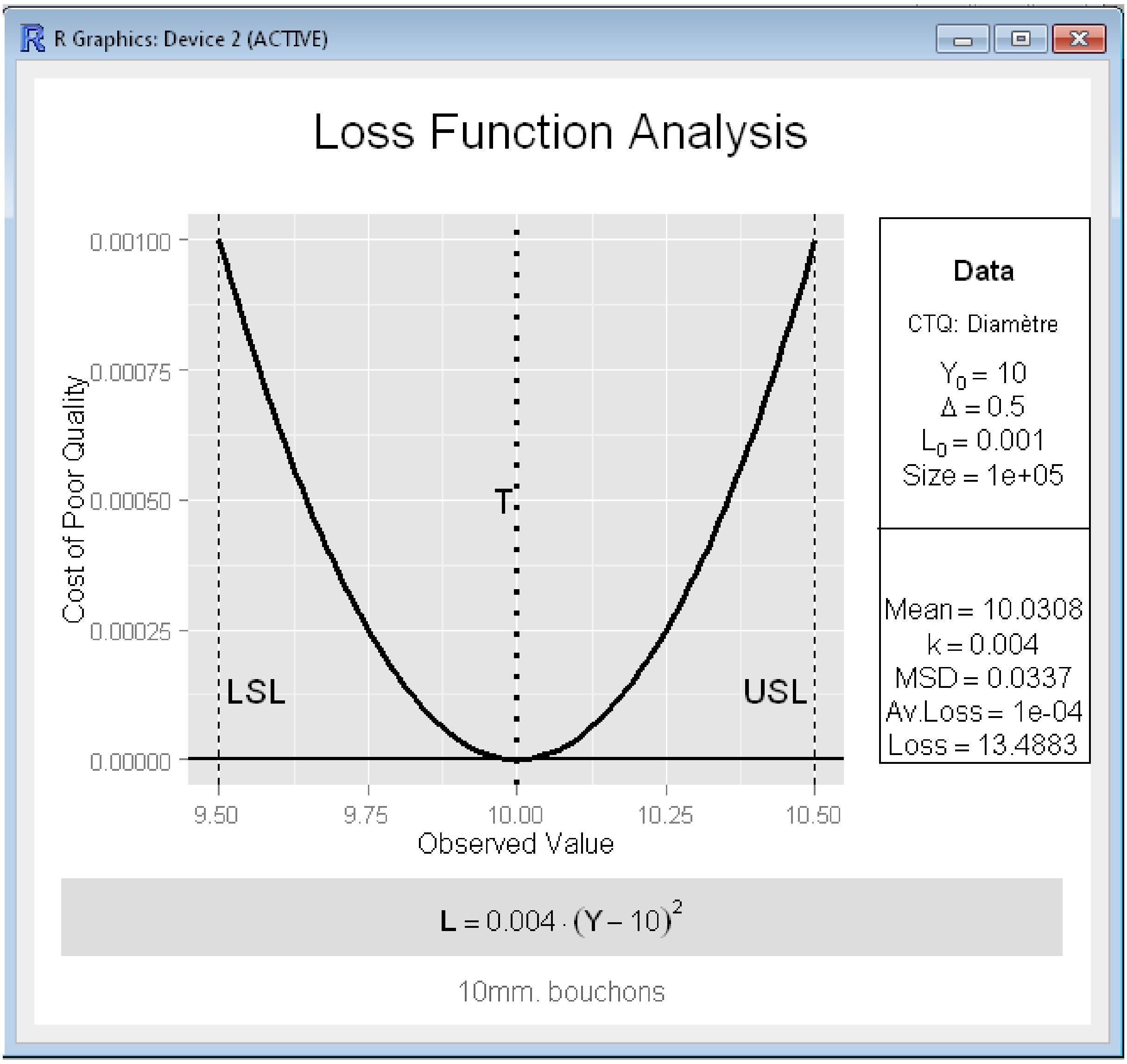


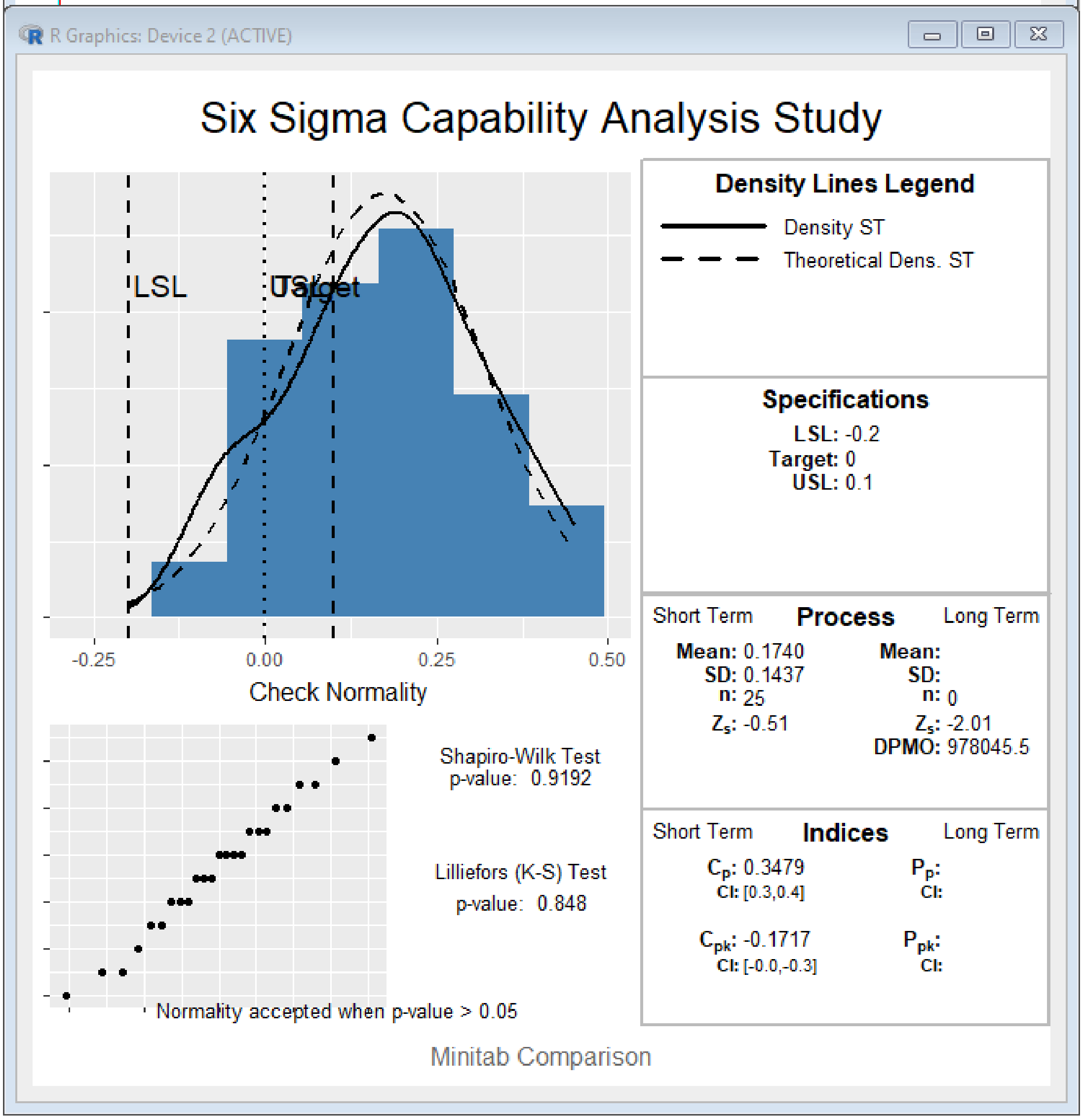
Figure 268 Graph fonction de perte de Taguchi

Exemple 294.: Analyse de capabilité de données non-groupées

R 3.6.2

Voyons si nous pouvons reproduire ici les calculs de capabilité fait à la main dans le cours théorique et avec Minitab à l'aidre de R et encore une fois du package **SixSigma** et sa fonction **ss.study.ca( )**:

```
> library("SixSigma")
> mesures <- c(0.45,0.40,0.35,0.35,0.30,0.30,
+ 0.25,0.25,0.25,0.20,0.20,0.20,0.20,
+ 0.15,0.15,0.15,0.1,0.1,0.1,0.05,0.05,
+ 0,-0.05,-0.05,-0.1)
> ss.study.ca(mesures, LSL = -0.2, USL = 0.1,
+             Target = 0,
+             alpha = 0.05,
+             f.su = "Minitab Comparison")
>
```



Ce qui est bien conforme!

Exemple 295.: Analyse SixPack de données groupées (+ analyse capabilités + intervalles de confiance des capabilités)

R 3.0.2

Encore une fois, nous allons contrôler que nous obtenons les mêmes données que celles calculées manuellement dans le cours théorique, ainsi qu'avec Microsoft Excel et Minitab.

La fonction que nous allons utiliser ici n'est plus disponible depuis la version 2.2 du package **qcc** (on se demande bien sous la pression de qui...). Il faut donc décompresser le fichier:

qcc_2.0.tar.gz

que je vous mets à disposition pendant le cours et téléchargeable ici:

http://cran.r-project.org/src/contrib/Archive/qcc/

y prendre les fichiers suivants:

capability.R
qcc.R
utilities.R

Nous travaillerons avec les mêmes données brutes, c'est-à-dire:

SixSigma.csv - Notepad

```
Echantillon;Mesure;Temps
1;-2;1
1;-4;2
1;-1;3
1;0;4
1;4;5
1;0;6
1;3;7
1;0;8
1;1;9
1;-1;10
2;0;1
2;-3;2
2;0;3
2;-2;4
2;1;5
2;-2;6
2;0;7
2;1;8
2;-1;9
2;2;10
3;-1;1
3;0;2
3;-3;3
```

Ensuite, nous utilisons les commandes **qcc.groups( )** pour préparer les données:

```
> source("C:/qcc.R")
> source("C:/capability.R")
> source("C:/utilities.R")
> mydata<-read.csv("C:/SixSigma.csv",header=T,sep=";")
> mydata<-qcc.groups(mydata$Mesure,mydata$Echantillon)
> mydata
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
1   -2   -4   -1    0    4    0    3    0    1    -1
2    0   -3    0   -2    1   -2    0    1   -1     2
3   -1    0   -3   -1    0    0   -1   -1    3     1
4    1    1   -2    2    2    0    1    0    4     0
5   -1   -1   -3    0    0    3    3    2    1     0
> 
```

Ensuite, nous sommes obligés de choisir au moins une carte de contrôle pour l'analyse six pack. Nous prendrons alors une carte X-barre en utilisant la commande **qcc( )** :

```
> obj<-qcc(t(mydata),type="xbar")
> summary(obj)

Call:
qcc(data = t(mydata), type = "xbar")

xbar chart for t(mydata)

Summary of group statistics:
   Min. 1st Qu. Median    Mean 3rd Qu.    Max.
  -1.80   -0.50   0.30    0.12    1.00    1.60

Group sample size:  5
Number of groups:  10
Center of group statistics:  0.12
Standard deviation:  1.676698

Control limits:
       LCL      UCL
 -2.129527 2.369527
> 
```

avec le graphique associé:

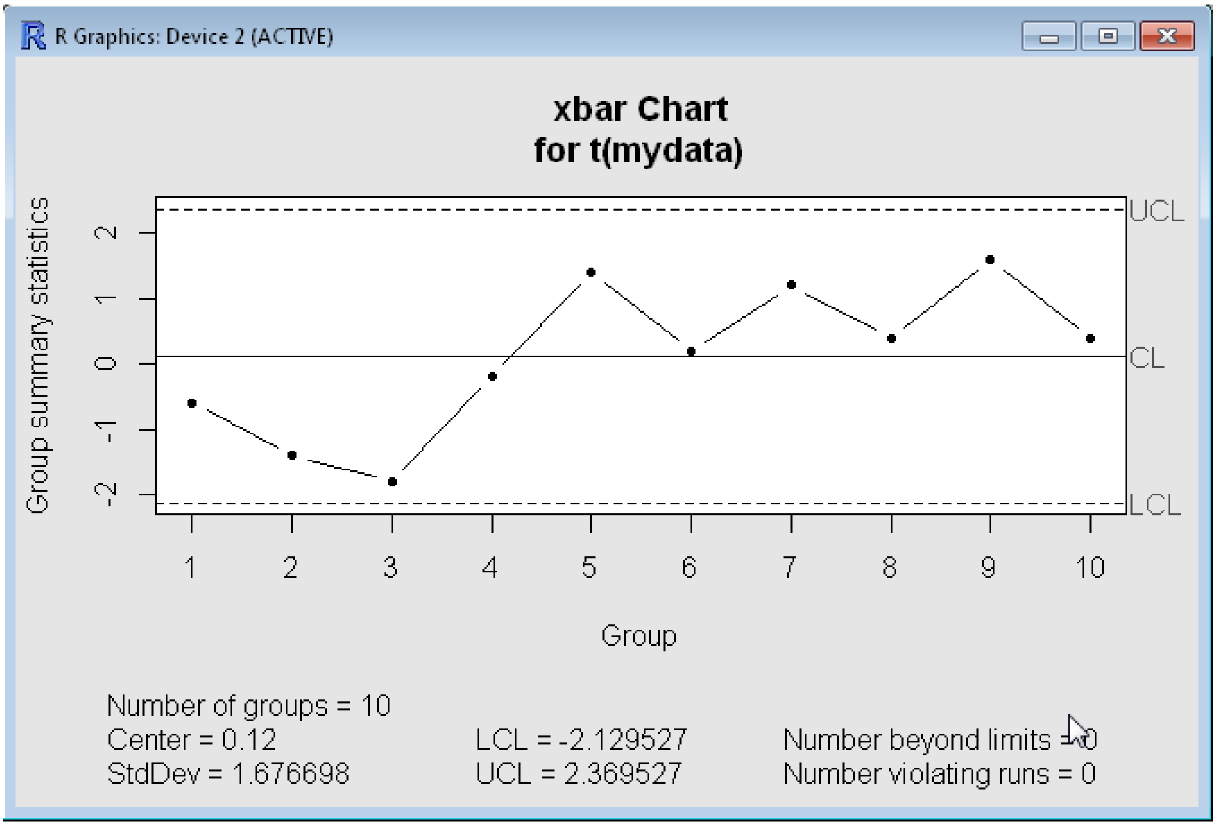


Figure 269 Graph carte de contrôle xbar

Donc à part l'esthétique qui est relativement médiocre mais améliorable en changeant le code source, nous retrouvons exactement les mêmes résultats que ceux calculés dans le cours théorique et dans Minitab excepté pour le LCL et l'UCL où il y a une différence de quelques pourcents.

Et c'est maintenant que vient la commande qui a disparue dans les nouvelles versions du package. Nous allons donc utiliser la commande **process.capability( )** :

```
R Console

> process.capability(obj,spec.limits=c(-7,7),target=0)

Process Capability Analysis

Call:
process.capability(object = obj, spec.limits = c(-7, 7), target = 0)

Number of obs = 50          Target = 0
       Center = 0.12           LSL = -7
       StdDev = 1.676698       USL = 7

Capability indices:

      Value   2.5%  97.5%
Cp    1.392  1.117  1.666
Cp_l  1.415  1.168  1.663
Cp_u  1.368  1.128  1.608
Cp_k  1.368  1.082  1.654
Cpm   1.388  1.116  1.660

Exp<LSL 0%   Obs<LSL 0%
Exp>USL 0%   Obs>USL 0%
>
```


avec le graphique:

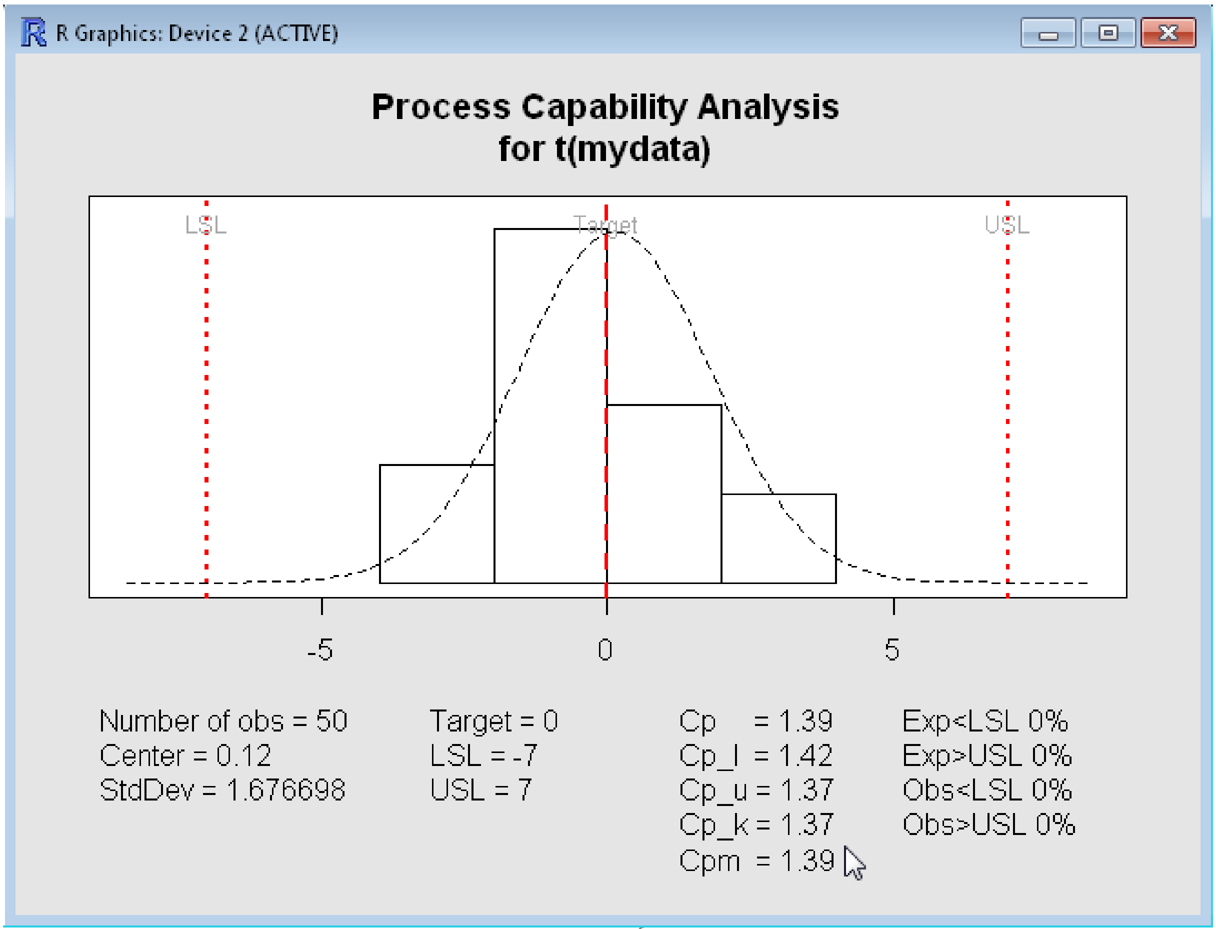
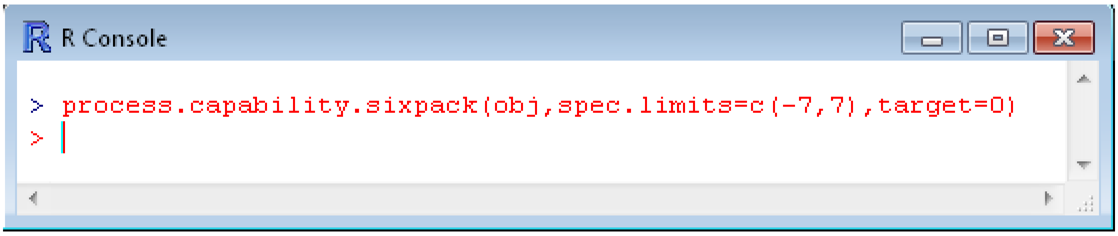


Figure 270 Graph d'analyse de capabilité

Là aussi les résultats ne sont pas tout à fait les mêmes que ceux calculés dans le cours théorique et que ceux calculés à la main mais il y a toutefois un avantage très intéressant avec R: nous avons les intervalles de confiance pour tous les indicateurs de capabilité :-)

Et avec la commande **process.capability.sixpack( )** qui a aussi disparue dans les dernières versions du package **qcc**, nous avons:



Avec le graphique associé (la encore on est loin derrière la qualité esthétique de Minitab où de la pertinence des informations que donne ce dernier sur le graphe SixPack:

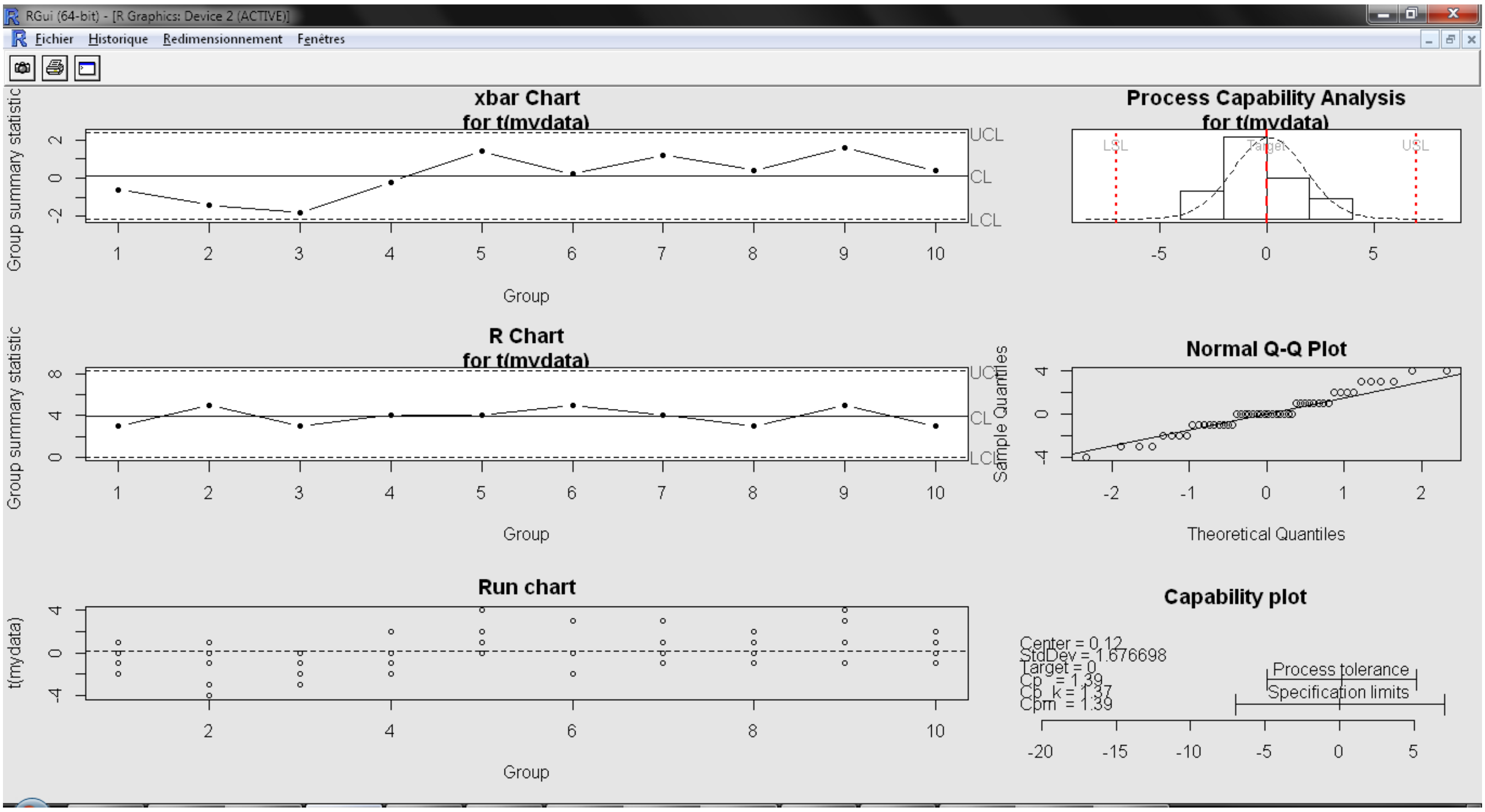
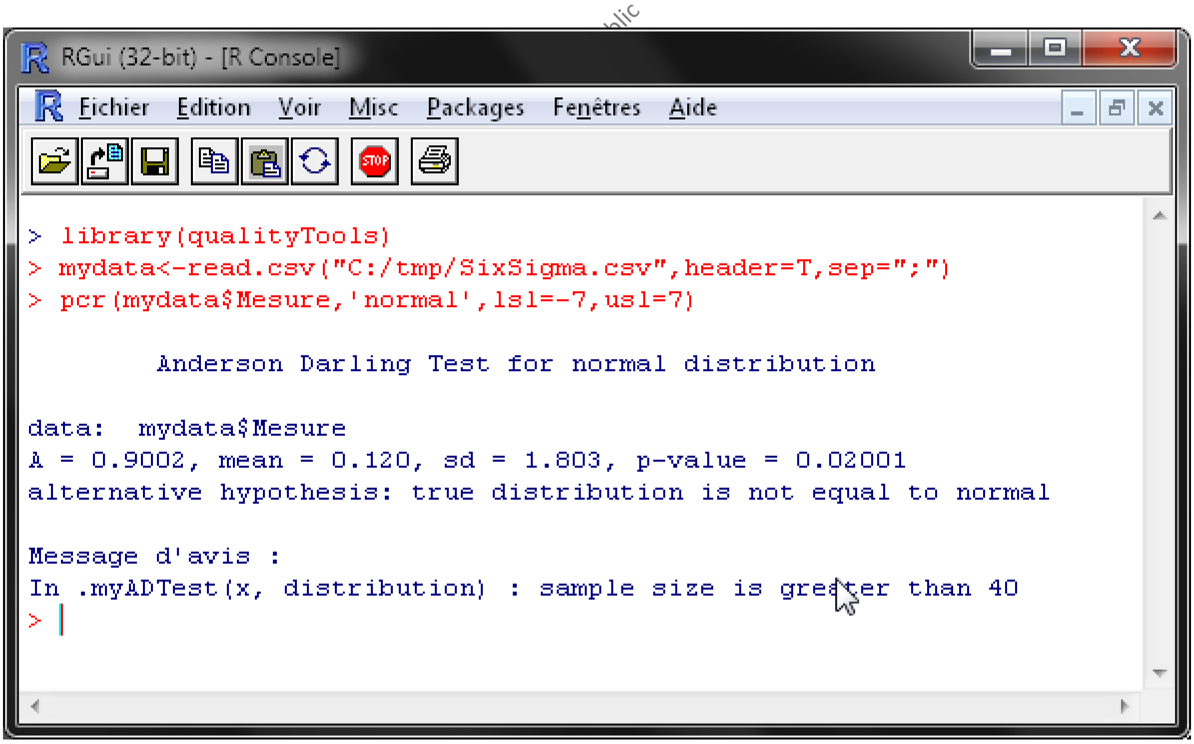


Figure 271 Graph d'analyse Six Pack

Maintenant, voyons ce que nous obtenons avec la fonction **pcr( )** du package **qualityTools**:



et avec le graphique et les indicateurs correspondants (dont nous voyons que ces derniers ne respectent pas la notation d'usage... d'où l'importance de suivre les normes pour ne pas se mélanger les pinceaux):

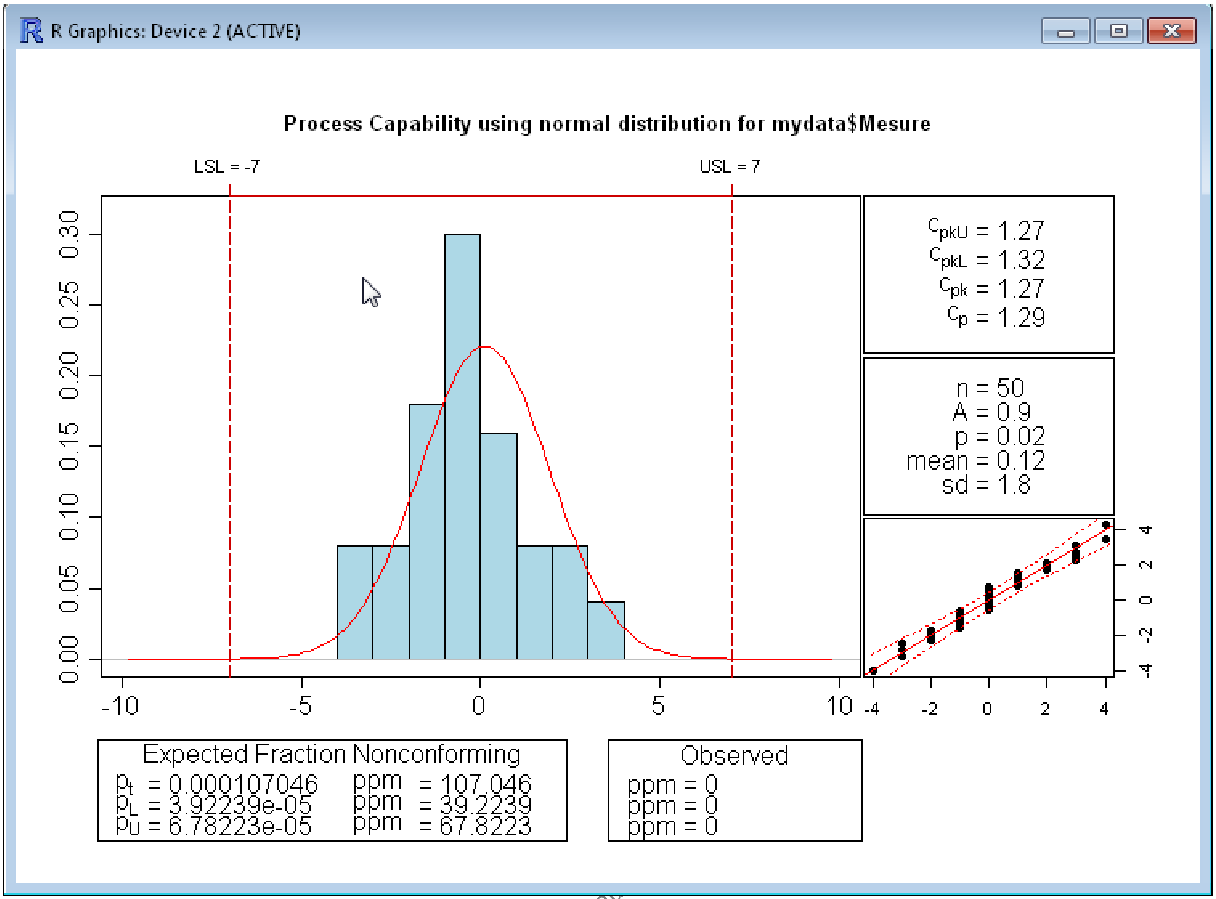


Figure 272 Graph d'Analyse de capabilité

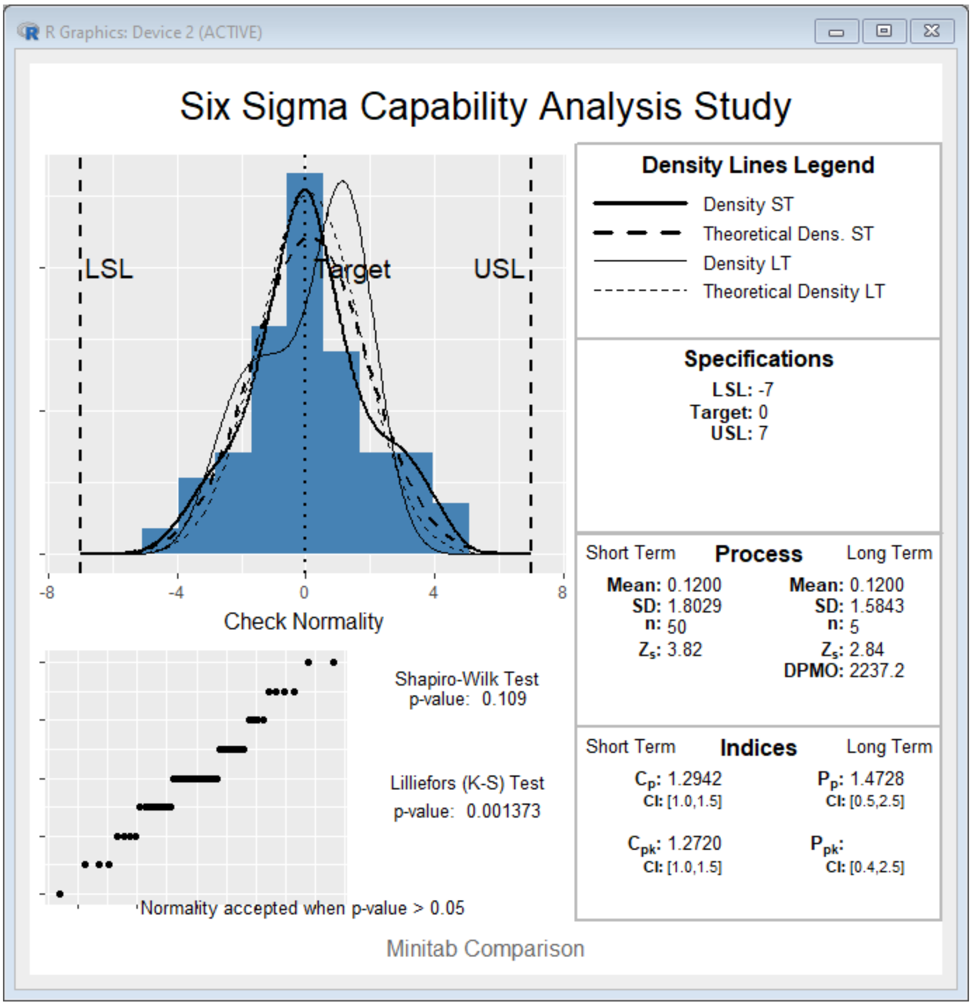
À comparer avec le graphique obtenu avec Minitab et les données calculées à la main dans le cours théorique !

Sinon avec le package **SixSigma** et la fonction **ss.study.ca( )** (**il semblerait – a priori… - que cette fonction ait un problème car il faut inverser les indicateurs courst et longs termes sur le graphique**[30]):

[30] Si un lecteur ou un de nos étudiants / apprenants sait comment corriger ce problème sans aller dans le code du package ou arrive à identifier notre erreur, qu'il n'hésite pas à se manifester!

```
> (unstacked<-unstack(mydata[,-1],form=Mesure~Temps))
  X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
1 -2 -4 -1  0  4  0  3  0  1  -1
2  0 -3  0 -2  1 -2  0  1 -1   2
3 -1  0 -3 -1  0  0 -1 -1  3   1
4  1  1 -2  2  2  0  1  0  4   0
5 -1 -1 -3  0  0  3  3  2  1   0
>
> moyenne_ST<-mean(mydata$Mesure)
> ecarttype_ST<-sqrt(sum(apply(unstacked, 2, var))/length(unstacked))
>
> rnorm_exact <- function(n,mean,sd) { mean+sd*scale(rnorm(n)) }
> r <- rnorm_exact(5,moyenne_ST,ecarttype_ST)
>
> ss.study.ca(xST=mydata$Mesure, xLT=r, LSL = -7, USL = +7,
+   Target = 0, alpha = 0.05, f.su = "Minitab Comparison")
>
```

Et le graphique correspondant:



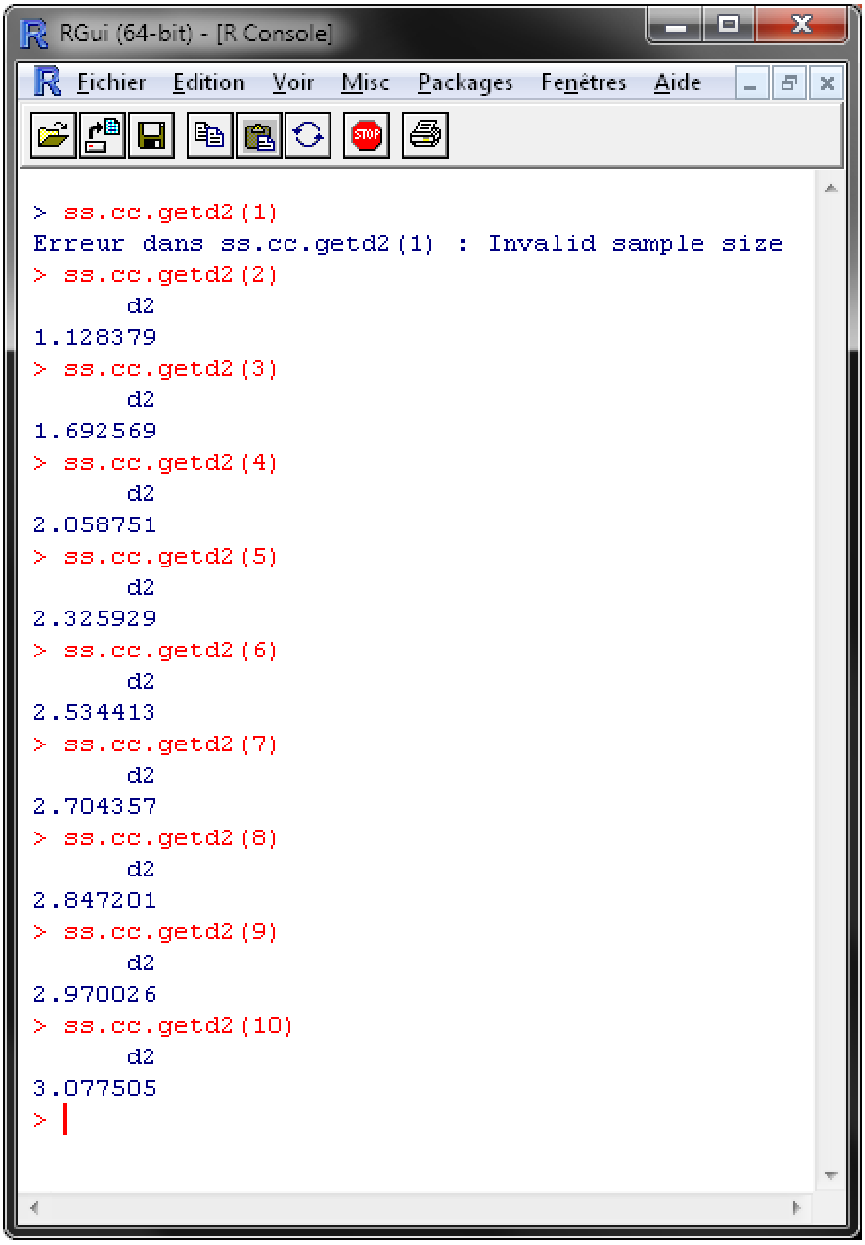
Outre la position a priori erronée des indicateurs, ce graphique nous donne des résultats qui sont beaucoup plus conformes à ce que nous avions obtenu avec Minitab et et les données calculées à la main dans le cours théorique !

Exemple 296.: Constantes de Hartley et autres constantes des cartes de contrôle

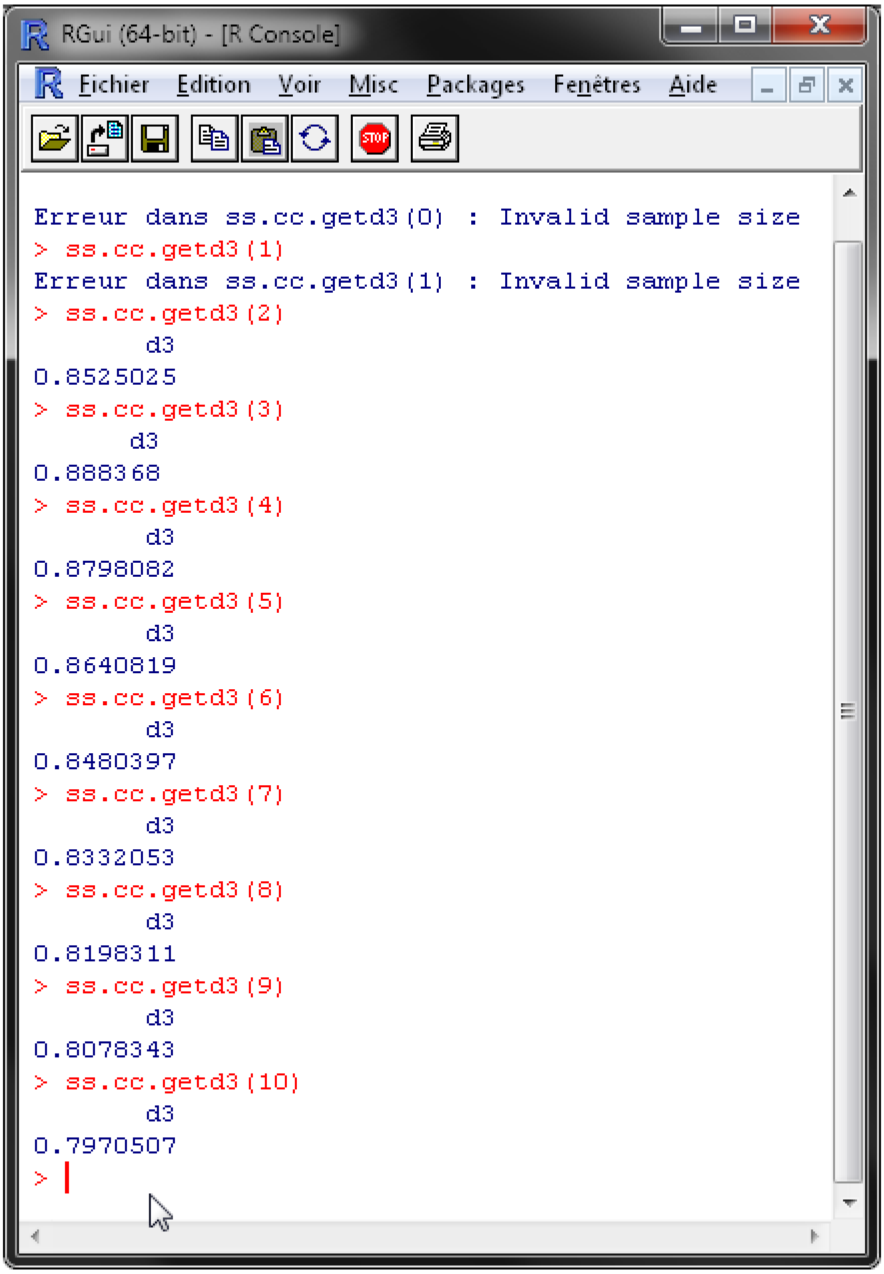
R 3.1.0

Lors de notre étude des statistiques de rangs nous avons démontré qu'une constante émergeait qui dépendant du nombre de rangs et qui nous était indispensable pour construite un certain nombre de cartes de contrôle.

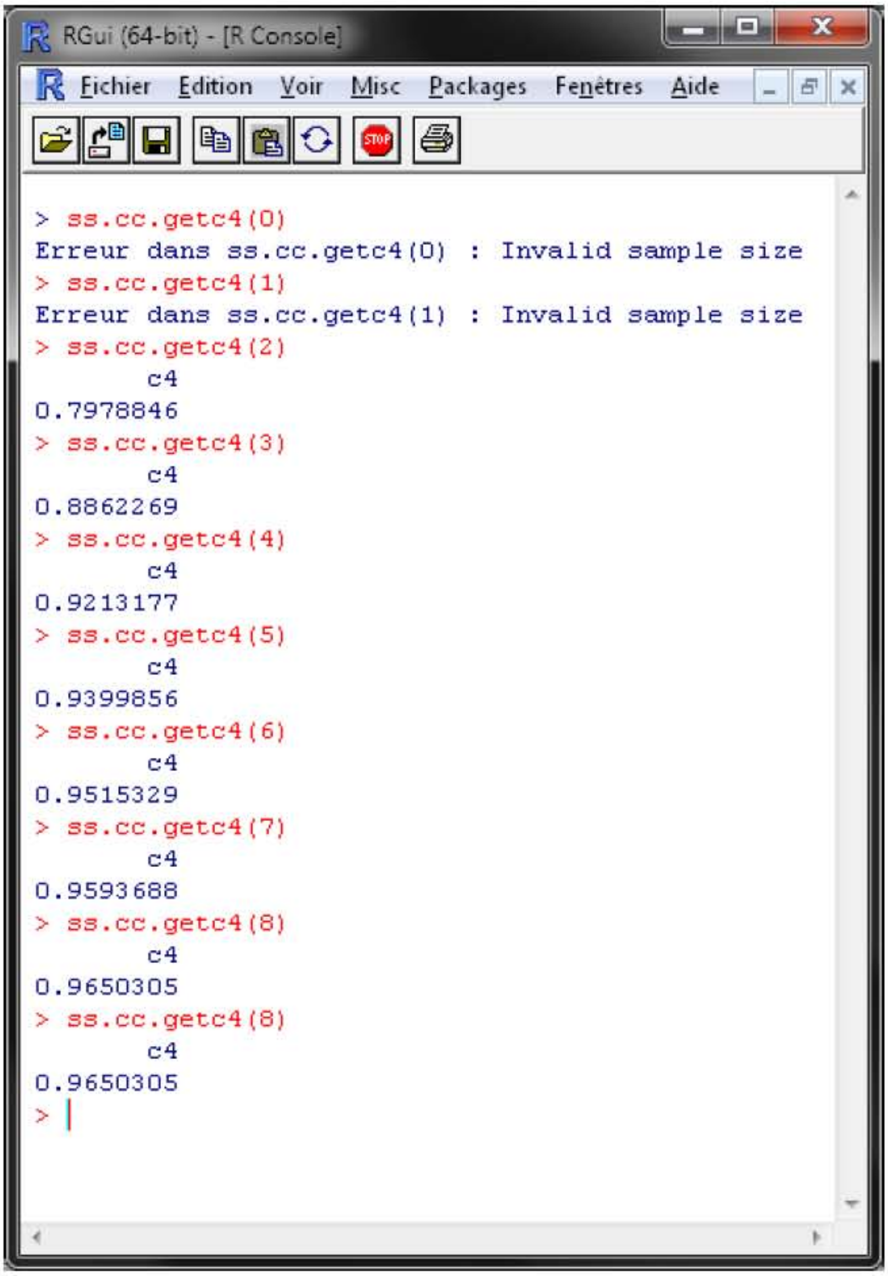
Plutôt que de faire référence à des tables, le package **SixSigma** contient une commande **ss.cc.getd2( )** qui nous permet de récuperer la valeur de la constante de Hartley pour un certain nombre de rangs (taille d'échantillon). Nous allons vérifier ici dans un premier temps que nous obtenons bien les valeurs obtenues dans le cours théorique pour la première constante de Harley d_2:



De même pour la deuxième constante de Harley d_3 avec la commande **ss.cc.getd3( )** :



De même pour la constante c_4 dont nous avons démontré l'origine dans le cours théorique de Génie Industriel:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> ss.cc.getc4(0)
Erreur dans ss.cc.getc4(0) : Invalid sample size
> ss.cc.getc4(1)
Erreur dans ss.cc.getc4(1) : Invalid sample size
> ss.cc.getc4(2)
c4
0.7978846
> ss.cc.getc4(3)
c4
0.8862269
> ss.cc.getc4(4)
c4
0.9213177
> ss.cc.getc4(5)
c4
0.9399856
> ss.cc.getc4(6)
c4
0.9515329
> ss.cc.getc4(7)
c4
0.9593688
> ss.cc.getc4(8)
c4
0.9650305
> ss.cc.getc4(8)
c4
0.9650305
>

Exemple 297.: Carte de contrôle par attributs p (proportion)

R 3.0.2

Dans le cadre de contrôle de lots nous avons obtenus les mesures suivantes:

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide

> library(qcc)
> mydata<-read.csv("C:/CCP.csv",header=T,sep=";")
> mydata
   IDLot TailleLot NonConformes
1      1        50            1
2      2        50            2
3      3        50            5
4      4        50            2
5      5        50            2
6      6        50            3
7      7        50            4
8      8        50            3
9      9        50            1
10    10        50            1
11    11        50            2
12    12        50            1
13    13        50            2
14    14        50            1
15    15        50            2
16    16        50            3
17    17        50            3
18    18        50            1
19    19        50            2
20    20        50            0
21    21        50            3
22    22        50            2
> 
```

Et nous utilisons encore une fois la commande **qcc( )** :

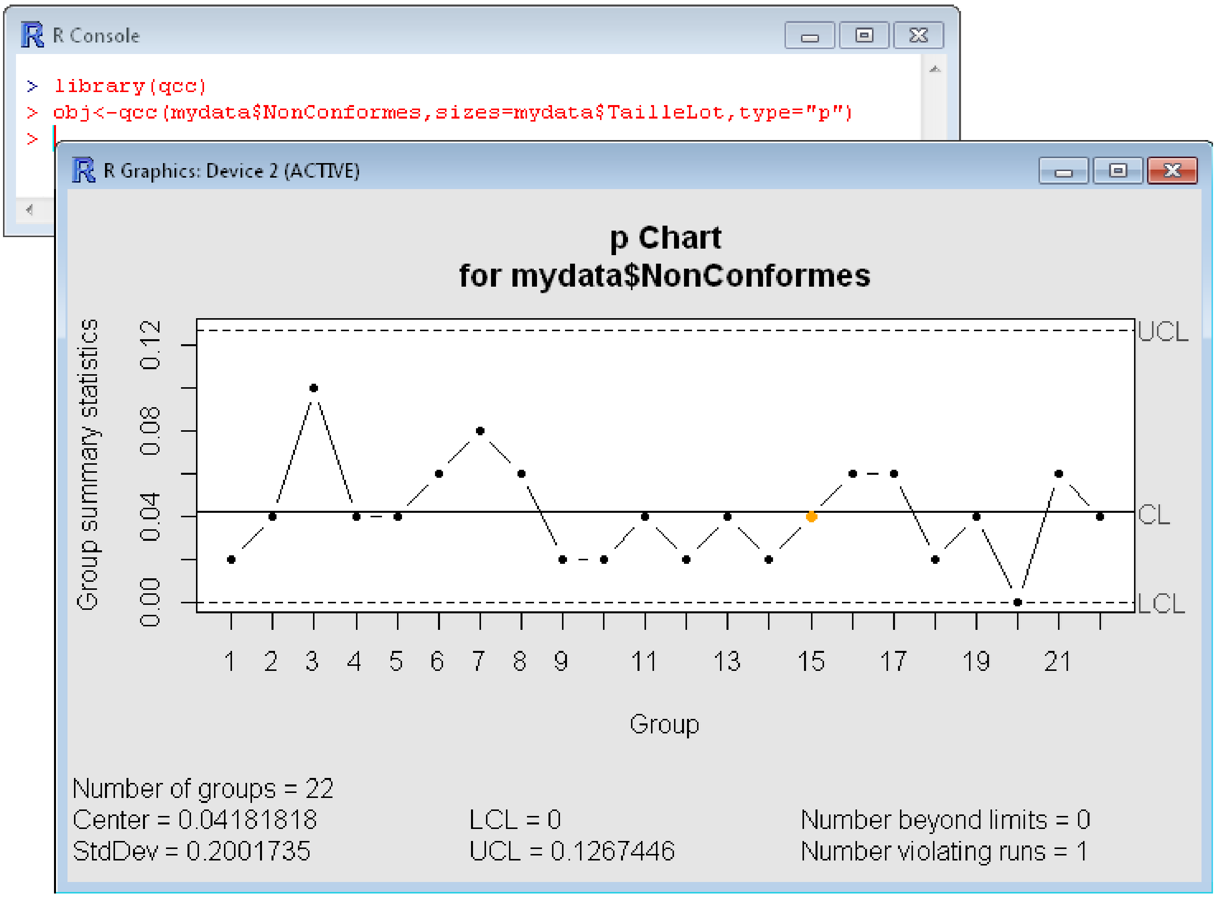


Figure 273 Graphique Carte de contrôle P

Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*. Sympa!!!

Exemple 298.: Carte de contrôle par attributs de type np (fréquences)

R 3.0.2

Dans le cadre de contrôle de lots nous avons obtenus les mesures suivantes:

```
> library(qcc)
> mydata<-read.csv("C:/CCNP.csv",header=T,sep=";")
> mydata
   IDLot TailleLot NonConformes
1      1        50            1
2      2        50            2
3      3        50            5
4      4        50            2
5      5        50            2
6      6        50            3
7      7        50            4
8      8        50            3
9      9        50            1
10    10        50            1
11    11        50            2
12    12        50            1
13    13        50            2
14    14        50            1
15    15        50            2
16    16        50            3
17    17        50            3
18    18        50            1
19    19        50            2
20    20        50            0
21    21        50            3
22    22        50            2
>
```

Et nous utilisons encore une fois la commande **qcc( )** :

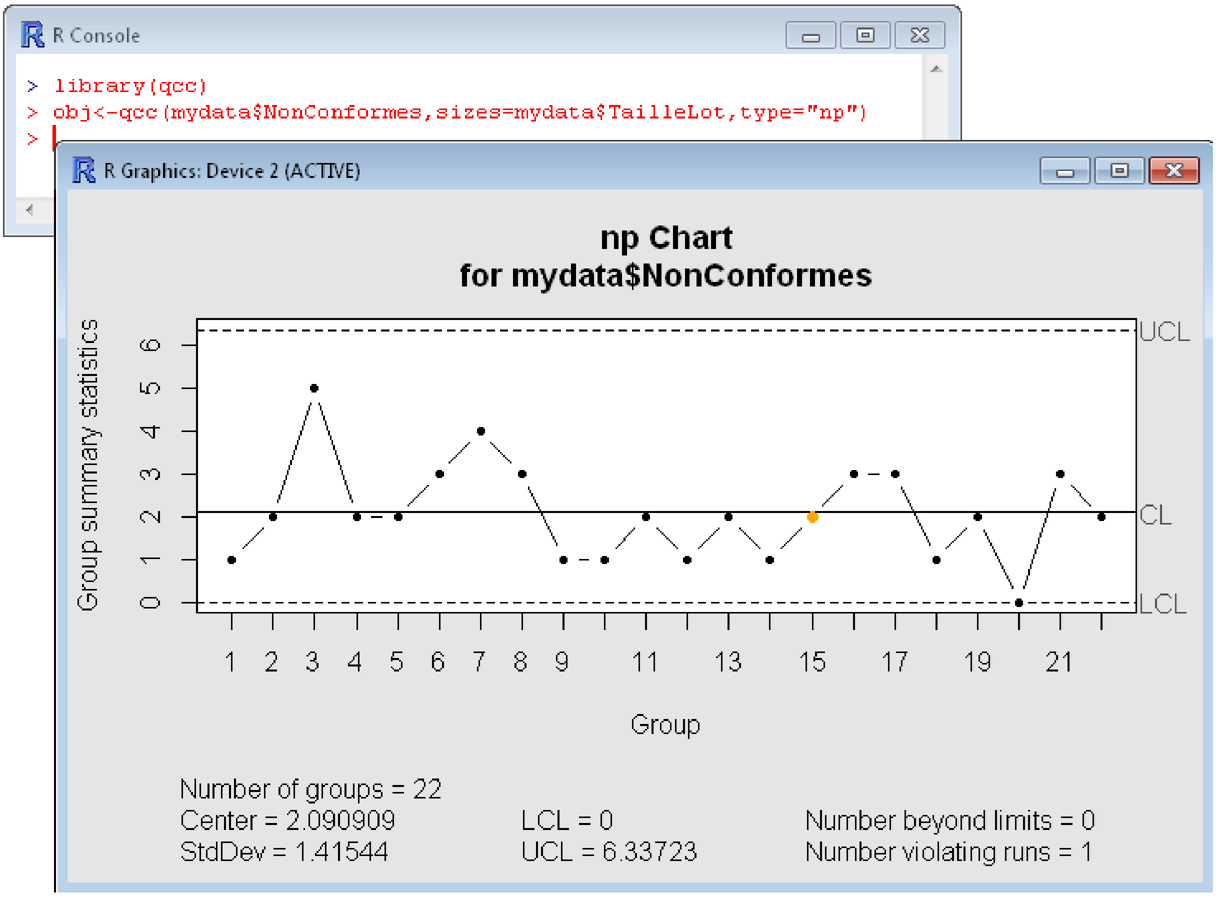


Figure 274 Graph Carte de contrôle NP

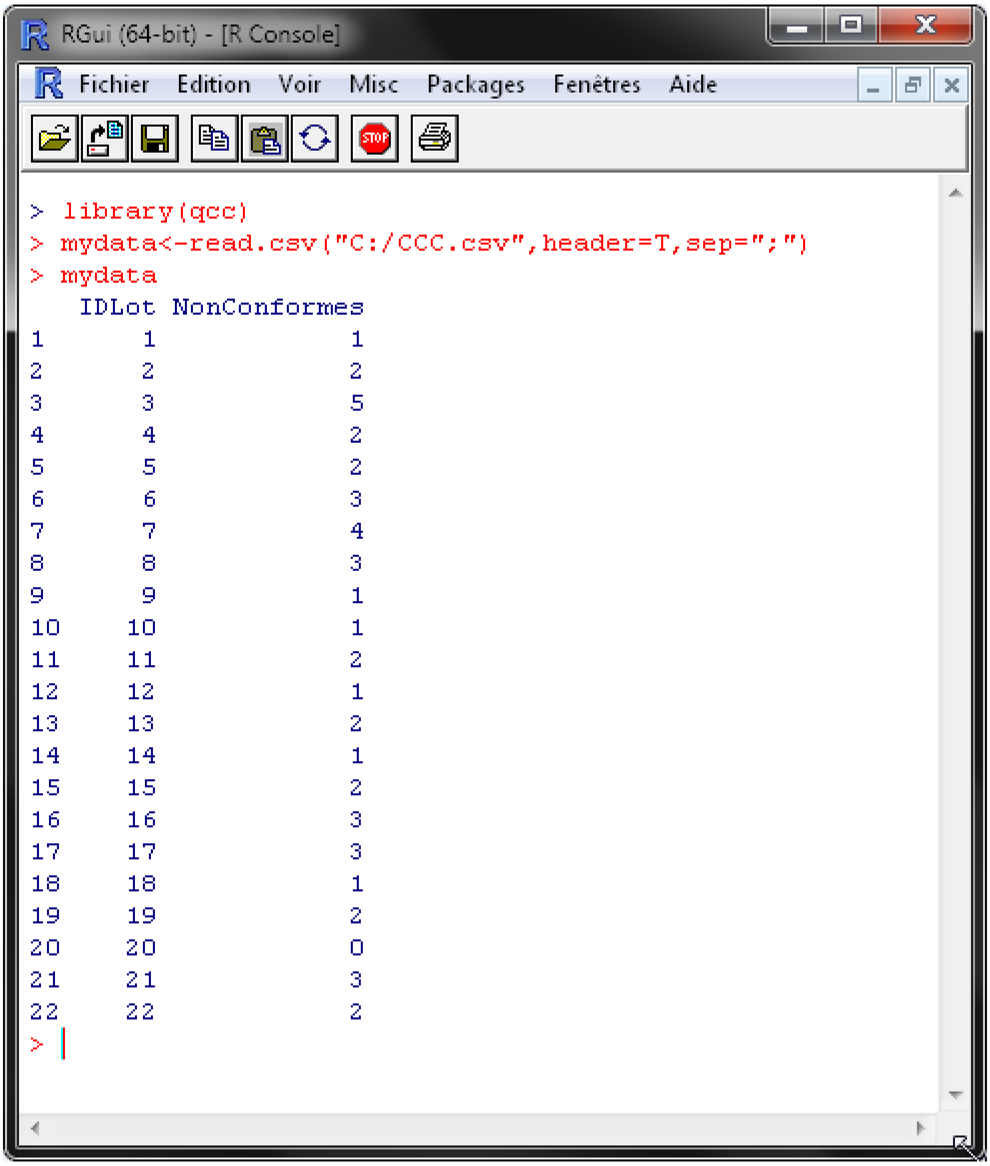
Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

Exemple 299.: Carte de contrôle par attributs de type c

R 3.0.2

Dans le cadre de contrôle de lots nous avons obtenu les mesures suivantes:



Et nous utilisons encore une fois la commande **qcc( )** :

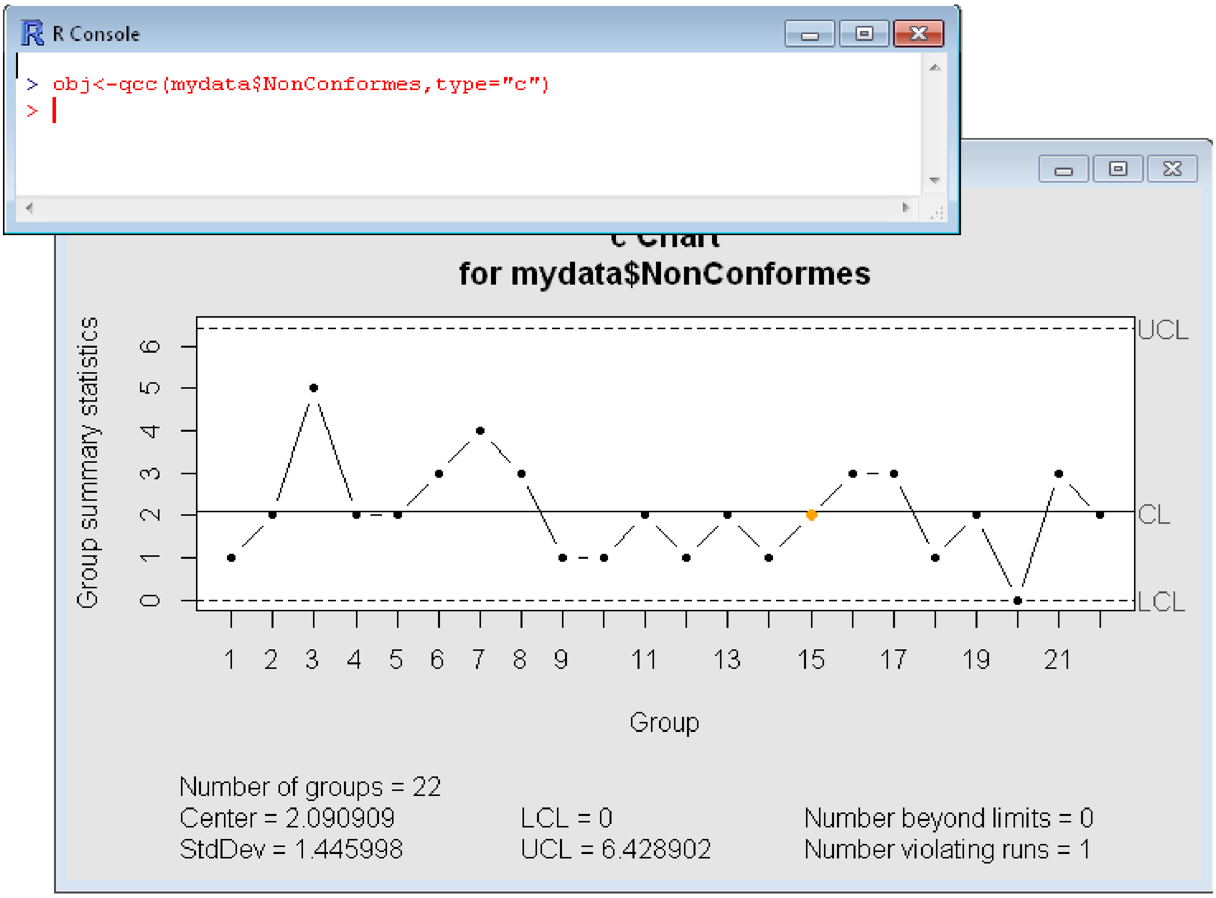


Figure 275 Graph Carte de contrôle C

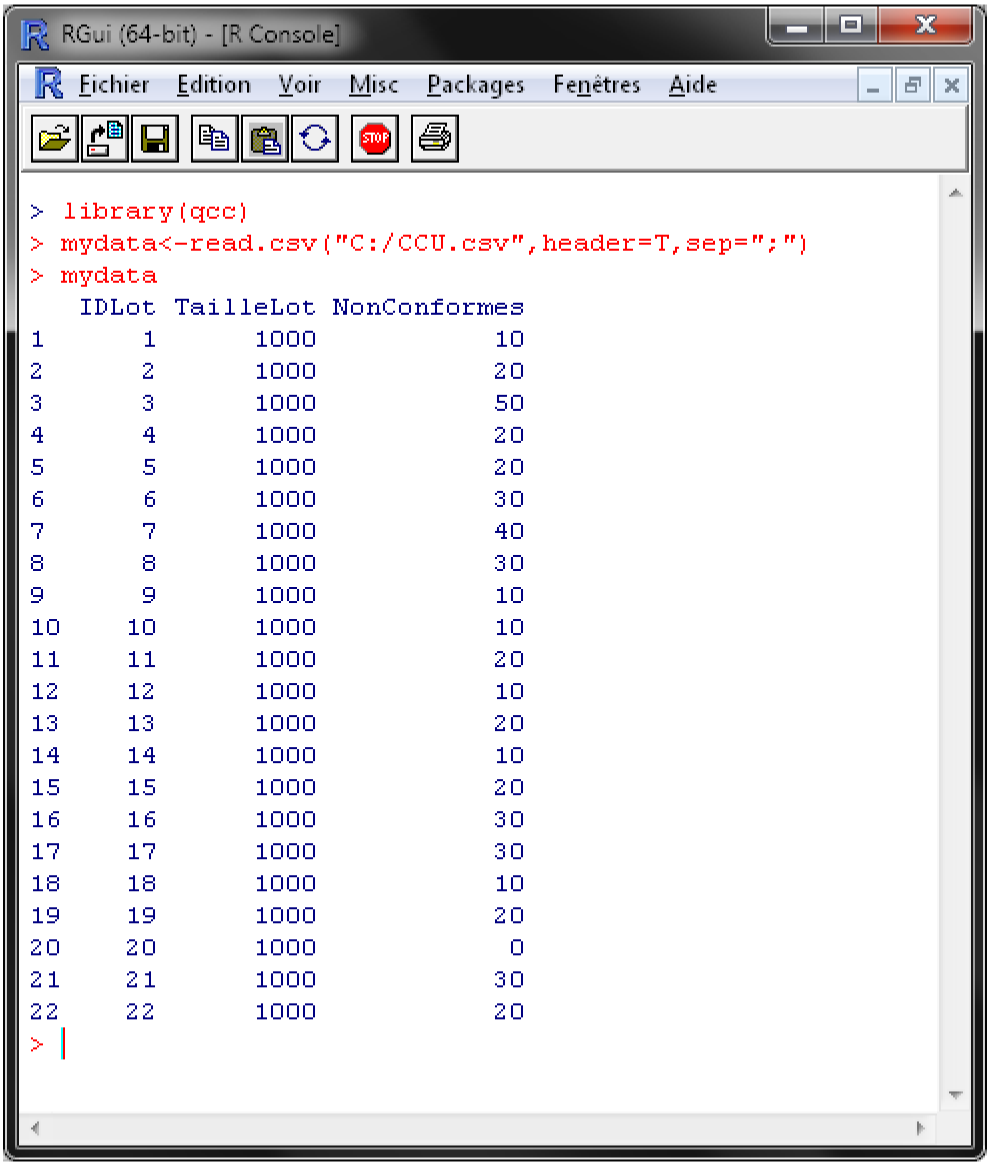
Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

Exemple 300.: Carte de contrôle par attributs de type u

R 3.0.2

Dans le cadre de contrôle de lots nous avons obtenu les mesures suivantes:



Et nous utilisons encore une fois la commande **qcc( )** :

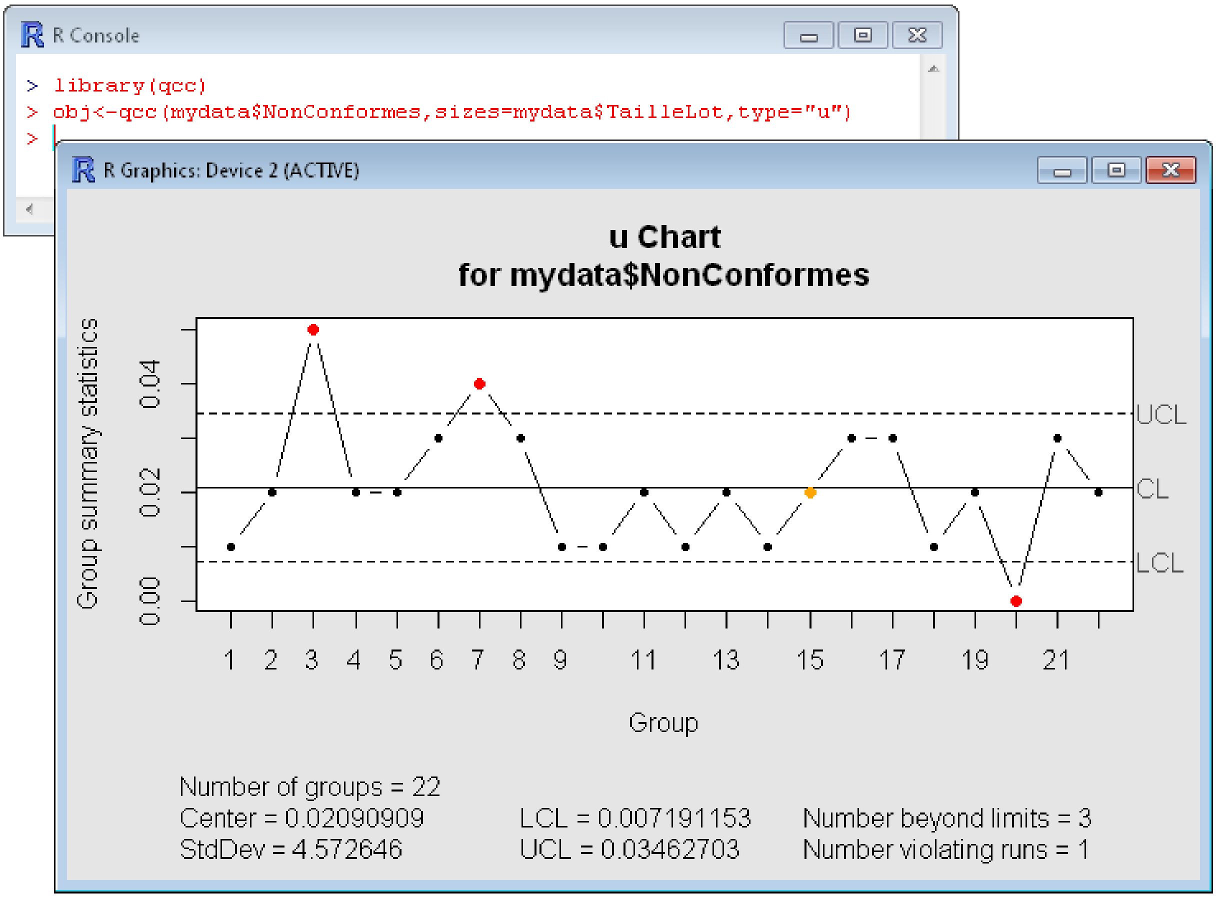


Figure 276 Graph Carte de contrôle U

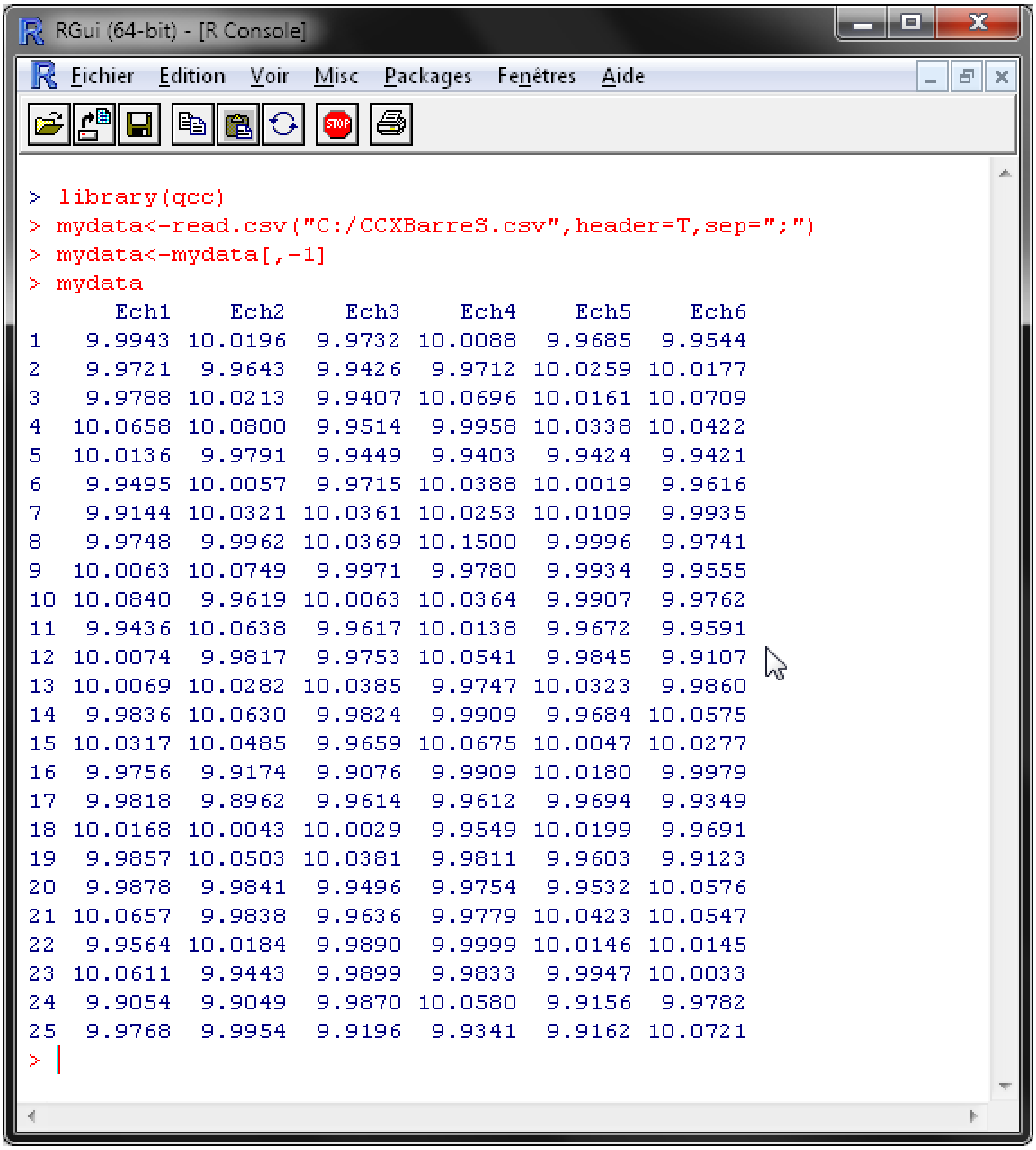
Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

Exemple 301.: Carte de contrôle par mesures X barre-S simple

R 3.0.2

Dans le cadre de contrôle d'une production sous contrôle statistique nous avons obtenu les mesures suivantes (chaque ligne est pour rappel une journée de mesures):




```
> library(qcc)
> mydata<-read.csv("C:/CCXBarreS.csv",header=T,sep=";")
> mydata<-mydata[,-1]
> mydata
      Ech1    Ech2    Ech3    Ech4    Ech5    Ech6
1   9.9943 10.0196  9.9732 10.0088  9.9685  9.9544
2   9.9721  9.9643  9.9426  9.9712 10.0259 10.0177
3   9.9788 10.0213  9.9407 10.0696 10.0161 10.0709
4  10.0658 10.0800  9.9514  9.9958 10.0338 10.0422
5  10.0136  9.9791  9.9449  9.9403  9.9424  9.9421
6   9.9495 10.0057  9.9715 10.0388 10.0019  9.9616
7   9.9144 10.0321 10.0361 10.0253 10.0109  9.9935
8   9.9748  9.9962 10.0369 10.1500  9.9996  9.9741
9  10.0063 10.0749  9.9971  9.9780  9.9934  9.9555
10 10.0840  9.9619 10.0063 10.0364  9.9907  9.9762
11  9.9436 10.0638  9.9617 10.0138  9.9672  9.9591
12 10.0074  9.9817  9.9753 10.0541  9.9845  9.9107
13 10.0069 10.0282 10.0385  9.9747 10.0323  9.9860
14  9.9836 10.0630  9.9824  9.9909  9.9684 10.0575
15 10.0317 10.0485  9.9659 10.0675 10.0047 10.0277
16  9.9756  9.9174  9.9076  9.9909 10.0180  9.9979
17  9.9818  9.8962  9.9614  9.9612  9.9694  9.9349
18 10.0168 10.0043 10.0029  9.9549 10.0199  9.9691
19  9.9857 10.0503 10.0381  9.9811  9.9603  9.9123
20  9.9878  9.9841  9.9496  9.9754  9.9532 10.0576
21 10.0657  9.9838  9.9636  9.9779 10.0423 10.0547
22  9.9564 10.0184  9.9890  9.9999 10.0146 10.0145
23 10.0611  9.9443  9.9899  9.9833  9.9947 10.0033
24  9.9054  9.9049  9.9870 10.0580  9.9156  9.9782
25  9.9768  9.9954  9.9196  9.9341  9.9162 10.0721
>
```

Et nous utilisons encore une fois la commande **qcc( )** :

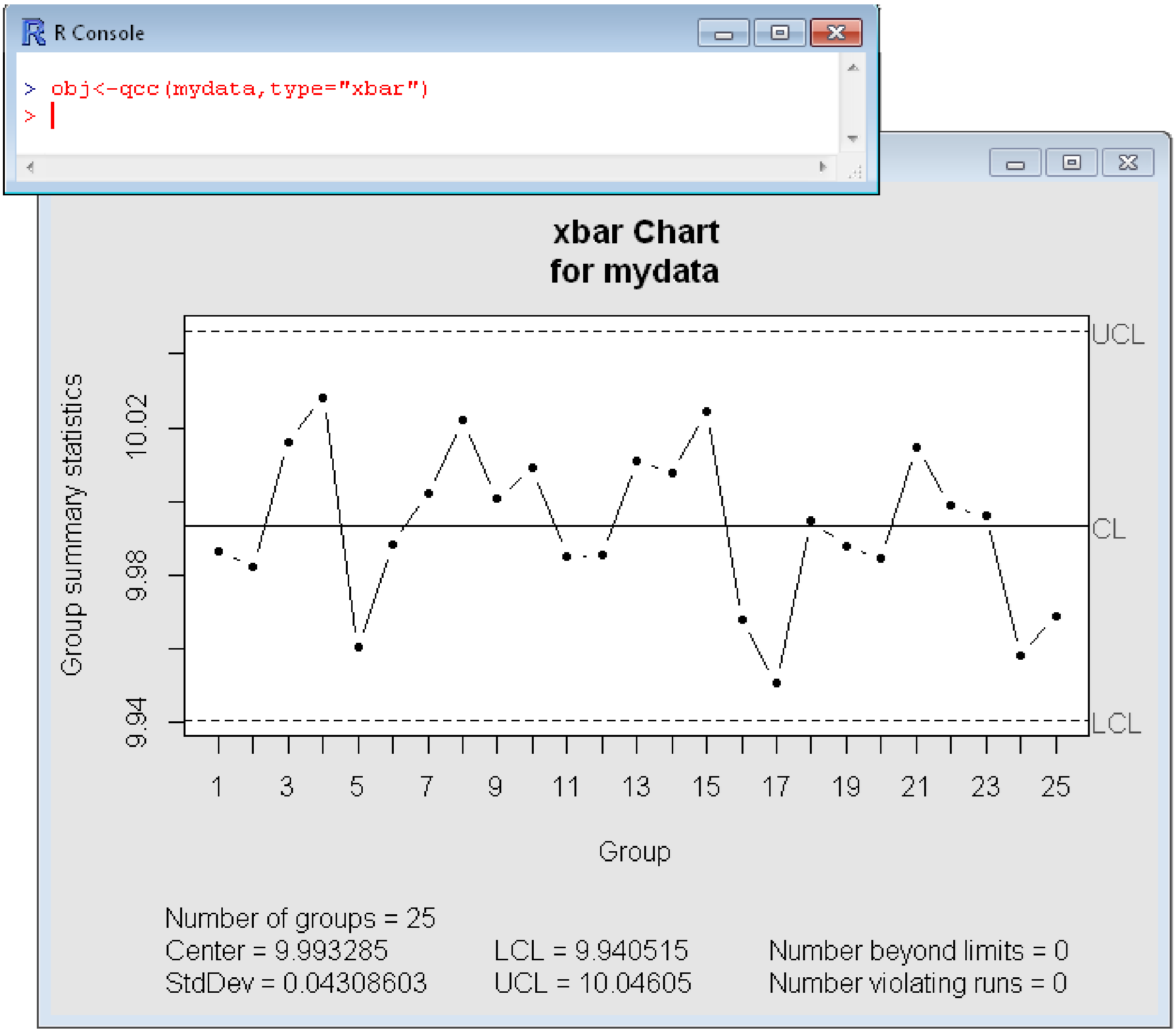


Figure 277 Carte de contrôle Xbar-S

Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

Courbe d'efficacité

Nous avons vu dans le cours théorique comment calculer l'erreur de type II d'une carte de contrôle de type X-barre. Le but ici est de vérifier que nous retrouvons les mêmes résultats sous les mêmes hypothèses.

Pour cela, nous utilisons la fonction **oc.curves( )** du package **qcc**:



et nous obtenons alors:

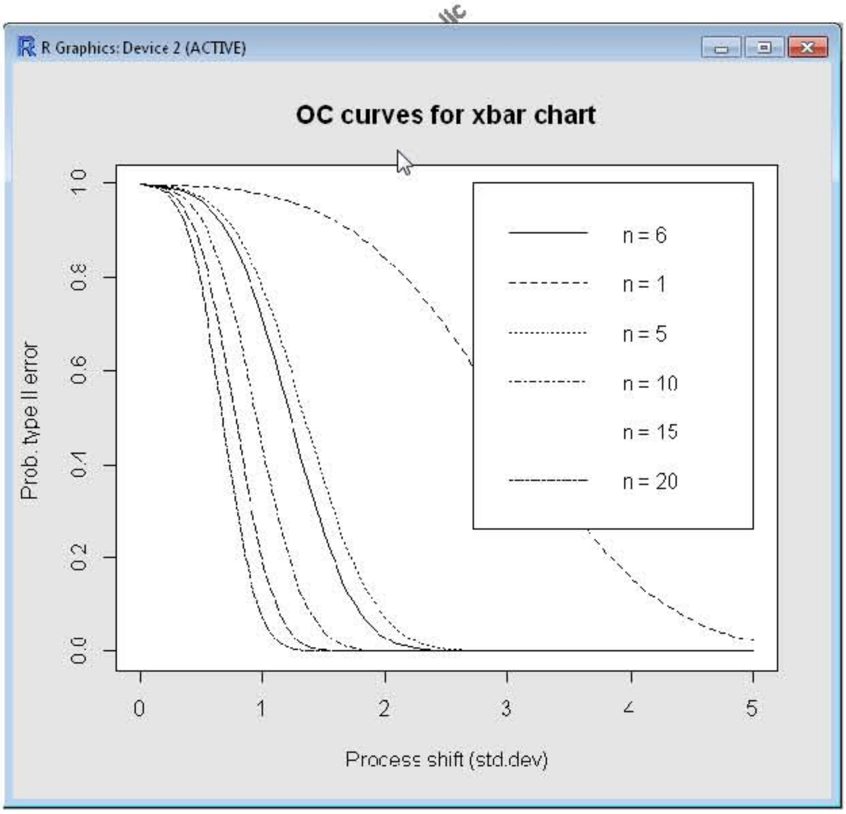


Figure 278 Graph de courbe de caractéristique opérationnelle (OC)

Ce qui correspond parfaitement aux calculs faits à la main.

Exemple 302.: Carte de contrôle par mesures S barre-S simple

R 3.0.2

Dans le cadre de contrôle d'une production sous contrôle statistique nous avons obtenu les mesures suivantes (chaque ligne est pour rappel une journée de mesures):

```
> library(qcc)
> mydata<-read.csv("C:/CCSBarreS.csv",header=T,sep=";")
> mydata<-mydata[,-1]
> mydata
      Ech1    Ech2    Ech3    Ech4    Ech5    Ech6
1   9.9943 10.0196  9.9732 10.0088  9.9685  9.9544
2   9.9721  9.9643  9.9426  9.9712 10.0259 10.0177
3   9.9788 10.0213  9.9407 10.0696 10.0161 10.0709
4  10.0658 10.0800  9.9514  9.9958 10.0338 10.0422
5  10.0136  9.9791  9.9449  9.9403  9.9424  9.9421
6   9.9495 10.0057  9.9715 10.0388 10.0019  9.9616
7   9.9144 10.0321 10.0361 10.0253 10.0109  9.9935
8   9.9748  9.9962 10.0369 10.1500  9.9996  9.9741
9  10.0063 10.0749  9.9971  9.9780  9.9934  9.9555
10 10.0840  9.9619 10.0063 10.0364  9.9907  9.9762
11  9.9436 10.0638  9.9617 10.0138  9.9672  9.9591
12 10.0074  9.9817  9.9753 10.0541  9.9845  9.9107
13 10.0069 10.0282 10.0385  9.9747 10.0323  9.9860
14  9.9836 10.0630  9.9824  9.9909  9.9684 10.0575
15 10.0317 10.0485  9.9659 10.0675 10.0047 10.0277
16  9.9756  9.9174  9.9076  9.9909 10.0180  9.9979
17  9.9818  9.8962  9.9614  9.9612  9.9694  9.9349
18 10.0168 10.0043 10.0029  9.9549 10.0199  9.9691
19  9.9857 10.0503 10.0381  9.9811  9.9603  9.9123
20  9.9878  9.9841  9.9496  9.9754  9.9532 10.0576
21 10.0657  9.9838  9.9636  9.9779 10.0423 10.0547
22  9.9564 10.0184  9.9890  9.9999 10.0146 10.0145
23 10.0611  9.9443  9.9899  9.9833  9.9947 10.0033
24  9.9054  9.9049  9.9870 10.0580  9.9156  9.9782
25  9.9768  9.9954  9.9196  9.9341  9.9162 10.0721
>
```

Et nous utilisons encore une fois la commande **qcc( )** :

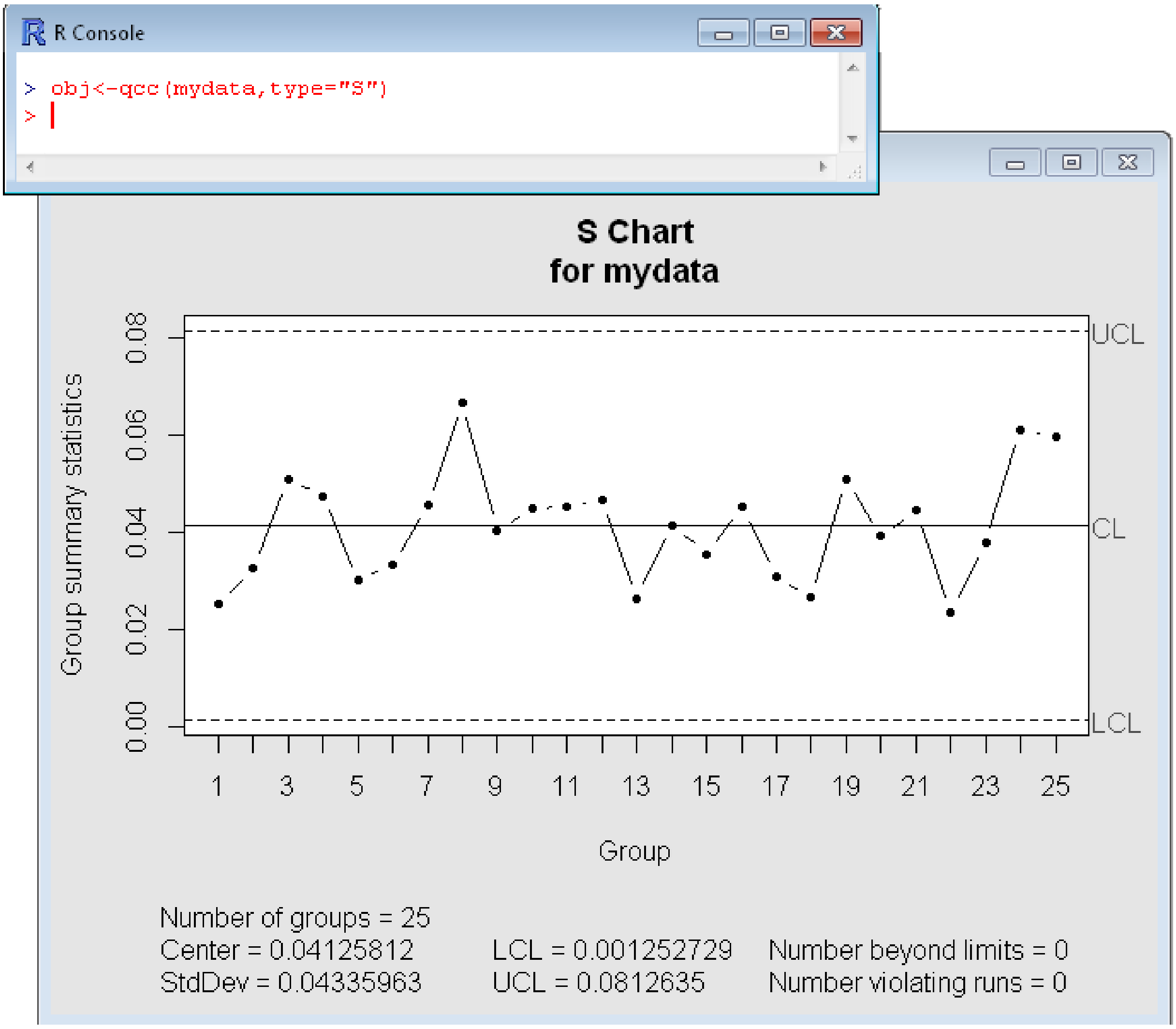


Figure 279 Carte de contrôle Sbar-S

Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

Malheureusement il n'est pas possible à ma connaissance de combiner dans une seule fenêtre la CC X barre-S et la CC S barre-S et ce même avec la commande par(mfrow=c(2,2)) contrairement à Minitab...

Il ne semble pas possible non plus, contrairement à Minitab, d'avoir une abscisse basée sur des dates (timbres).

Exemple 303.: Carte de contrôle par mesures R barre-R simple

R 3.0.2

Dans le cadre de contrôle d'une production sous contrôle statistique nous avons obtenu les mesures suivantes (chaque ligne est pour rappel une journée de mesures):

```
> library(qcc)
> mydata<-read.csv("C:/CCRBarreR.csv",header=T,sep=";")
> mydata<-mydata[,-1]
> mydata
      Ech1    Ech2    Ech3    Ech4    Ech5    Ech6
1   9.9943 10.0196  9.9732 10.0088  9.9685  9.9544
2   9.9721  9.9643  9.9426  9.9712 10.0259 10.0177
3   9.9788 10.0213  9.9407 10.0696 10.0161 10.0709
4  10.0658 10.0800  9.9514  9.9958 10.0338 10.0422
5  10.0136  9.9791  9.9449  9.9403  9.9424  9.9421
6   9.9495 10.0057  9.9715 10.0388 10.0019  9.9616
7   9.9144 10.0321 10.0361 10.0253 10.0109  9.9935
8   9.9748  9.9962 10.0369 10.1500  9.9996  9.9741
9  10.0063 10.0749  9.9971  9.9780  9.9934  9.9555
10 10.0840  9.9619 10.0063 10.0364  9.9907  9.9762
11  9.9436 10.0638  9.9617 10.0138  9.9672  9.9591
12 10.0074  9.9817  9.9753 10.0541  9.9845  9.9107
13 10.0069 10.0282 10.0385  9.9747 10.0323  9.9860
14  9.9836 10.0630  9.9824  9.9909  9.9684 10.0575
15 10.0317 10.0485  9.9659 10.0675 10.0047 10.0277
16  9.9756  9.9174  9.9076  9.9909 10.0180  9.9979
17  9.9818  9.8962  9.9614  9.9612  9.9694  9.9349
18 10.0168 10.0043 10.0029  9.9549 10.0199  9.9691
19  9.9857 10.0503 10.0381  9.9811  9.9603  9.9123
20  9.9878  9.9841  9.9496  9.9754  9.9532 10.0576
21 10.0657  9.9838  9.9636  9.9779 10.0423 10.0547
22  9.9564 10.0184  9.9890  9.9999 10.0146 10.0145
23 10.0611  9.9443  9.9899  9.9833  9.9947 10.0033
24  9.9054  9.9049  9.9870 10.0580  9.9156  9.9782
25  9.9768  9.9954  9.9196  9.9341  9.9162 10.0721
>
```

Et nous utilisons encore une fois la commande **qcc( )** :

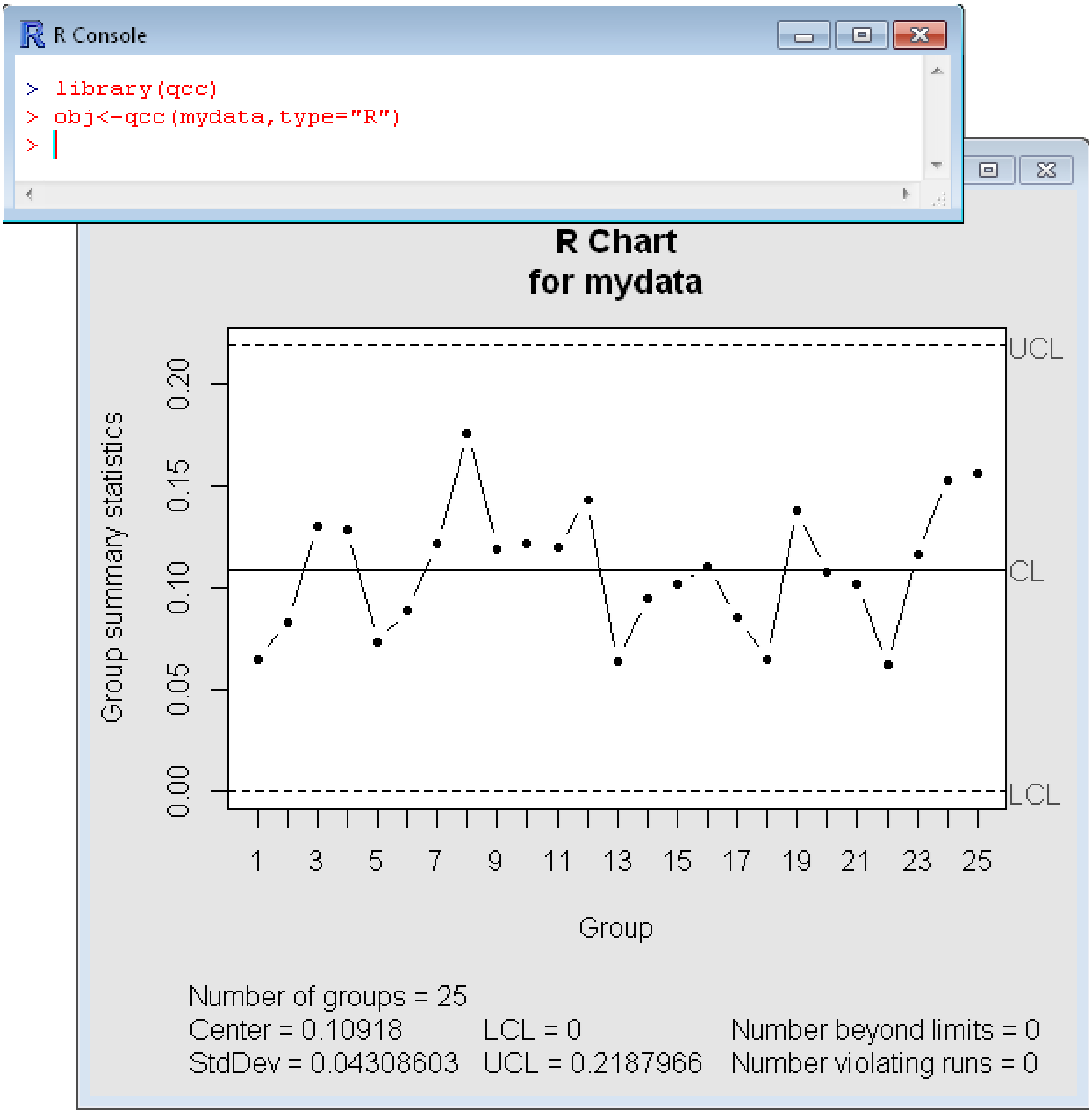


Figure 280 Carte de contrôle Rbar-R (simple)

Nous obtenons donc les mêmes valeurs et la même carte que dans le cours de statistique théorique et Microsoft Excel ainsi que Minitab.

Cependant avec un plus: R détecte les anomalies de séquences sous la dénomination *Number violating runs*.

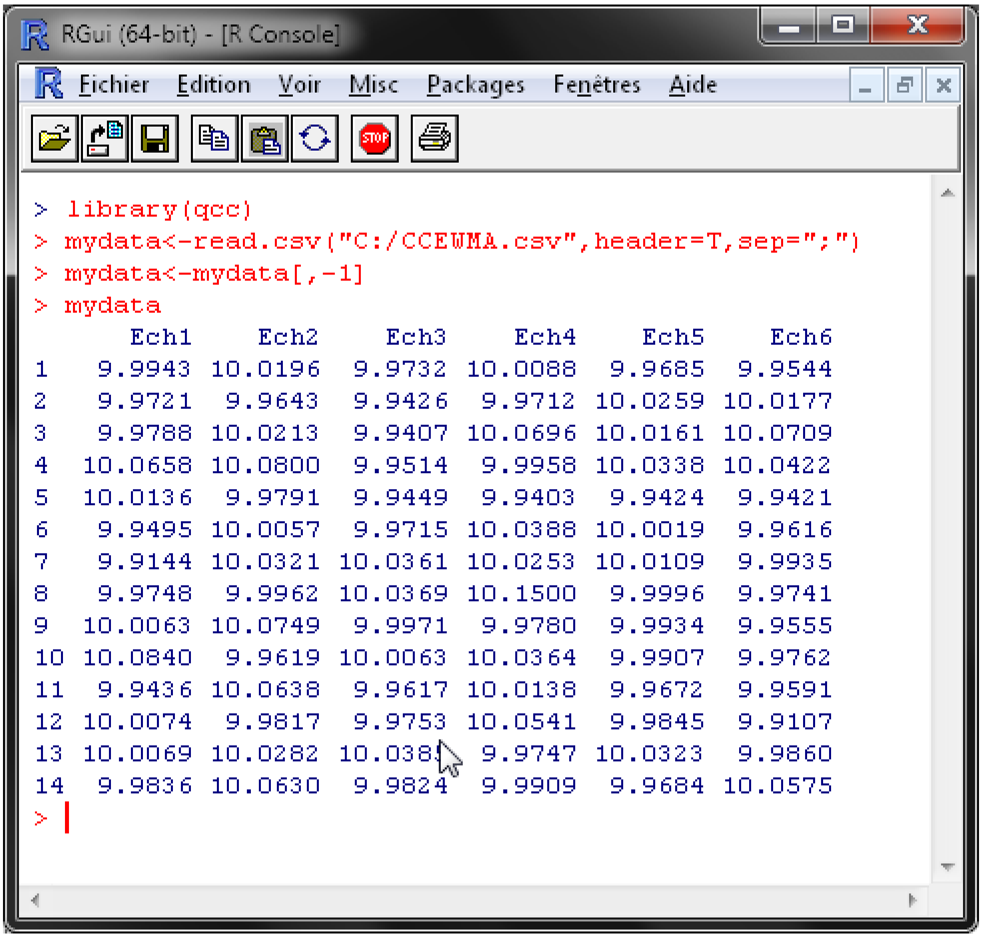
Malheureusement il n'est pas possible à ma connaissance de combiner dans une seule fenêtre la CC X barre-R (n'existe pas à ce jour) et la CC R barre-R et ce même avec la commande par(mfrow=c(2,2)) contrairement à Minitab...

Il ne semble pas possible non plus, contrairement à Minitab, d'avoir une abscisse basée sur des dates (timbres).

Exemple 304.: Carte de contrôle EWMA avec échantillons

R 3.0.2

Dans le cadre de contrôle d'une production sous contrôle statistique nous avons obtenu les mesures suivantes (chaque ligne est pour rappel une journée de mesures):



```
> library(qcc)
> mydata<-read.csv("C:/CCEWMA.csv",header=T,sep=";")
> mydata<-mydata[,-1]
> mydata
      Ech1    Ech2    Ech3    Ech4    Ech5    Ech6
1   9.9943 10.0196  9.9732 10.0088  9.9685  9.9544
2   9.9721  9.9643  9.9426  9.9712 10.0259 10.0177
3   9.9788 10.0213  9.9407 10.0696 10.0161 10.0709
4  10.0658 10.0800  9.9514  9.9958 10.0338 10.0422
5  10.0136  9.9791  9.9449  9.9403  9.9424  9.9421
6   9.9495 10.0057  9.9715 10.0388 10.0019  9.9616
7   9.9144 10.0321 10.0361 10.0253 10.0109  9.9935
8   9.9748  9.9962 10.0369 10.1500  9.9996  9.9741
9  10.0063 10.0749  9.9971  9.9780  9.9934  9.9555
10 10.0840  9.9619 10.0063 10.0364  9.9907  9.9762
11  9.9436 10.0638  9.9617 10.0138  9.9672  9.9591
12 10.0074  9.9817  9.9753 10.0541  9.9845  9.9107
13 10.0069 10.0282 10.038   9.9747 10.0323  9.9860
14  9.9836 10.0630  9.9824  9.9909  9.9684 10.0575
>
```

Et nous utilisons encore une fois la commande **ewma( )** :

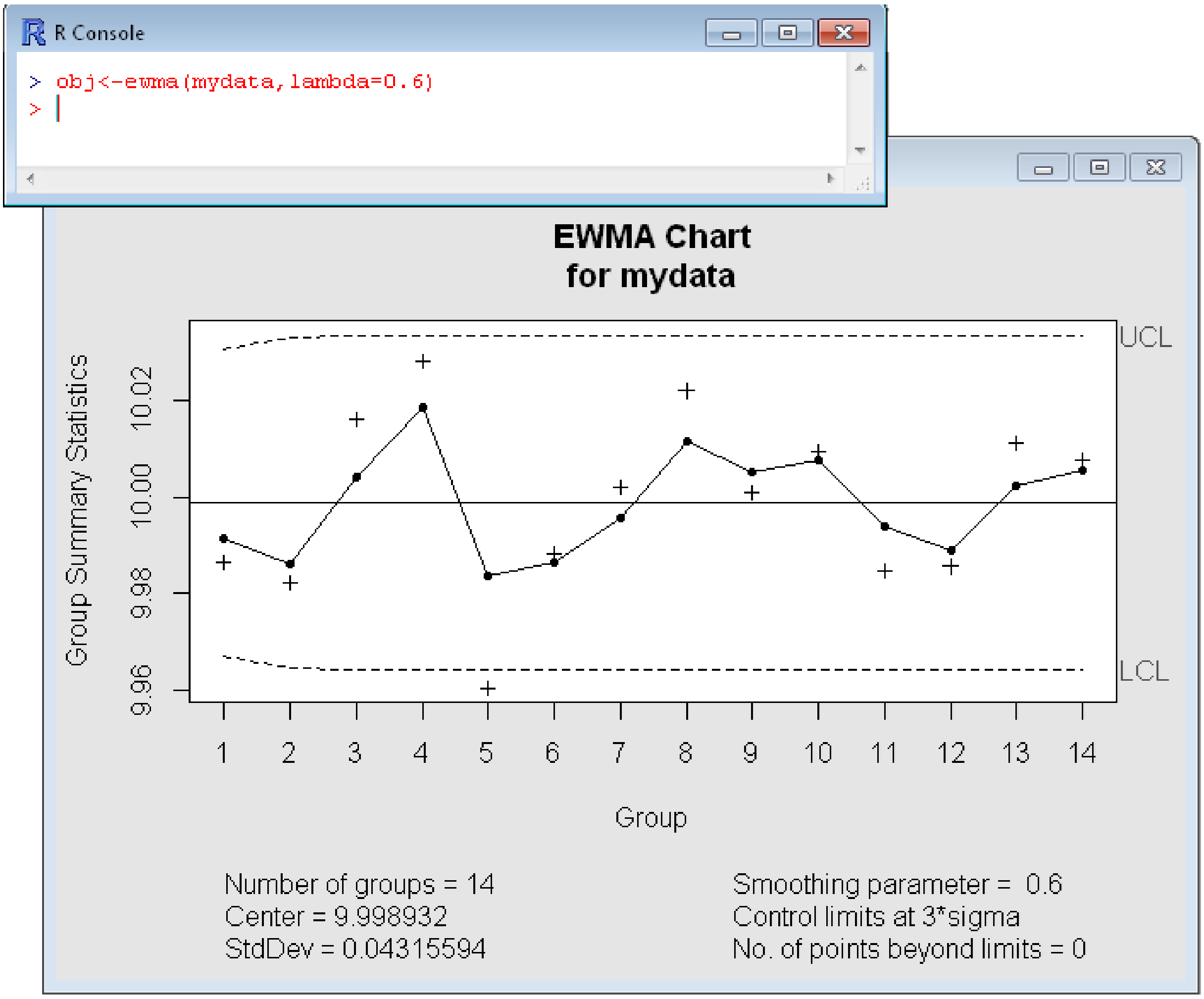


Figure 281 Carte de contrôle EWMAd

Nous retrouvons donc quasiment les mêmes valeurs que dans le cours théorique de statistiques et que le cours Microsoft Excel et Minitab. Effectivement, nous y avions fait une approximation en ce qui concerne les limites de contrôle (indépendantes du temps *t*) et la valeur du point de départ d'où la petite différence.

Exemple 305.: Carte de contrôle CUSUM

R 3.0.2

Dans le cadre de contrôle d'une production sous contrôle statistique nous avons obtenu les mesures suivantes (chaque ligne est pour rappel une journée de mesures):

```
> library(qcc)
> mydata<-read.csv("C:/CCCusum.csv",header=T,sep=";")
> mydata<-mydata[,-1]
> mydata
      Ech1    Ech2    Ech3    Ech4    Ech5    Ech6
1   9.9943 10.0196  9.9732 10.0088  9.9685  9.9544
2   9.9721  9.9643  9.9426  9.9712 10.0259 10.0177
3   9.9788 10.0213  9.9407 10.0696 10.0161 10.0709
4  10.0658 10.0800  9.9514  9.9958 10.0338 10.0422
5  10.0136  9.9791  9.9449  9.9403  9.9424  9.9421
6   9.9495 10.0057  9.9715 10.0388 10.0019  9.9616
7   9.9144 10.0321 10.0361 10.0253 10.0109  9.9935
8   9.9748  9.9962 10.0369 10.1500  9.9996  9.9741
9  10.0063 10.0749  9.9971  9.9780  9.9934  9.9555
10 10.0840  9.9619 10.0063 10.0364  9.9907  9.9762
11  9.9436 10.0638  9.9617 10.0138  9.9672  9.9591
12 10.0074  9.9817  9.9753 10.0541  9.9845  9.9107
13 10.0069 10.0282 10.0385  9.9747 10.0323  9.9860
14  9.9836 10.0630  9.9824  9.9909  9.9684 10.0575
15 10.0317 10.0485  9.9659 10.0675 10.0047 10.0277
16  9.9756  9.9174  9.9076  9.9909 10.0180  9.9979
17  9.9818  9.8962  9.9614  9.9612  9.9694  9.9349
18 10.0168 10.0043 10.0029  9.9549 10.0199  9.9691
19  9.9857 10.0503 10.0381  9.9811  9.9603  9.9123
20  9.9878  9.9841  9.9496  9.9754  9.9532 10.0576
21 10.0657  9.9838  9.9636  9.9779 10.0423 10.0547
22  9.9564 10.0184  9.9890  9.9999 10.0146 10.0145
23 10.0611  9.9443  9.9899  9.9833  9.9947 10.0033
24  9.9054  9.9049  9.9870 10.0580  9.9156  9.9782
25  9.9768  9.9954  9.9196  9.9341  9.9162 10.0721
>
```

Et nous utilisons encore une fois la commande **cusum( )** :

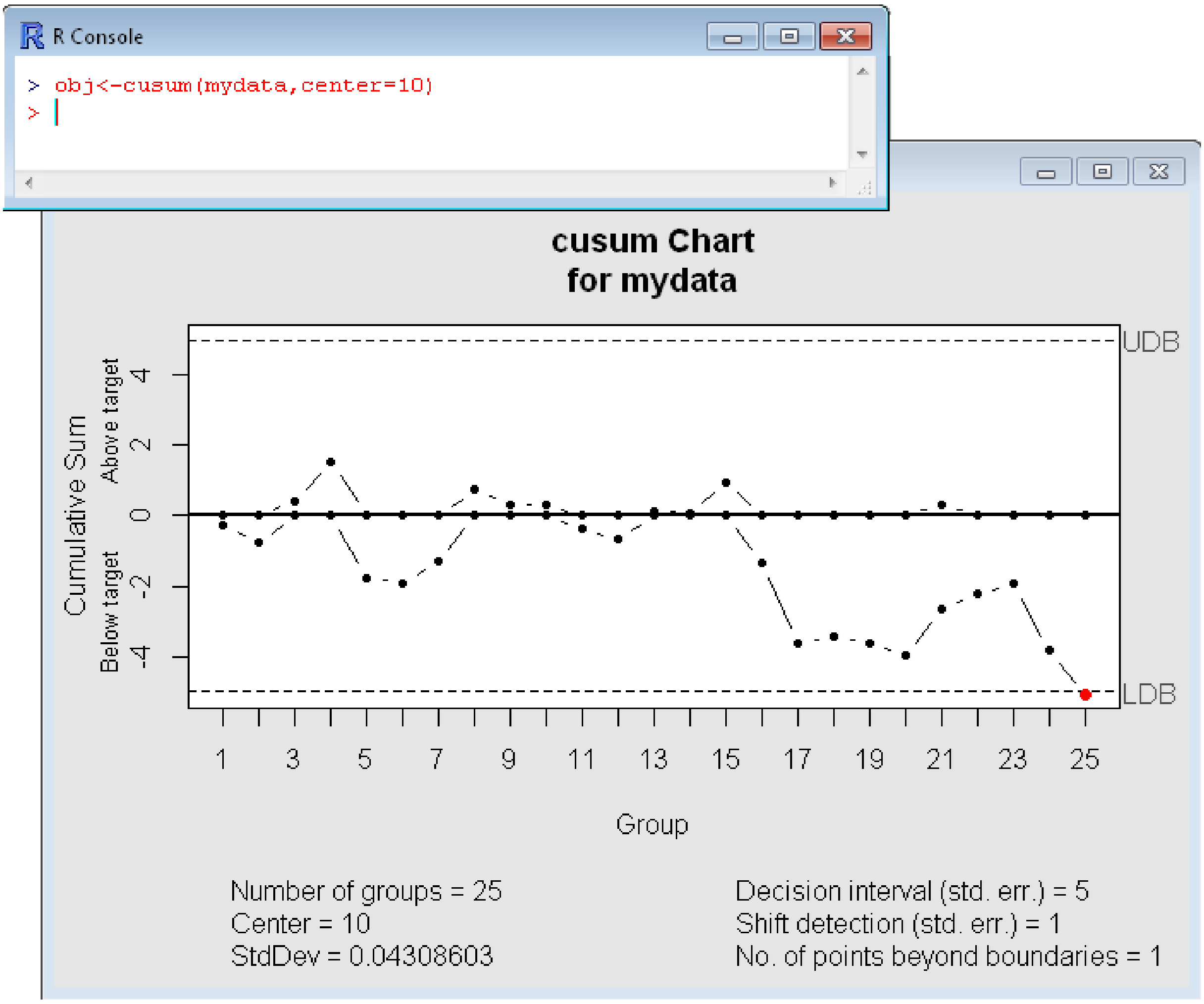


Figure 282 Graph Carte de contrôle CUSUM

Nous voyons alors plusieurs curiosités. Bien que le forme de la carte CUSUM soit conforme, impossible de définir a priori un V-masque et il semblerait qu'une deuxième ligne de mesures est représentée dans la carte de contrôle (Ligne d'origine inconnue!). Les limites ne sont pas non plus conformes à ce que nous avons calculé dans le cours théorique ou même obtenu avec Minitab (le package prend le modèle des limites fixes – que Minitab propose aussi – mais cela serait toutefois mieux d'avoir le V-masque puisque c'est l'intérêt principal de cette carte selon moi).

La commande ne demande in extenso pas non plus les valeurs de h et δ (ou in extenso de K).

A priori il vaudrait mieux mettre cette carte de côté à ce jour... Donc affaire à suivre!

Exemple 306.: Carte de contrôle G

R 3.0.2

Nous reprenons les mêmes données que dans le cours théorique et que dans le cours Minitab:

```
R Console
> mydata<-read.csv("C:/tmp/CCG.csv",header=T,sep=";")
> mydata
        Dates Jours
1  05.02.2009    NA
2  03.04.2009    57
3  25.04.2009    22
4  06.05.2009    11
5  15.09.2009   132
6  21.09.2009     6
7  07.10.2009    16
8  22.11.2009    46
9  01.12.2009     9
10 05.01.2010    35
11 30.01.2010    25
12 07.04.2010    67
13 06.04.2010    58
14 24.06.2010    20
15 05.07.2010    11
16 03.08.2010    49
17 08.10.2010    46
18 04.12.2010    57
> 
```

Et nous faisons une analyse:

```
> qcc(mydata$Jours[-1], type = "g")
List of 11
 $ call      : language qcc(data = mydata$Jours[-1], type = "g")
 $ type      : chr "g"
 $ data.name : chr "mydata$Jours[-1]"
 $ data      : int [1:17, 1] 57 22 11 132 6 16 46 9 35 25 ...
  ..- attr(*, "dimnames")=List of 2
 $ statistics: Named int [1:17] 57 22 11 132 6 16 46 9 35 25 ...
  ..- attr(*, "names")= chr [1:17] "1" "2" "3" "4" ...
 $ sizes     : int [1:17] 1 1 1 1 1 1 1 1 1 1 ...
 $ center    : num 39.2
 $ std.dev   : num 38.7
 $ nsigmas   : num 3
 $ limits    : num [1, 1:2] 0 155
  ..- attr(*, "dimnames")=List of 2
 $ violations:List of 2
 - attr(*, "class")= chr "qcc"
Message d'avis :
In limits.g(center = 39.2352941176471, std.dev = 38.7320669520828,  :
  The Geometric distribution is quite skewed, it is better to set conf$
> 
```

Nous obtenons alors graphiquement:

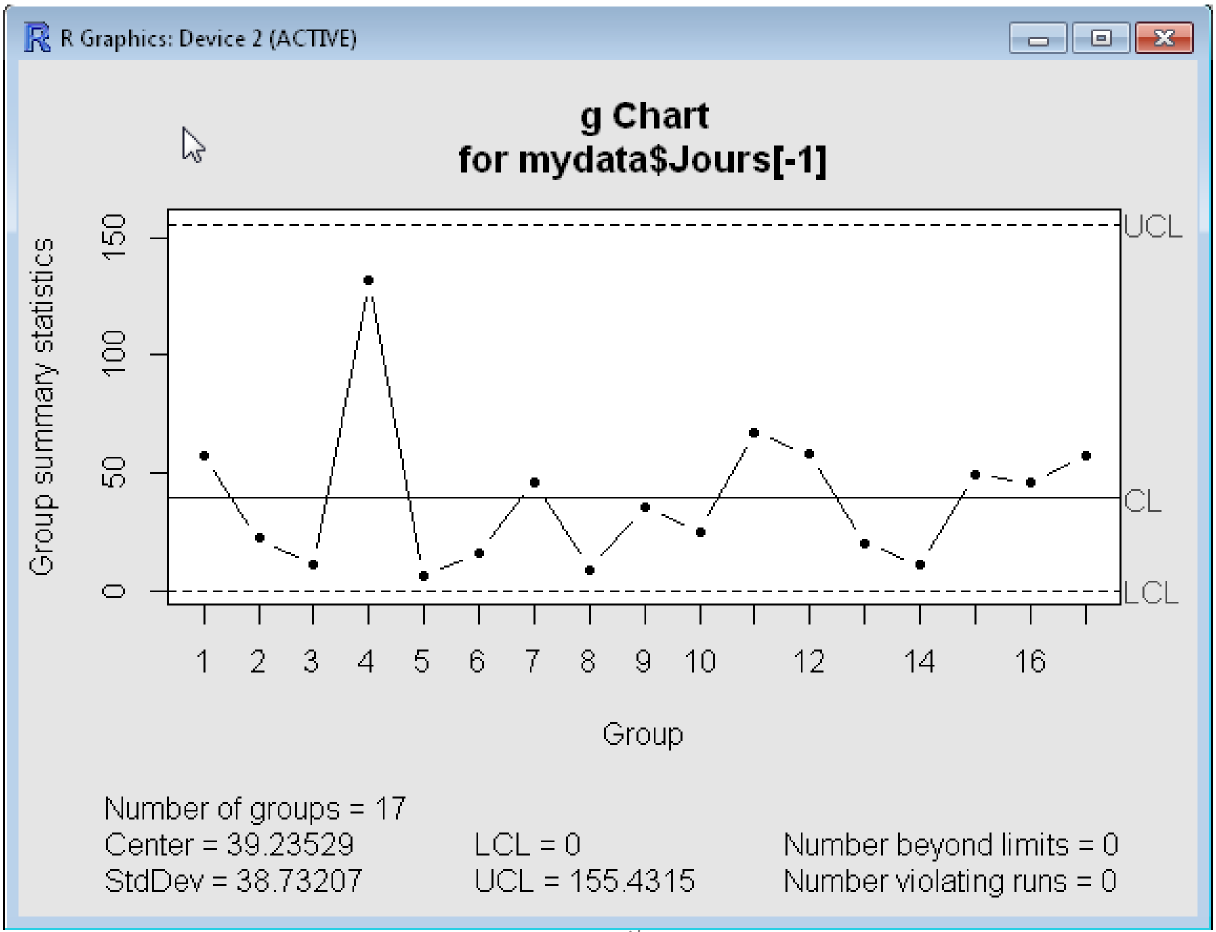
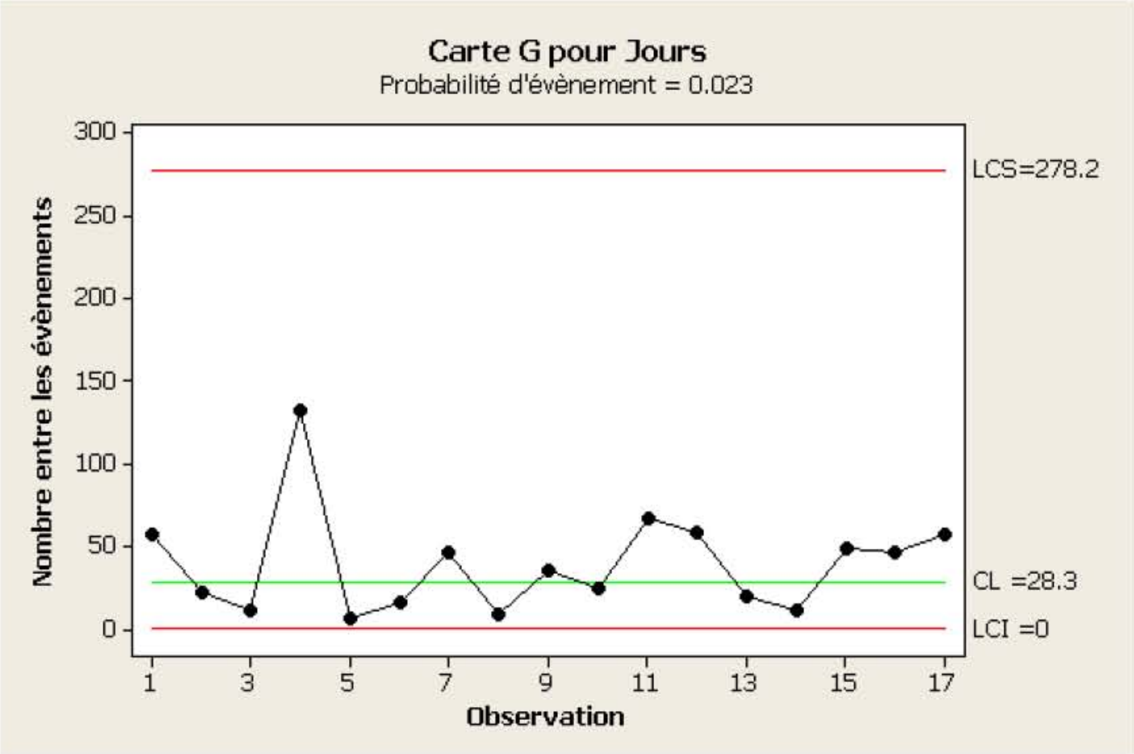


Figure 283 Graph Carte de contrôle G

L'allure de la carte est conforme à ce que nous avons fait dans Microsoft Excel et Minitab.

Nous avons:

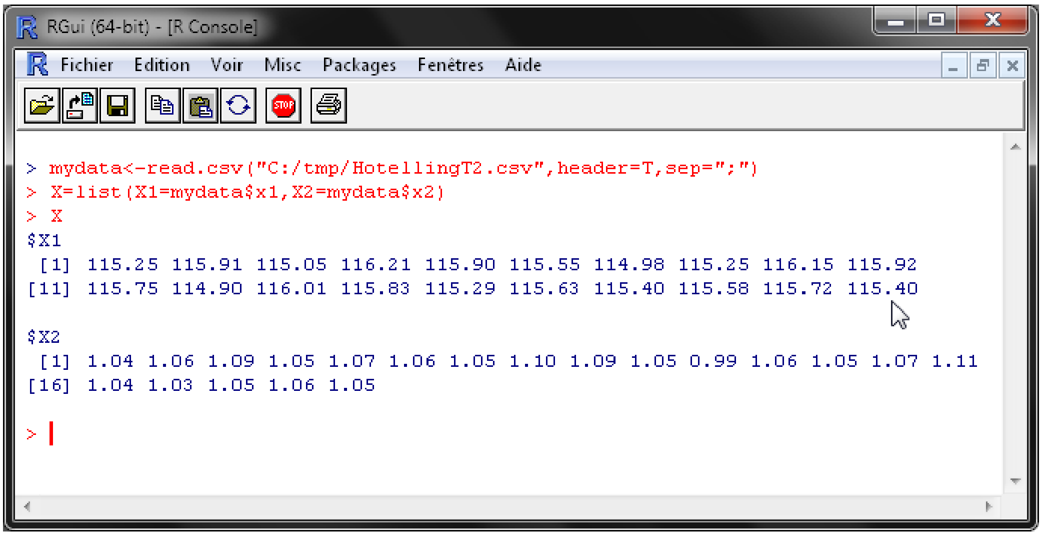
- La limite supérieure qui est de loin inférieure à celle de 278.2 donnée par Minitab mais elle correspond parfaitement à la valeur calculée à la main dans le cours théorique ☺
- La valeur centrale de 39.23 qui est bien supérieure à celle de 28.3 calculée par Minitab mais elle correspond parfaitement à la valeur calculée à la main dans le cours théorique ☺


Carte G pour Jours
Probabilité d'évènement = 0.023
Nombre entre les évènements
Observation
LCS=278.2
CL =28.3
LCI =0
300
250
200
150
100
50
0
1
3
5
7
9
11
13
15
17

Exemple 307.: Carte de contrôle multivariée T^2 de Hotelling

R 3.1.2

Nous allons vérifier ici si nous retrouvons les mêmes valeurs que dans le cours théorique et que Minitab. Nous partons donc des mêmes données:

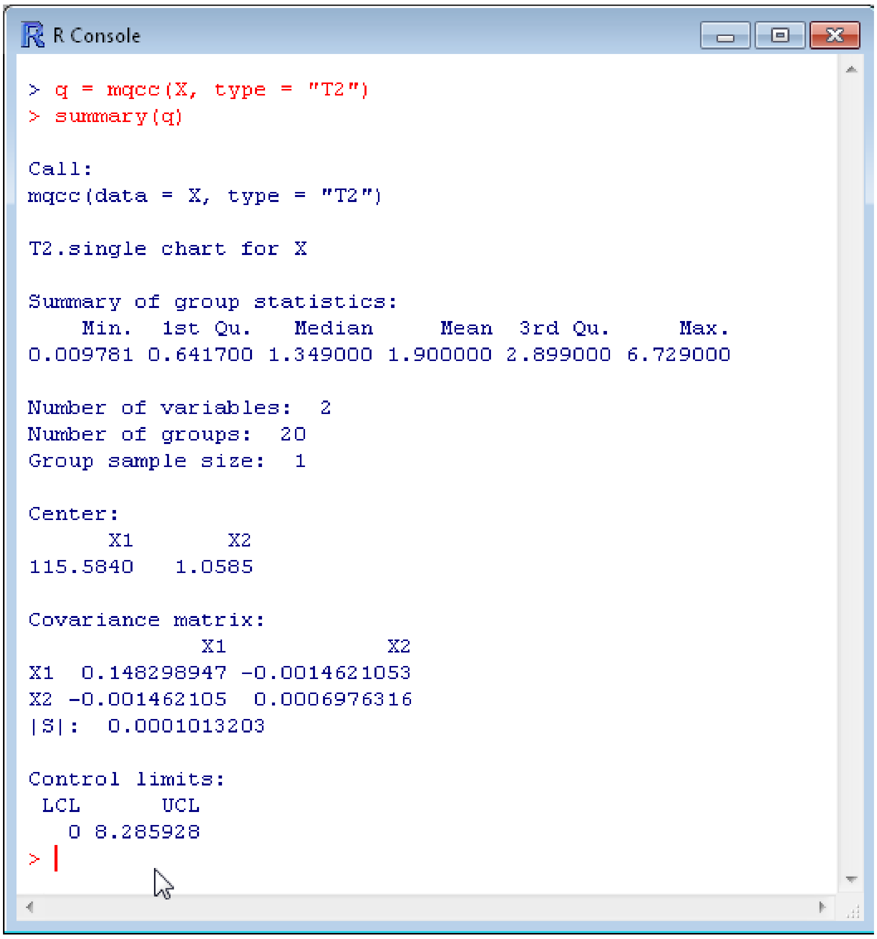


```
> mydata<-read.csv("C:/tmp/HotellingT2.csv",header=T,sep=";")
> X=list(X1=mydata$x1,X2=mydata$x2)
> X
$X1
 [1] 115.25 115.91 115.05 116.21 115.90 115.55 114.98 115.25 116.15 115.92
[11] 115.75 114.90 116.01 115.83 115.29 115.63 115.40 115.58 115.72 115.40

$X2
 [1] 1.04 1.06 1.09 1.05 1.07 1.06 1.05 1.10 1.09 1.05 0.99 1.06 1.05 1.07 1.11
[16] 1.04 1.03 1.05 1.06 1.05

> 
```

Ensuite, nous lançons une analyse quantitative:



```
> q = mqcc(X, type = "T2")
> summary(q)

Call:
mqcc(data = X, type = "T2")

T2.single chart for X

Summary of group statistics:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.009781 0.641700 1.349000 1.900000 2.899000 6.729000

Number of variables:  2
Number of groups:  20
Group sample size:  1

Center:
      X1        X2
115.5840    1.0585

Covariance matrix:
             X1            X2
X1  0.148298947 -0.0014621053
X2 -0.001462105  0.0006976316
|S|:  0.0001013203

Control limits:
 LCL      UCL
   0 8.285928
> 
```

Nous retrouvons la même matrice de variances-covariances, la même limite LCL mais par contre la limite UCL diffère considérablement et en plus nous ne savons pas pour quel niveau de confiance il est calculé?!

Qualitativement nous avons:

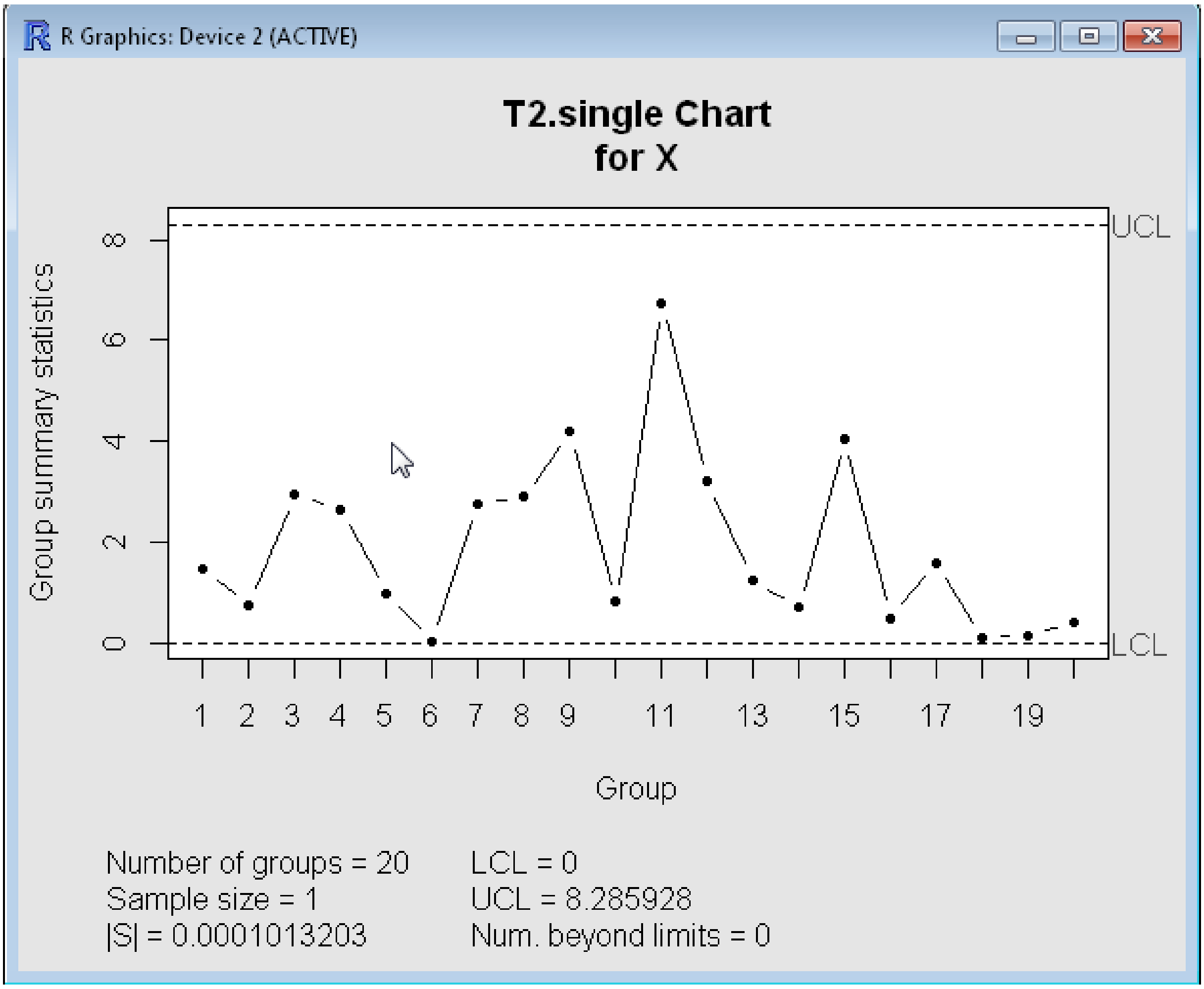
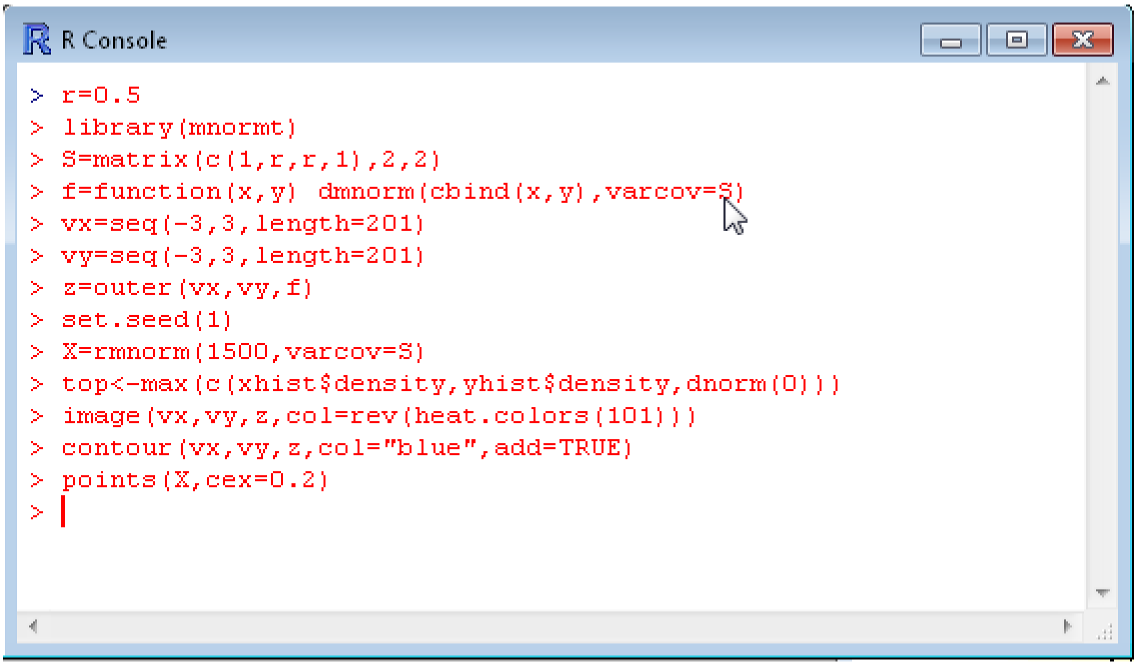


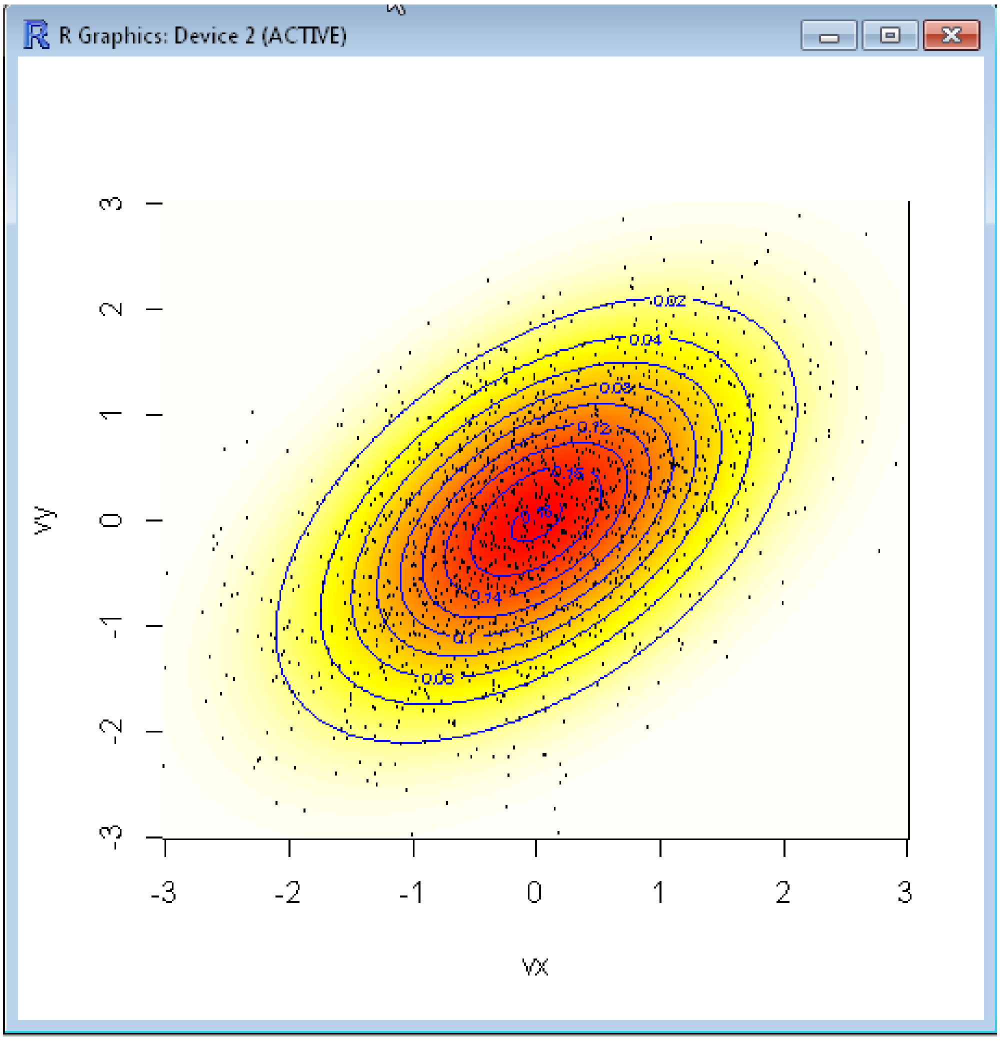
Figure 284 Graph Carte de contrôle T2 Hotelling

Ce qui est le même graphique que celui obtenu à la main à part la limite supérieure qui est très loin de celle calculée à la main ou de Minitab!

Pour ceux qui veulent l'ellipse de contrôle correspondant, ils peuvent se rendre à la page 1204. Mais au besoin voici une version simplifiée ici même:


R Console
> r=0.5
> library(mnormt)
> S=matrix(c(1,r,r,1),2,2)
> f=function(x,y) dmnorm(cbind(x,y),varcov=S)
> vx=seq(-3,3,length=201)
> vy=seq(-3,3,length=201)
> z=outer(vx,vy,f)
> set.seed(1)
> X=rmnorm(1500,varcov=S)
> top<-max(c(xhist$density,yhist$density,dnorm(0)))
> image(vx,vy,z,col=rev(heat.colors(101)))
> contour(vx,vy,z,col="blue",add=TRUE)
> points(X,cex=0.2)
>

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
0.02
0.04
0.08
0.12
0.1
0.06
vy
vx
-3
-2
-1
0
1
2
3

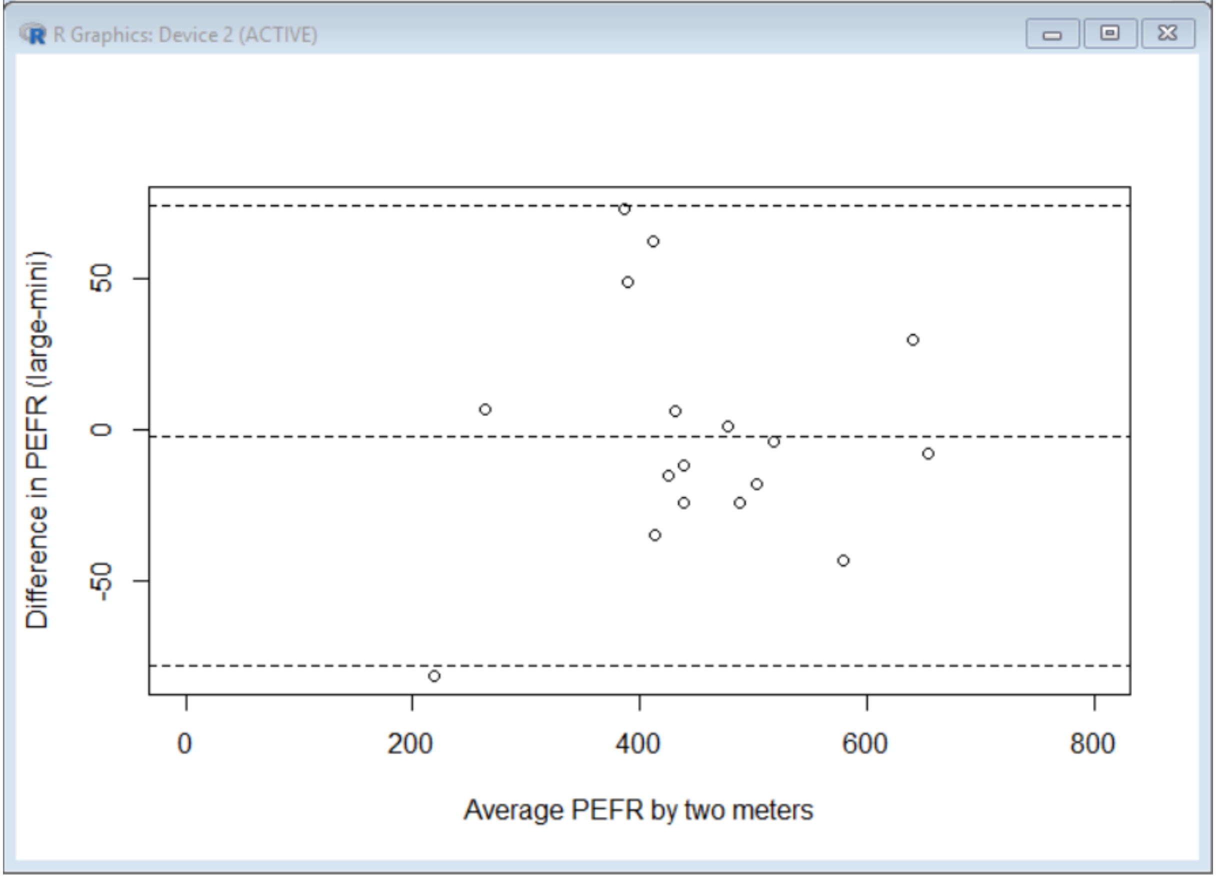
Exemple 308.: Carte Bland-Altman

R 3.4.2

Nous avons mentionné dans le cours théorique, lors de notre étude de la régression de Deming, les cartes de Bland-Altman qui sont une version visuelle de qualifier la régression orthogonale correspondante.

Alors voyons comment effectuer un tel graphique avec R avec les données de l'article d'origine de Bland et Altman en 1986 en utilisant la fonction **bland.altman.plot( )** du package **BlandAltmanLeh**:

```
> library("BlandAltmanLeh")
> PEFR1 <- bland.altman.PEFR[,1]
> PEFR2 <- bland.altman.PEFR[,3]
> bland.altman.plot(PEFR1, PEFR2, silent=TRUE, xlim=c(0,800),
+ xlab="Average PEFR by two meters",
+ ylab="Difference in PEFR (large-mini)")
NULL
>
```



Exemple 309.: Étude R&R (reproductibilité/répétabilité) pour données continues

R 3.0.2

Une étude R&R mesure l'erreur de précision en faisant mesurer par exemple une pièce par plusieurs fois et par des personnes différentes.

Étant donné que cette pièce ne change pas de taille, une variation dans les résultats doit représenter la répétabilité de la mesure et sa reproductibilité par différentes personnes.

Cette étude est très simple et basée sur l'ANOVA à deux facteurs avec répétition dont on peut déduire les données de l'étude R&R à partir du tableau de l'ANOVA (TAV). Pour plus d'exemples, le lecteur pour se référer à la norme ISO/TR 12888:2011.

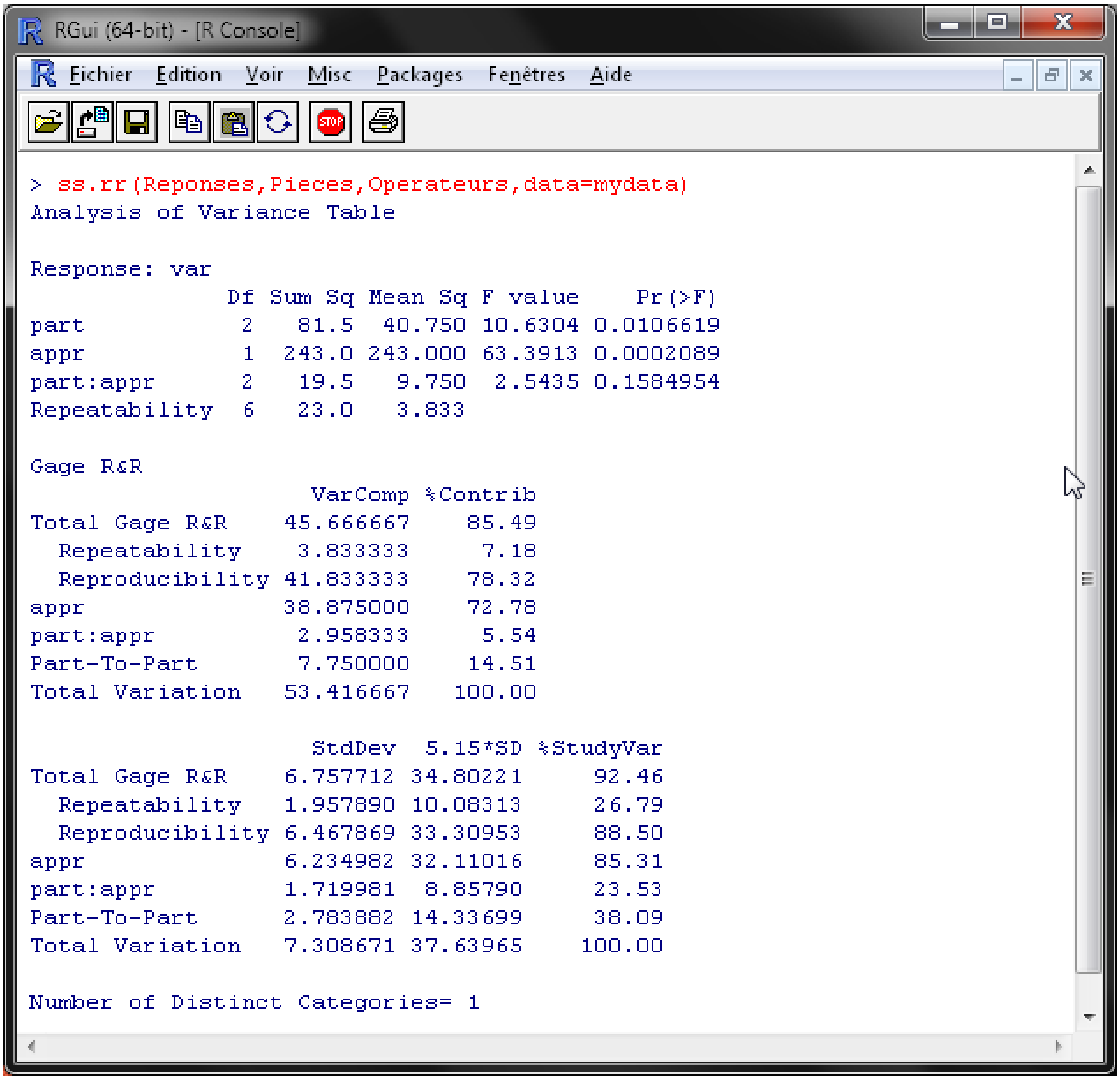
Considérons donc les données suivantes qui correspondent à une ANOVA à deux facteurs avec répétition (il s'agit des mêmes valeurs numériques que l'exercice de l'ANOVA à deux facteurs sans répétition vu plus haut mais détourné).

Nous reprenons donc les données du cours Minitab pour vérifier la concordance:

```
> library(SixSigma)
> mydata<-read.csv("C:/EtudeRRContinu.csv",header=T,sep=";")
> mydata
    Operateurs  Pieces  Reponses
1  Operateur 1 Piece 1       256
2  Operateur 1 Piece 1       251
3  Operateur 1 Piece 2       246
4  Operateur 1 Piece 2       248
5  Operateur 1 Piece 3       254
6  Operateur 1 Piece 3       251
7  Operateur 2 Piece 1       242
8  Operateur 2 Piece 1       240
9  Operateur 2 Piece 2       238
10 Operateur 2 Piece 2       240
11 Operateur 2 Piece 3       246
12 Operateur 2 Piece 3       246
> 
```

et du package **SixSigma**[31], nous utilisons la commande **ss.rr( )** :

[31] Voir aussi la fonction **gageRRDesign( )** du package **qualityTools**



```
> ss.rr(Reponses,Pieces,Operateurs,data=mydata)
Analysis of Variance Table

Response: var
              Df Sum Sq Mean Sq F value    Pr(>F)
part           2   81.5  40.750 10.6304 0.0106619
appr           1  243.0 243.000 63.3913 0.0002089
part:appr      2   19.5   9.750  2.5435 0.1584954
Repeatability  6   23.0   3.833

Gage R&R
                    VarComp %Contrib
Total Gage R&R    45.666667    85.49
  Repeatability    3.833333     7.18
  Reproducibility 41.833333    78.32
appr              38.875000    72.78
part:appr          2.958333     5.54
Part-To-Part       7.750000    14.51
Total Variation   53.416667   100.00

                    StdDev  5.15*SD %StudyVar
Total Gage R&R    6.757712 34.80221     92.46
  Repeatability   1.957890 10.08313     26.79
  Reproducibility 6.467869 33.30953     88.50
appr              6.234982 32.11016     85.31
part:appr         1.719981  8.85790     23.53
Part-To-Part      2.783882 14.33699     38.09
Total Variation   7.308671 37.63965    100.00

Number of Distinct Categories= 1
```

Nous retrouvons donc les mêmes valeurs que dans le cours Minitab. Le graphique nous donne quant à lui (attention!! parfois il faut exécuter la commande ss.rr( ) plusieurs fois pour que le graphique s'affiche sans erreurs):

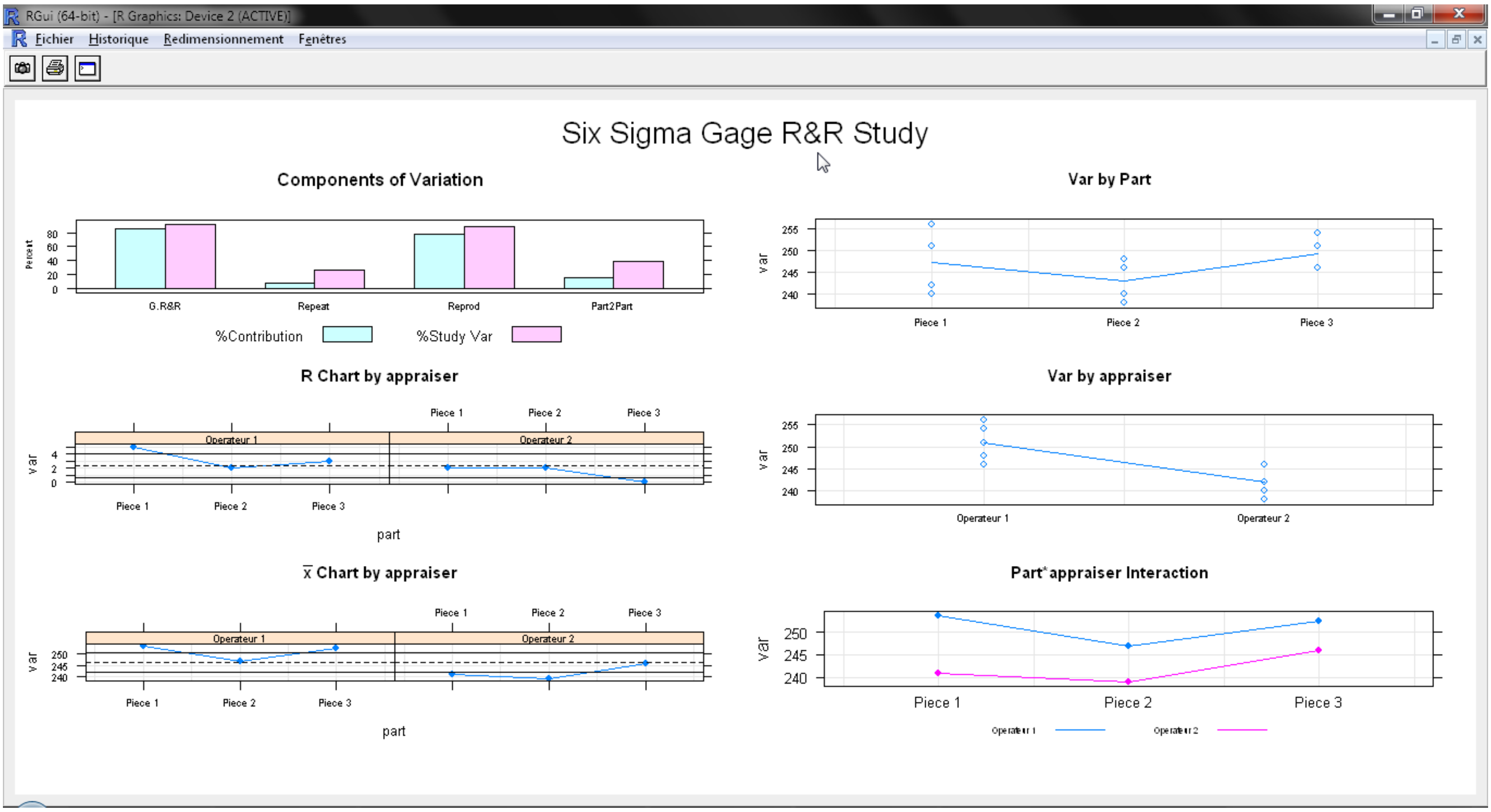


Figure 285 Graph étude Gage R&R (Six Sigma)

La lisibilité du graphique n'est pas des meilleures sur un petit écran mais à part cela, nous retrouvons bien les mêmes informations que celles renvoyées par Minitab.

Exemple 310.: Coefficient de corrélation intra-classe ICC1

R 3.1.1.

Le but comme ici est comme à l'habitude de vérifier les calculs obtenus à la main dans le cours théorique suite à la démonstration mathématique de l'origine du coefficient de corrélation intra-classe ICC1.

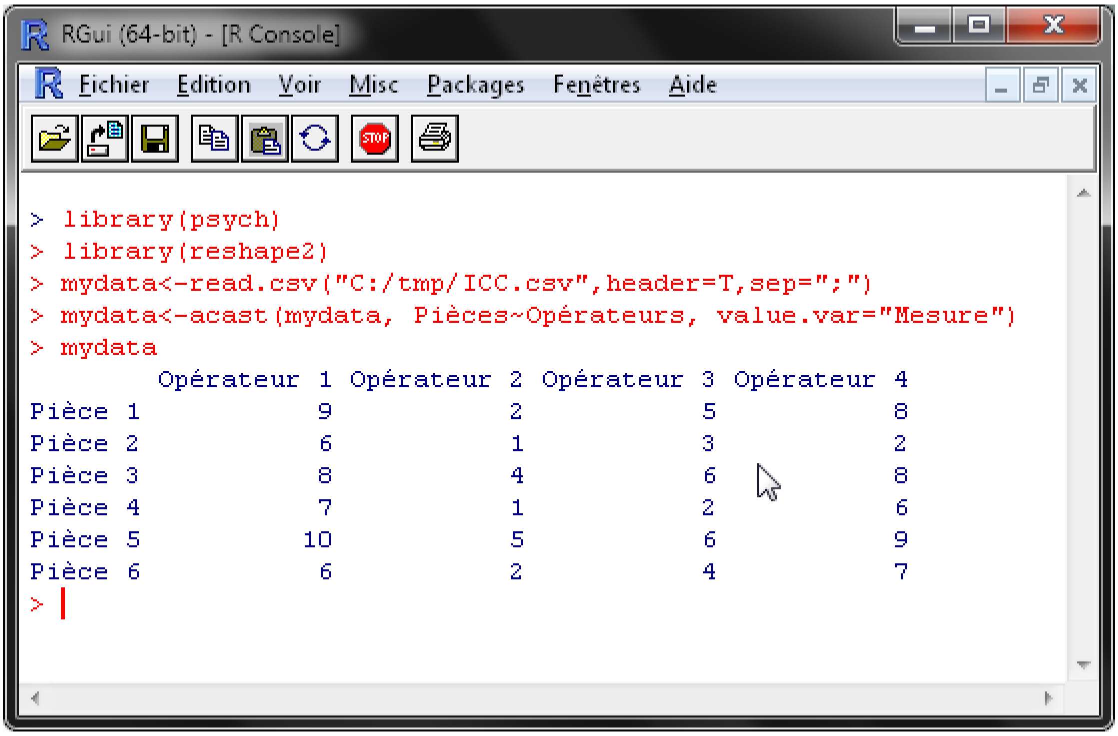
Nous partons donc des mêmes données que dans le cours théorique:


ICC.csv - Bloc-notes
Fichier Edition Format Affichage ?

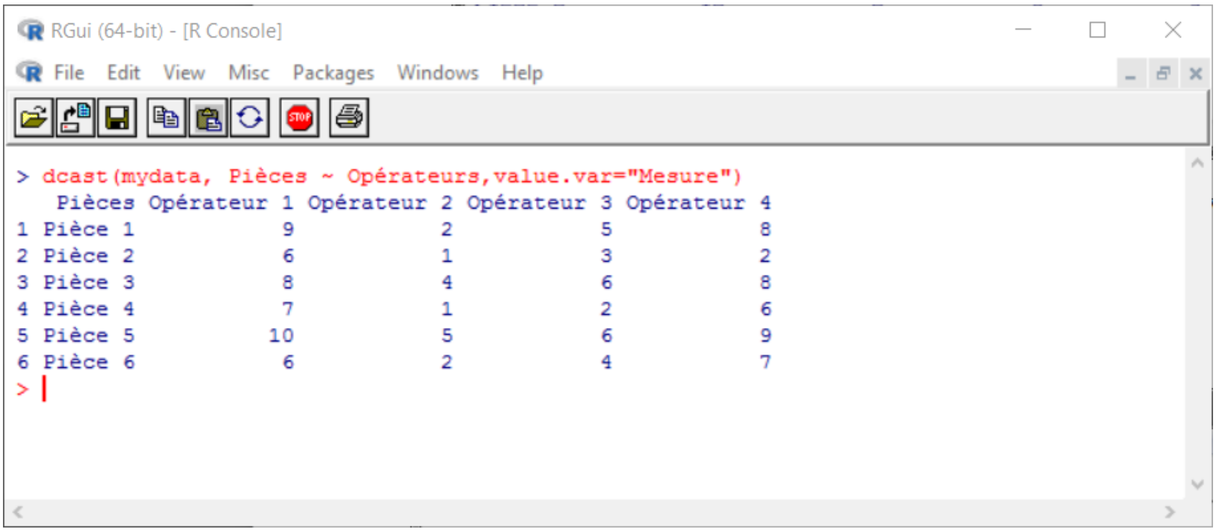
```
Pièces;Mesure;Opérateurs
Pièce 1;9;Opérateur 1
Pièce 2;6;Opérateur 1
Pièce 3;8;Opérateur 1
Pièce 4;7;Opérateur 1
Pièce 5;10;Opérateur 1
Pièce 6;6;Opérateur 1
Pièce 1;2;Opérateur 2
Pièce 2;1;Opérateur 2
Pièce 3;4;Opérateur 2
Pièce 4;1;Opérateur 2
Pièce 5;5;Opérateur 2
Pièce 6;2;Opérateur 2
Pièce 1;5;Opérateur 3
Pièce 2;3;Opérateur 3
Pièce 3;6;Opérateur 3
Pièce 4;2;Opérateur 3
Pièce 5;6;Opérateur 3
Pièce 6;4;Opérateur 3
Pièce 1;8;Opérateur 4
Pièce 2;2;Opérateur 4
Pièce 3;8;Opérateur 4
Pièce 4;6;Opérateur 4
Pièce 5;9;Opérateur 4
Pièce 6;7;Opérateur 4
```

Nous utilisons le package **psych** pour faire l'analyse avec la fonction **ICC( )** et le package **reshape2** avec la fonction **acast( )**[32] pour restructurer (pivoter) les données:

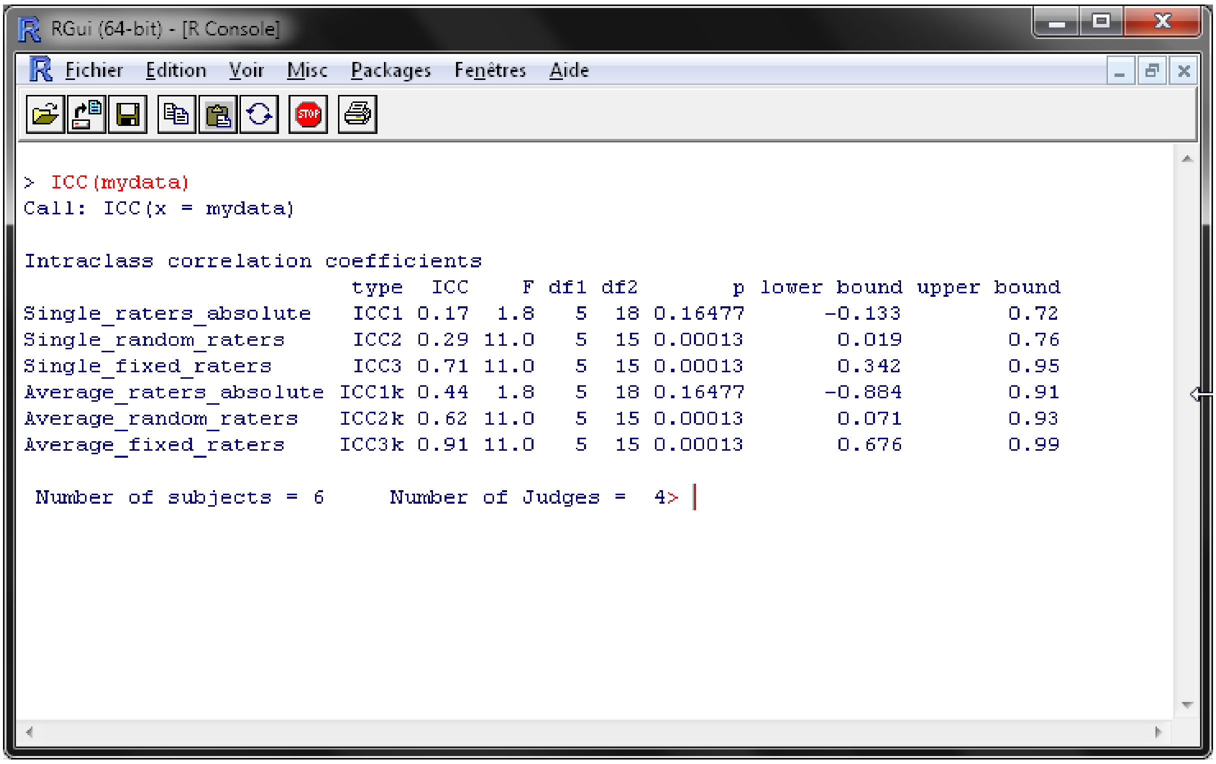
[32] Le "a" de acast( ) signifie "array" car la fonction dcast( ) du même package renvoie un "data frame".



Nous avons donc besoin d'un array (le "a" de acast( )) mais si un jour vous veniez à avoir besoin d'un data frame vous pouvez utiliser **dcast( )**:



Une fois ceci fait, nous lançons l'analyse:



Nous retrouvons bien pour le ICC1 celui du calcul fait à la main suite à la démonstration mathématique faite dans le cours théorique. Pour les autres ainsi que les intervalles de confiance il faudra attendre la rédaction d'une démonstration mathématique élégante pour qu'elle soit présentée en classe.

Il est d'usage de présenter un ICC avec son graphique classique utilisant la fonction **dotplot( )** du package **lattice**:

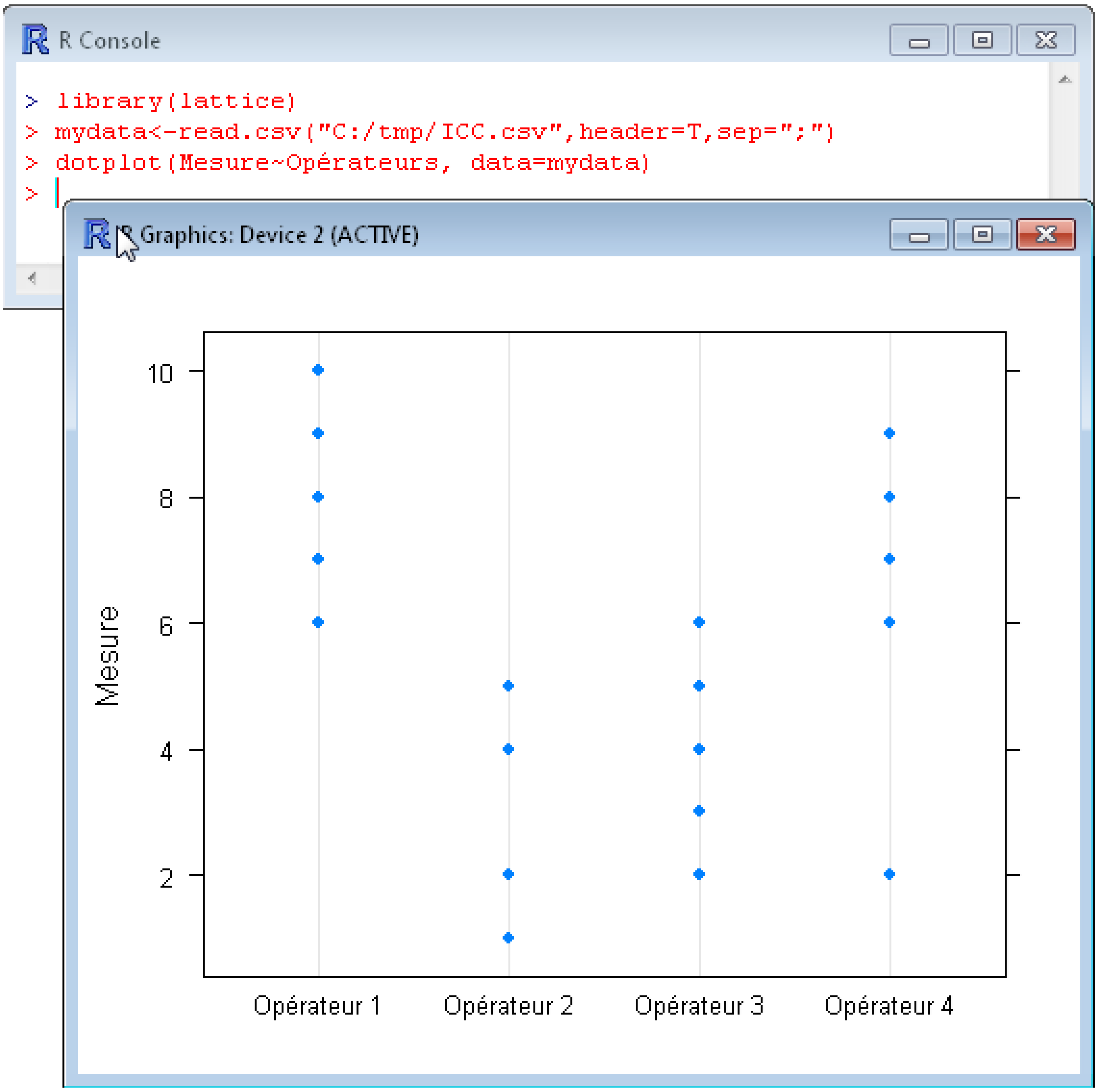


Figure 286 Graph d'analyse CCI (coefficient corrélation intra-classe)

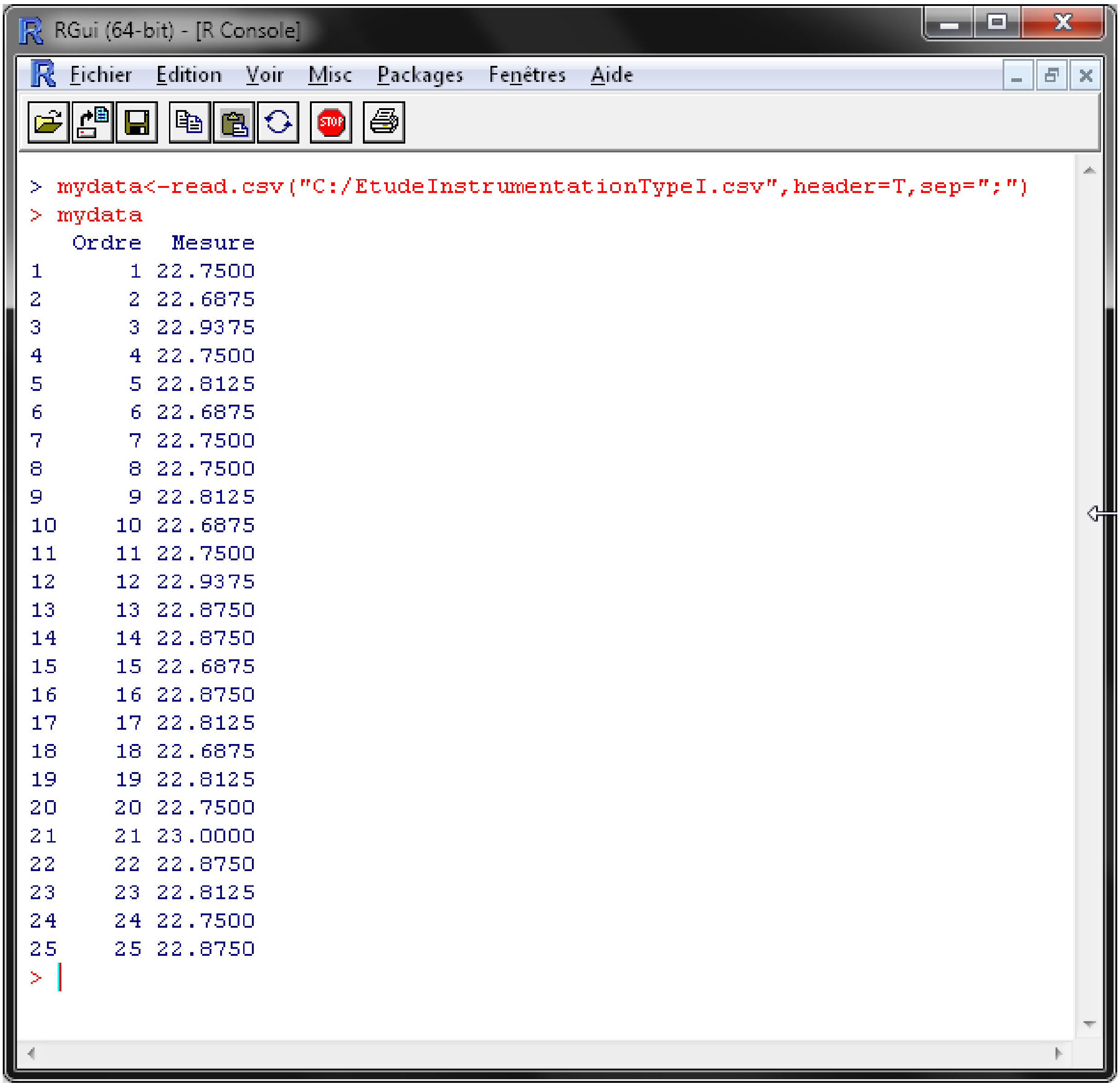
Voilà résultat superbe obtenu!

Exemple 311.: Étude de l'instrumentation de type I

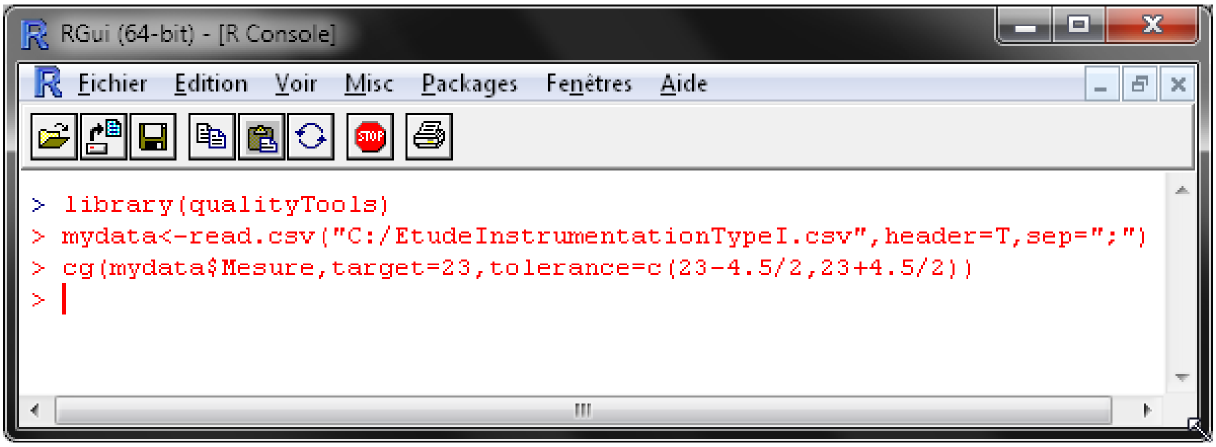
R 3.0.2

Bon revoilà le sujet qui fait mal et dont nous avons pas mal parlé dans le cours théorique de MSP bien que la mathématique (empirique) sous-jacente soit triviale.

Nous travaillons avec le même jeu de données:



Et nous restons avec le package **qualityTools** en utilisant la commande **cg( )** :



Ce qui donne:

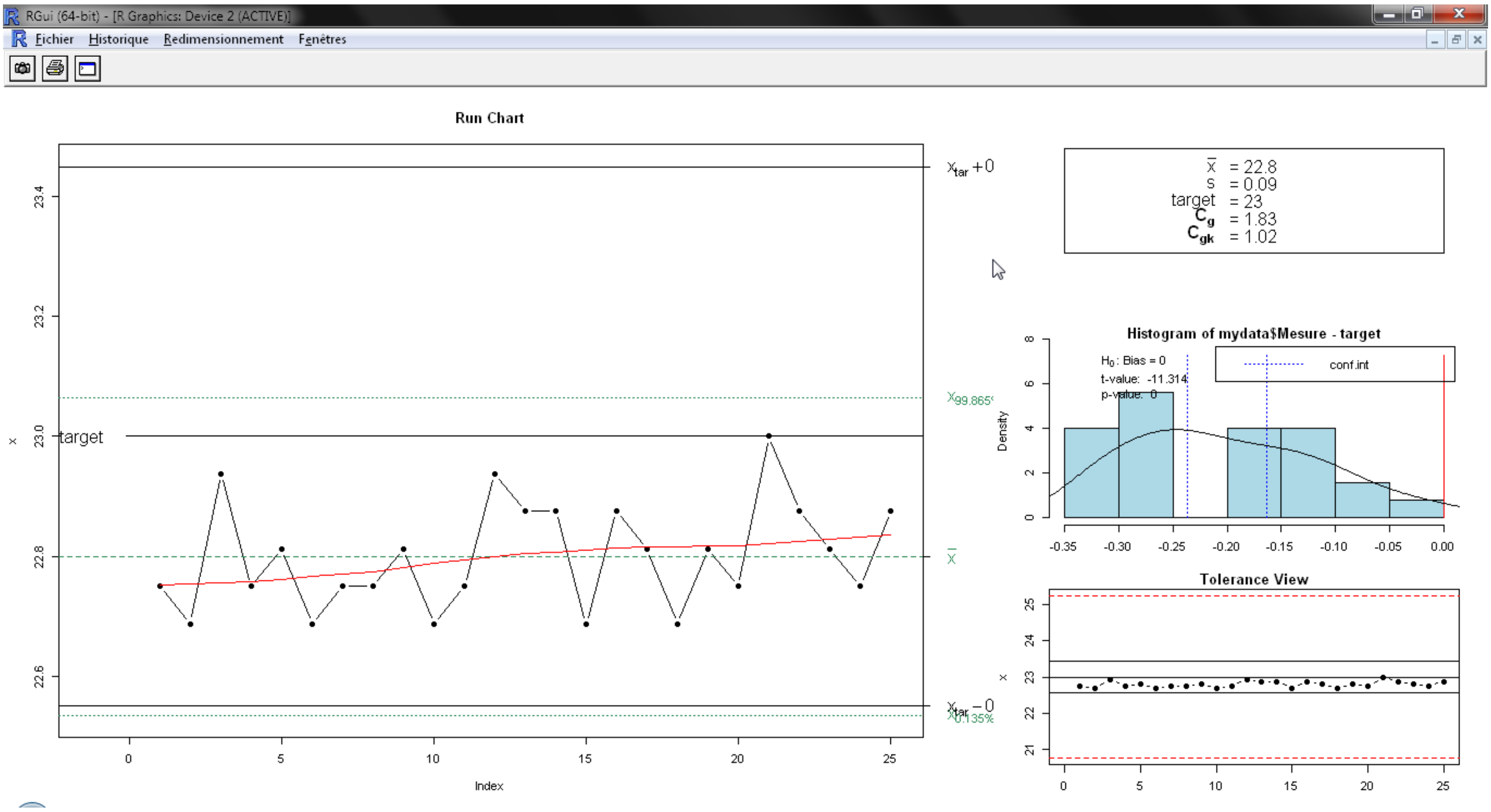


Figure 287 Graph étude de l'instrumentation de type I

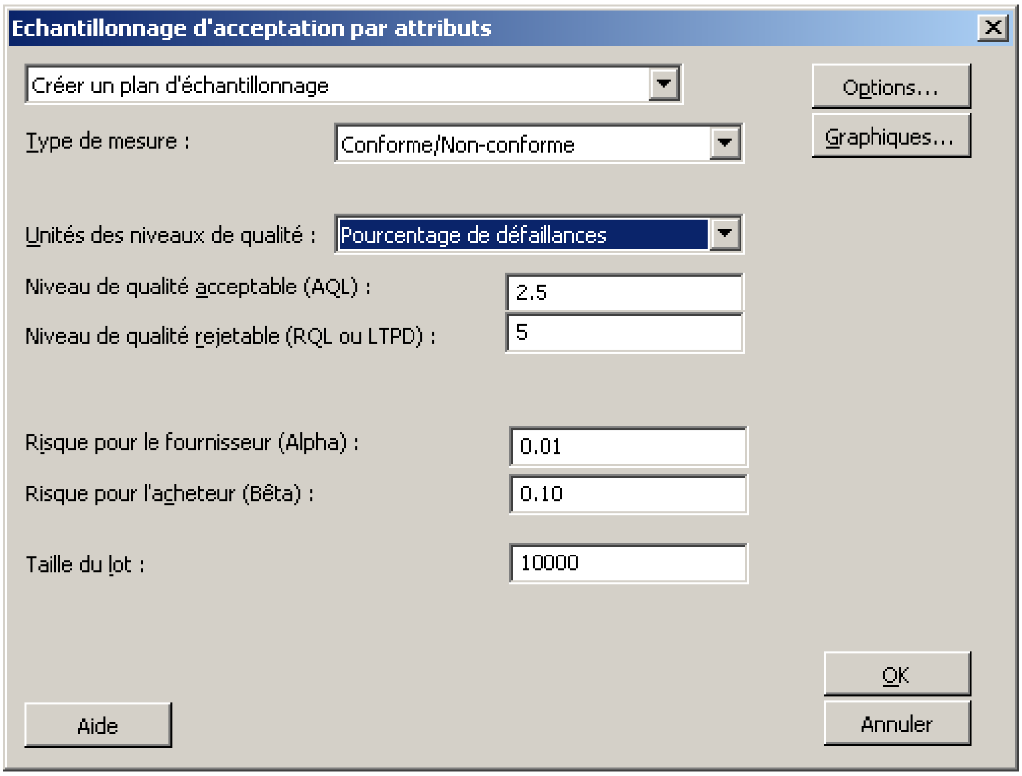
Bon ce n'est pas surprenant on retrouve à un tout petit pourcentage près les mêmes valeurs qu'avec Minitab mais rappelez-vous une chose. Dans l'état actuel, trois logiciels (Minitab, R et QIMacros) utilisent trois définitions différentes (à ce jour) de la capabilité de gabarit. Donc le jour où il y aura une norme internationale qui mettra tout le monde d'accord on en rediscutera.

Exemple 312.: Plan d'échantillonnage par attributs

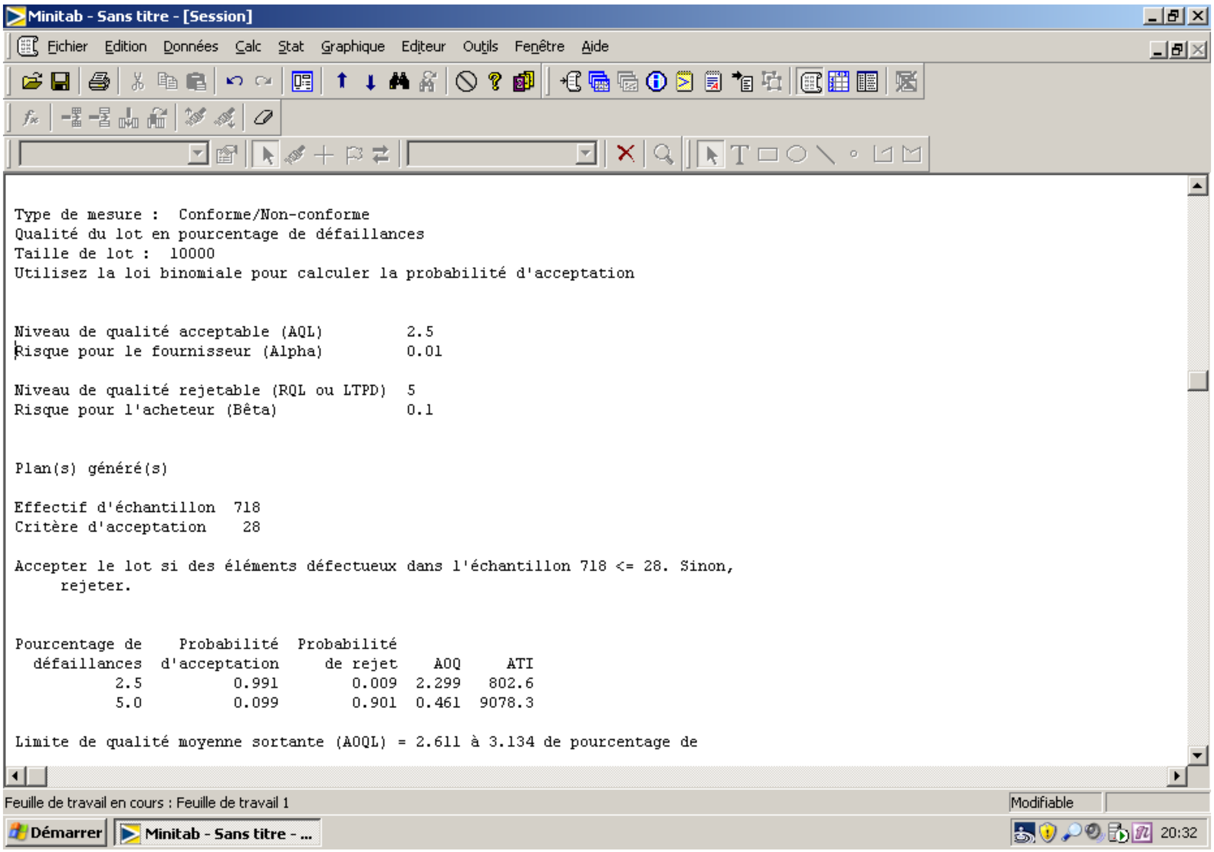
R 3.0.2

Bon ce sujet est comme nous le savons délicat étant donné qu'il existe plusieurs normes nationales qui sont utilisées à l'instar des normes internationales ISO… Nous avons déjà pu le vérifier avec Minitab où nous avions un écart d'environ 3% pour la taille de l'échantillon entre les calculs théoriques et le logiciel et un écart de 7% pour le nombre d'éléments définissant si le lot est à rejeter ou non.

Alors d'abord rappelons juste que dans Minitab nous avions pour les paramètres d'entrées ci-dessous (pour un plan par attributs!):



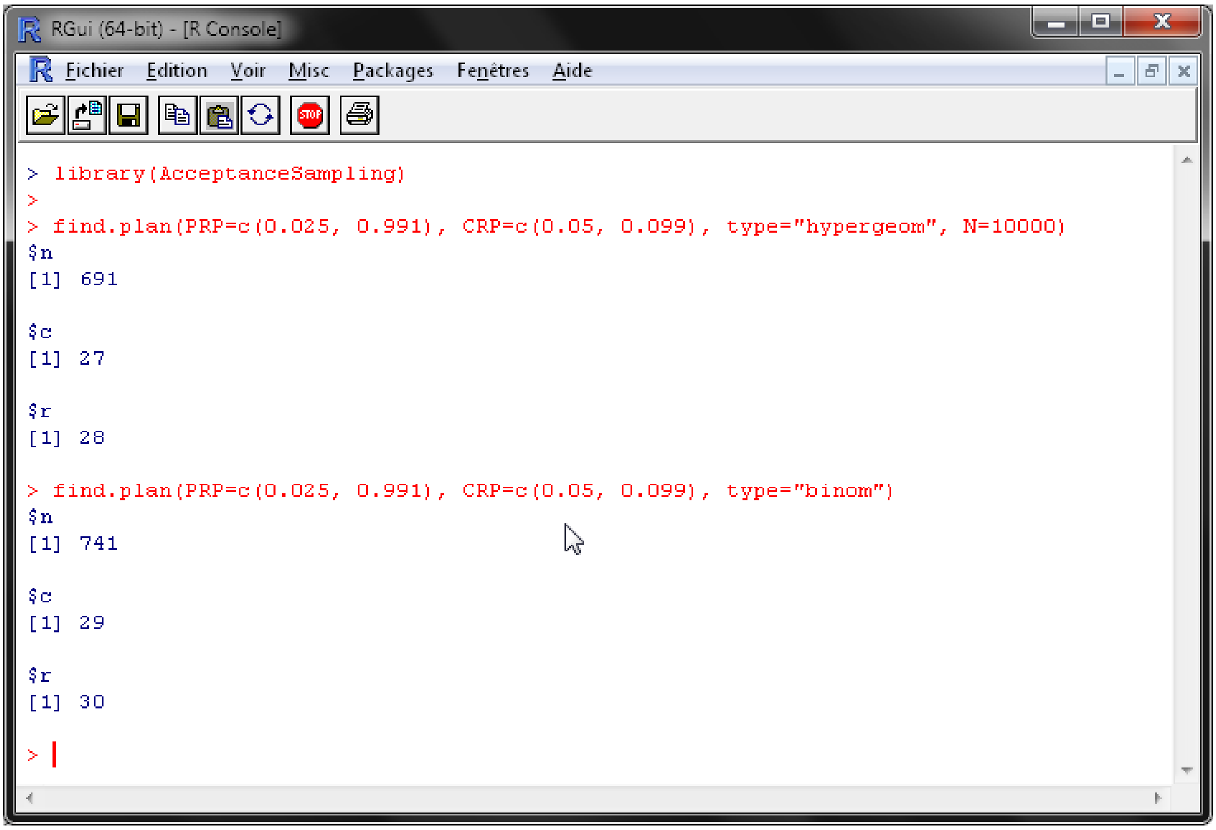
Ce qui nous avait donné:



Avec le package **AcceptanceSampling** il n'est pas possible de trouver le plan avec la même approche. Ce package exige que nous utilisions les probabilités d'acceptation et les pourcentages de défaillance. C'est-à-dire que nous supposons que nous avons à l'avance les données suivantes:

Pourcentage de défaillances	Probabilité d'acceptation	Probabilité de rejet	AOQ	ATI
2.5	0.991	0.009	2.299	802.6
5.0	0.099	0.901	0.461	9078.3

Dans R, il nous faut alors utiliser la fonction **find.plan( )** du package **AcceptanceSampling** et nous pouvons comparer le modèle binomial au modèle hypergéométrique très facilement par la même occasion:



où:

- PRP signifie *Producer Risk Point*, le premier argument est le niveau de qualité auquel nous voulons évaluer le plan (AQL) et le second la probabilité <u>minimale</u> *d'acceptation* du plan.
- CRP signifie *Consumer Risk Point*, le premier argument est le risque de qualité auquel nous voulons évaluer le plan (RQL) et le second la probabilité maximale *d'acceptation* du plan.

Nous retrouvons alors des valeurs "relativement" proches du plan proposé par Minitab:

```
Plan(s) généré(s)

Effectif d'échantillon  718
Critère d'acceptation    28

Accepter le lot si des éléments défectueux dans l'échantillon 718 <= 28. Sinon,
     rejeter.
```

Exemple 313.: Intervalle de Tolérance paramétrique pour loi Normale

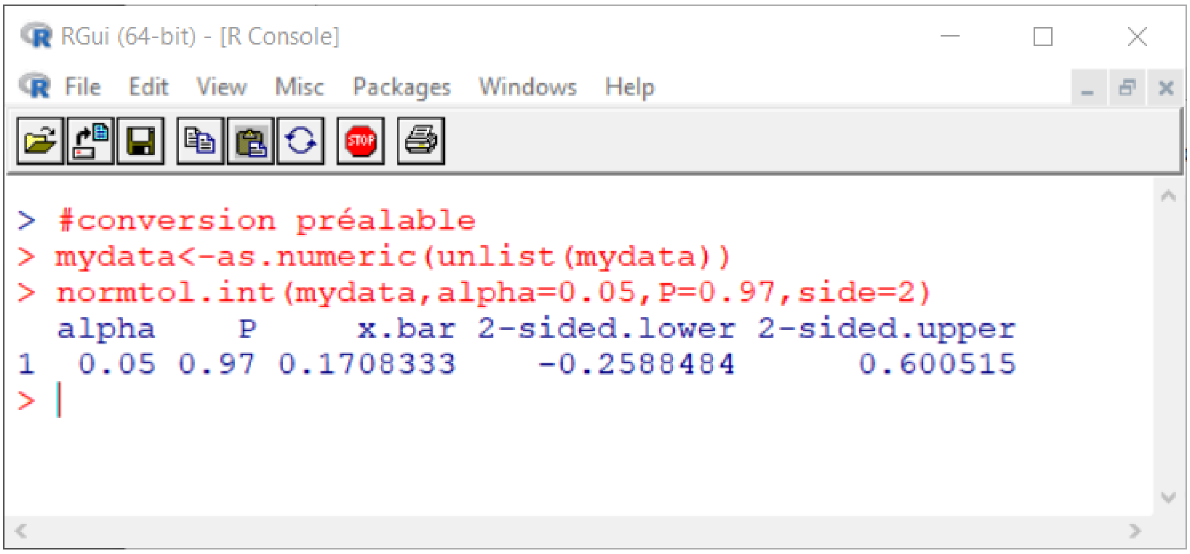
R 3.4.3

Nous avons étudié en détails dans le cours théorique ce qu'était l'intervalle de tolérance (paramétrique) pour une loi Normale et à quoi cela pouvait être extrêmement utile dans l'industrie et les services. Voyons donc de quoi il s'agit sans oublier que les données de l'intégrale qui ne peut être résolue à ce jour que numériquement est tabulée dans la norme ISO 16269-6.

Nous chargeons le package **tolerance** qui nous sera nécessaire et le jeu de données avec lequel nous voulons jouer afin de comparer avec Minitab:

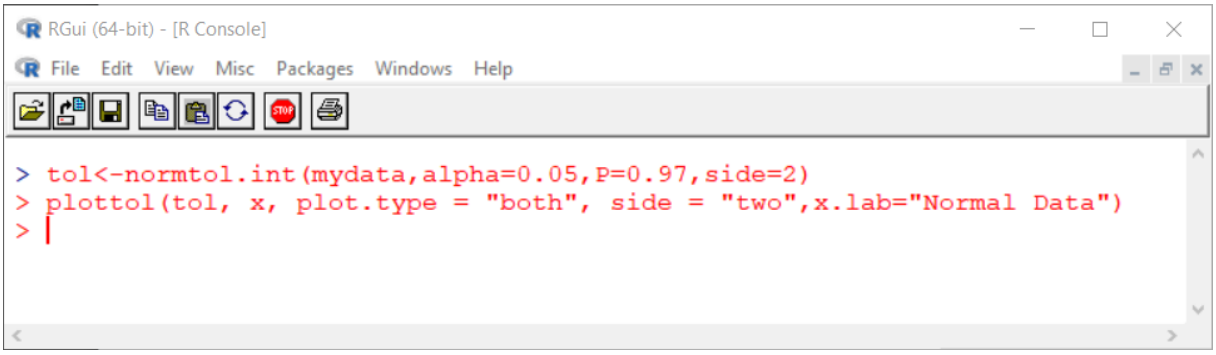
```
> library("tolerance")
> mydata<-read.csv("c:/tmp/intervalle_tolerance.csv",header=T,sep=";")
> mydata
   Ecarts
1    0.45
2    0.40
3    0.35
4    0.35
5    0.30
6    0.30
7    0.25
8    0.25
9    0.20
10   0.20
11   0.20
12   0.20
13   0.15
14   0.15
15   0.15
16   0.10
17   0.10
18   0.10
19   0.05
20   0.05
21   0.00
22  -0.05
23  -0.05
24  -0.10
> 
```

Nous avons donc avec la fonction **normaltol.int( )** :



Nous retrouvons donc ainsi exactement les mêmes valeurs qu'avec Minitab!

Voyons le graphique correspondant avec la fonction **plottol( )** :



Ce qui donne:

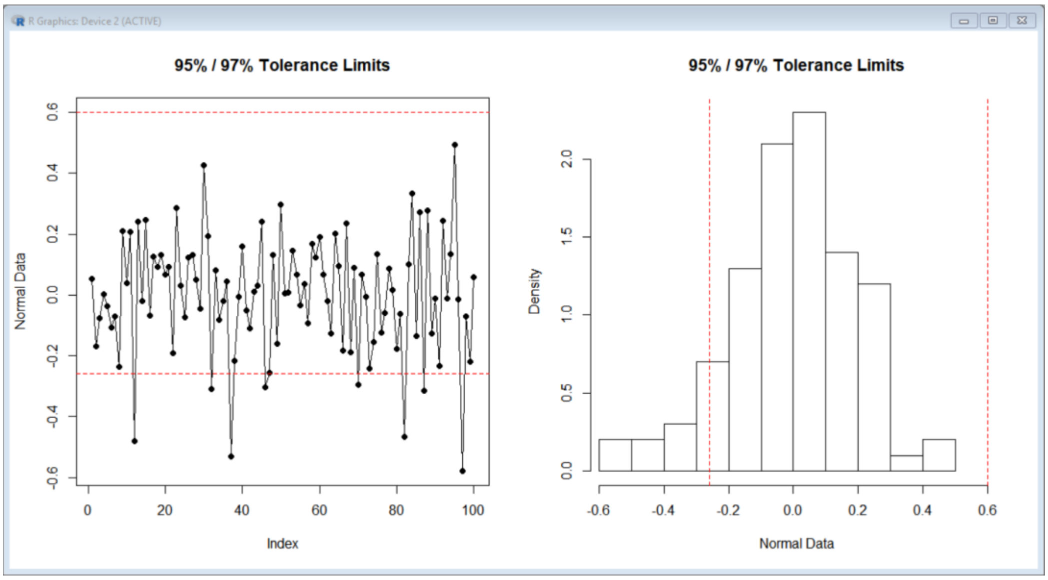


Figure 288 Graph d'analyse d'intervalle de tolérance

Il faut avouer… nous sommes très loin de la qualité de sortie du graphique correspondant dans Minitab…

21. Méthodes brutes d'inférences

Le but ici va être de voir la puissance de frappe de R par rapport à des logiciels de simulations numériques comme Matlab, @Risk (inutile de mentionner les tableurs qui dans ces cas business réels avec plus d'une dizaine de variables sont compléments hors course).

Nous allons voir comment utiliser les méthodes de Monte-Carlo (méthodes pseudo/stratifiées ou de Fauré), du Bootstrapping, du Jacknife et du Latin Hypercube que dont nous avons étudié les bases théoriques dans le cours.

Exemples 314.: Monte-Carlo

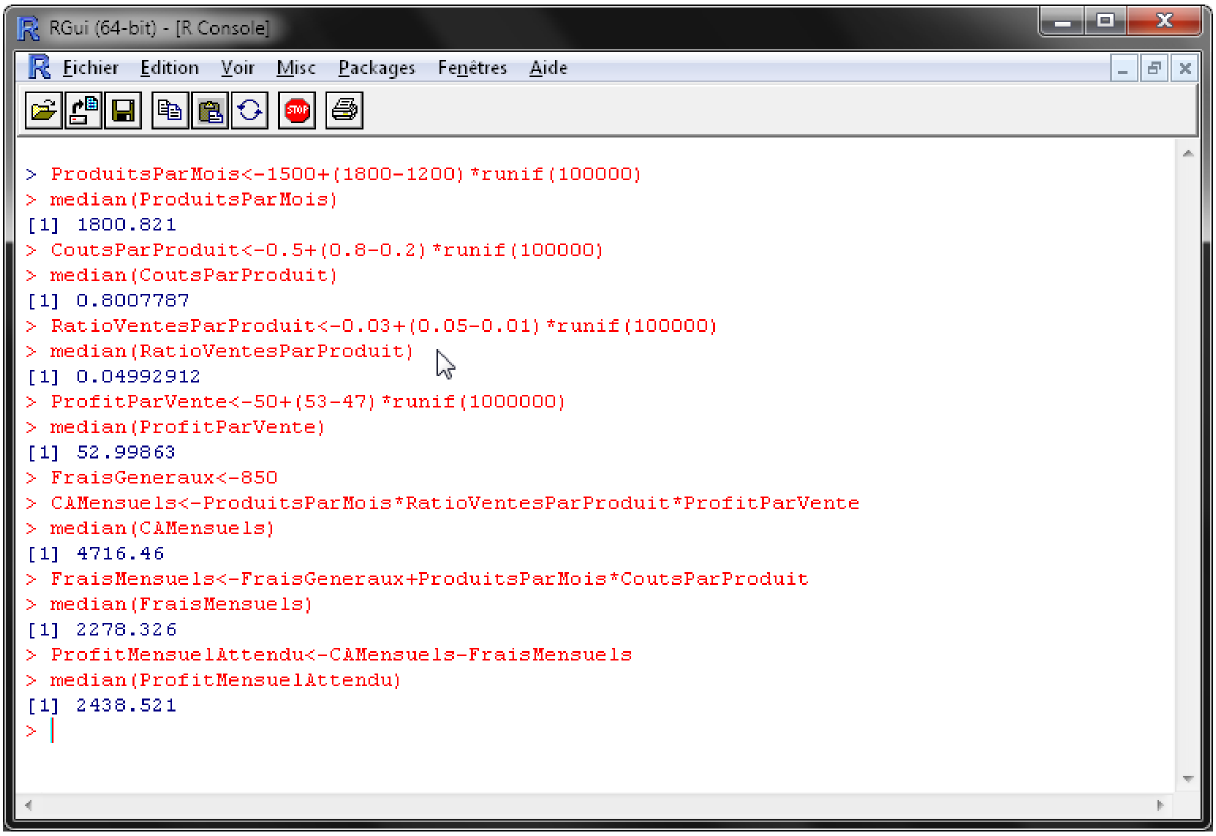
R 3.0.2

Comme nous l'avons mentionné dans le cours théorique, un l'aspect central des méthodes de Monte-Carlo est surtout relatif à la génération de variables aléatoires indépendantes ou non.

Méthode classique (variables pseudo-aléatoires)

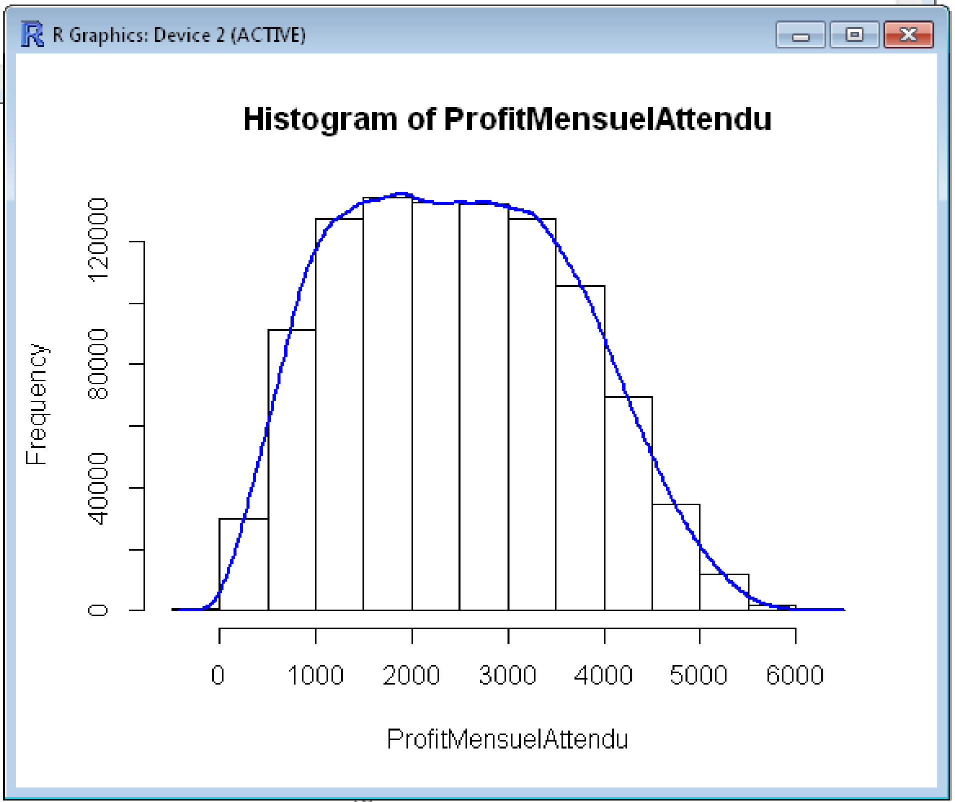
Nous retrouverons les méthodes de Monte-Carlo lors de l'étude des méthodes de scripting dans R (programmation) à partir de la page 2682.

Nous allons ici utiliser la puissance du C++ derrière R et la simplicité de sa syntaxe pour reproduire en quelques secondes un exercice pour lequel il nous a fallu un peu plus de 45 minutes à mettre en place dans le tableur MS Excel:



Avec un premier petit graph à l'aide de nos connaissances cumulée sur les graphiques:

```
> h<-hist(ProfitMensuelAttendu)
> d<-density(ProfitMensuelAttendu)
> lines(x=d$x,y=d$y*length(ProfitMensuelAttendu)*diff(h$breaks)[1],lwd=2,col="blue")
>
```



et une deuxième petit graph encore plus important dans ce cas business:

```
> j<-ecdf(ProfitMensuelAttendu)
> plot(j,xlab="Profits",main="fonction de répartition empirique")
> grid()
> abline(h=0.95,col="red");abline(v=quantile(ProfitMensuelAttendu,0.95),col="red")
>
```

Ce qui donne:

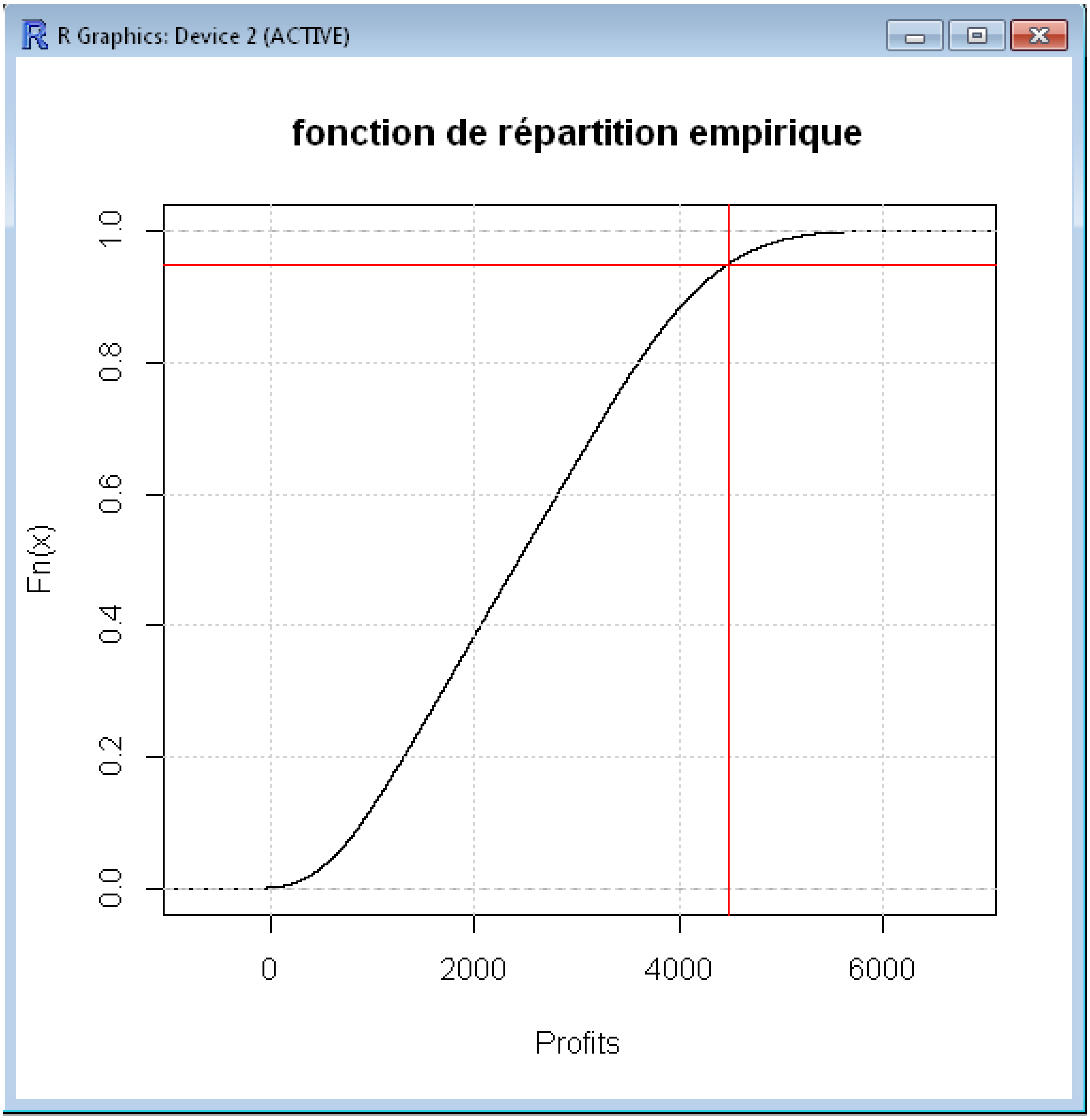


Figure 289 Graph de fonction de répartition avec ligne verticale et horizontale

et ainsi de suite avec tous les graphs que l'on veut en quelques secondes. Personnellement je trouve donc R plus intéressant et plus puissant que n'importe que tableur et même plus rapide que @Risk.

Exemple de calcul de la surface du quart de disque

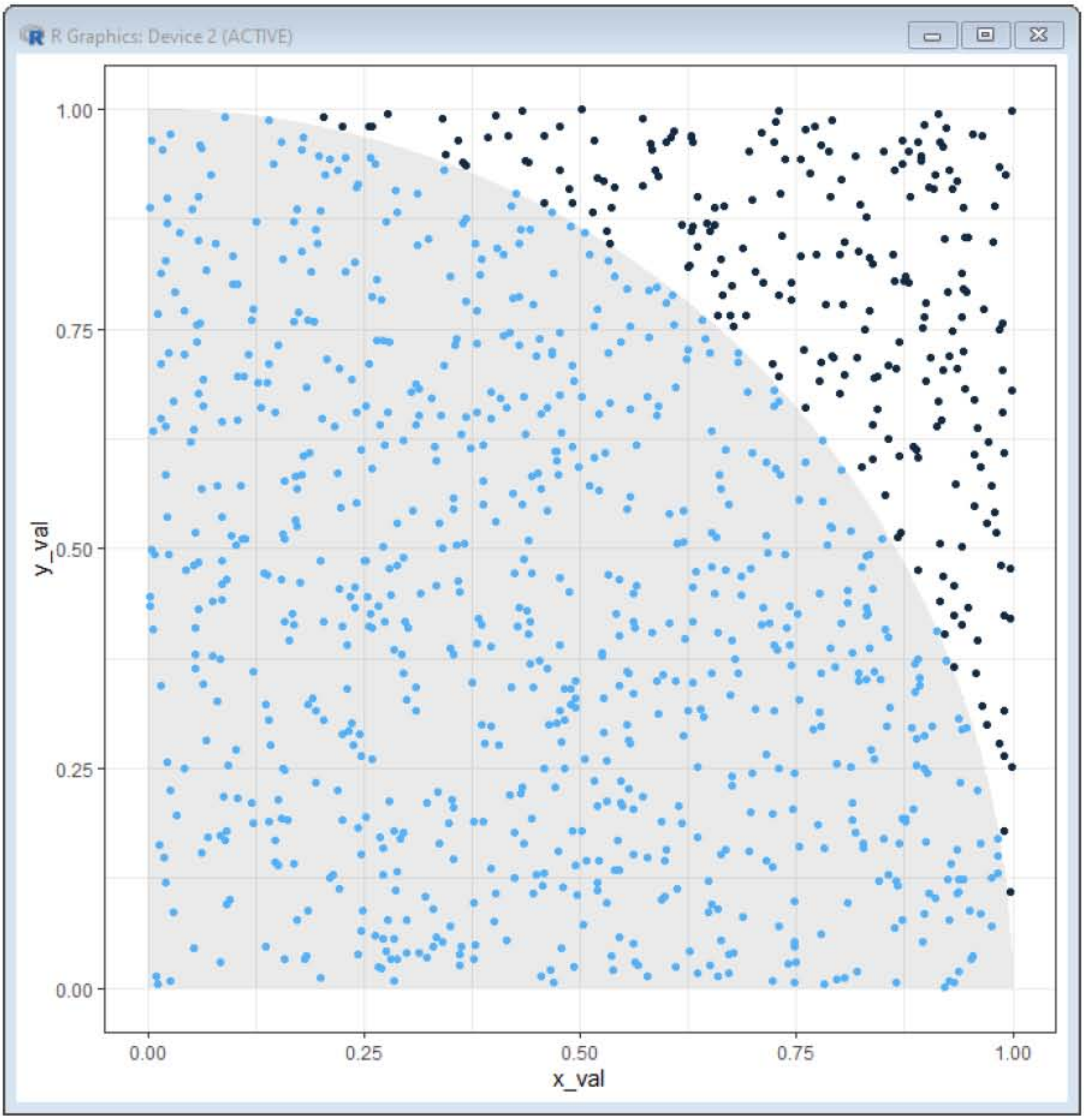
Nous allonrs reproduire ici un exemple scolaire fameux dans le domaine scolaire qui est l'approximation de Pi.

Le code est repris de :

https://berkorbay.github.io/fe522/02_Monte_Carlo_Simulation.html

```
> number_simulations<-1000
>
> #Create the points necessary to build a polygon
> circle_data<-data.frame(x_val=c(0,seq(0,1,length.out=1000)),
+                         y_val=c(0,sqrt(1-seq(0,1,length.out=1000)^2)))
>
> library(ggplot2) #Load ggplot2
> #create the shaded area for the quarter circle
> plot_pi <- ggplot() +
+ geom_polygon(data=circle_data,aes(x=x_val,y=y_val),alpha=0.1) + theme_bw()
> plot_pi
>
> #Create random points by declaring dot positions
> #with 2 random uniformly distributed values for x and y.
> dot_data<-data.frame(x_val=runif(1000),y_val=runif(1000))
> #Define in with 1 and out with 0.
> #Since circle radius is 1. We calculate in/out with the distance from origin.
> dot_data$in_or_out<-ifelse(sqrt(dot_data$x_val^2+dot_data$y_val^2)<=1,1,0)
>
> plot_pi + geom_point(data=dot_data,aes(x=x_val,y=y_val,color=in_or_out)) +
+ theme(legend.position="none")
>
> #Simulated value of pi
> 4*sum(dot_data$in_or_out)/nrow(dot_data)
[1] 3.092
>
```

Ce qui donne visuellement:

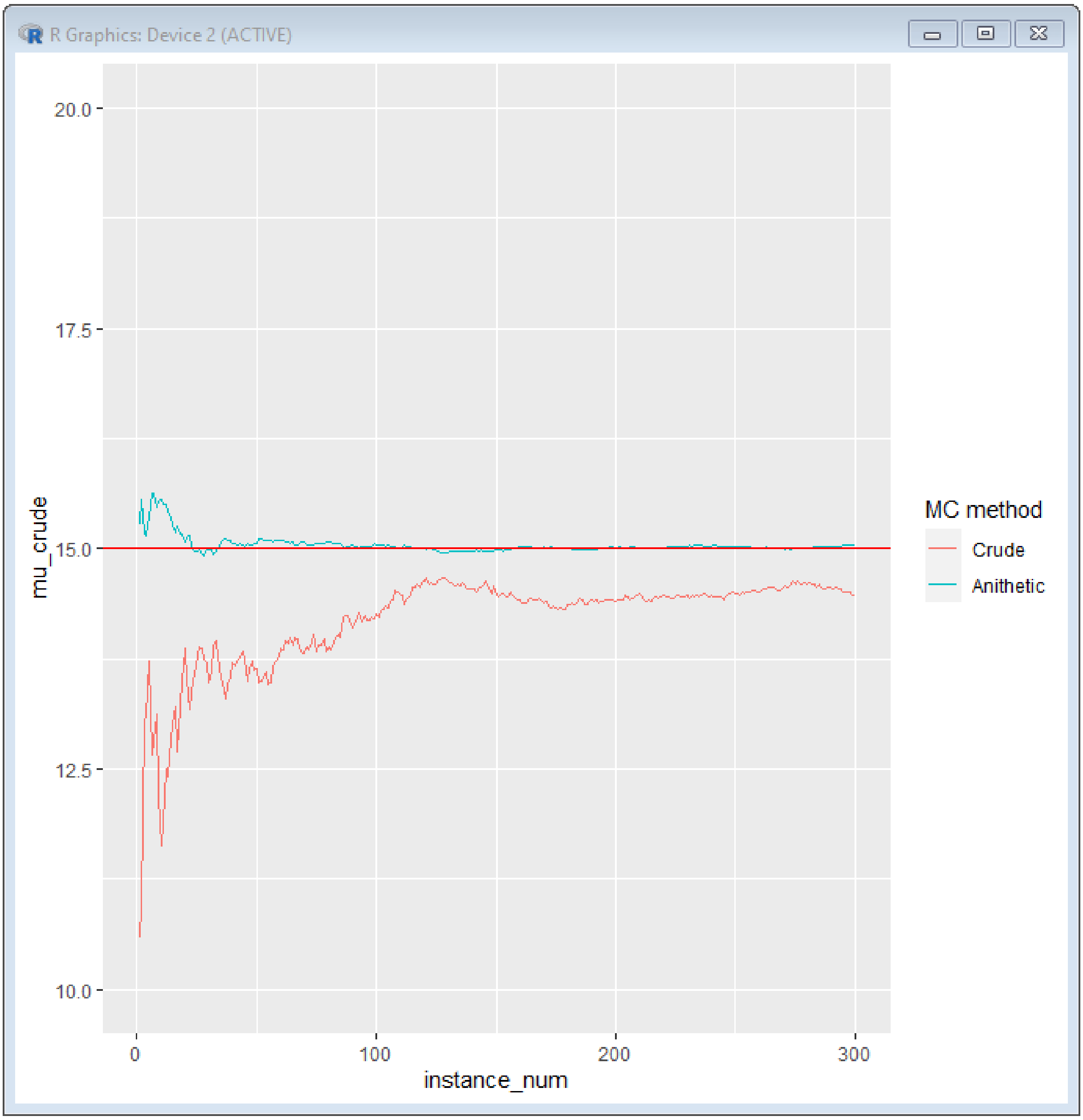

R Graphics: Device 2 (ACTIVE)
1.00
0.75
0.50
0.25
0.00
y_val
0.00
0.25
0.50
0.75
1.00
x_val

Méthode de réduction de la variance avec variable antithétique

Nous allons reproudire ici l'exemple à l'identique que celui fait dans le cours MATLAB (évidemment cela reste relié à ce que nous avons vu dans le cours théorique):

```
> nombre_simulations<-300
>
> #méthode brute
> x<-runif(nombre_simulations)
> y<-runif(nombre_simulations)
>
> Nx<-qnorm(x,mean=5,sd=3)
> Ny<-qnorm(y,mean=10,sd=4)
>
> mc_process_crude<-data.frame(instance_num=1,
+                              mu_crude=mean(Nx[1]+Ny[1]))
> for(i in 2:nombre_simulations){
+     mc_process_crude[i,]<-data.frame(instance_num=i,
+                                  mu=mean(Nx[1:i]+Ny[1:i]))
+ }
>
> #méthode antithétique
> #on ne redéfinit pas x car on reprendre celui généré plus haut!
> y<-1-x
> Ny<-qnorm(y,mean=10,sd=4)
> muNxy<-mean(Nx+Ny)
>
> mc_process_antithetic<-data.frame(instance_num=1,
+                              mu_antithetic=mean(Nx[1]+Ny[1]))
> for(i in 2:nombre_simulations){
+     mc_process_antithetic[i,]<-data.frame(instance_num=i,
+                                  mu_antithetic=mean(Nx[1:i]+Ny[1:i]))
+ }
>
> mc_process<-cbind(mc_process_crude,mu_antithetic<-mc_process_antithetic[,2])
>
> library("ggplot2")
> ggplot(data=mc_process) +
+ geom_line(aes(x=instance_num,y=mu_crude,color="blue")) +
+ geom_line(aes(x=instance_num,y=mu_antithetic,color="green")) +
+ geom_hline(yintercept=15,color="red") + ylim(c(10,20)) +
+ scale_color_discrete(name = "MC method", labels = c("Crude", "Anithetic"))
>
```

Ce qui donne:



Nous voyons donc bien que la méthode antithéatique converge plus rapidement (sans surprises!).

On peut faire une exemple plus orientié "finance" et donc business avec le calcul d'un prix d'un call européen en comparant les deux méthodes:

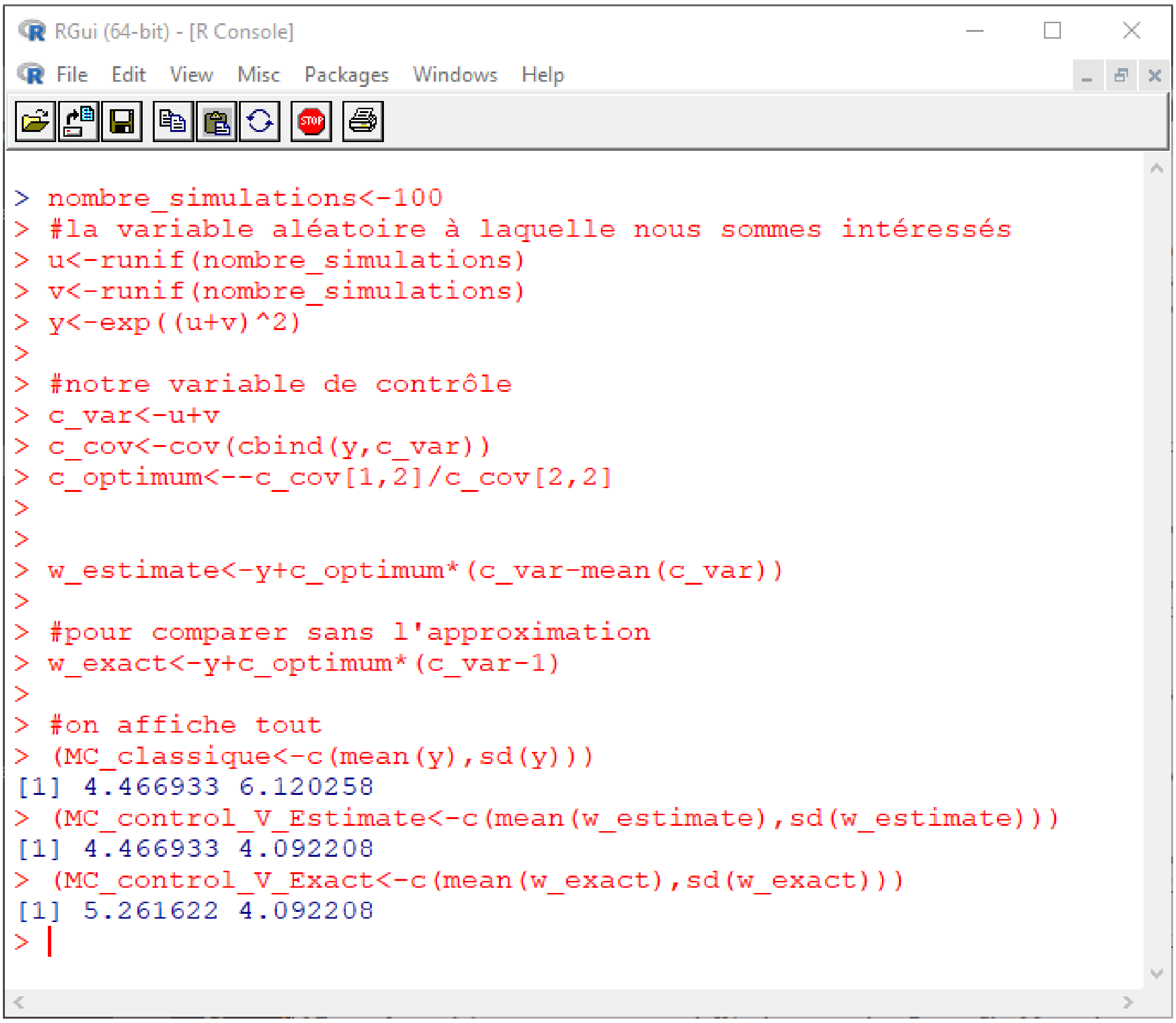
RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

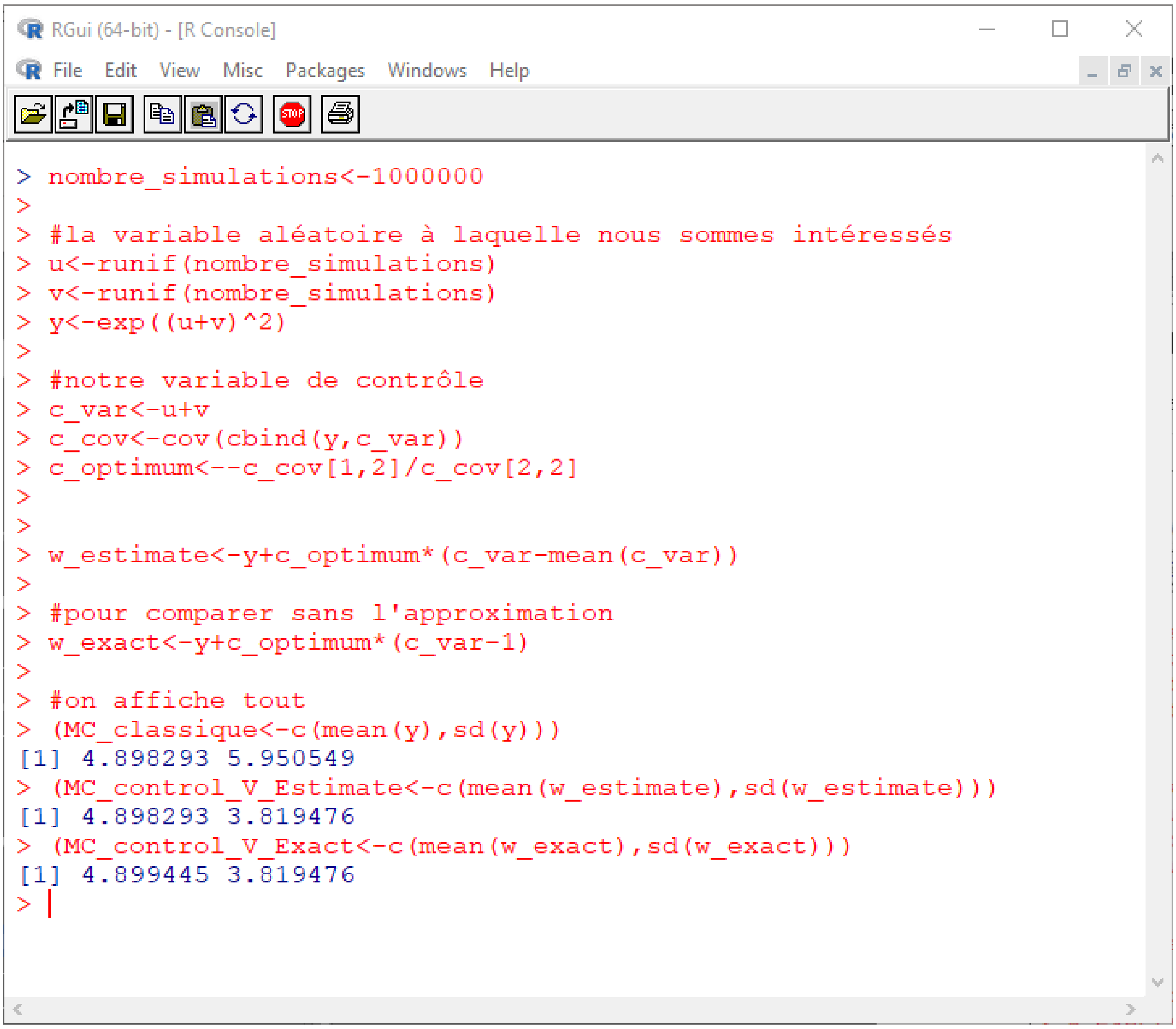
```
> #Simulation MC Call européen
> sim_european_call<-function(S_0=100,K=100,vol=0.25,T_years=1,r=0.02,n=10^4){
+     #Simulate the stock
+     sim_S_T<-S_0*exp((r-0.5*vol^2)*T_years + vol*rnorm(n)*sqrt(T))
+     #Calculate payoffs
+     payoffs<-pmax(sim_S_T-K,0)*exp(-r*T_years)
+     #Simulate results and bounds
+     Price<-mean(payoffs)
+     SE<-1.96*sd(payoffs)/sqrt(n)
+     LowerB <- Price - SE
+     UpperB <- Price + SE
+     return(c(Price=Price,SE=SE,Lower=LowerB,Upper=UpperB))
+ }
> #Simulation MC Call européen par variables antithétiques
> sim_european_call_av<-function(S_0=100,K=100,vol=0.25,T_years=1,r=0.02,n=10^4){
+     z_1 <- rnorm(n/2)
+     z_2 <- -z_1
+     #Simulate the payoffs with both processes
+     sim_payoff_1<-exp(-r*T)*pmax(S_0*exp((r-0.5*vol^2)*T_years + vol*z_1*sqrt(T))-K,0)
+     sim_payoff_2<-exp(-r*T)*pmax(S_0*exp((r-0.5*vol^2)*T_years + vol*z_2*sqrt(T))-K,0)
+     sim_payoff <- (sim_payoff_1 + sim_payoff_2)/2
+     #Calculate results and bounds
+     Price<-mean(sim_payoff)
+     SE<-1.96*sd(sim_payoff)/sqrt(n/2)
+     LowerB <- Price - SE
+     UpperB <- Price + SE
+     return(c(Price=Price,SE=SE,Lower=LowerB,Upper=UpperB))
+ }
> ##On compare
> #Méthode naive
> sim_european_call(n=10^4)
     Price         SE      Lower      Upper
11.1377908  0.3503062 10.7874846 11.4880969
> sim_european_call_av(n=10^4)
     Price         SE      Lower      Upper
10.9632675  0.2779489 10.6853186 11.2412164
>
> #comparaisona avec méthode exacte: 10.87056
> 
```

Méthode de réduction de la variance avec variables de contrôle

Nous allons reproduire ici l'exemple à l'identique que celui fait dans le cours MATLAB (évidemment cela reste relié à ce que nous avons vu dans le cours théorique):



et avec 1'000'000 de simulations:



Nous retombons bien sur des résultats similaires à ceux obtenus avec MATLAB.

Nous pouvons à nouveau faire un exemple plus orienté "business" avec encore une fois la simulation de Monte Carlo du prix d'un Call européen et le comparer avec les autres méthodes:

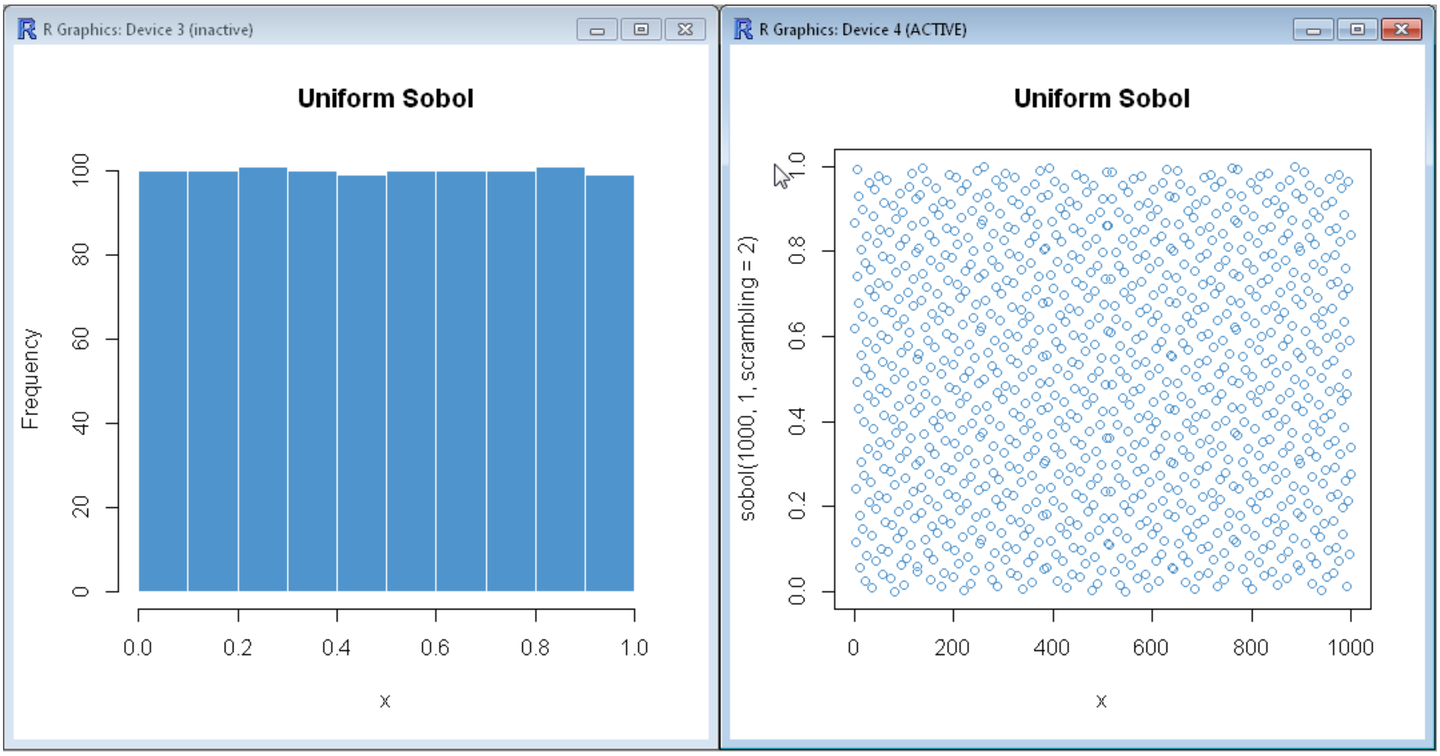
```
> sim_european_call_cv<-function(S_0=100,K=100,vol=0.25,T_years=1,r=0.02,n=10^4,with_naive=TRUE){
+     #Simulate S_T values
+     S_T_est <-S_0*exp((r-0.5*vol^2)*T_years + vol*rnorm(n)*sqrt(T))
+     #Simulate the payoffs with both processes
+     sim_payoff<-exp(-r*T)*pmax(S_T_est-K,0)
+     #Calculate theta_star
+     theta_star<-cov(S_T_est,sim_payoff)/var(S_T_est)
+     #Calculate CV effect
+     payoff_cv <- sim_payoff - theta_star*(S_T_est - S_0*exp(r*T_years))
+     #Calculate the output of naive simulation as well
+     if(with_naive){
+         Price<-mean(sim_payoff)
+         SE<-1.96*sd(sim_payoff)/sqrt(n)
+         LowerB <- Price - SE
+         UpperB <- Price + SE
+         print(c(Price_naive=Price,SE_naive=SE,Lower=LowerB,Upper=UpperB))
+     }
+     Price<-mean(payoff_cv)
+     SE<-1.96*sd(payoff_cv)/sqrt(n)
+     LowerB <- Price - SE
+     UpperB <- Price + SE
+     return(c(Price_CV=Price,SE_CV=SE,Lower=LowerB,Upper=UpperB))
+ }
> sim_european_call_av(n=10^4)
     Price         SE      Lower      Upper 
10.8727955  0.2736292 10.5991663 11.1464247 
> sim_european_call_cv(n=10^4)
Price_naive    SE_naive       Lower       Upper 
 11.0223709   0.3492367  10.6731341  11.3716076 
  Price_CV      SE_CV      Lower      Upper 
10.7706475  0.1435762 10.6270714 10.9142237 
> 
```

Méthode de réduction de la variance avec séquence quasi-aléatoire de Sobol

Nous avons mentionné dans le cours théorique que ces nombres pseudo-aléatoires dont l'origine est empirique permet souvent une meilleure convergence des calculs et nous en avions fait l'expérience avec du VBA dans Microsoft Excel avec par exemple les nombres quasi-aléatoires de Fauré. Nous allons voir ici un autre type empirique de nombre quasi-aléatoire que sont les séquences de Sobol en utilisant la fonction **sobol( )** du package **randtoolbox**:

R Console
```
> library(randtoolbox)
> hist(sobol(1000,1,scrambling=2),main="Uniform Sobol",xlab="x",col="steelblue3",border="white")
> windows()
> plot(sobol(1000,1,scrambling=2),main="Uniform Sobol",xlab="x",col="steelblue3")
>
```

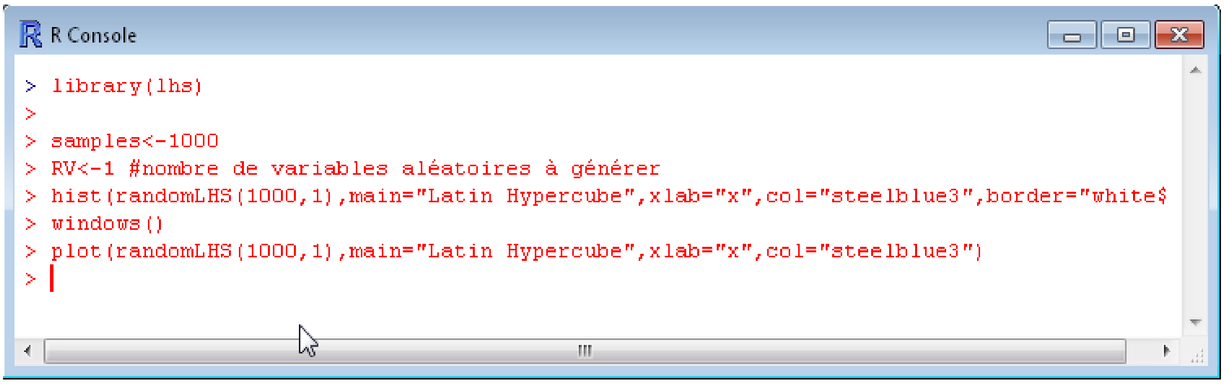
Ce qui donne visuellement:



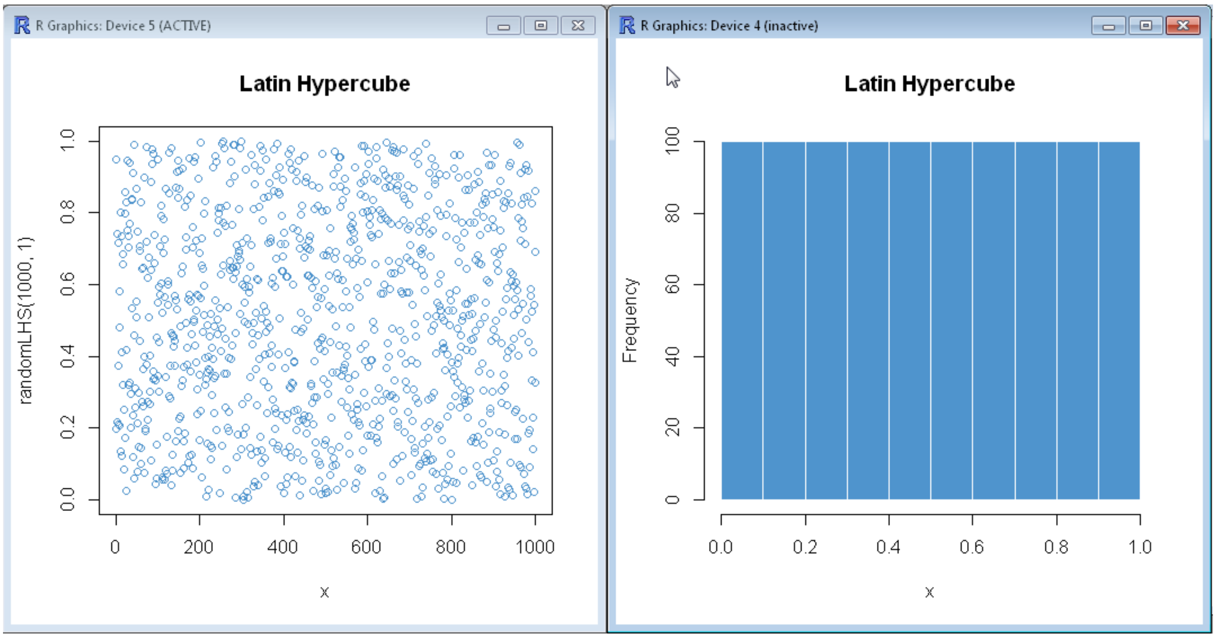
Méthode de réduction de la variance avec Latin Hypercube

Nous avons aussi étudié théoriquement le principe mathématique empirique des jeux de variables aléatoires Latin Hypercube qui n'ont normalement qu'un intérêt qu'à partir d'un nombre de variables aléatoires supérieur 1. Évidemment, au même titre que la séquence quasi-aléatoire de Sobol il ne sera pas possible de vérifier la conformité avec la génération fait dans Microsoft Excel mais voyons le principe puisque dans la pratique l'enjeu est important (des logiciels comme @Risk utilisant ce type de simulation).

Nous allons pour cela utiliser la fonction **randomLHS( )** du package **lhs**:

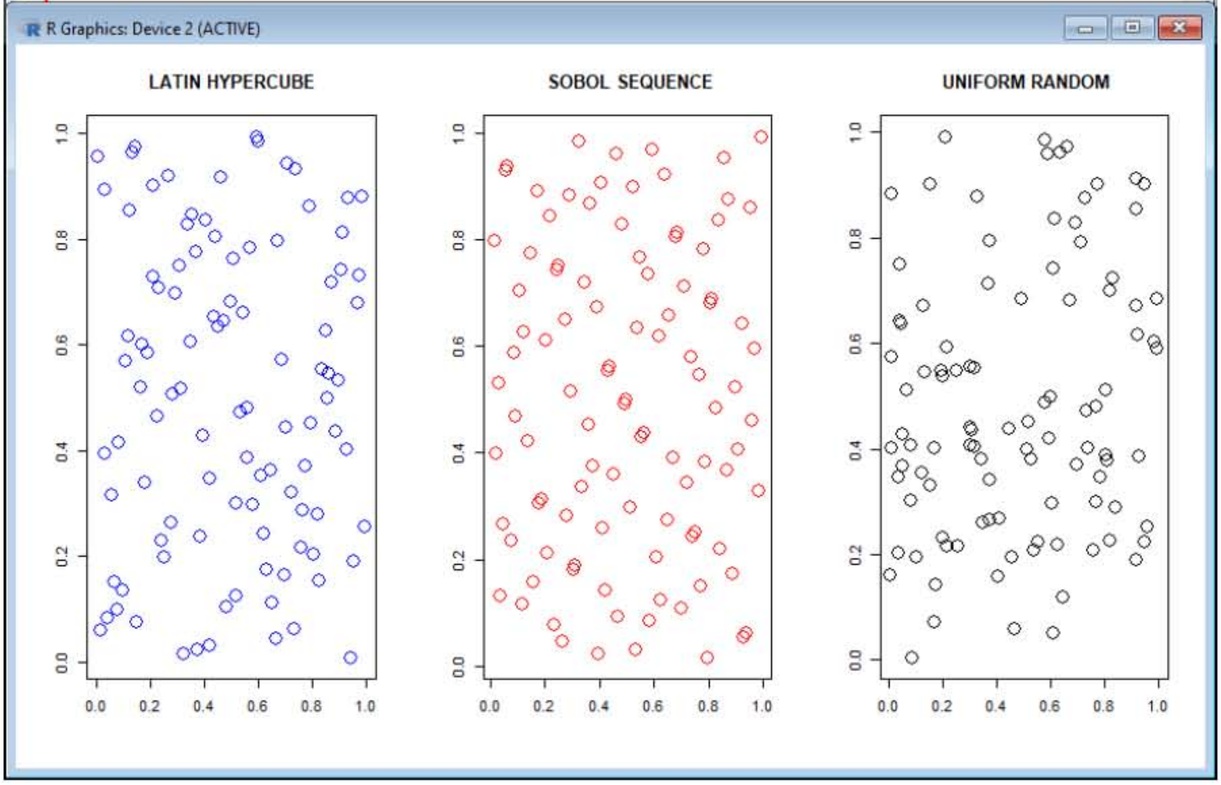


Ce qui donne:



Pour le fun nous pouvons comparer quelques méthodes:

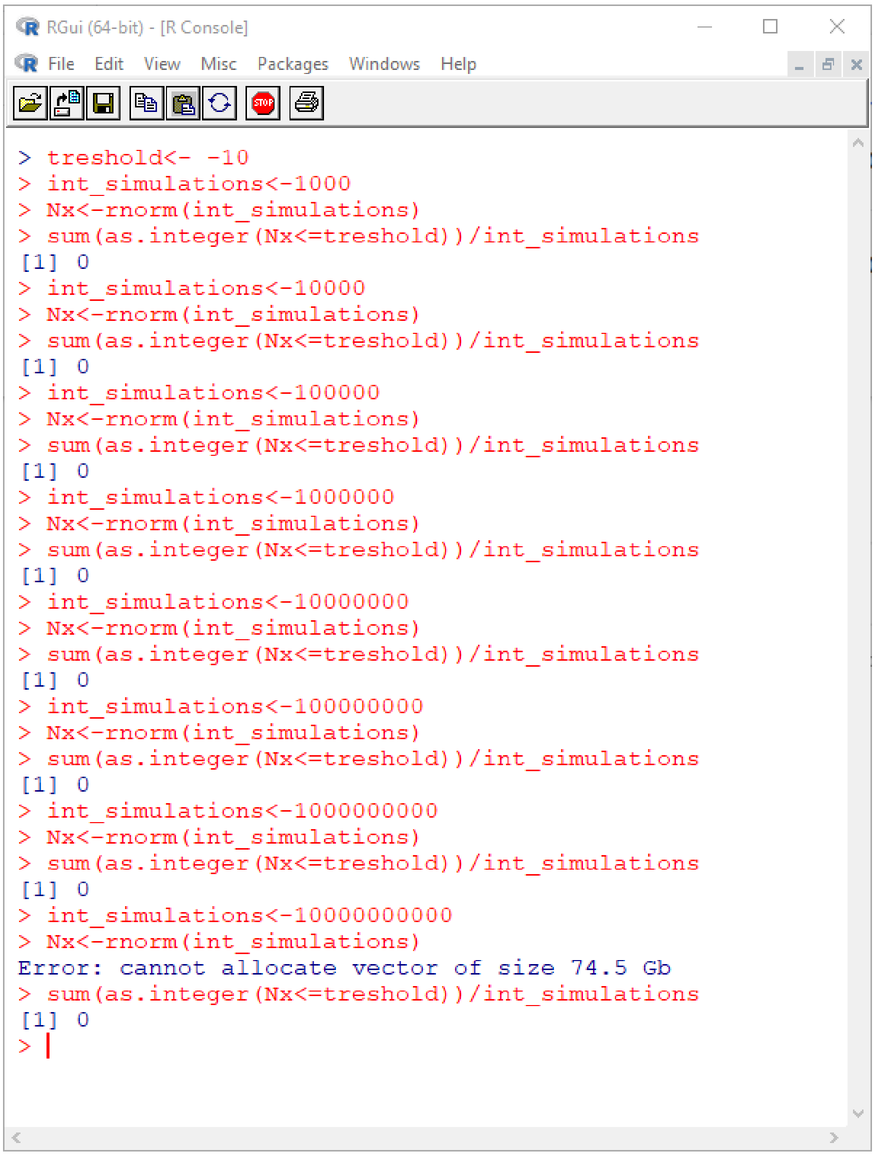
```
> unifm_2d <- function(n, seed) {
+   set.seed(seed)
+   return(replicate(2, runif(n)))
+ }
>
> sobol_2d <- function(n, seed) {
+   return(randtoolbox::sobol(n, dim = 2, scrambling = 3, seed = seed))
+ }
>
> latin_2d <- function(n, seed) {
+   set.seed(seed)
+   return(lhs::randomLHS(n, k = 2))
+ }
>
> par(mfrow = c(1, 3))
> plot(latin_2d(100, 2019), main = "LATIN HYPERCUBE", xlab = '', ylab = '', cex = 2, col = "blue")
> plot(sobol_2d(100, 2019), main = " SOBOL SEQUENCE", xlab = '', ylab = '', cex = 2, col = "red")
Warning message:
In randtoolbox::sobol(n, dim = 2, scrambling = 3, seed = seed) :
  scrambling is currently disabled.
> plot(unifm_2d(100, 2019), main = " UNIFORM RANDOM", xlab = '', ylab = '', cex = 2, col = "black")
>
```



Méthode de réduction de la variance par échantillonnage préférentiel (importance sampling)

Nous allons ici reprendre le même exemple que dans le cours théorique et que celui dans le cours MATLAB.

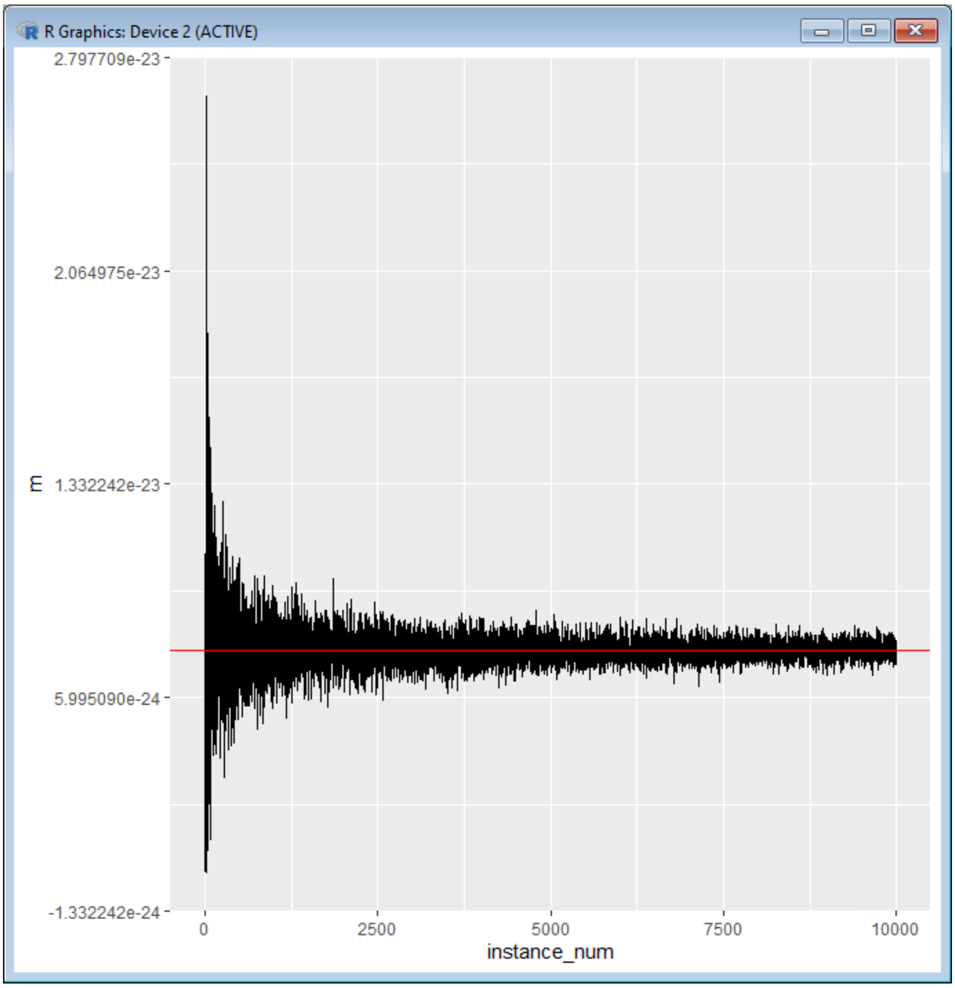
D'abord nous montrons le problème:



Et la solution y relative utilisant l'échantillonnage préférentiel (importance sampling):

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

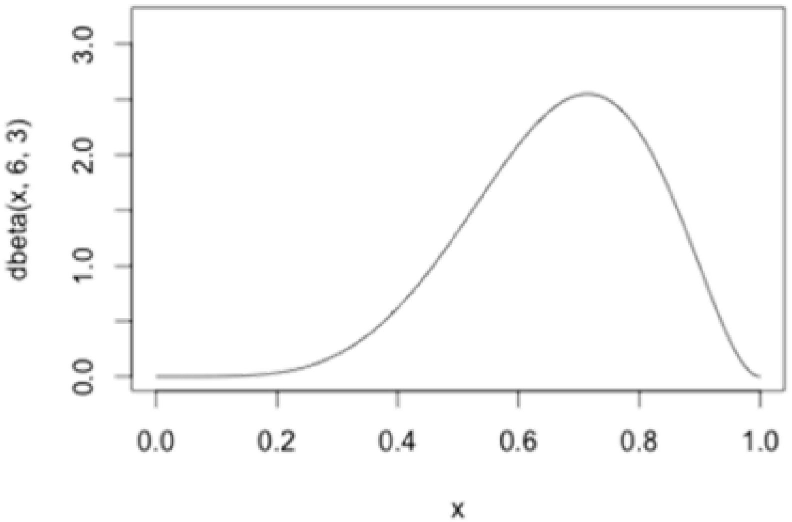
> int_simulations<-10000
> treshold<- -10
>
> Y<-rnorm(1)+treshold
> h<-as.integer(Y<=treshold)
> mc_process_echant_prefer<-data.frame(instance_num=1,
+                         m=mean(h[1]*exp((-treshold)*Y+0.5*(treshold)^2)))
>
> for(k in 2:int_simulations){
+     Y<-rnorm(k)+treshold
+     h<-as.integer(Y<=treshold)
+     mc_process_echant_prefer[k,]<-data.frame(instance_num=k,
+                             m=mean(h*exp((-treshold)*Y+0.5*(treshold)^2)))
+ }
>
> mean(h*exp((-treshold)*Y+0.5*(treshold)^2))
[1] 7.552098e-24
>
> library("ggplot2")
> ggplot(data=mc_process_echant_prefer) +
+ geom_line(aes(x=instance_num,y=m))+
+ geom_hline(yintercept=7.6199E-24,color="red")
> 
```


R Graphics: Device 2 (ACTIVE)
2.797709e-23
2.064975e-23
1.332242e-23
5.995090e-24
-1.332242e-24
m
0
2500
5000
7500
10000
instance_num

Méthode d'acceptation/rejet

Comme nous l'avons vu dans le cours théorique il s'agit d'une méthode dont l'algorithme permet de générer des variables aléatoires d'une fonction de densité difficilement inversible est très simple à mettre en place.

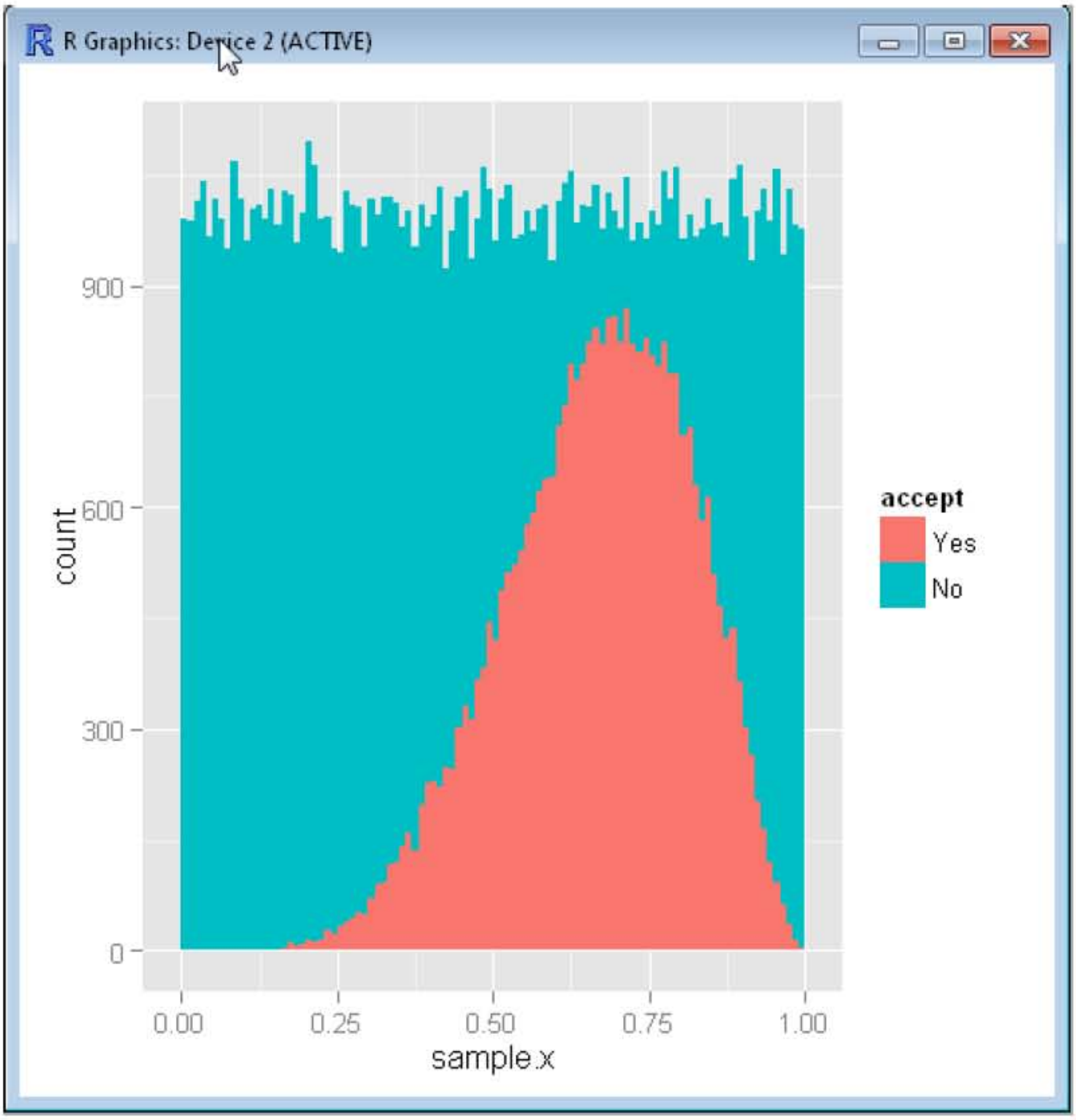
L'idée très académique mais néanmoins plus réaliste qu'avec l'exemple pris dans la formation MATLAB est de générer un vecteur de variables aléatoires d'une distribution beta de paramètres 6 et 3:



Nous avons alors trivialement:

```
R Console
> sample.x = runif(100000,0,1)
> accept = c()
>
> for(i in 1:length(sample.x)){
+  U = runif(1, 0, 1)
+  if(dunif(sample.x[i], 0, 1)*3*U <= dbeta(sample.x[i], 6, 3)) {
+    accept[i] = 'Yes'
+  }
+  else if(dunif(sample.x[i],0,1)*3*U > dbeta(sample.x[i], 6, 3)) {
+   accept[i] = 'No'
+  }
+ }
>
> T = data.frame(sample.x, accept = factor(accept, levels= c('Yes','No')))
>
> library(ggplot2)
> print(qplot(sample.x,data=T,geom='histogram',fill=accept,binwidth=0.01))
Message d'avis :
position_stack requires constant width: output may be incorrect
>
```

Ce qui donne visuellement:



Méthode de Monte-Carlo par chaînes de Markov

Comme déjà vu dans le cours théorique, les méthodes de Monte-Carlo par chaînes de Markov, ou méthodes MCMC pour Markov chain Monte Carlo en anglais, sont une classe de méthodes d'échantillonnage à partir de distributions de probabilité. Ces méthodes de Monte-Carlo se basent sur le parcours de chaînes de Markov qui ont pour lois stationnaires les distributions à échantillonner.

Certaines méthodes utilisent des marches aléatoires sur les chaînes de Markov (algorithme de Metropolis, de Metropolis-Hastings, échantillonnage de Gibbs), alors que d'autres algorithmes, plus complexes, introduisent des contraintes sur les parcours pour essayer d'accélérer la convergence (Monte Carlo Hybride, Surrelaxation successive).

Méthode Chaîne de Markov continue

Échantillonnage de Gibbs

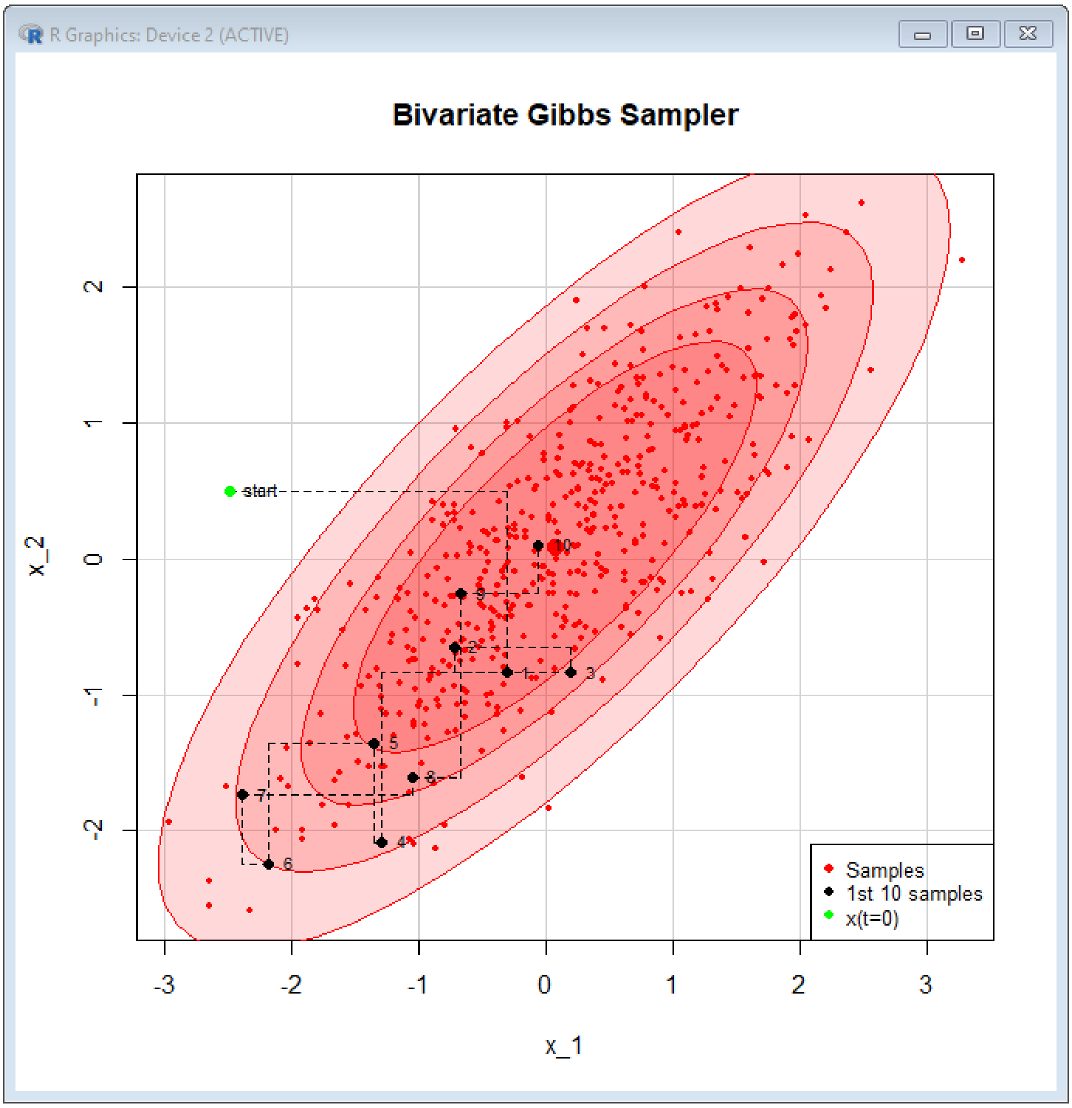
Comme nous l'avons vu dans le cours théorique, l'échantillonnage de Gibbs est une **C**haîne de **M**arkov de **M**onte **C**arlo (MCMC) qui permet de générer des variables aléatoires pour des fonctions multivariées dont les distributions conditionnelles sont connues analytiquement. Nous avons également vu que l'algorithme converge typiquement vers la distribution conjointe uniquement si la fonction de distribution est unimodale ou n'a pas de forts pics (hautement leptokurtique).

Nous allors d'abord reproduire ici le même exemple académique que celui étudié dans le cours théorique tiré du joli exemple de Dustin Stansbury:

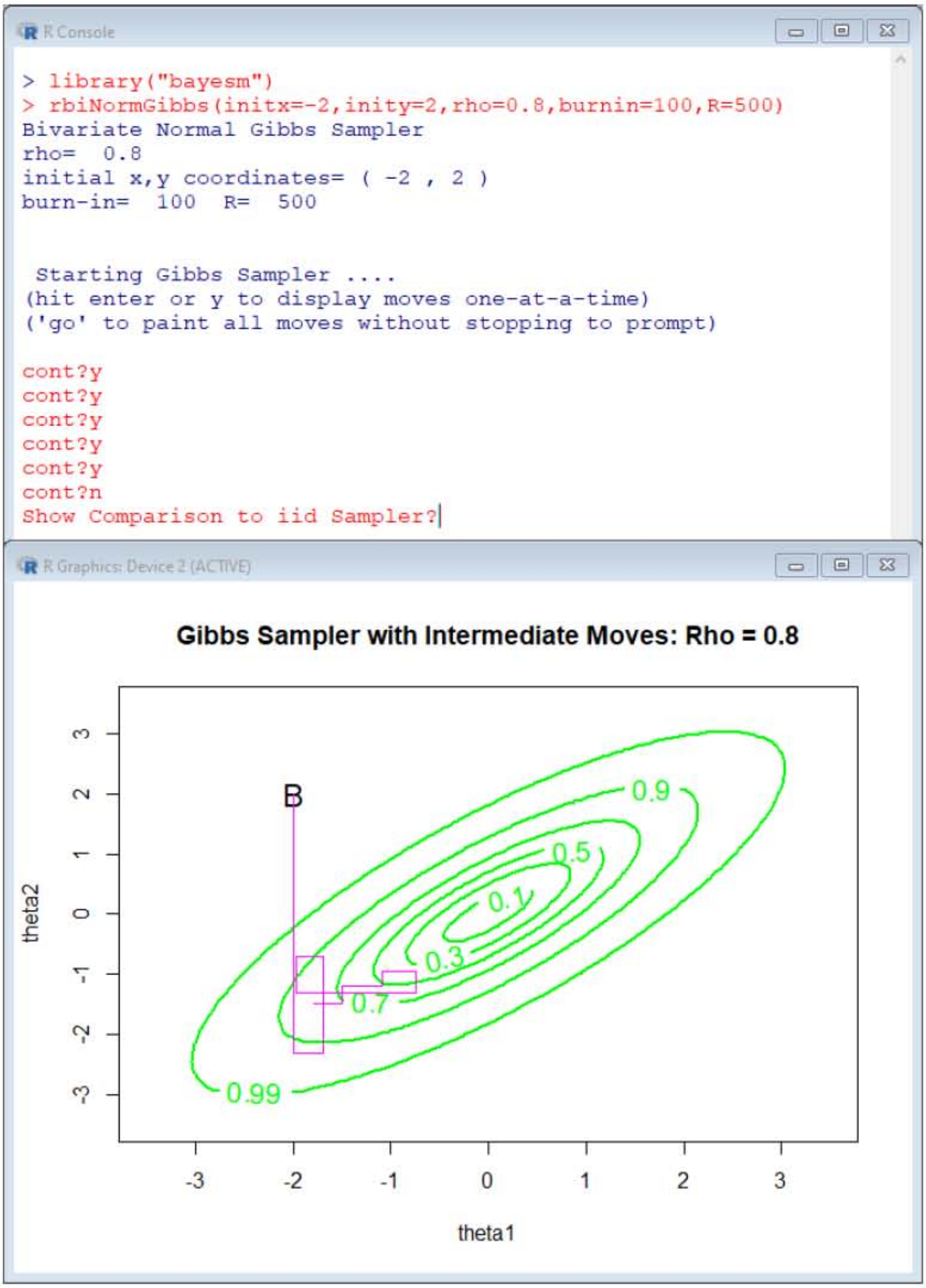
https://theclevermachine.wordpress.com/2012/11/05/mcmc-the-gibbs-sampler/

```
> library("car") #pour les ellipses de confiance plus tard
>
> Nsamples<-500
> mu<-c(0,0) #moyennes cibles des fonctions Normales
> rho<-c(0.8,0.8) #rho_21 rho_12
>
> #initialisation de l'échantillon de Gibbs
> propSigma<-1
> minn<-c(-3,-3);maxx<-c(+3,+3)
>
> #on initialise les échantillons
> #x<-phonTools::zeros(Nsamples, 2)
> x<-matrix( ncol=2, rep( 0, len=2*Nsamples))
> x[1,1]<-runif(1, min = minn[1], max = maxx[1])
> x[1,2]<-runif(1, min = minn[2], max = maxx[2])
>
> dims<- 1:2 #index dans chaque dimesion
>
> #on exécute l'échantillonnage de Gibbs
> t<-1
> while (t < Nsamples) {
+    t<-t + 1
+    T<-c(t-1,t)
+    for (iD in 1:2) { #on boucle sur les dimensions
+      #on met à jour les échantillons
+      nIx<-(dims!=iD)
+      #moyenne conditionnelle
+      muCond <- mu[iD] + rho[iD]*(x[T[iD],nIx]-mu[nIx]);
+      #variance conditionnelle
+      varCond <- sqrt(1-rho[iD]^2)
+      x[t,iD] <-rnorm(1, mean=muCond, sd=varCond)
+    }
+ }
>
> #on affiche le graph
> stepsToDisplay<-10
> car::dataEllipse(x[,1], x[,2],xlab = "x_1", ylab = "x_2",col="red",
+ pch=19,cex = 0.5,levels=c(0.70,0.85,0.95,0.99),fill=TRUE,
+ fill.alpha=0.15, lty=1, lwd=1,main="Bivariate Gibbs Sampler")
>
> for(t in 1:stepsToDisplay){
+ lines(c(x[t,1],x[t+1,1]),c(x[t,2],x[t,2]), lty=2,cex = 1)
+ lines(c(x[t+1,1],x[t+1,1]),c(x[t,2],x[t+1,2]), lty=2,cex = 1)
+ points(x[t+1,1],x[t+1,2], pch=16, col="black", lty=2,cex = 1)
+ text(x[t+1,1], x[t+1,2], labels=t, cex= 0.7, pos=4)
+ }
> points(x[1,1],x[1,2], pch=16, col="green", lty=2,cex = 1)
> text(x[1,1], x[1,2], labels="start", cex= 0.7, pos=4)
>
> legend("bottomright", legend=c("Samples", paste("1st",stepsToDisplay,"samples"),"x(t=0)"),
+        col=c("red", "black","green"), pch = c(16,16,16), cex=0.8)
```

Ce qui donne typiquement:



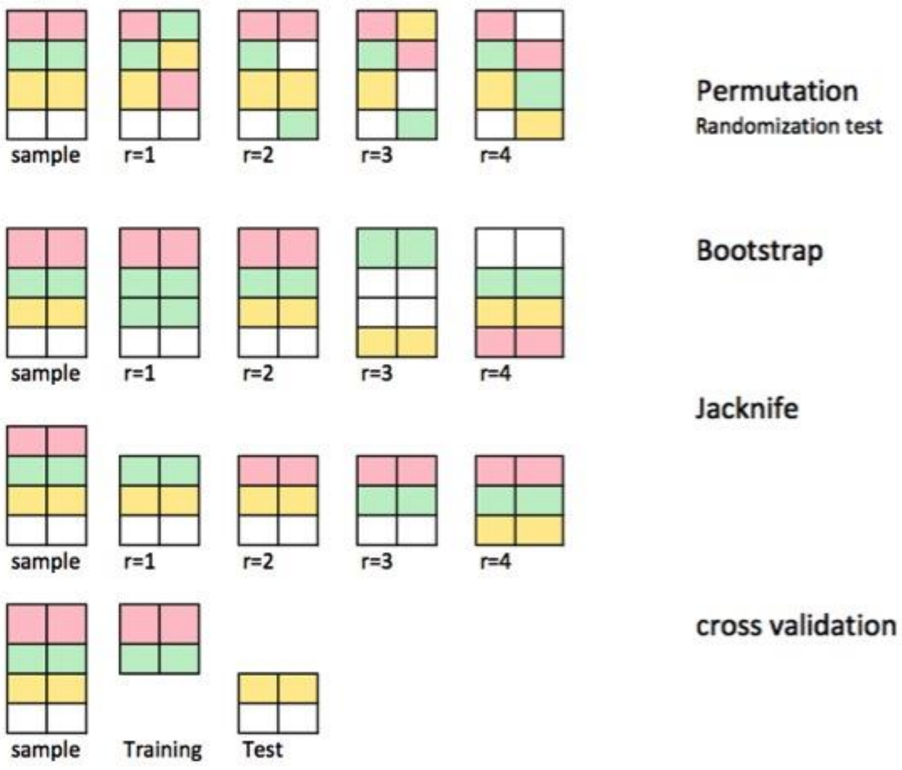
Il existe même un package qui implémente une version animée et donc pédagogique du l'échantillonneur de Gibbs pour deux lois Normales centrées réduites. Il s'agit du package **bayesm** et de sa fonction **rbiNormGibb( )**:


R Console
> library("bayesm")
> rbiNormGibbs(initx=-2,inity=2,rho=0.8,burnin=100,R=500)
Bivariate Normal Gibbs Sampler
rho= 0.8
initial x,y coordinates= (-2 , 2)
burn-in= 100 R= 500
Starting Gibbs Sampler ....
(hit enter or y to display moves one-at-a-time)
('go' to paint all moves without stopping to prompt)
cont?y
cont?y
cont?y
cont?y
cont?y
cont?n
Show Comparison to iid Sampler?
R Graphics: Device 2 (ACTIVE)
Gibbs Sampler with Intermediate Moves: Rho = 0.8
B
0.9
0.5
0.1
0.3
0.7
0.99
theta2
theta1
-3
-2
-1
0
1
2
3

Exemples 315.: Test de permutation

R 3.4.2

Nous avons vu dans le cours théorique la figure suivante (le dernier cas est plutôt pour le machine learning que nous traiterons plus dans le chapitre correspondant!):



Nous allons donc aborder dans ce qui suit et dans l'ordre un exemple de test de permutation, des tests de Bootstrap et des tests jacknifes!

Donc rappelons que le test de permutation s'applique pour les types de test ou l'hypothèse nulle est qu'il n'y a pas de différence entre les jeux de données à comparer (condition sous laquelle on peut les mélanger pour continuer avec la même hypothèse).

```
> #Heterozygous (BA)
> a<-c(86,88,89,89,92,93,94,94,94,95,95,96,96,97,97,98,98,99,99,101,106,107,110,113,116,118)
>
> #Homozygous (BB)
> b<-c(89,90,92,93,93,96,99,99,99,102,103,104,105,106,106,107,108,108,110,110,112,114,116,116)
>
> #Fusion des deux jeux de données puisque sous l'hypothèse nulle
> #il n'y pas de différence entre les deux groupes
> combined<-c(a,b)
>
> #Différence observée initiale
> diff.observed<-mean(b) - mean(a)
>
> number_of_permutations<-1000
>
> diff.random<-NULL
> for (i in 1 : number_of_permutations) {
+
+     # Sample from the combined dataset without replacement
+     shuffled<-sample (combined, length(combined))
+
+     a.random<-shuffled[1 : length(a)]
+     b.random<-shuffled[(length(a) + 1) : length(combined)]
+
+     # Null (permuated) difference
+     diff.random[i]<-mean(b.random) - mean(a.random)
+ }
>
> #La p-value est la fraction du nombre de fois les différences permutées sont égales
> #ou plus grandes que la différence observées
> pvalue<- sum(abs(diff.random) >= abs(diff.observed))/number_of_permutations
> pvalue
[1] 0.04
>
```

Exemples 316.: Bootstrap

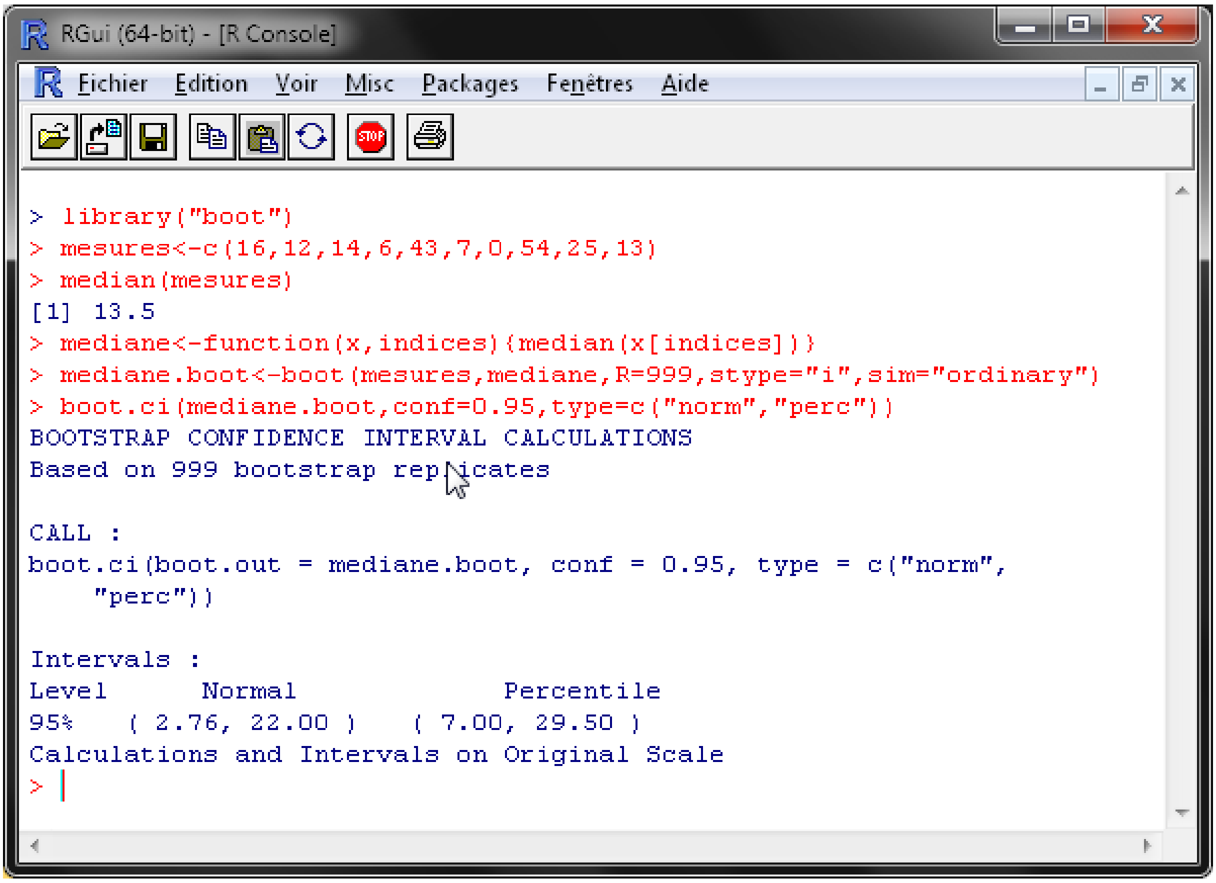
R 3.0.2

Nous allons voir ici une des techniques de statistiques dites par "permutations" que nous avons traité dans le cours théorique. Et comme à l'habitude… par difficulté croissante!

Bootstrapping univarié

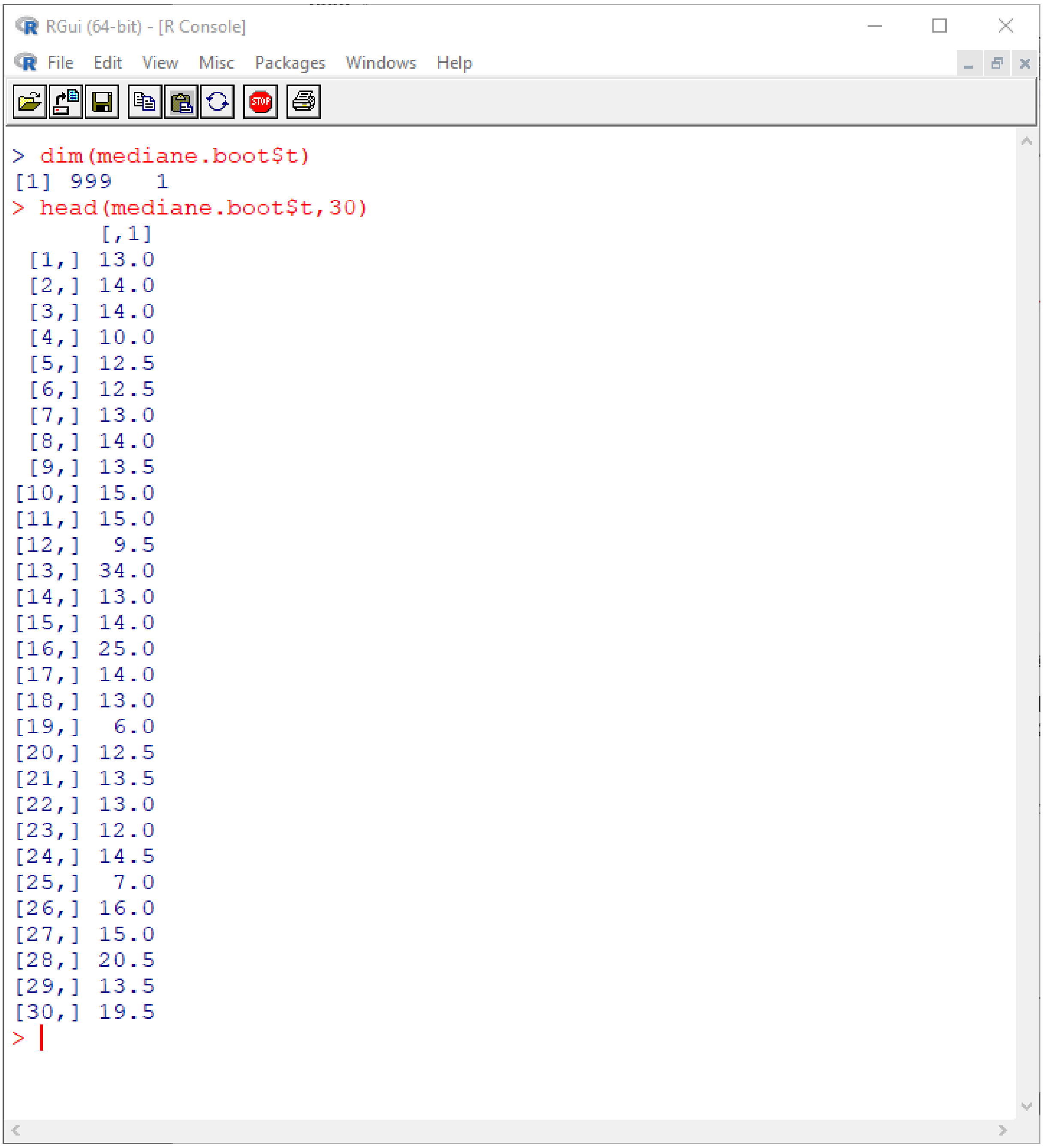
Nous allons ici mettre en oeuvre avec une facilité extrême l'exemple d'inférence par méthode brute que nous avions mis en place avec MS Excel.

Nous partons du même jeu de données et allons aussi faire de l'inférence sur la médiane en utilisant les commandes **boot( )** et **boot.ci( )** du package **boot**:

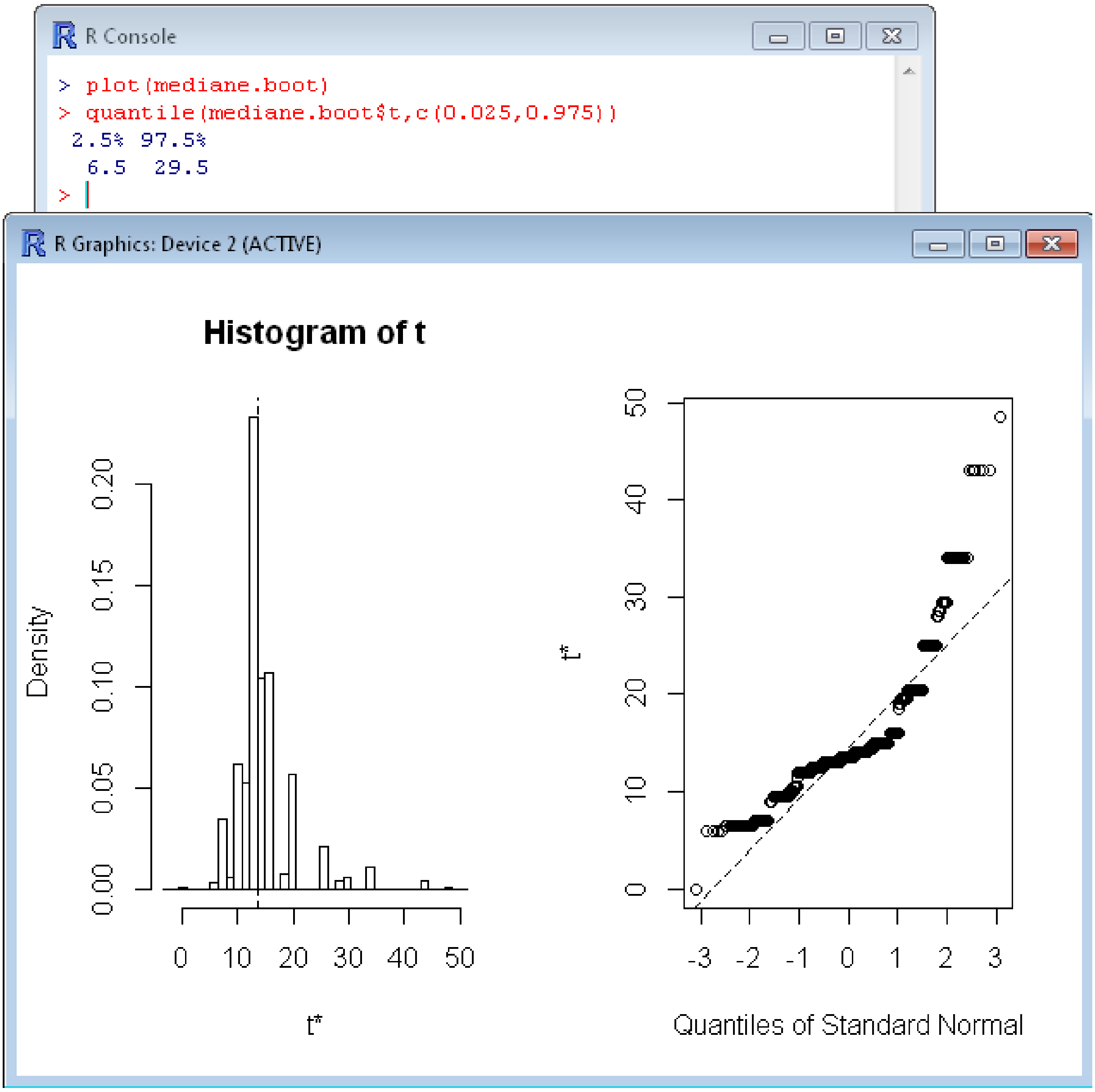


Donc comme nous pouvons le voir pour la partie percentile que nous avions calculé avec Microsoft Excel nous ne sommes pas loin puisque nous avions obtenu (7.00, 29.5).... mais pas sans peine!

Nous pouvons facilement accéder au valeurs bootstrappés de la variable mediane.boot. Pour cela il suffit d'écrire:



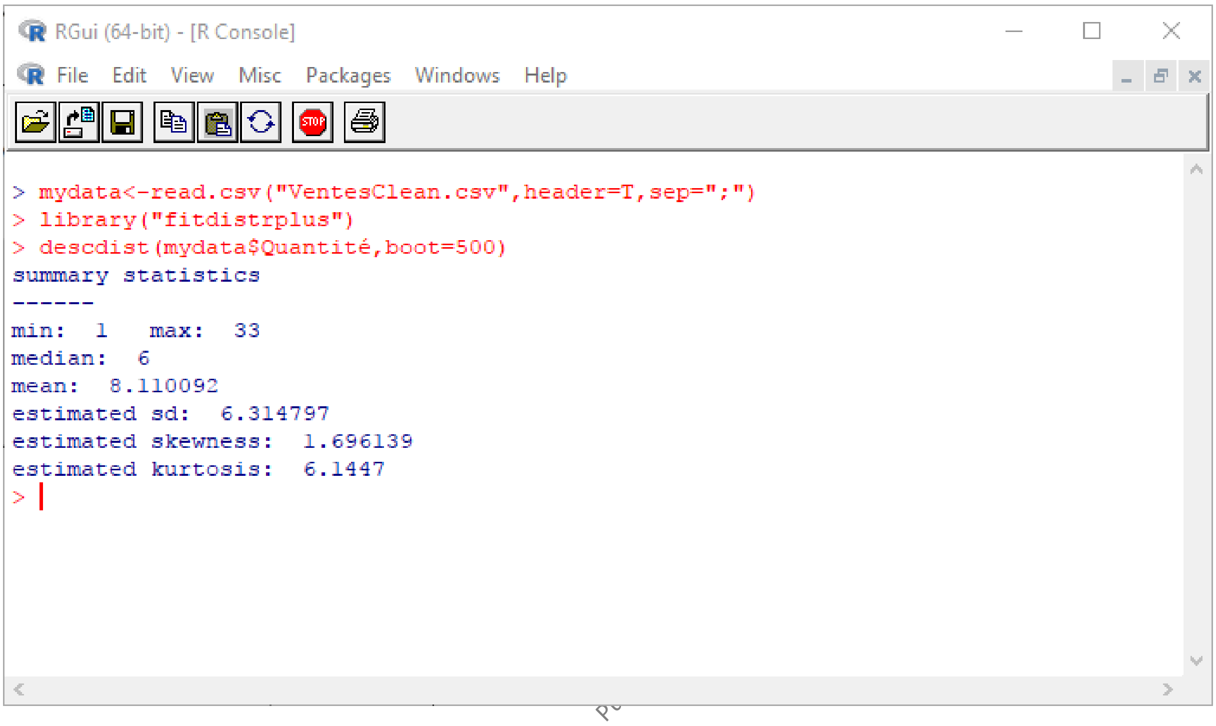
Nous pouvons avoir un petit complément graphique:



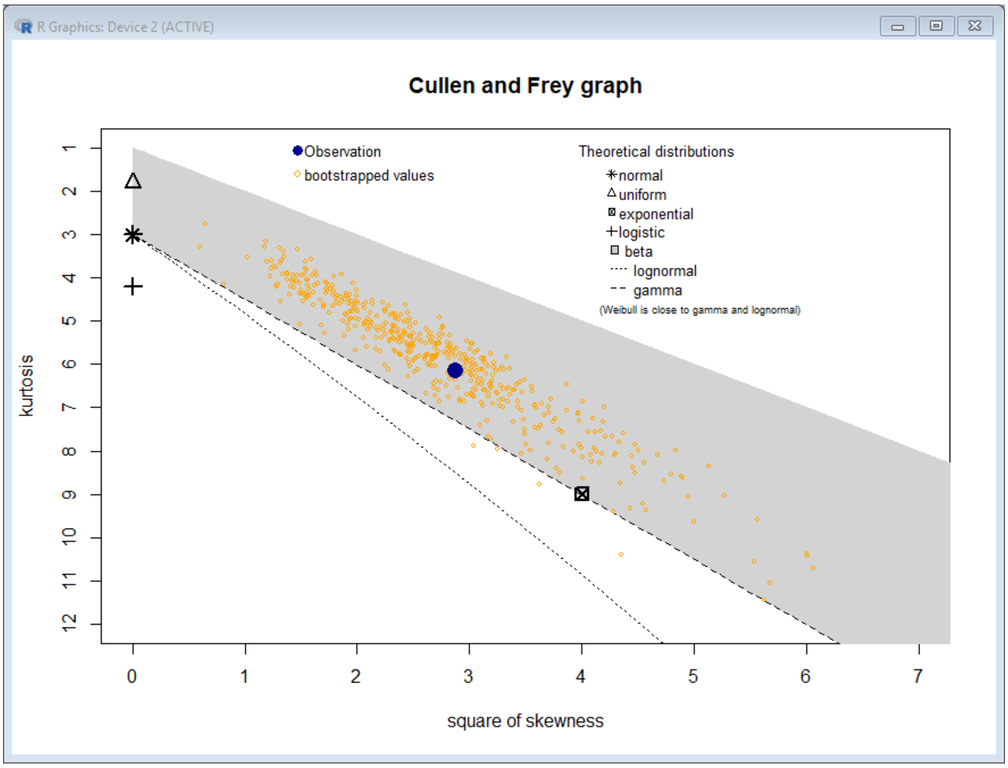
Et ainsi de suite pour toute autre statistique possible et imaginable!

Bootstrap Skewness et Kurtosis

Il est bien évidemment aussi possible de faire un aussi une analyse bootstrap du kurtosis et skewness. Pour cela, pour nous simplifier la vie et avoir une certaine esthétique prémâchée, nous pouvons faire usage de la fonction **descdist( )** déjà vue plus haut mais cette fois-ci avec un paramètre *boot* comme illustré ci-dessous:



Ce qui donne visuellement:



Bootstrapping multivarié

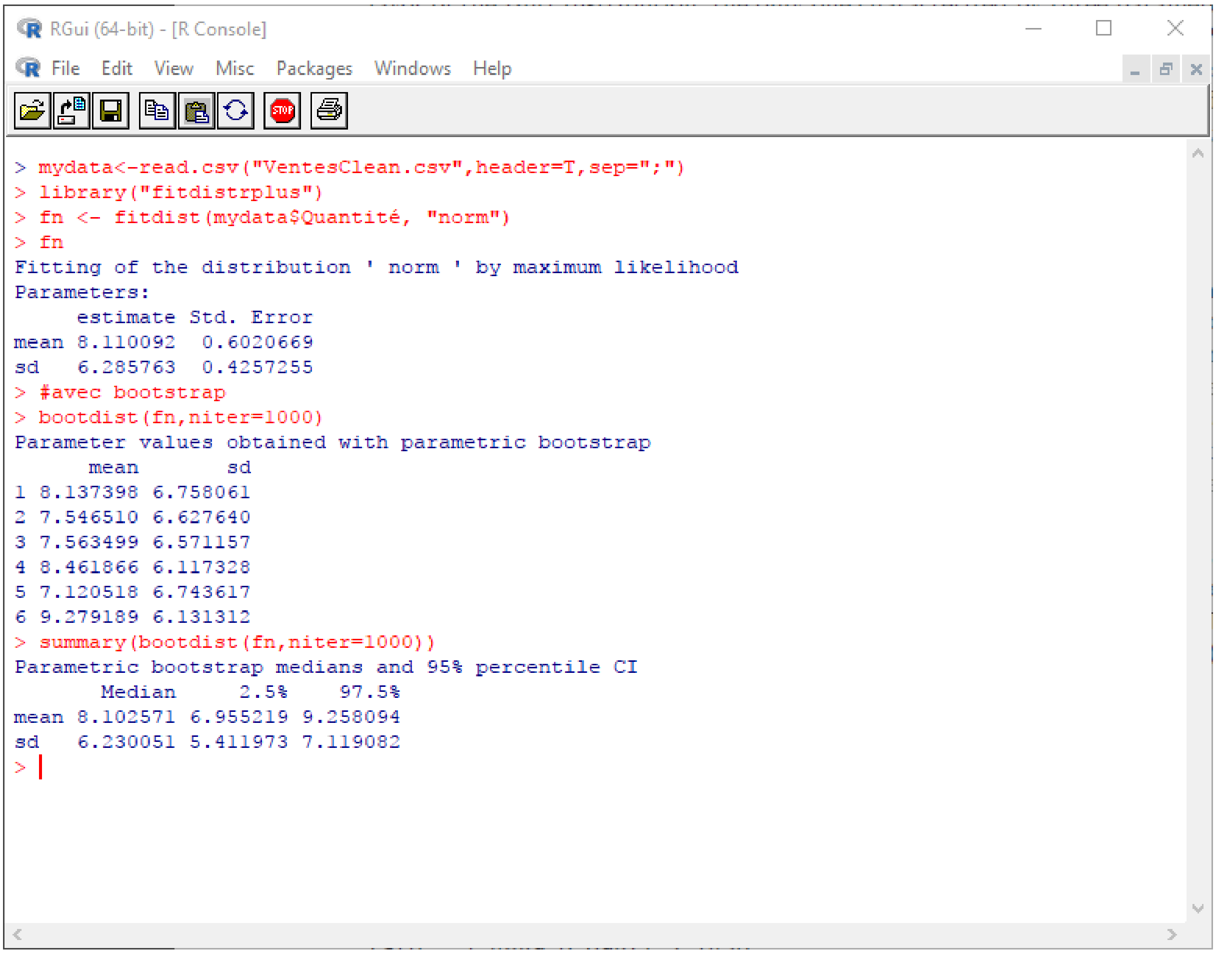
Le but ici va être de reprendre l'exemple que nous avons fait lors de notre étude plus haut d'un test de permutation:

```
> # Heterozygous (BA)
> a = c(86, 88, 89, 89, 92, 93, 94, 94, 94, 95, 95, 96, 96, 97, 97, 98, 98, 99, 99, 101, 106, 107, 110, 113, 116, 118)
>
> # Homozygous (BB)
> b = c(89, 90, 92, 93, 93, 96, 99, 99, 99, 102, 103, 104, 105, 106, 106, 107, 108, 108, 110, 110, 112, 114, 116, 116)
>
> # Différence des moyennes entre les deux jeus de données
> diff.observed = mean(b) - mean(a)
>
> # Niveau de confidence
> alpha = 0.05
>
> # Nombre de réplique
> n = 1000
>
> # On initialise la variable (vecteur) de différences (n réplicats)
> diff.bootstrap = NULL
>
> for (i in 1 : n) {
+     # échantillon avec remplacement
+     a.bootstrap = sample  (a, length(a), TRUE)
+     b.bootstrap = sample  (b, length(b), TRUE)
+
+     diff.bootstrap[i] = mean(b.bootstrap) - mean(a.bootstrap)
+ }
>
> # Intervalle de confiance (exemple parmi d'autres)
> quantile(diff.bootstrap, c(alpha/2, 1 - alpha/2))
     2.5%     97.5%
0.2938301 9.4879808
>
```

Donc rien à dire de spécial. C'est de la simple algorithmique!

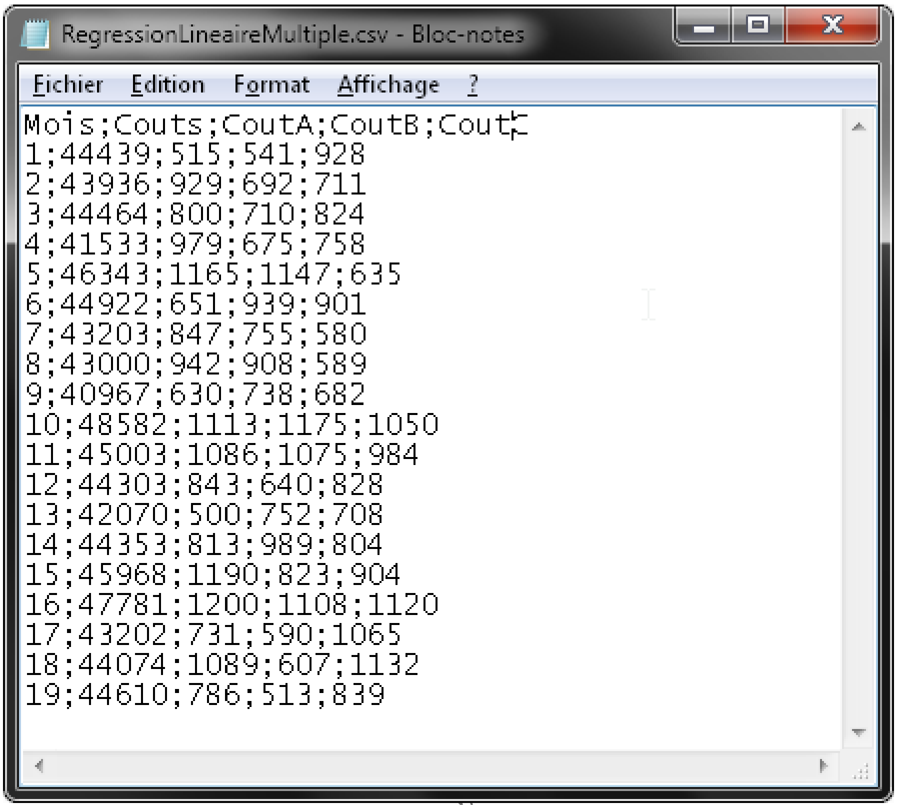
Bootstrap d'adéquation

Avec les fonctions **fitdist( )** et **bootdist( )** du package **fitdistrplus** nous pouvons faire du bootstrapping sur les paramètres des lois (distributions) ajustées comme l'illustre bien l'exemple ci-dessous:



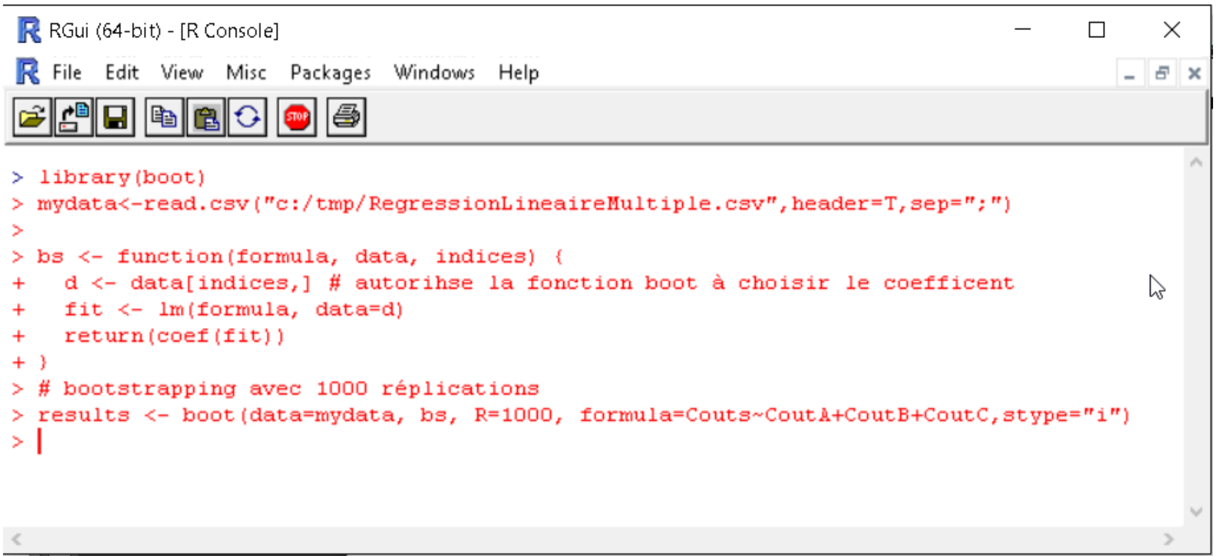
Régression Bootstrap

Nous allons faire une régression bootstrap multivariée de notre fameux jeu de données:



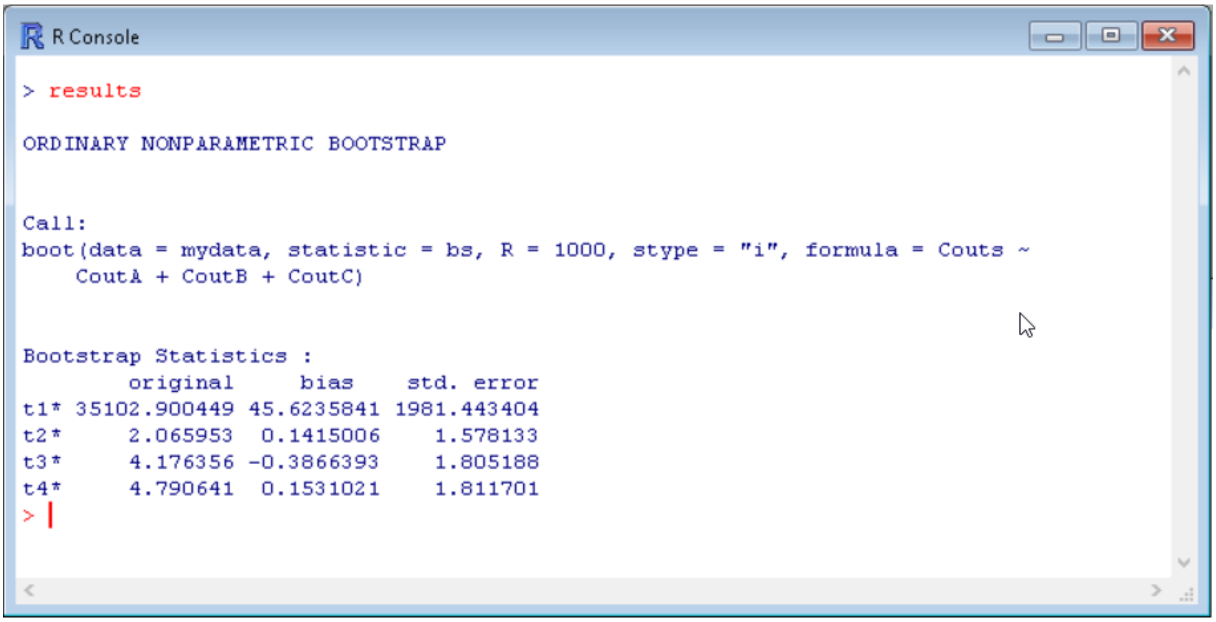
```
Mois;Couts;CoutA;CoutB;CoutC
1;44439;515;541;928
2;43936;929;692;711
3;44464;800;710;824
4;41533;979;675;758
5;46343;1165;1147;635
6;44922;651;939;901
7;43203;847;755;580
8;43000;942;908;589
9;40967;630;738;682
10;48582;1113;1175;1050
11;45003;1086;1075;984
12;44303;843;640;828
13;42070;500;752;708
14;44353;813;989;804
15;45968;1190;823;904
16;47781;1200;1108;1120
17;43202;731;590;1065
18;44074;1089;607;1132
19;44610;786;513;839
```

Nous allons faire un bootstrap sur tous les coefficients avec un intervalle de 95%:



```
> library(boot)
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
>
> bs <- function(formula, data, indices) {
+   d <- data[indices,] # autorihse la fonction boot à choisir le coefficent
+   fit <- lm(formula, data=d)
+   return(coef(fit))
+ }
> # bootstrapping avec 1000 réplications
> results <- boot(data=mydata, bs, R=1000, formula=Couts~CoutA+CoutB+CoutC,stype="i")
>
```

Ensuite nous lançons l'analyse quantitative:



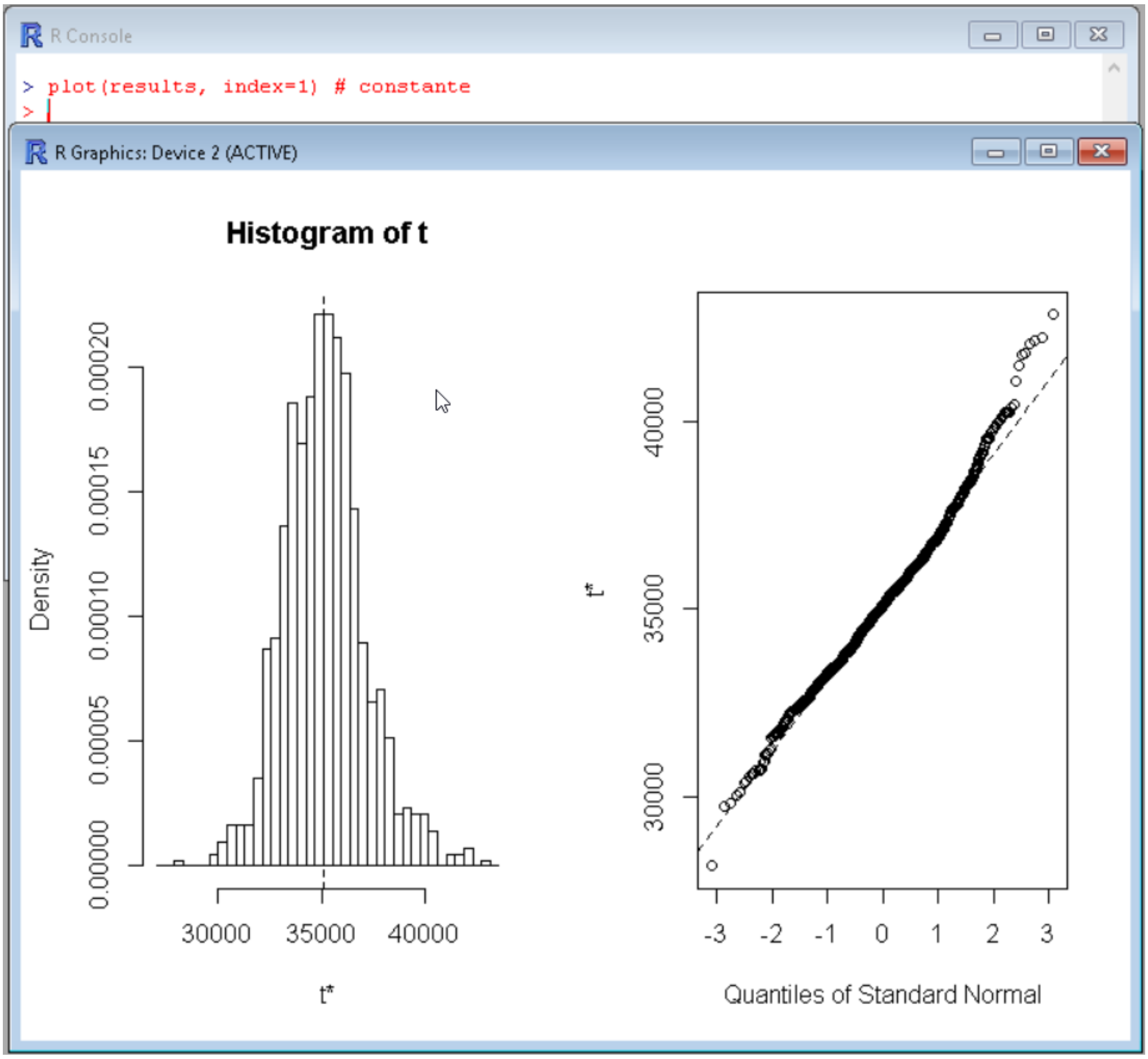
```
R Console
> results

ORDINARY NONPARAMETRIC BOOTSTRAP

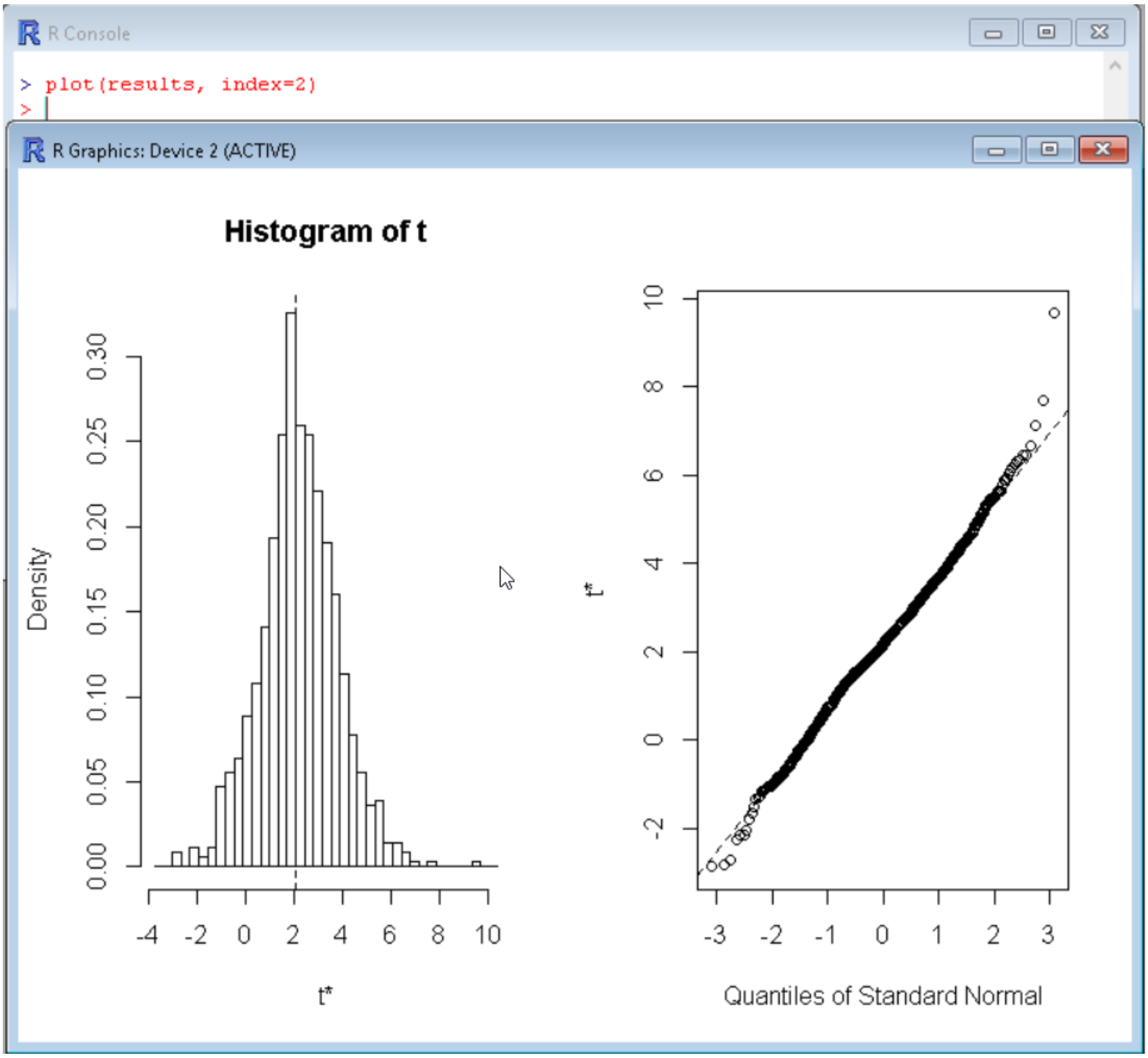

Call:
boot(data = mydata, statistic = bs, R = 1000, stype = "i", formula = Couts ~
    CoutA + CoutB + CoutC)


Bootstrap Statistics :
        original      bias    std. error
t1* 35102.900449 45.6235841 1981.443404
t2*     2.065953  0.1415006    1.578133
t3*     4.176356 -0.3866393    1.805188
t4*     4.790641  0.1531021    1.811701
>
```

et la qualitative de la constante:



Du premier coefficient:



et ainsi de suite jusqu'à *index=4*.

Exemples 317.: Jacknife

R 3.0.2

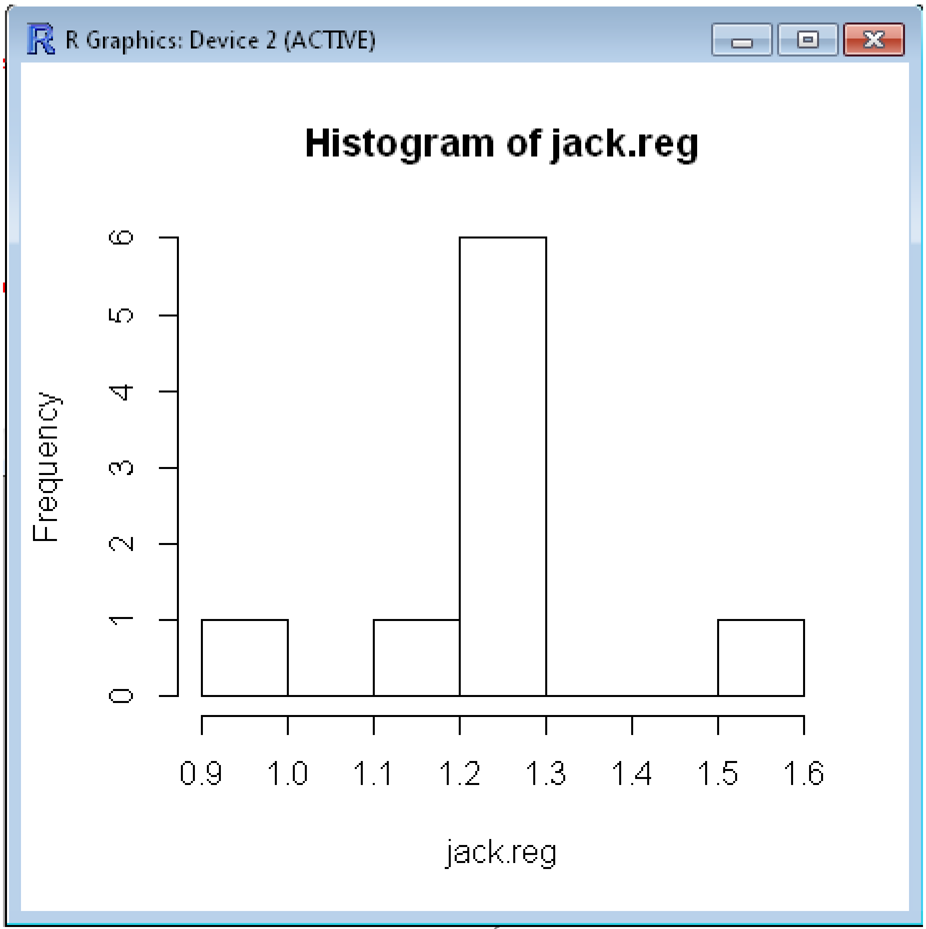
Bootstrapping univarié

Le but ici va être de mettre en pratique la technique jacknife mentionnée dans le cours théorique de Méthodes Numériques avec une petite régression univariée.

Nous allons de suite voir avec cet exemple pourquoi la taille de l'échantillon a une importance plus grande avec le Jacknife qu'avec le Bootstrap que nous verrons juste après:

```
> mydata<-read.csv("C:/RegressionLineaireUnivariee.csv",header=T,sep=";")
> #on regarde combien nous avons de mesures
> v.length<-length(mydata$Ventes)
> #on préalloue un vecteur de même dimension car il y aura autant de simulation
> jack.reg<-numeric(v.length)
> #on fait la simulation
> for(i in 1:v.length){
+ model<-lm(mydata$Ventes[-i]~mydata$Mois[-i])
+ jack.reg[i]<-coef(model)[2]
+ }
> #et on observe que dans ce cas particulier on pourra pas faire grande chose
> #comme inférence sur la pente de la droite de régression
> hist(jack.reg)
>
```

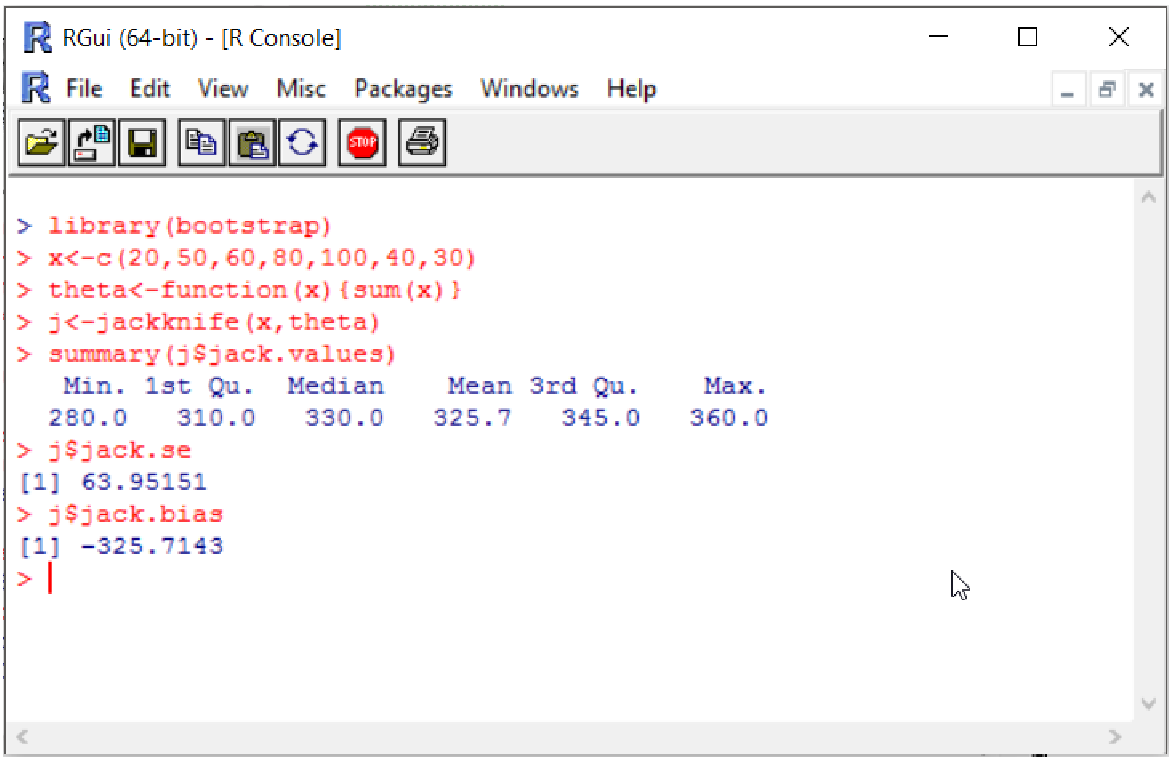
Cela donne donc:



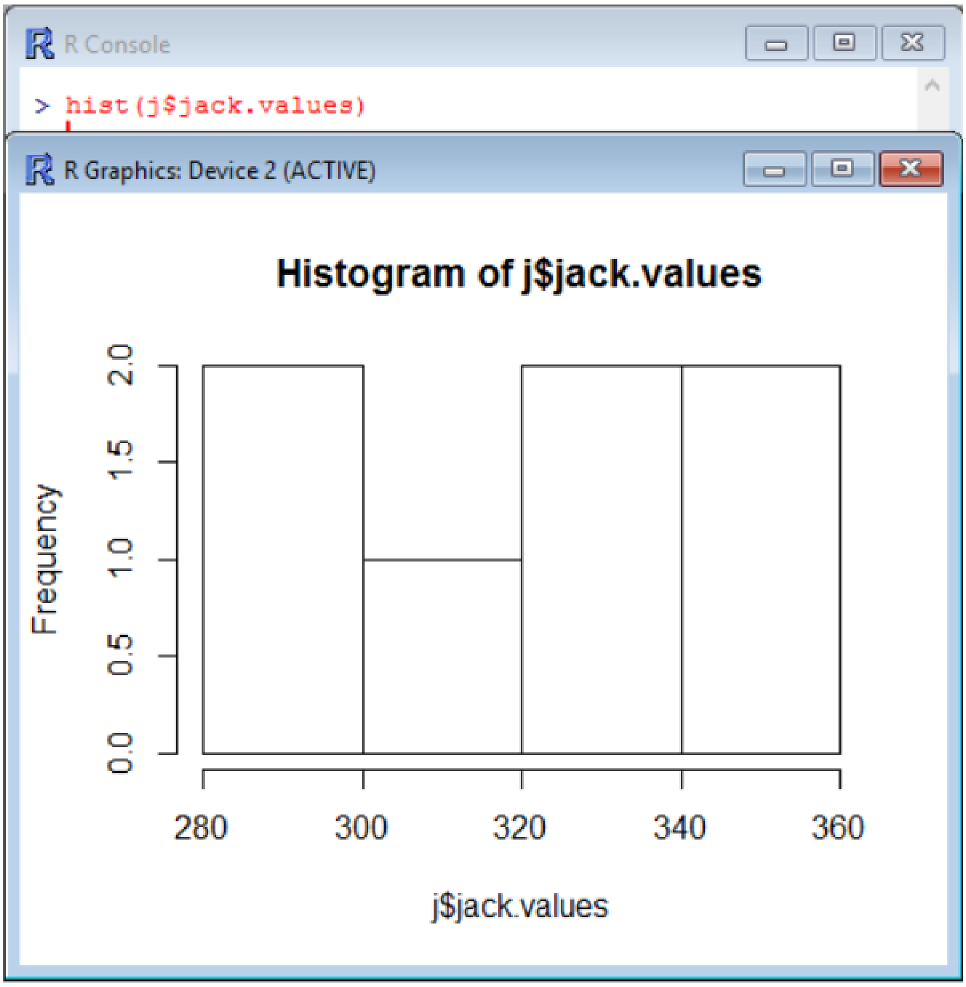
ou en utilisant directement la commande **jackknife( )** du package **bootstrap**:

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(bootstrap)
> mydata<-read.csv("C:/RegressionLineaireUnivariee.csv",header=T,sep=";")
> l<-dim(mydata)[1]
> mySlope<-function(indice,data)
+ {
+ coef(lm(data[indice,2]~data[indice,1]))[2]
+ }
> j=jackknife(1:l,mySlope,data=mydata)
> summary(j$jack.values)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.916   1.207   1.211   1.223   1.230   1.597
> j$jack.se
[1] 0.4598815
> j$jack.bias
data[indice, 1]
     0.02051068
>
```

et pour le jacknife d'une simple somme:



et pour avoir une petit histogramme vite fait mal fait:



Régression Jacknife

Nous allons faire une régression bootstrap univariée sur le coefficient de pente (mais le principe est identique pour le cas multivarié) de notre fameux jeu de données:

```
RegressionLineaireUnivariee.csv - Notepad
File Edit Format View Help
Mois;Ventes
3;4
6;9
7;12
8;15
9;17
10;16
11;17
12;18
15;18
```

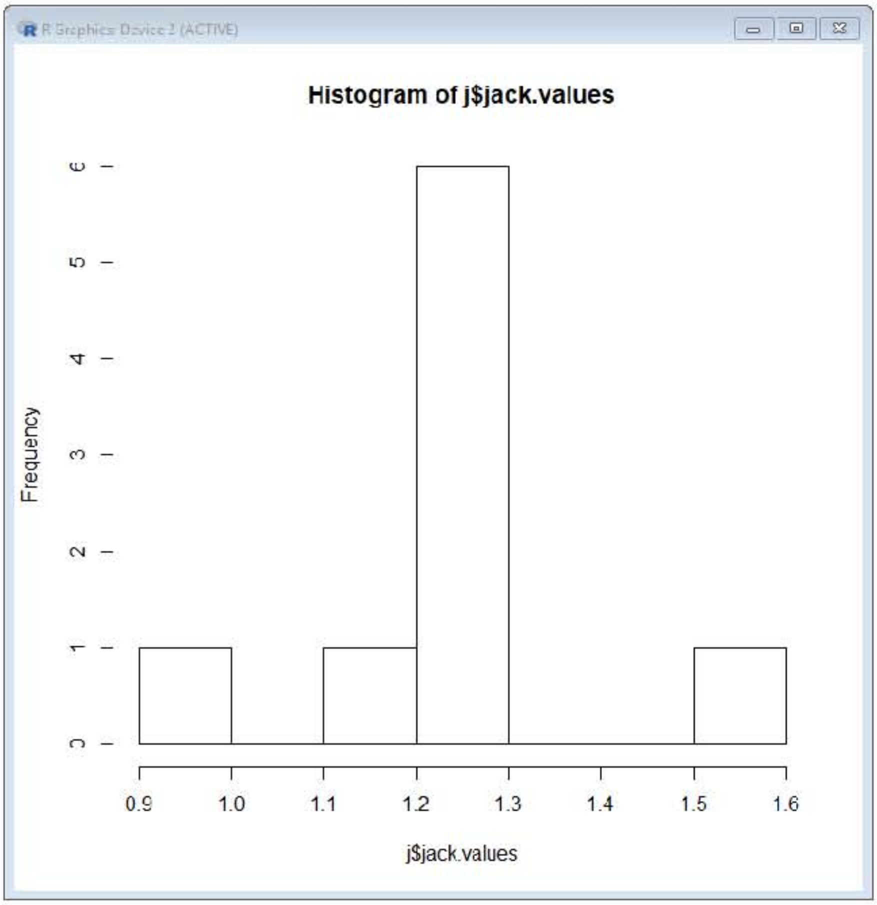
Et allons-y pour le code avec l'intervalle de confiance à 99% calculé via les quantiles:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("bootstrap")
> mydata<-read.csv("C:/tmp/RegressionLineaireUnivariee.csv",header=T,sep=";")
>
> slope <- function(x,xdata){
+   fit <- lm(formula=Ventes~Mois, data=xdata[x,])
+   return(coef(fit)[2]) #pente
+ }
>
> j<-jackknife(1:nrow(mydata),slope,mydata)
> j$jack.values
[1] 0.9159664 1.1696801 1.2303665 1.2452592 1.2200000 1.2111252 1.2068063 1.2072323
[9] 1.5966387
> hist(j$jack.values)
> median(j$jack.values)
[1] 1.211125
> (LB<-quantile(j$jack.values,0.005))
     0.5%
0.9261149
> (UB<-quantile(j$jack.values,0.995))
   99.5%
1.582583
> |
```

Notez qu'il y a bien 9 valeurs Jacknife!

Et l'histogramme correspondant:


R Graphics Device 2 (ACTIVE)
Histogram of j$jack.values
Frequency
0
1
2
3
4
5
6
0.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
j$jack.values

22. Méta-analyses

Exemple: p-valeurs ajustées

Dans le cours théorique nous avons traiter de différentes techniques d'ajustement des p-values dans le cadre des méta-analyses.

Voyons un cas simple d'application:

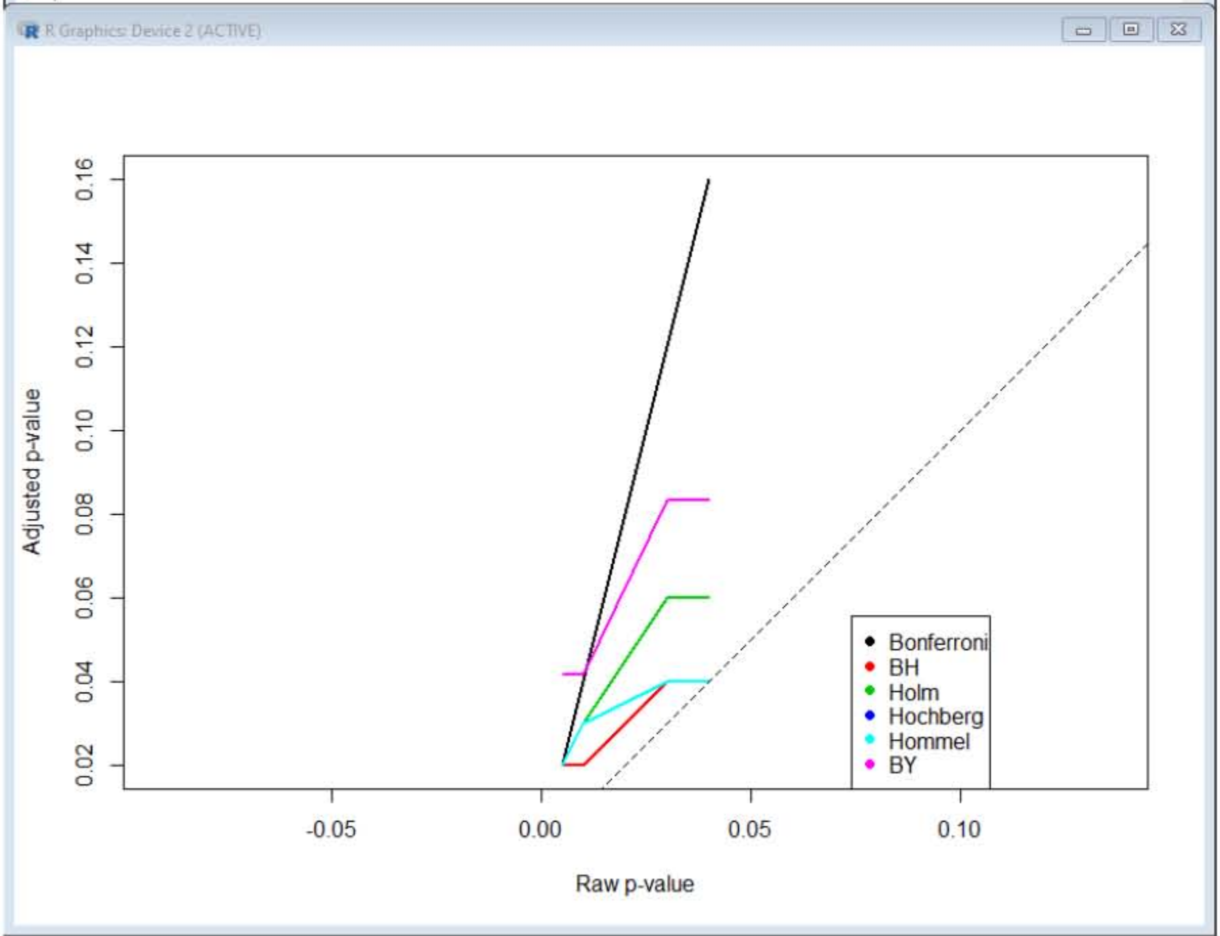
```
> #on crée un petit data frame de p-value à la main
> Input<-("
+ Test                  Raw.p
+ H1                    0.01
+ H2                    0.04
+ H3                    0.03
+ H4                    0.005
+ ")
>
> mydata<-read.table(textConnection(Input),header=TRUE)
>
> ### on trie par p-value croissant
>
> (mydata<-mydata[order(mydata$Raw.p),])
  Test Raw.p
4   H4 0.005
1   H1 0.010
3   H3 0.030
2   H2 0.040
>
```

Utilison maintenant la fonction **p.adjust( )** du package **FSA**:

```
> library("FSA") #Fisheries Stock Assessment
> (mydata$Bonferroni<-p.adjust(mydata$Raw.p,method = "bonferroni"))
[1] 0.02 0.04 0.12 0.16
> (mydata$BH<-p.adjust(mydata$Raw.p,method = "BH"))
[1] 0.02 0.02 0.04 0.04
> (mydata$Holm<-p.adjust(mydata$Raw.p,method = "holm"))
[1] 0.02 0.03 0.06 0.06
> (mydata$Hochberg<-p.adjust(mydata$Raw.p,method = "hochberg"))
[1] 0.02 0.03 0.04 0.04
> (mydata$Hommel<-p.adjust(mydata$Raw.p,method = "hommel"))
[1] 0.02 0.03 0.04 0.04
> (mydata$BY<-p.adjust(mydata$Raw.p,method = "BY"))
[1] 0.04166667 0.04166667 0.08333333 0.08333333
>
> mydata
  Test Raw.p Bonferroni   BH Holm Hochberg Hommel         BY
4   H4 0.005       0.02 0.02 0.02     0.02   0.02 0.04166667
1   H1 0.010       0.04 0.02 0.03     0.03   0.03 0.04166667
3   H3 0.030       0.12 0.04 0.06     0.04   0.04 0.08333333
2   H2 0.040       0.16 0.04 0.06     0.04   0.04 0.08333333
>
> #ou:
> stats::p.adjust(c(0.01,0.04,0.03,0.005), method ="bonferroni")
[1] 0.04 0.16 0.12 0.02
> stats::p.adjust(c(0.01,0.04,0.03,0.005), method ="BH")
[1] 0.02 0.04 0.04 0.02
> stats::p.adjust(c(0.01,0.04,0.03,0.005), method ="hochberg")
[1] 0.03 0.04 0.04 0.02
> stats::p.adjust(c(0.01,0.04,0.03,0.005), method ="holm")
[1] 0.03 0.06 0.06 0.02
> #etc..
> 
```

Nous pouvons aussi faire un graphique:

```
> X <- mydata$Raw.p
> Y <- cbind(mydata$Bonferroni,mydata$BH,mydata$Holm,mydata$Hochberg,
+           mydata$Hommel,mydata$BY)
>
> matplot(X,Y,xlab="Raw p-value",ylab="Adjusted p-value",type="l",
+         asp=1,col=1:6,lty=1,lwd=2)
>
> legend('bottomright', legend = c("Bonferroni", "BH", "Holm", "Hochberg",
+ "Hommel", "BY"),col = 1:6,cex = 1,pch = 16)
>
> abline(0, 1,col=1,lty=2,lwd=1)
> |
```



23. Décisionnel

Bon pour l'instant pour être honnête je n'ai pas trouvé grand-chose au niveau décisionnel dans R donc ce chapitre a un contenu qui est négligeable. Cependant et comme à l'habitude cela permet de vérifier des résultats calculés à la main dans le cours théorique.

Exemple 318.: Business Process Analysis

R 3.4.1

Pour étudier les fonctions de business process analysis, nous allons nous baser sur les packages **bupaR**, **xesreadR** et **processmapR** (tous du même auteur!) et en utiliser un fichier *.xes (eXtensible Event Stream[33]) avec le contenu suivant:

	A	B	C	D	E	F
1	Case ID	Event ID	dd-MM-yyy	Activity	Resource	Costs
2	1	35654423	30-12-2010:1	register req	Pete	50
3	1	35654424	31-12-2010:1	examine th	Sue	400
4	1	35654425	05-01-2011:1	check ticke	Mike	100
5	1	35654426	06-01-2011:1	decide	Sara	200
6	1	35654427	07-01-2011:1	reject reque	Pete	200
7	2	35654483	30-12-2010:1	register req	Mike	50
8	2	35654485	30-12-2010:1	check ticke	Mike	100
9	2	35654487	30-12-2010:1	examine ca	Sean	400
10	2	35654488	05-01-2011:1	decide	Sara	200
11	2	35654489	08-01-2011:1	pay compe	Ellen	200
12	3	35654521	30-12-2010:1	register req	Pete	50
13	3	35654522	30-12-2010:1	examine ca	Mike	400
14	3	35654524	30-12-2010:1	check ticke	Ellen	100
15	3	35654525	06-01-2011:0	decide	Sara	200
16	3	35654526	06-01-2011:1	reinitiate re	Sara	200
17	3	35654527	06-01-2011:1	examine th	Sean	400
18	3	35654530	08-01-2011:1	check ticke	Pete	100
19	3	35654531	09-01-2011:0	decide	Sara	200
20	3	35654533	15-01-2011:1	pay compe	Ellen	200
21	4	35654641	06-01-2011:1	register req	Pete	50
22	4	35654643	07-01-2011:1	check ticke	Mike	100
23	4	35654644	08-01-2011:1	examine th	Sean	400
24	4	35654645	09-01-2011:1	decide	Sara	200
25	4	35654647	12-01-2011:1	reject reque	Ellen	200
26	5	35654711	06-01-2011:0	register req	Ellen	50
27	5	35654712	07-01-2011:1	examine ca	Mike	400
28	5	35654714	08-01-2011:1	check ticke	Pete	100
29	5	35654715	10-01-2011:1	decide	Sara	200
30	5	35654716	11-01-2011:16	reinitiate re	Sara	200
31	5	35654718	14-01-2011:1	check ticke	Ellen	100
32	5	35654719	16-01-2011:1	examine ca	Mike	400
33	5	35654720	19-01-2011:11	decide	Sara	200
34	5	35654721	20-01-2011:1	reinitiate re	Sara	200
35	5	35654722	21-01-2011:0	examine ca	Sue	400
36	5	35654724	21-01-2011:11	check ticke	Pete	100
37	5	35654725	23-01-2011:1	decide	Sara	200
38	5	35654726	24-01-2011:1	reject reque	Mike	200
39	6	35654871	06-01-2011:1	register req	Mike	50
40	6	35654873	06-01-2011:1	examine ca	Ellen	400
41	6	35654874	07-01-2011:1	check ticke	Mike	100
42	6	35654875	07-01-2011:1	decide	Sara	200
43	6	35654877	16-01-2011:11	pay compe	Mike	200

Donc nous chargeons d'abord les packages et ensuite le fichier *.xes d'exemple à l'aide de la fonction **read_xes( )** :

[33] Il s'agit du format normalisé des logiciels qui stockent en temps réel ou manuellement les étapes d'avancement de processus business.

```
> library("xesreadR")
> library("processmapR")
> eventlog<- read_xes("c:/tmp/business_process_event_analysis.xes")
Warning messages:
1: In read_xes("c:/tmp/business_process_event_analysis.xes") :
  No lifecycle transition id specified in xes-file
2: In read_xes("c:/tmp/business_process_event_analysis.xes") :
  No activity instance identifier specified in xes-file. By default considered each event a$
>
> eventlog
Event log consisting of:
42 events
6 traces
6 cases
8 activities
42 activity instances

# A tibble: 42 x 10
   CASE_concept_name   CASE_creator           Activity        activity_id Costs
               <chr>          <chr>              <chr>              <chr> <chr>
 1                 3 Fluxicon Nitro   register request   register request    50
 2                 3 Fluxicon Nitro   examine casually   examine casually   400
 3                 3 Fluxicon Nitro       check ticket       check ticket   100
 4                 3 Fluxicon Nitro             decide             decide   200
 5                 3 Fluxicon Nitro reinitiate request reinitiate request   200
 6                 3 Fluxicon Nitro examine thoroughly examine thoroughly   400
 7                 3 Fluxicon Nitro       check ticket       check ticket   100
 8                 3 Fluxicon Nitro             decide             decide   200
 9                 3 Fluxicon Nitro   pay compensation   pay compensation   200
10                 2 Fluxicon Nitro   register request   register request    50
# ... with 32 more rows, and 5 more variables: resource_id <chr>,
#   Resource <chr>, timestamp <dttm>, lifecycle_id <lgl>,
#   activity_instance_id <int>
>
```

Nous avons d'abord le graphique suivant nommé bien évidemment **dotted_chart**:

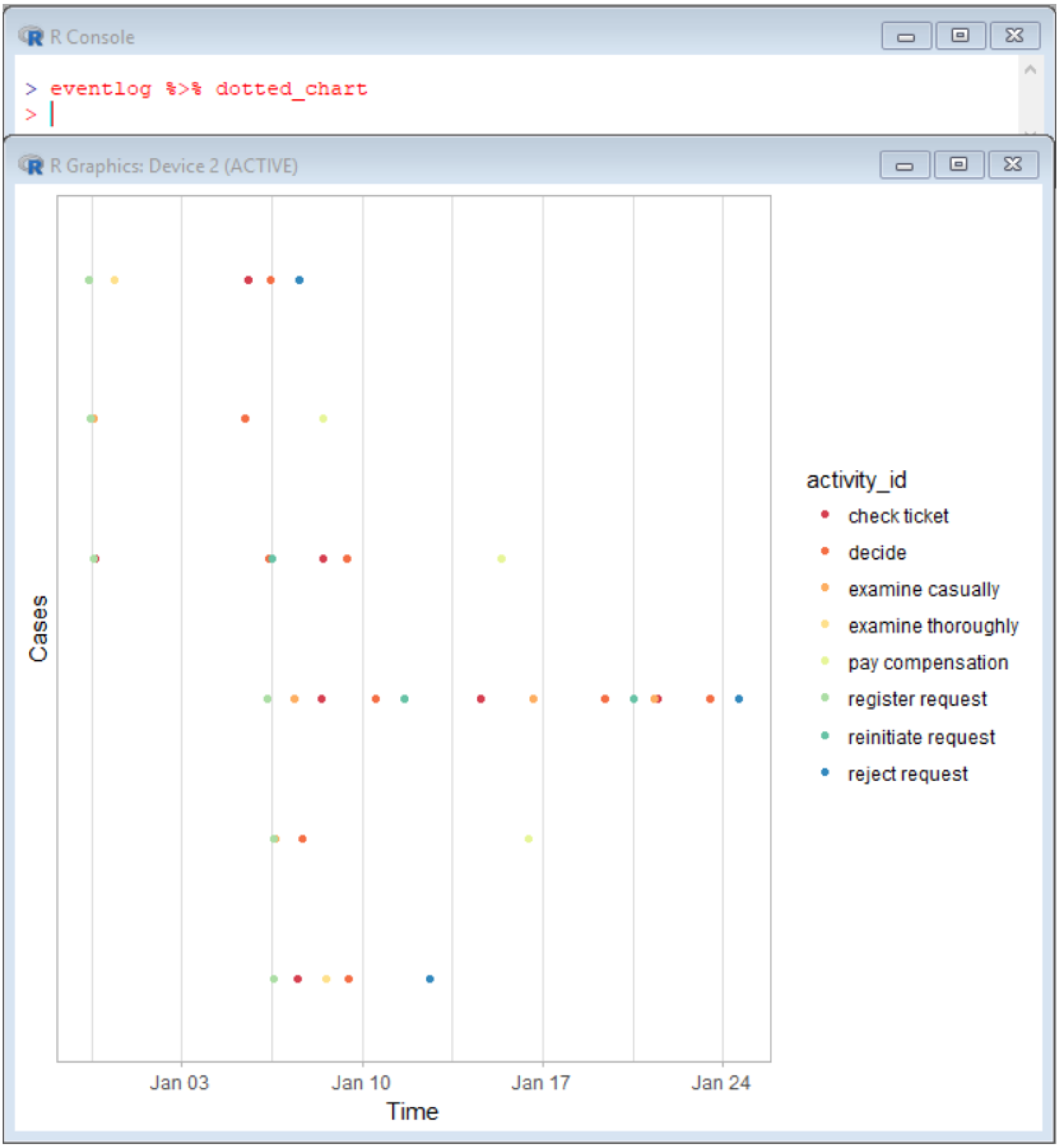
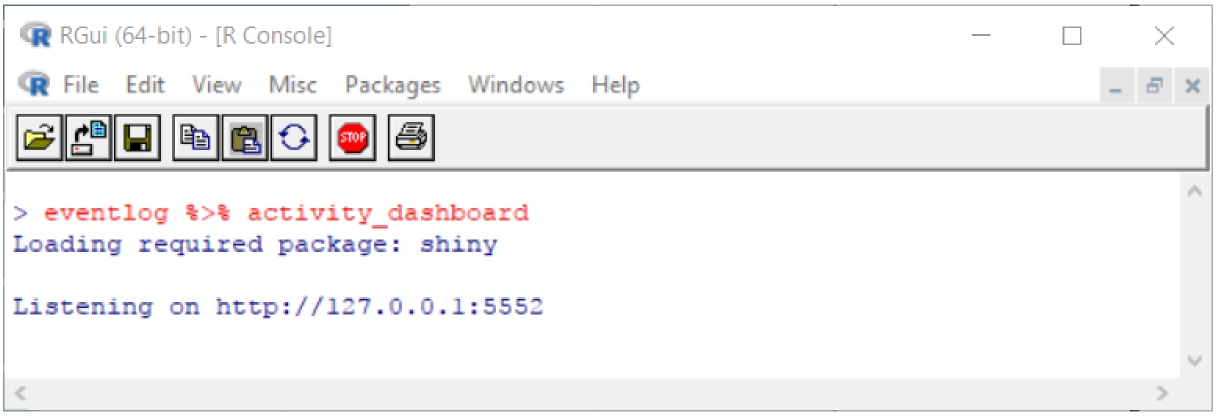


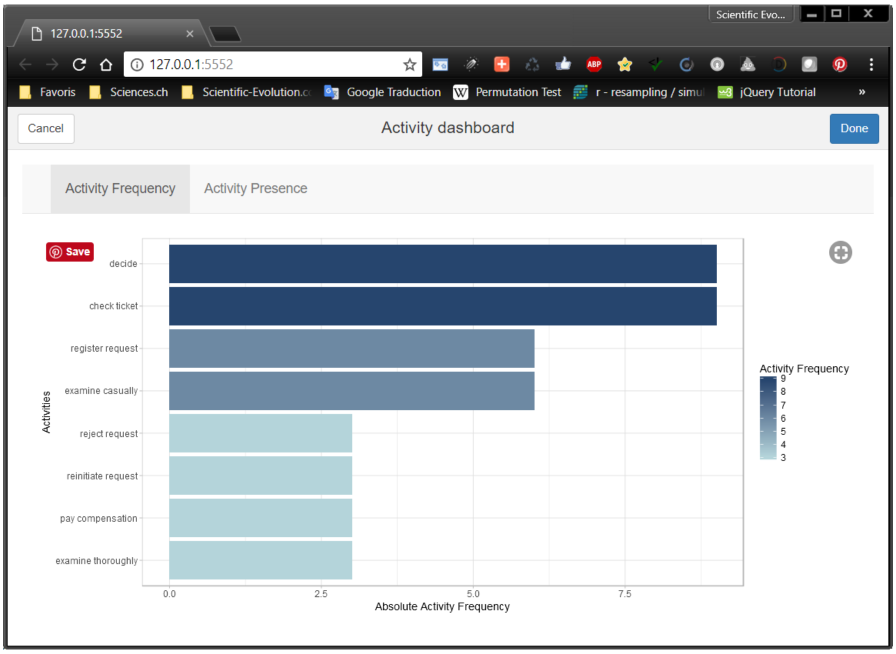
Figure 290 Graph évolution processus

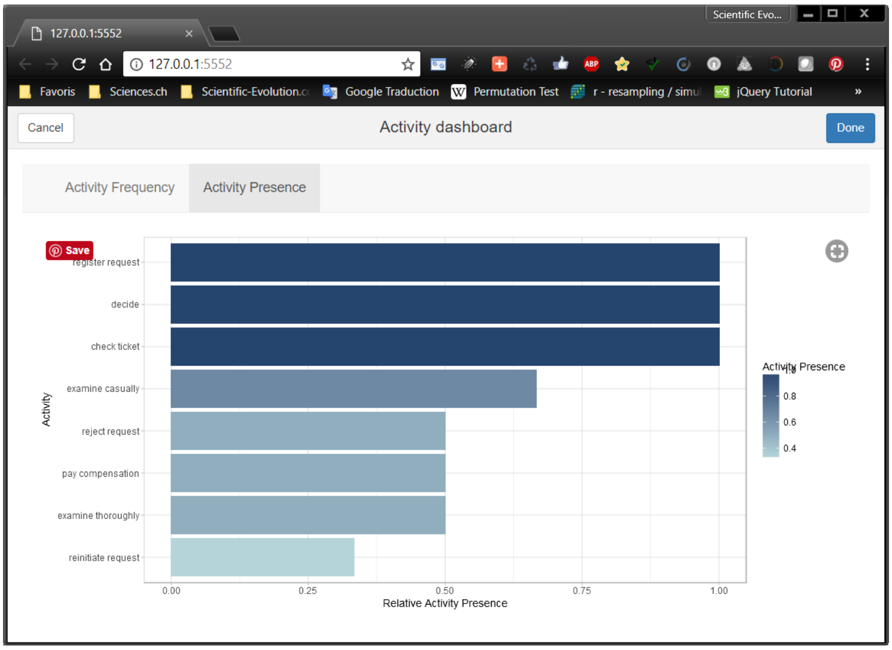
Qui par *Case_ID* donne donc la répartition dans le temps du type d'activités identifiées ou déclenchées.

Ensuite, on a la commande **activity_dashboard**:

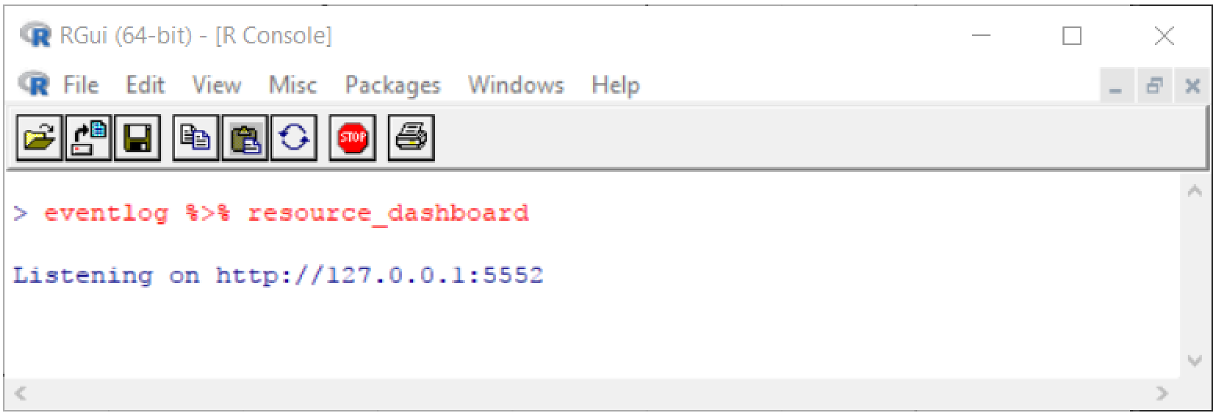


Qui donne bien l'impression de lancer un tableau de bord R Shiny:

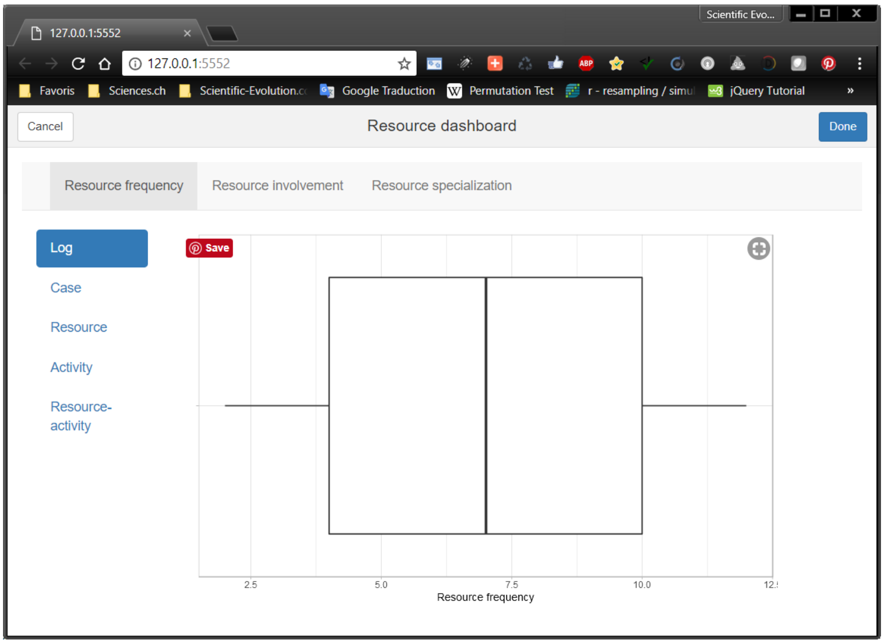


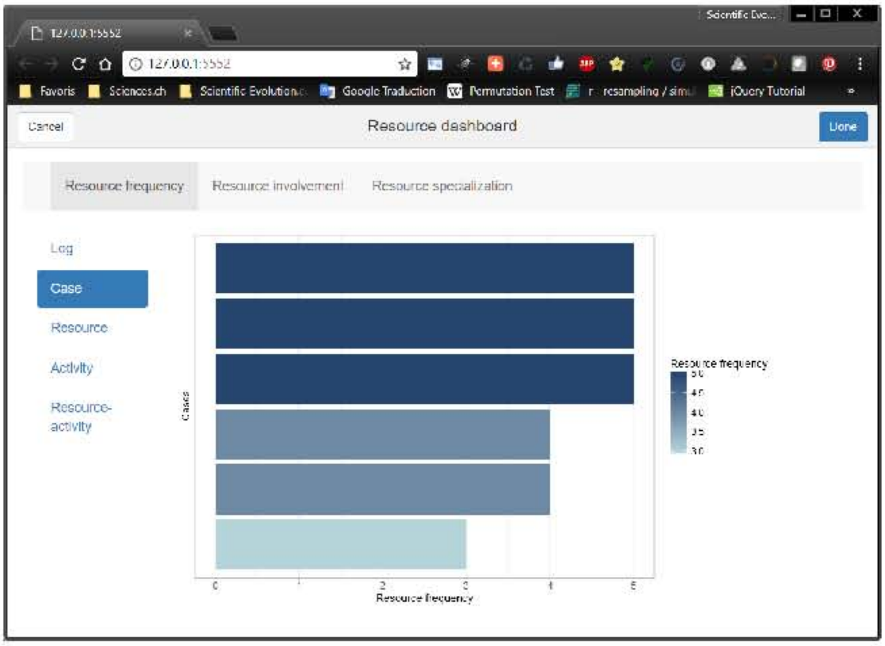


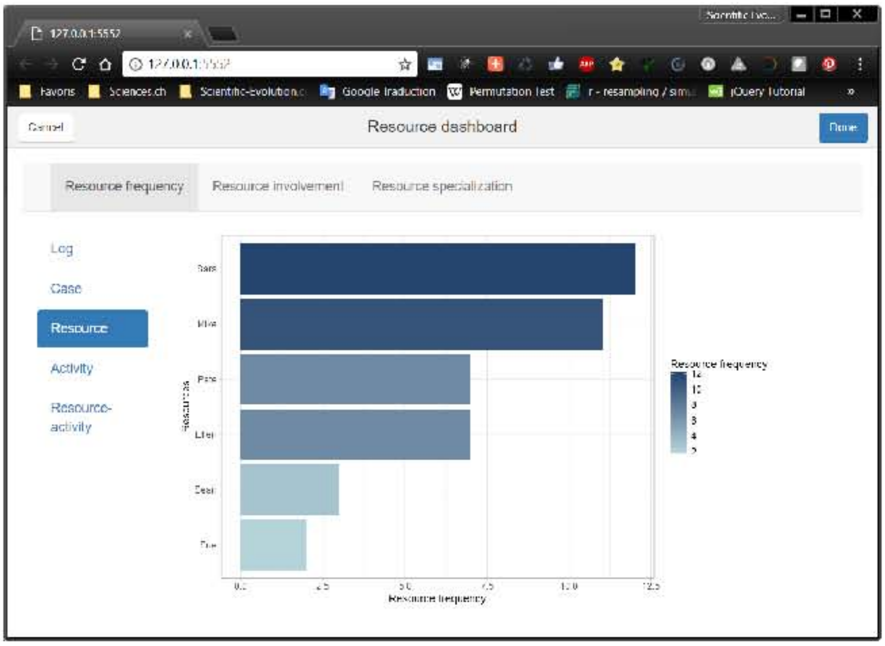
Et la commande **resource_dashboard**:

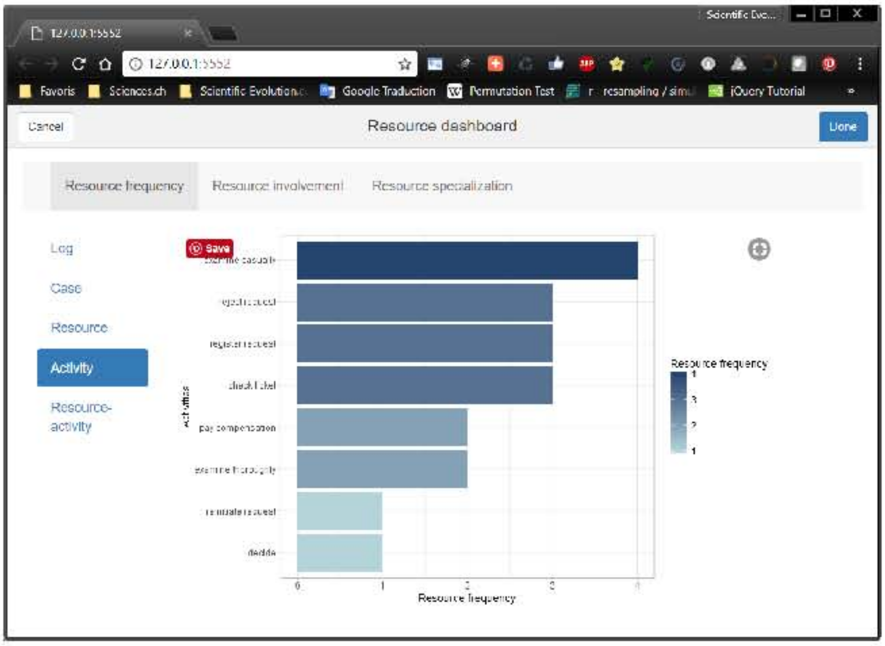


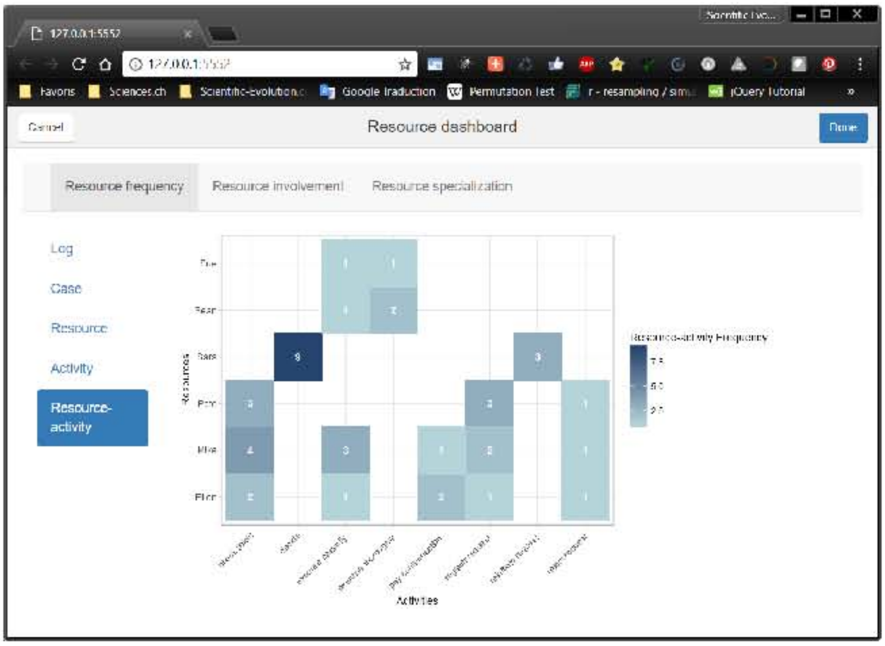
Qui donne une multitude de graphiques (dont certains sont redondants):

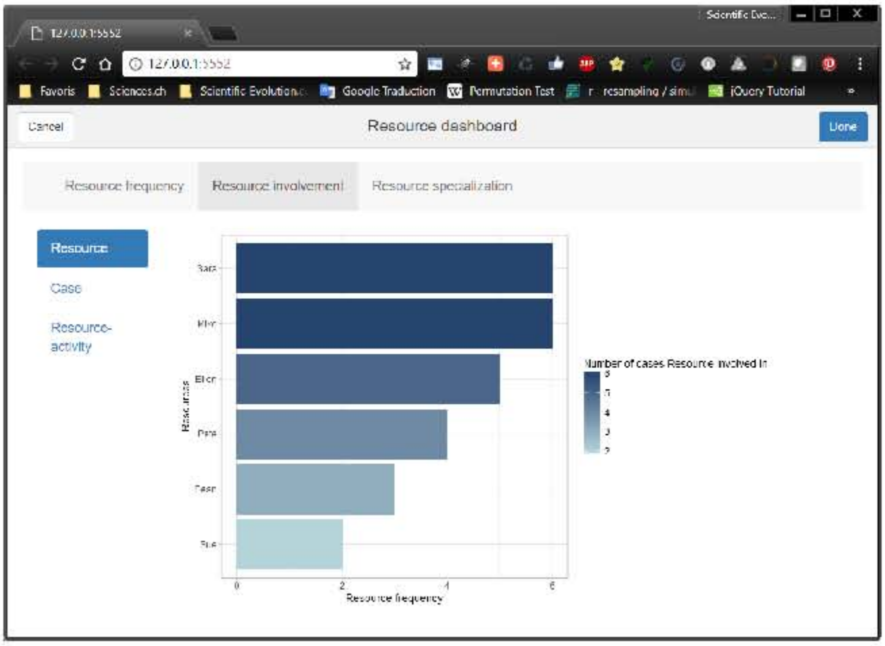


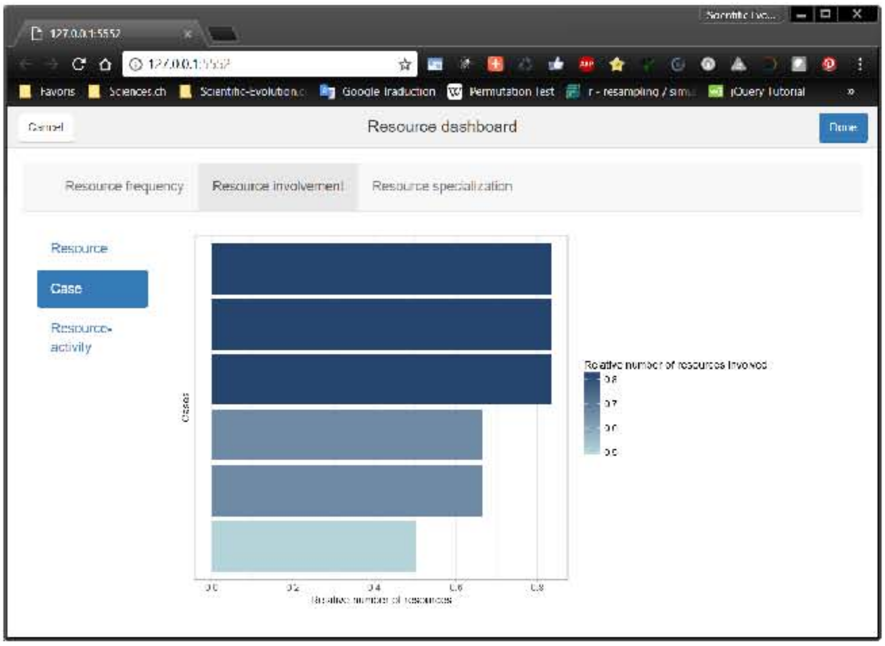

Resource dashboard
Cancel
Done
Resource frequency
Resource involvement
Resource specialization
Log
Case
Resource
Activity
Resource-activity
Resource frequency

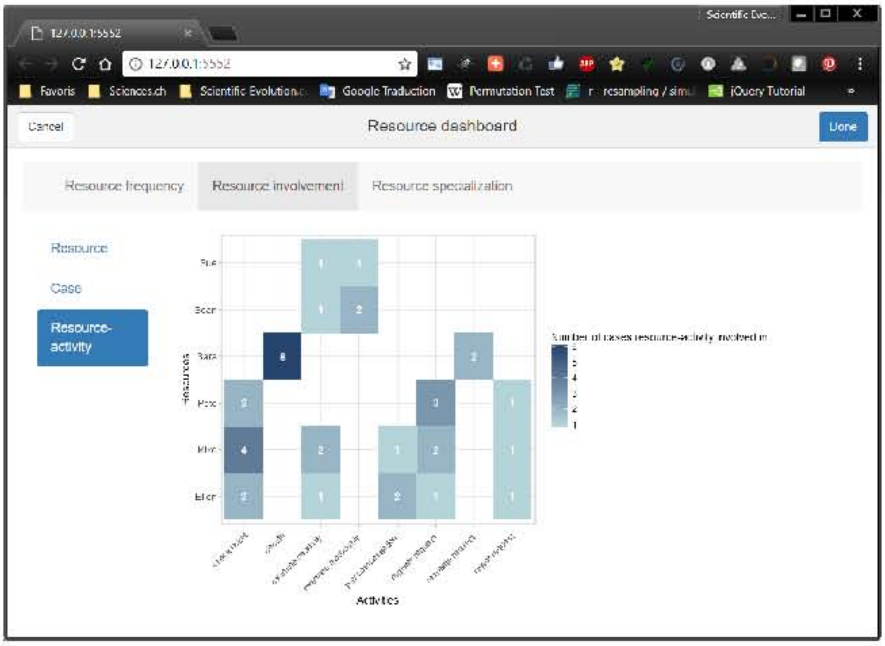

Resource dashboard
Done
Resource frequency
Resource involvement
Resource specialization
Log
Case
Resource
Activity
Resource-activity
Resource frequency

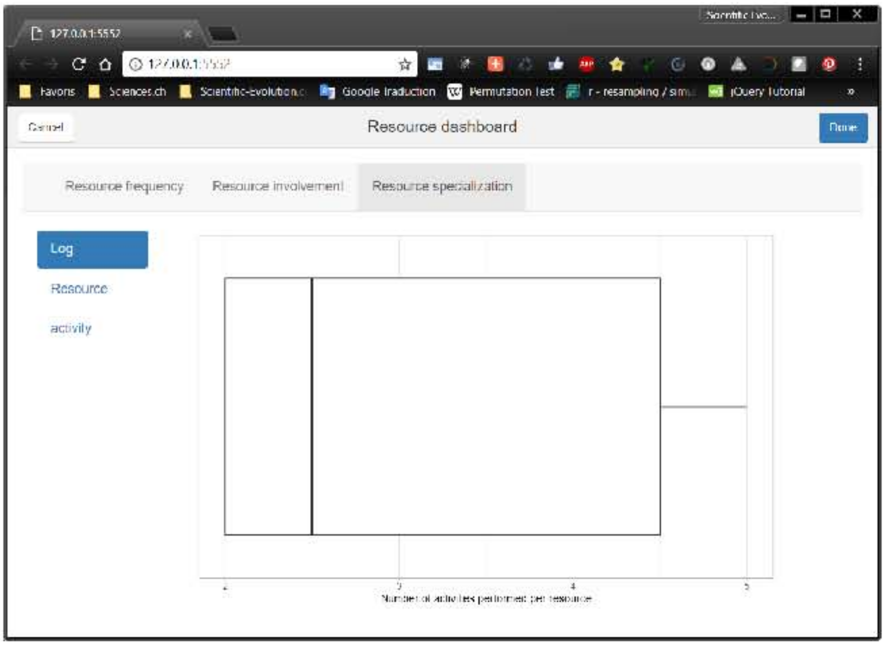

Cancel
Resource dashboard
Done
Resource frequency
Resource involvement
Resource specialization
Log
Case
Resource
Activity
Resource-activity
Resource frequency

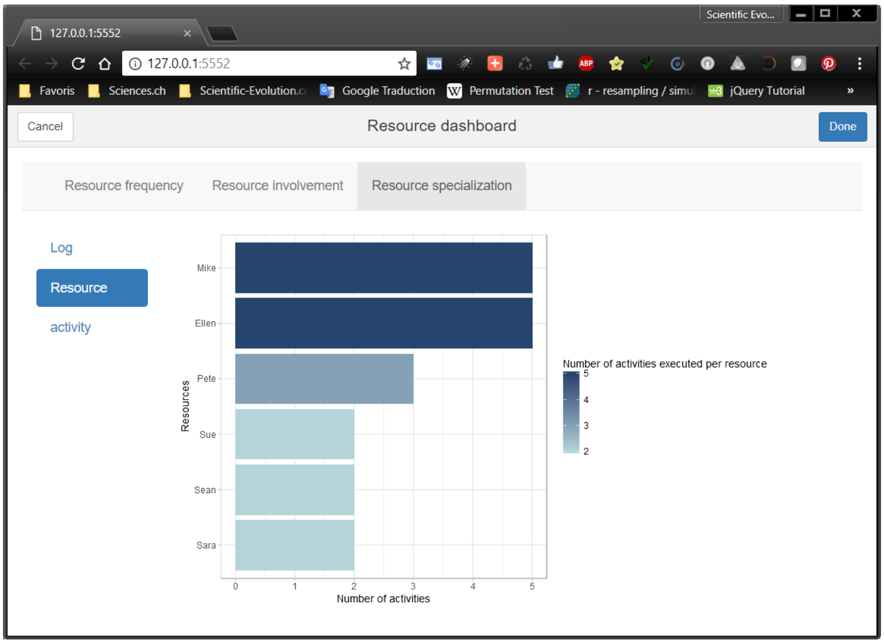

Cancel
Resource dashboard
Done
Resource frequency
Resource involvement
Resource specialization
Log
Case
Resource
Activity
Resource-activity
Activities

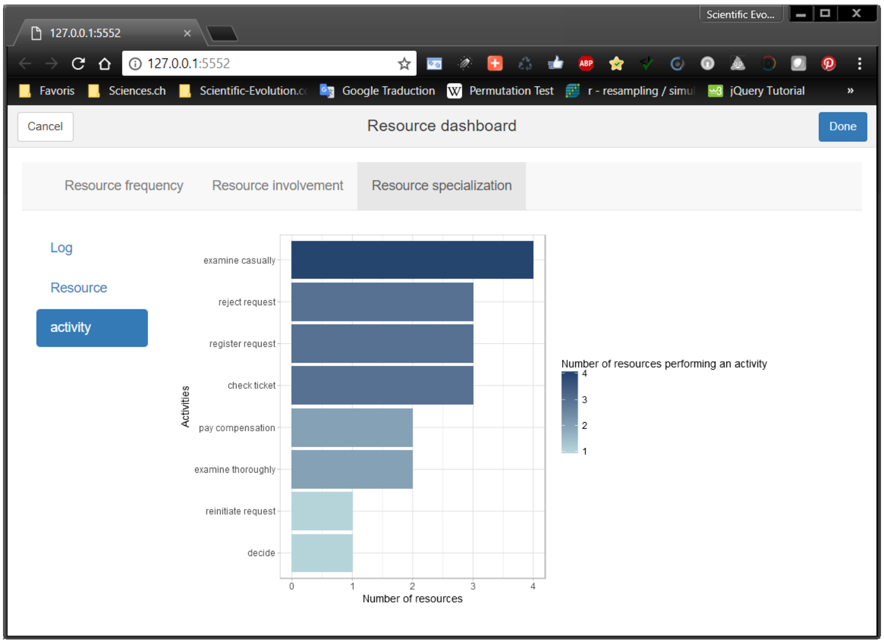

Favoris
Sciences.ch
Google Traduction
Permutation Test
jQuery Tutorial
Cancel
Resource dashboard
Done
Resource frequency
Resource involvement
Resource specialization
Resource
Case
Resource-activity
Number of cases Resource involved in
Resource frequency


Favoris
Sciences.ch
Google Traduction
Permutation Test
jQuery Tutorial
Cancel
Resource dashboard
Done
Resource frequency
Resource involvement
Resource specialization
Resource
Case
Resource-activity
Relative number of resources involved
Cases

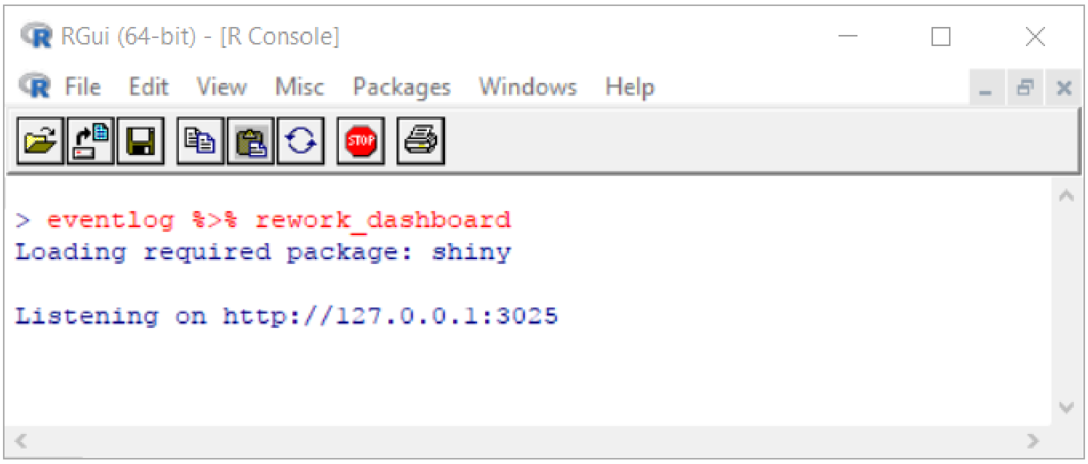

Resource dashboard
Cancel
Done
Resource frequency
Resource involvement
Resource specialization
Resource
Case
Resource-activity
Activities


Resource dashboard
Cancel
Done
Resource frequency
Resource involvement
Resource specialization
Log
Resource
activity

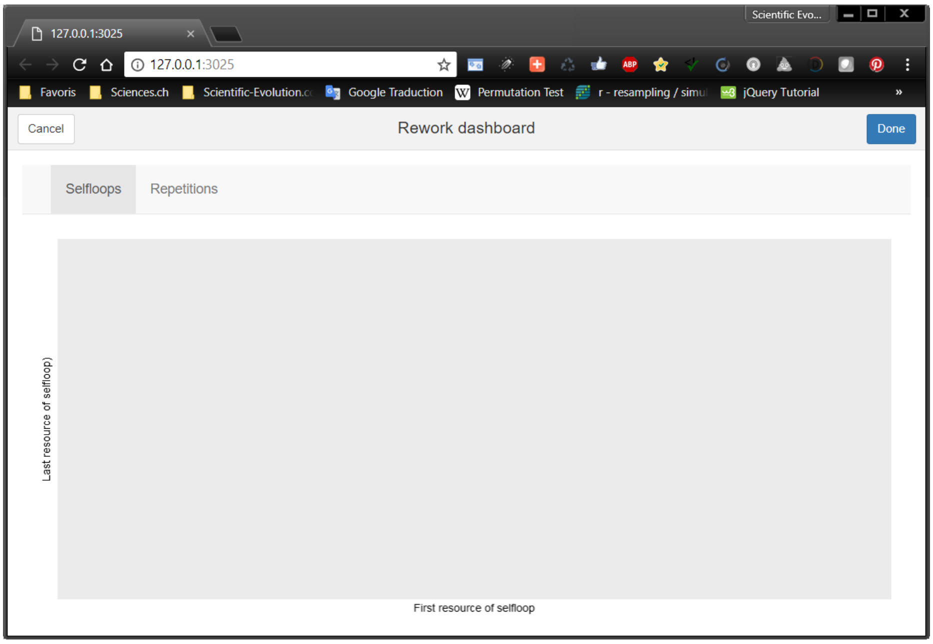


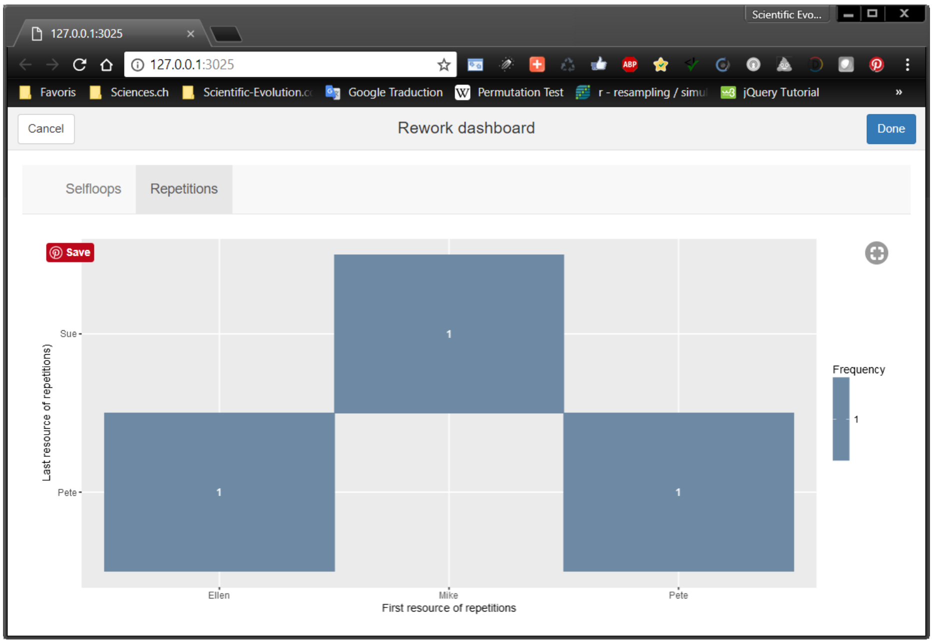


ou encore avec la commande **rework_dashboard**:

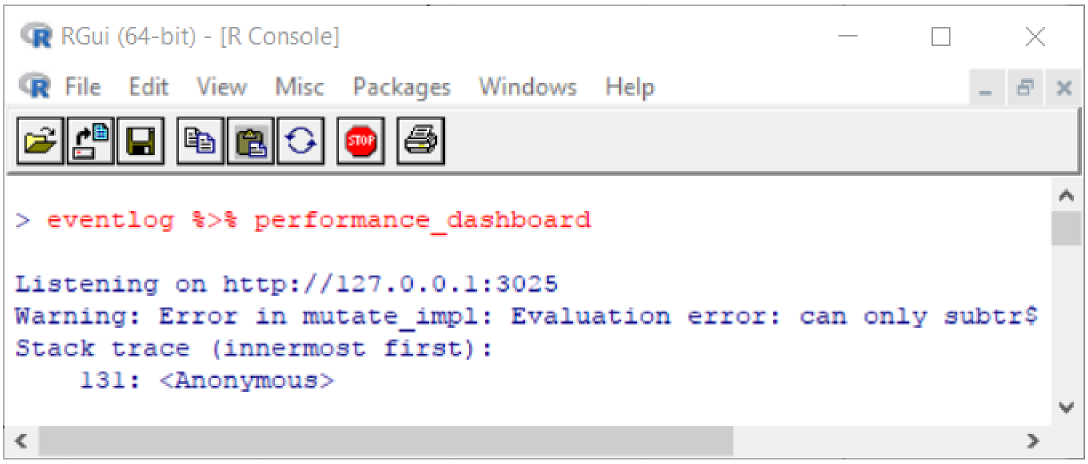


Ce qui donne:

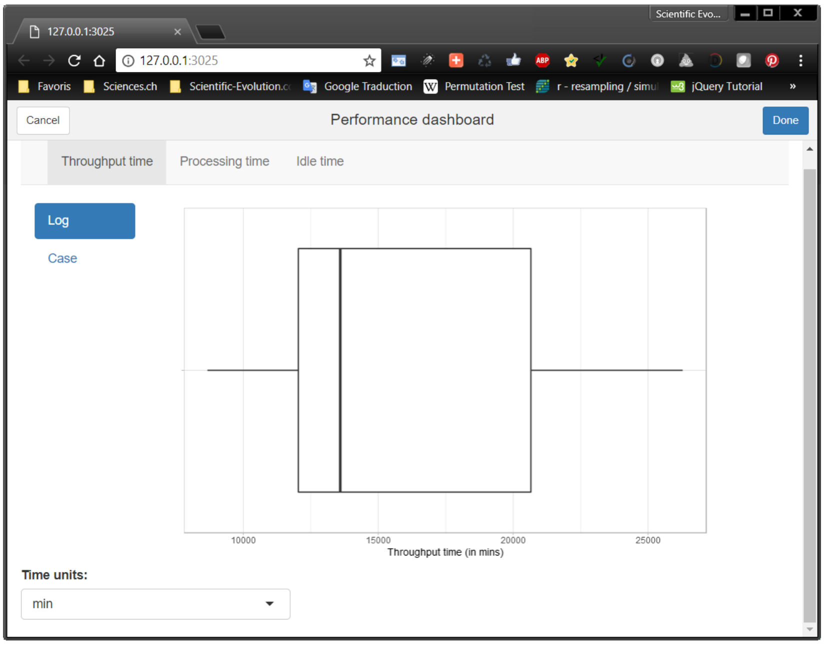


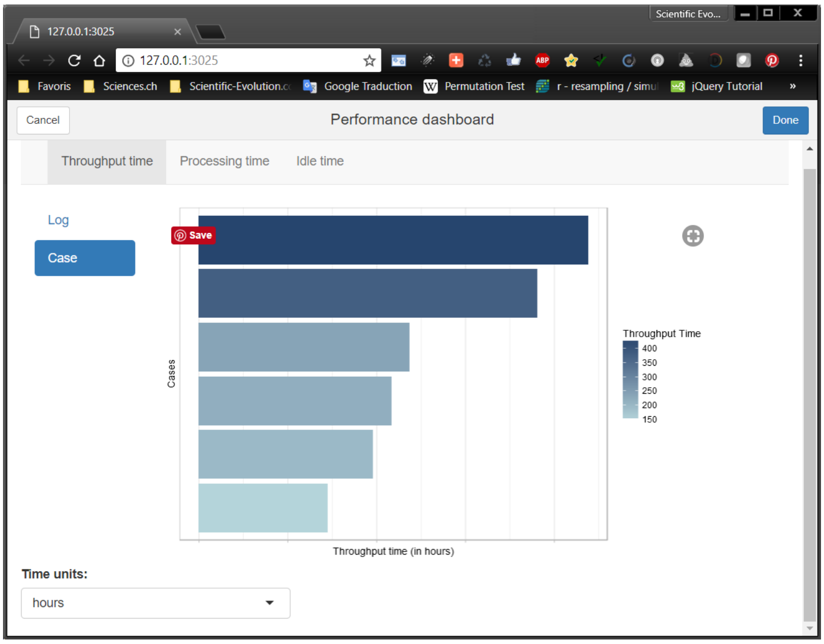


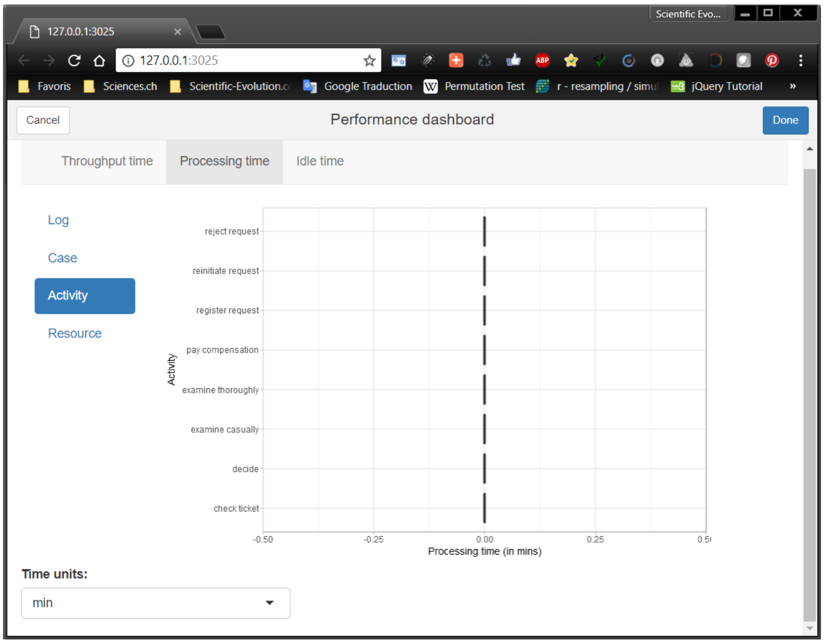
Et pour la commande **performance_dashboard**:

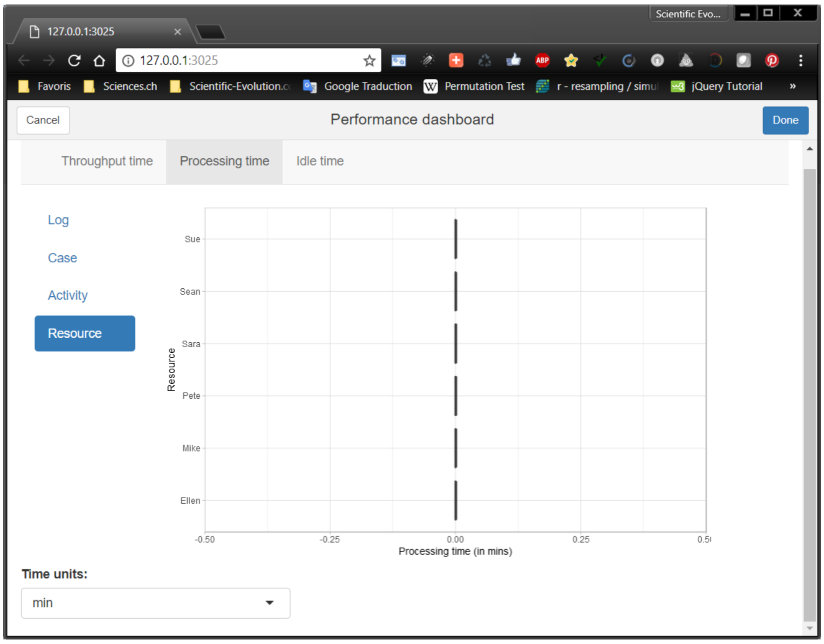


Ce qui donne (nous n'avons pas représenté les graphiques vides!):

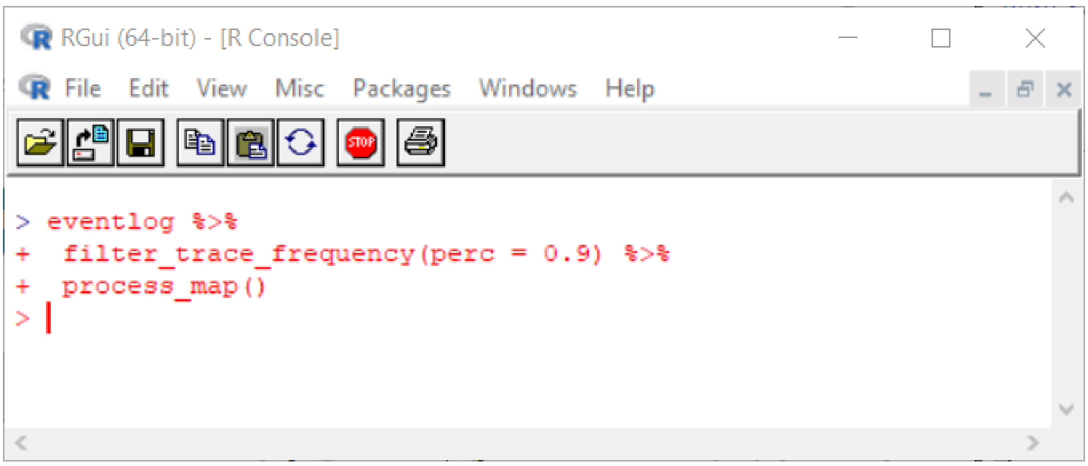








Avec la commande **process_map( )** :



Ce qui donne:

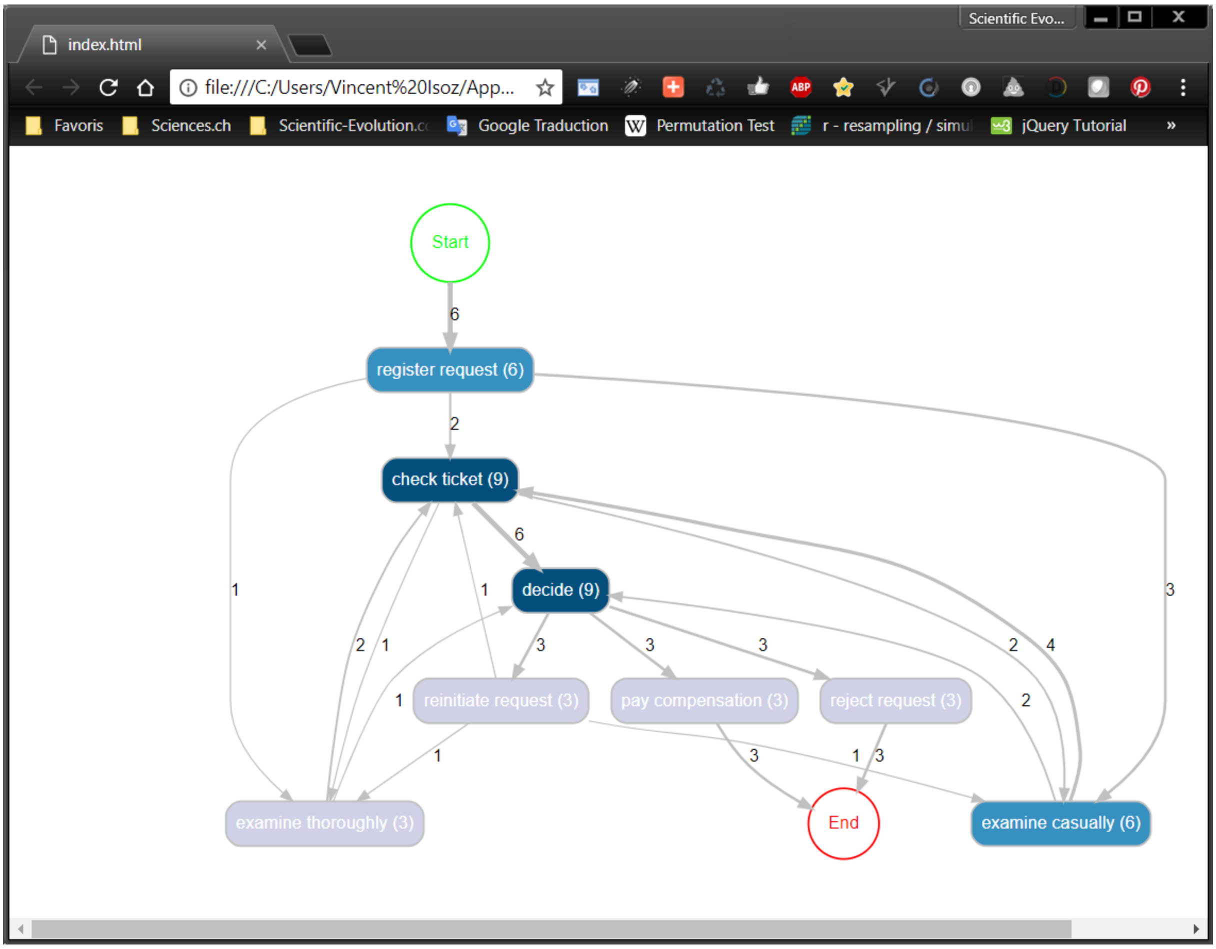
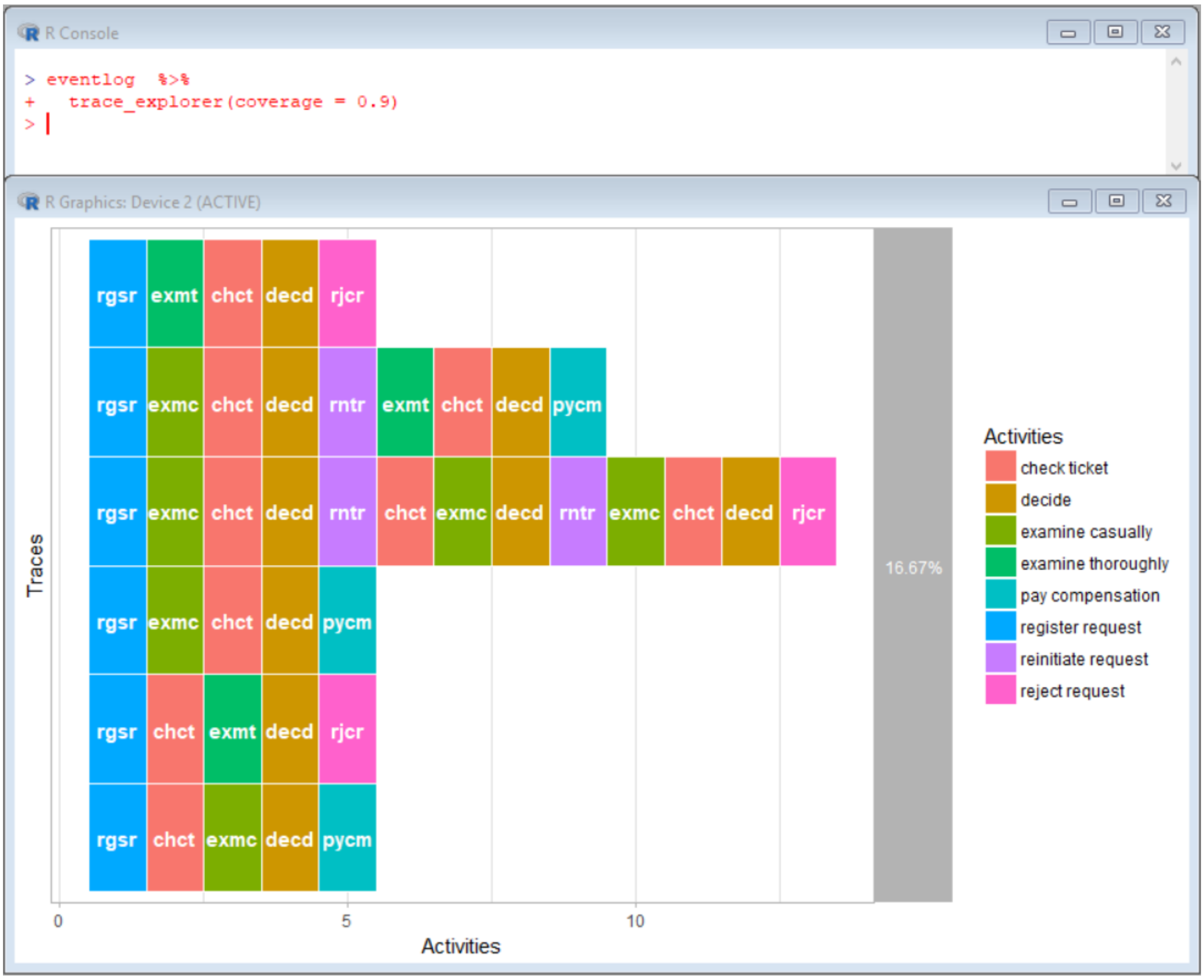
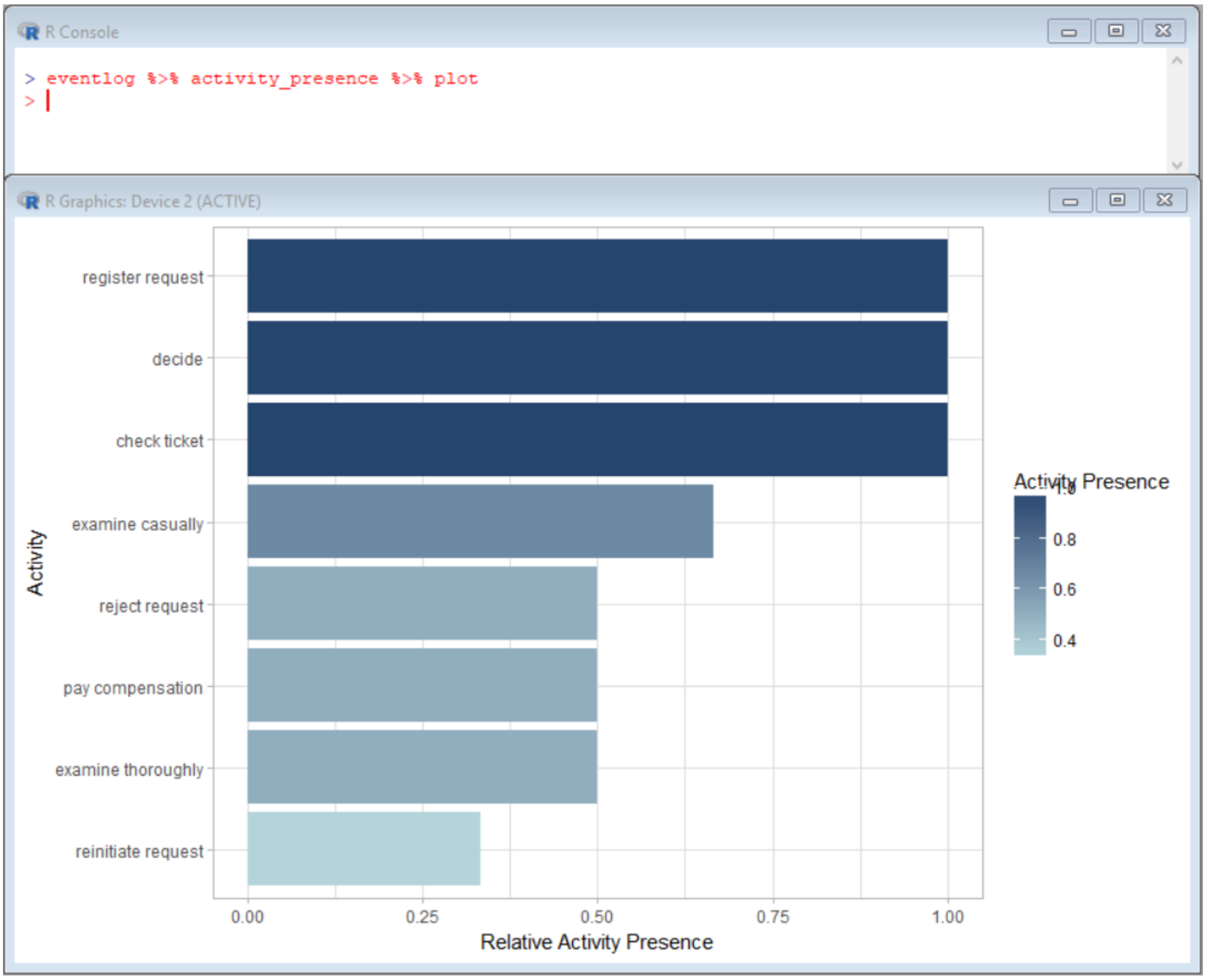


Figure 291 Graph de résumé de flux de processus

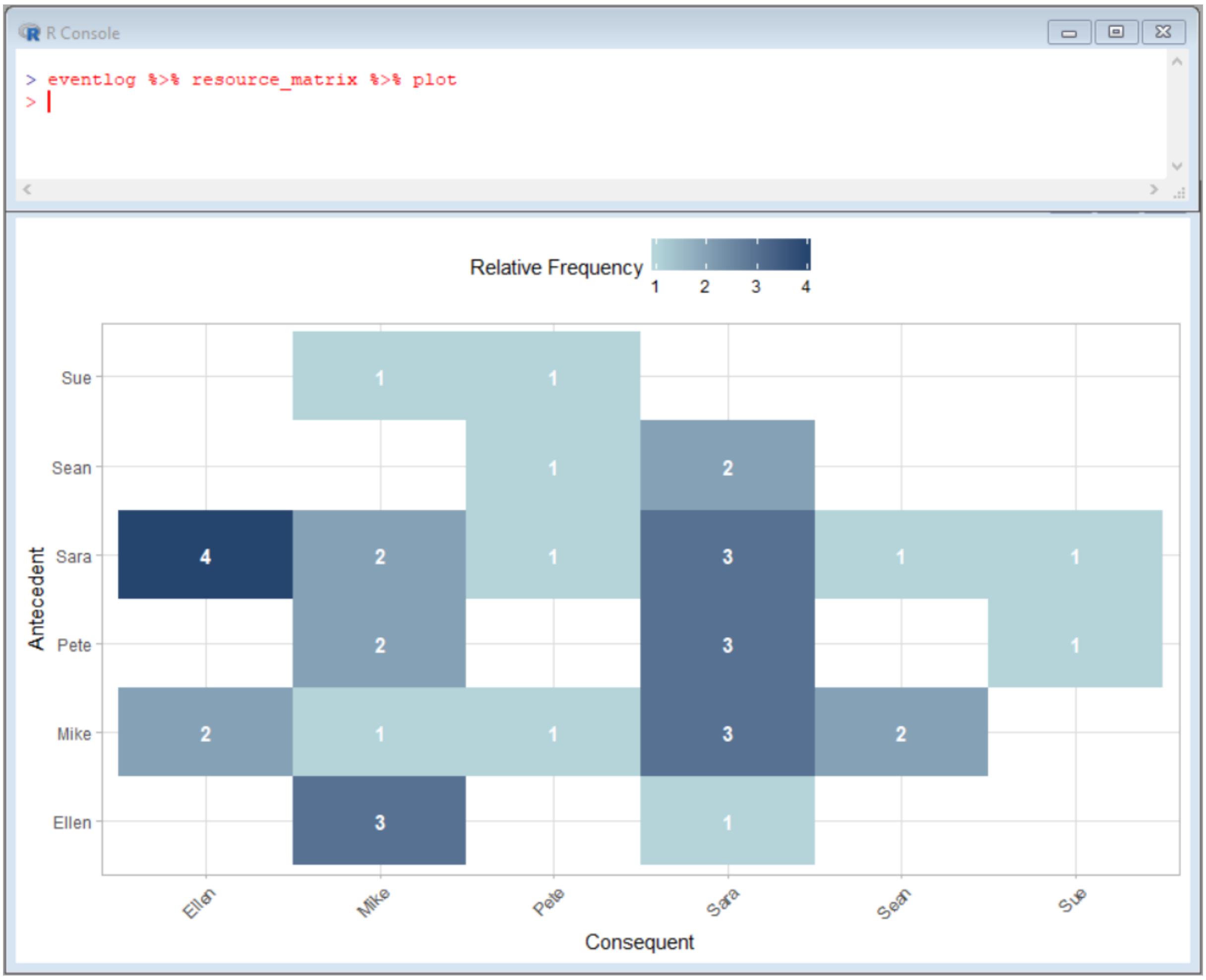
Ou encore avec la commande **trace_explorer( )** :



et la commande **activity_presence**:



et la commande **resource_matrix**:



et la commande **precedence_matrix**:



R Console
> eventlog %>% precedence_matrix %>% plot
Relative Frequency
1 2 3 4 5 6
reinitiate request
register request
examine thoroughly
examine casually
decide
check ticket
Antecedent
check ticket
decide
examine casually
examine thoroughly
pay compensation
reinitiate request
reject request
Consequent

Et pour ceux qui aiment les cheatsheets, voici cella du package bupaR:

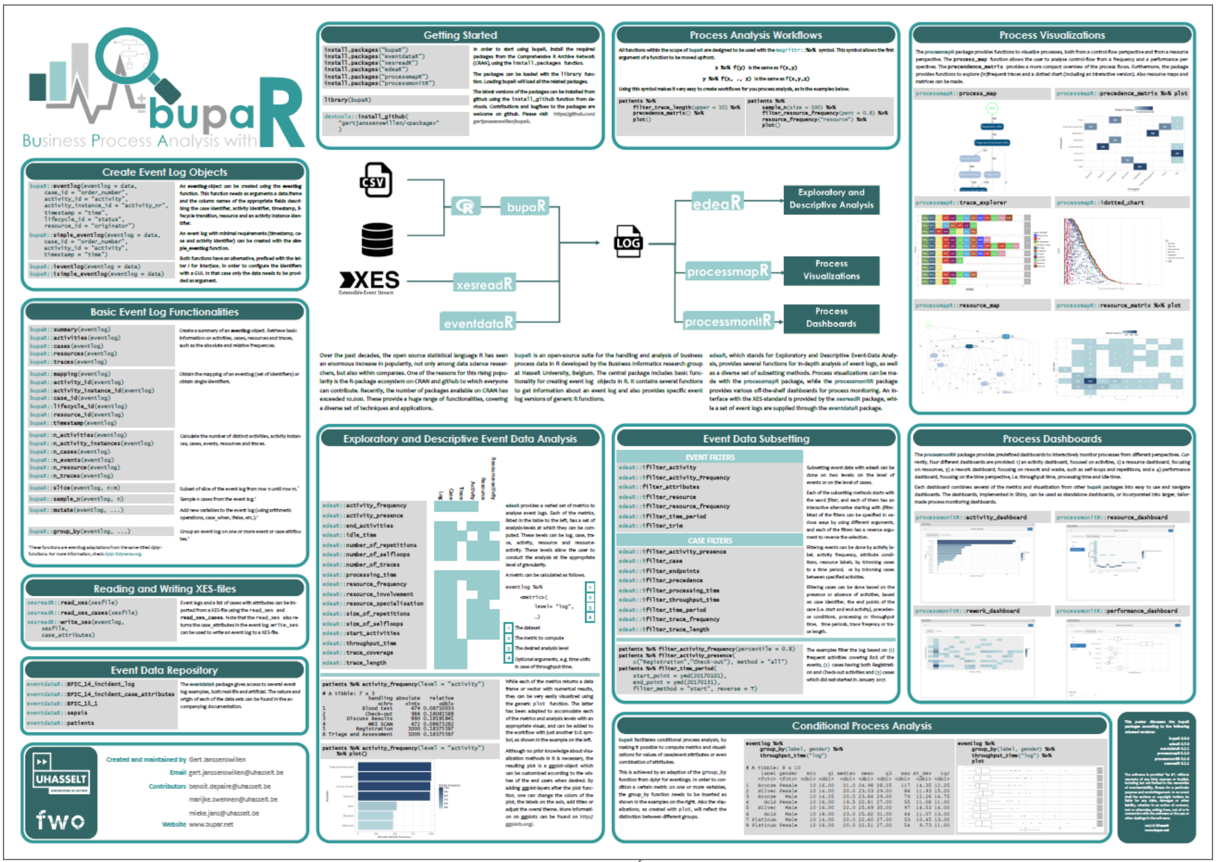


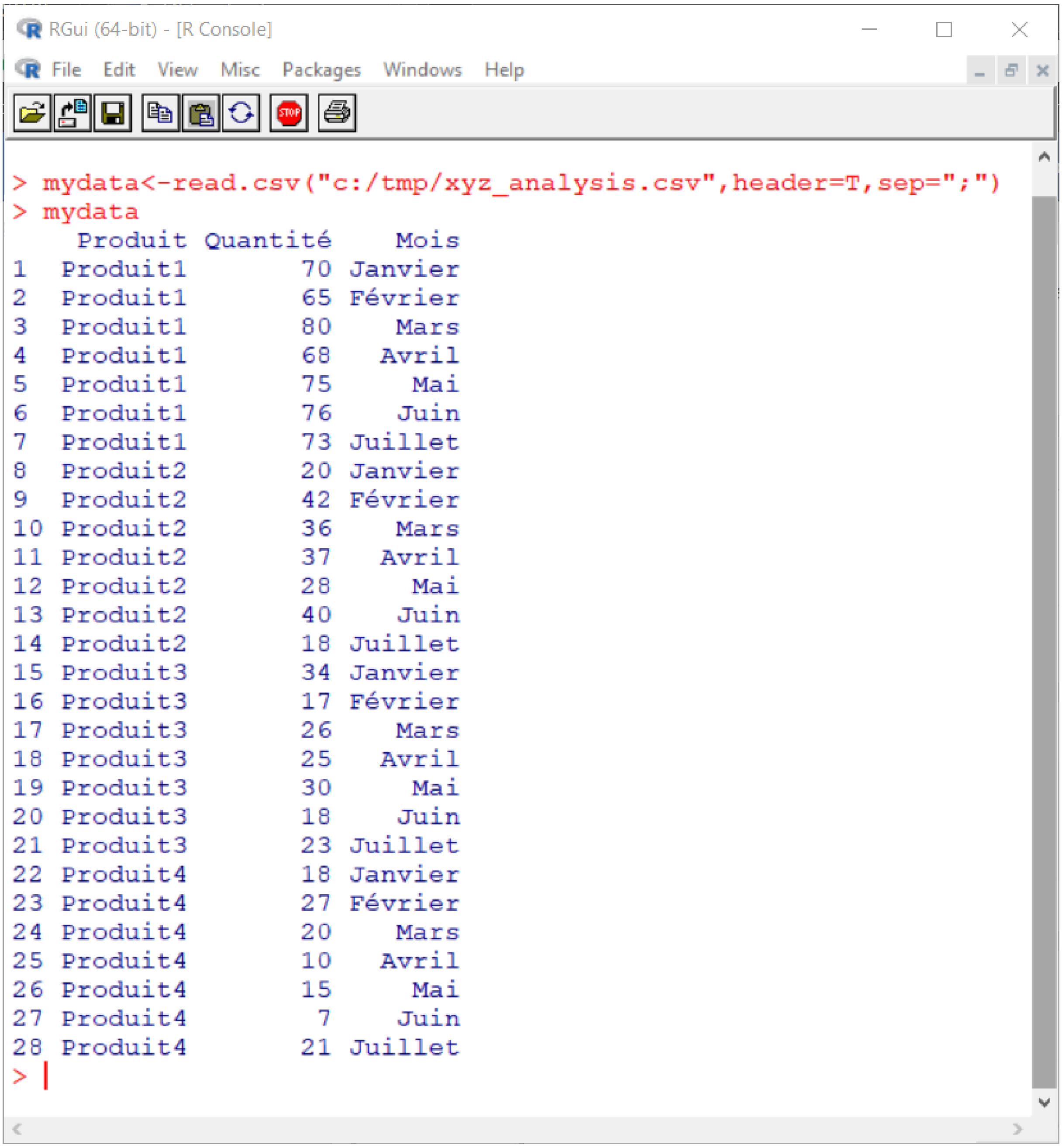
Figure 292 Cheatsheet bupaR

Exemple 319.: Analyse ABC, XYZ et ABC-XYZ

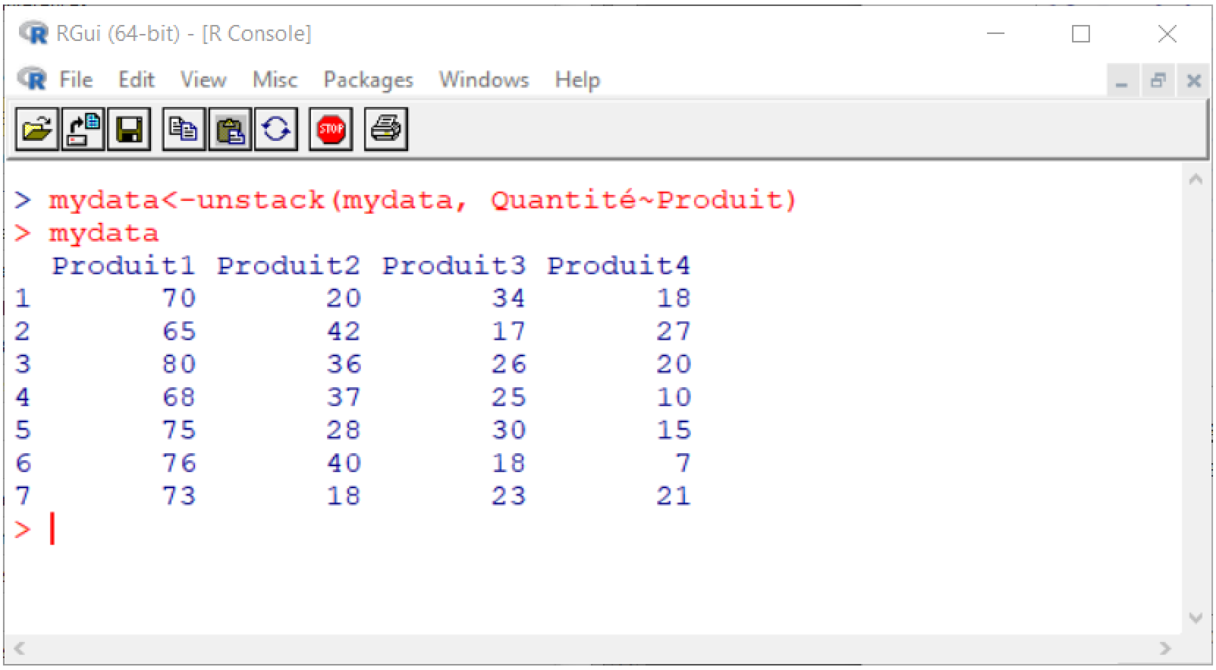
R 3.4.3

Nous avons étudié dans le cours théorique dans le cadre de la logistique des deux analyses semi-quantitatives naïves que sont l'analyse ABC et XYZ ainsi que la mixture ABC-XYZ. Voyons donc si nous obtenons ici les mêmes résultats avec le même jeu de données.

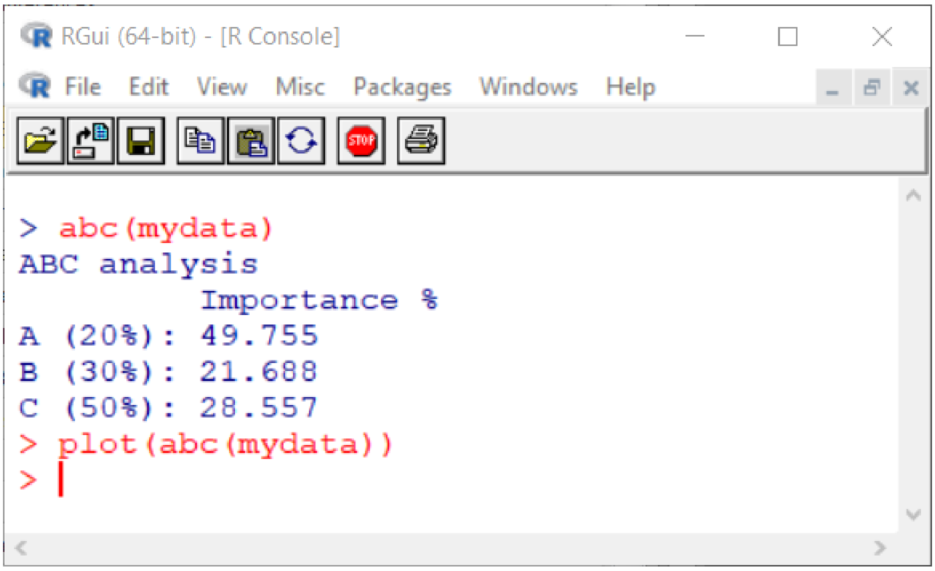
D'abord nous chargeons le package **TStools** et le jeu de données correspondant:



Ensuite nous réarrangeons le jeu de données comme il convient pour les fonctions du package **TStools** avec la fonction **unstack( )** native à R:



Faisons maintenant d'abord une analyse ABC avec la fonction **abc( )** :



Ce résultat venant pour information du calcul suivant (tout à fait discutable!):

```
> mydata<-unstack(mydata,Quantité~Produit)
> mydata
  Produit1 Produit2 Produit3 Produit4
1       70       20       34       18
2       65       42       17       27
3       80       36       26       20
4       68       37       25       10
5       75       28       30       15
6       76       40       18        7
7       73       18       23       21
> x.mean <- colMeans(mydata, na.rm = TRUE)
> x.mean
Produit1 Produit2 Produit3 Produit4
72.42857 31.57143 24.71429 16.85714
> x.rank <- order(x.mean, decreasing=TRUE)
> x.rank
[1] 1 2 3 4
> x.sort <- x.mean[x.rank]
> x.sort
Produit1 Produit2 Produit3 Produit4
72.42857 31.57143 24.71429 16.85714
> x.sort <- (x.sort/sum(x.sort))*100
> x.sort
Produit1 Produit2 Produit3 Produit4
49.75466 21.68793 16.97743 11.57998
>
```

Ce qui donne visuellement:

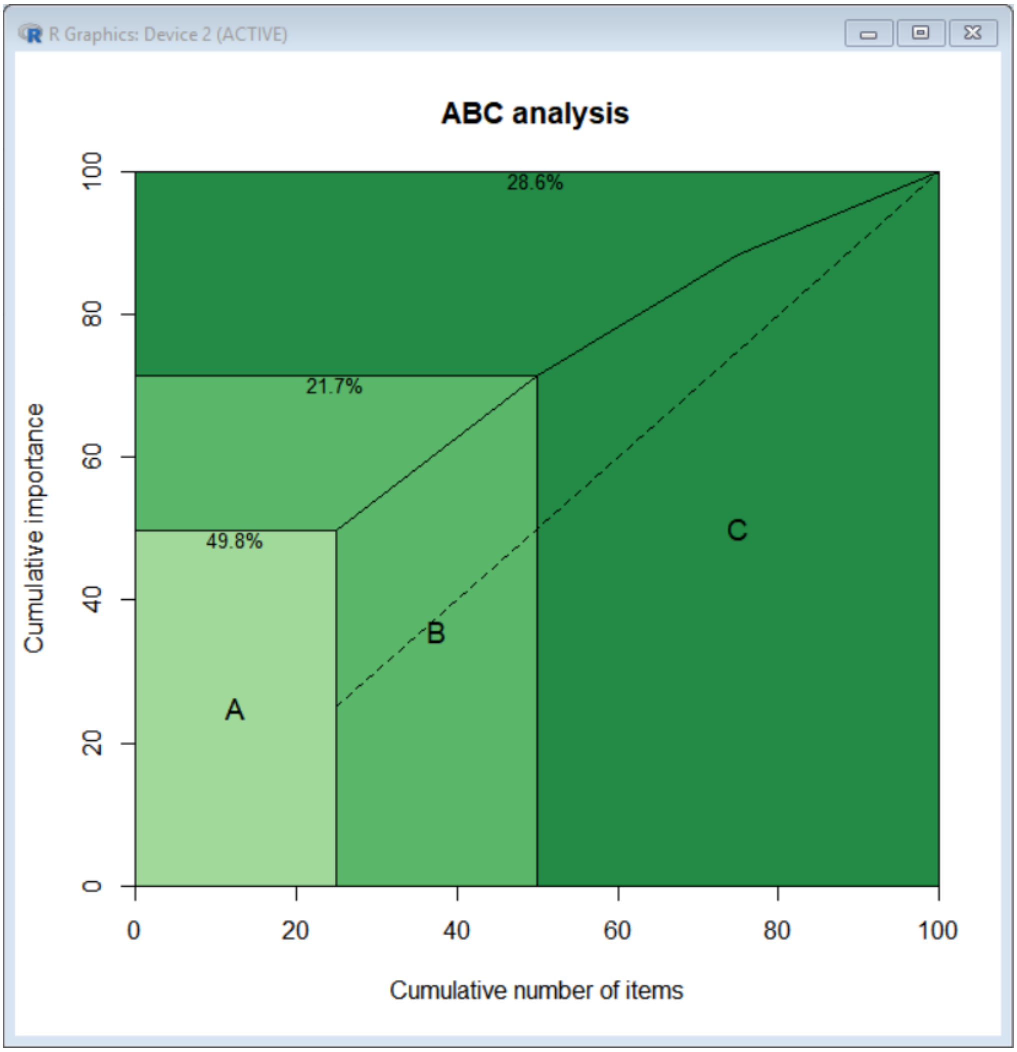


Figure 293 Graph d'analyse ABC

Maintenant faisons l'analyse XYZ avec la fonction **xyz( )** en supposant 2 saisonnalités et la technique naïve (moyenne divisée par RMS):

```
> xyz(mydata,m=2,type="naive")
XYZ analysis
          Errors %
Z (20%):  44.468
Y (30%):  24.284
X (50%):  31.248
> plot(xyz(mydata,m=2,type="naive"))
>
```

Ce qui donne:

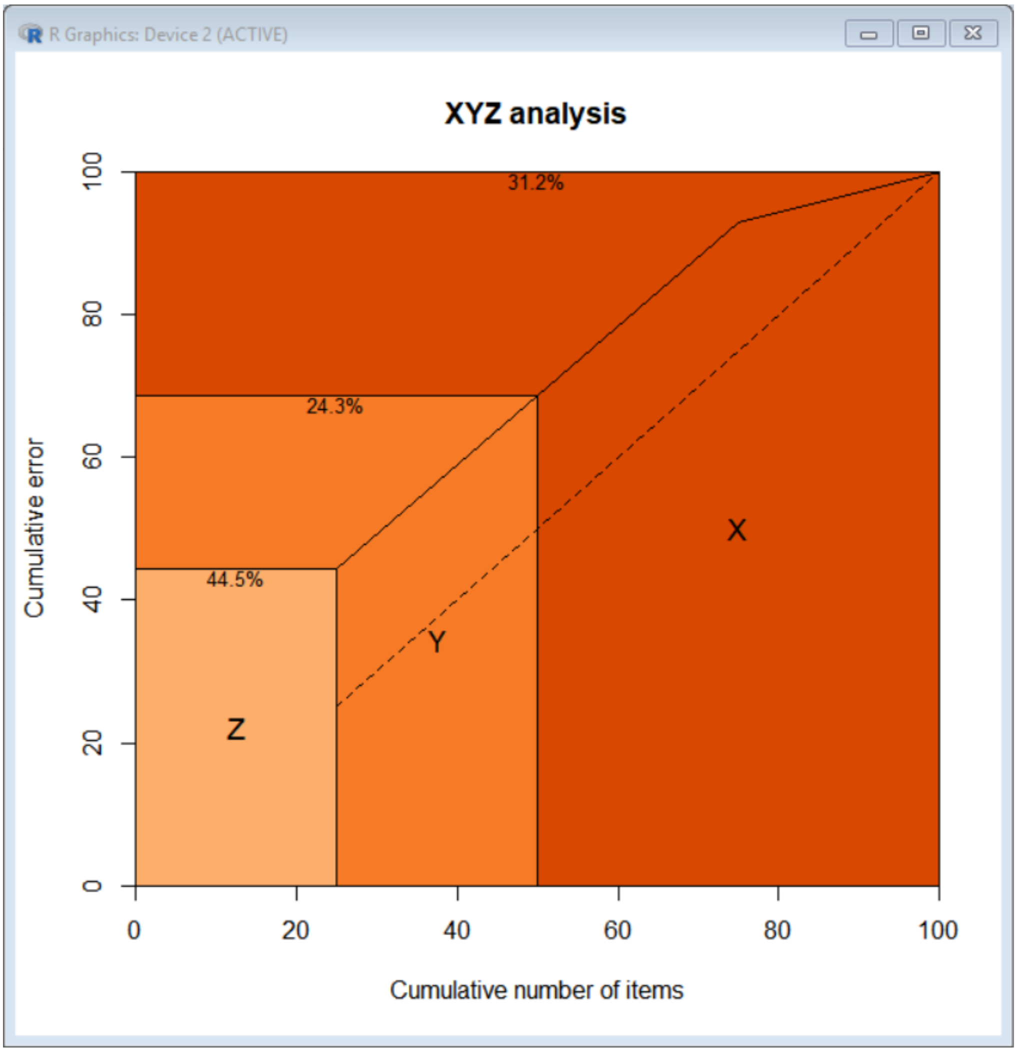
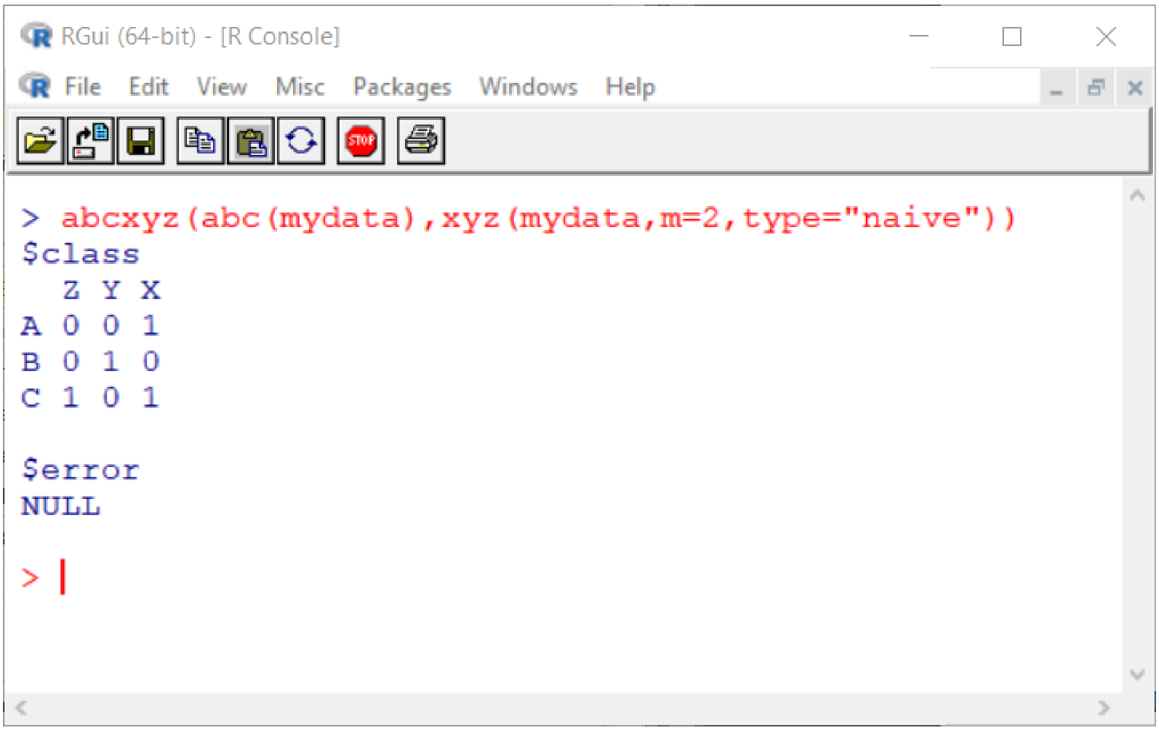


Figure 294 Graph d'analyse XYZ

Maintenant faisons l'analyse mixte avec la fonction **abcxyz( )** :



Ce qui donne:

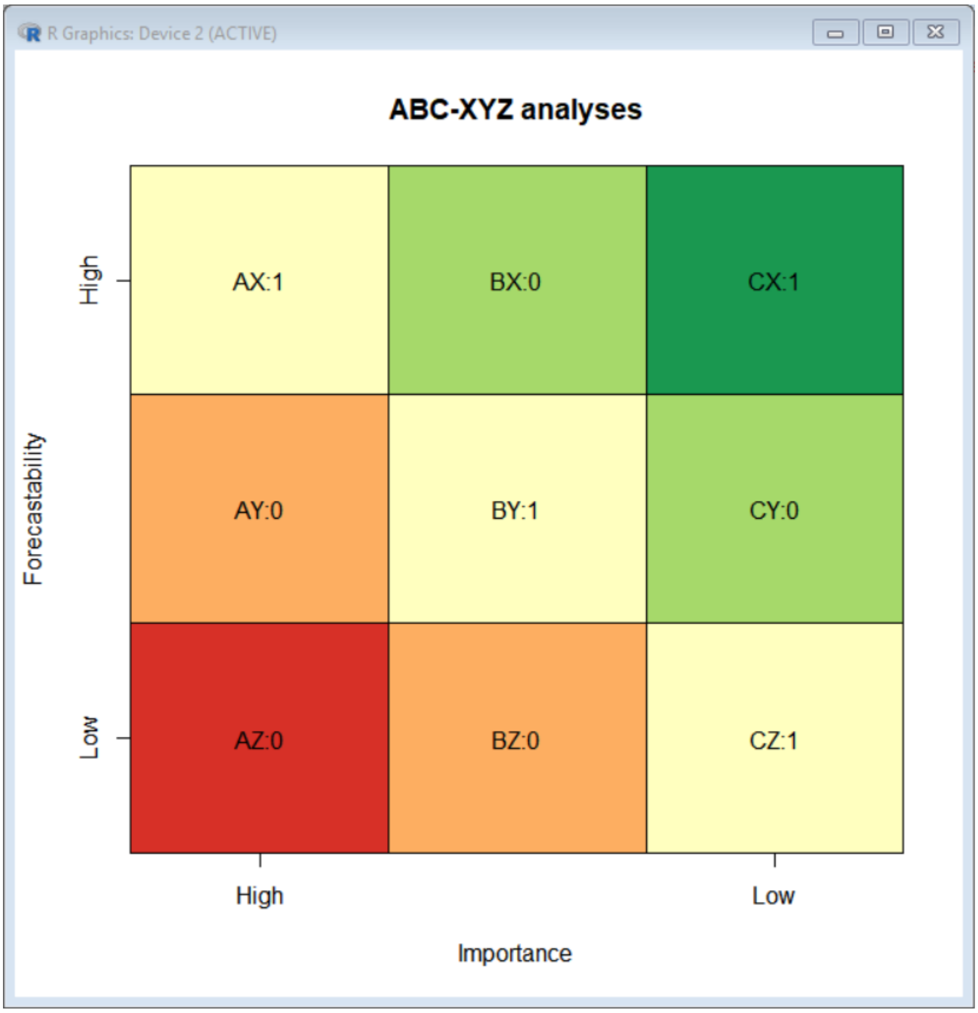


Figure 295 Graph Analyse ABC-XZY

Exemple: Analyse RFM (récence, fréquence, monétaire)

R 4.1

L'analyse RFM (récence, fréquence, monétaire) est une technique basée sur le comportement utilisée pour segmenter les clients en examinant l'historique de leurs transactions telles que

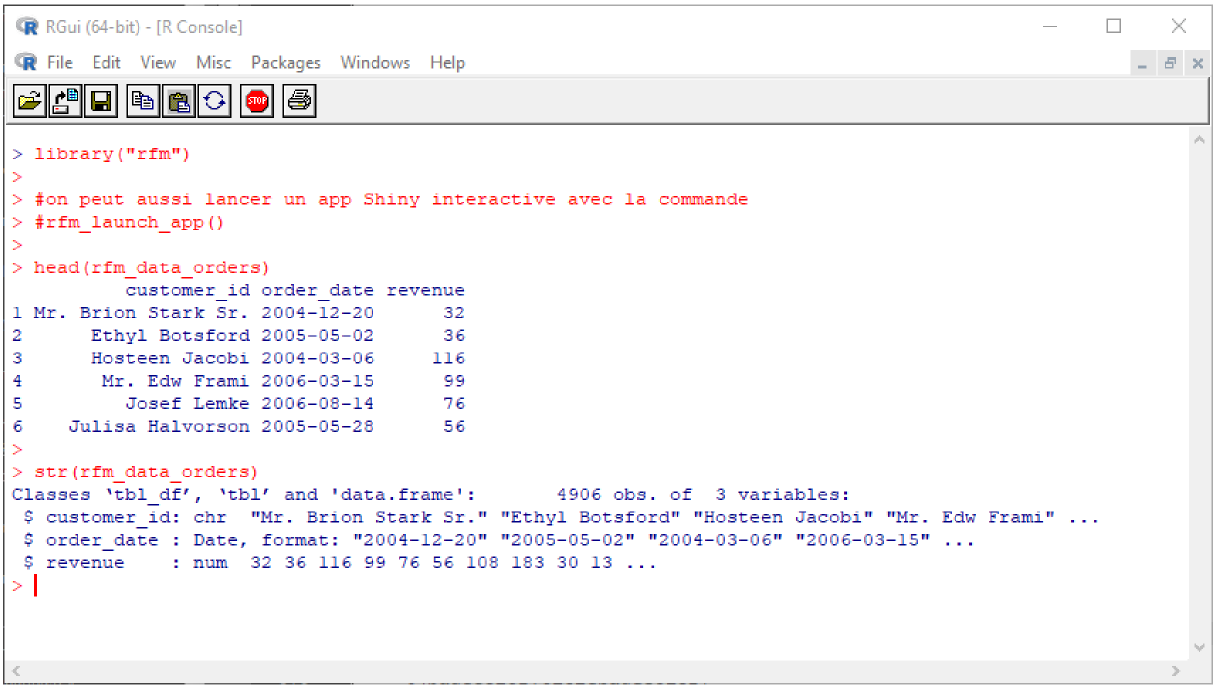
- depuis combien de temps un client a acheté (récence)
- combien de fois ils achètent (fréquence)
- combien le client dépense (monétaire)

Il est basé sur l'axiome marketing selon lequel 80% de votre entreprise provient de 20% de vos clients. RFM aide à identifier les clients les plus susceptibles de répondre aux promotions en les segmentant en différentes catégories.

Pour calculer le score RFM de chaque client, nous avons besoin de données de transaction qui doivent inclure les éléments suivants :

- un identifiant client unique
- date de la transaction/commande
- montant de la transaction/commande

Pour cela nous allons utiliser le package **rfm**:



Alors, comment le score RFM est-il calculé pour chaque client ? Les étapes ci-dessous expliquent le processus :

- Un score de récence est attribué à chaque client en fonction de la date de l'achat le plus récent. Le score est généré en classant les valeurs de récence dans un certain nombre de catégories (la valeur par défaut est 5). Par exemple, si vous utilisez quatre catégories, les clients avec les dates d'achat les plus récentes reçoivent un classement

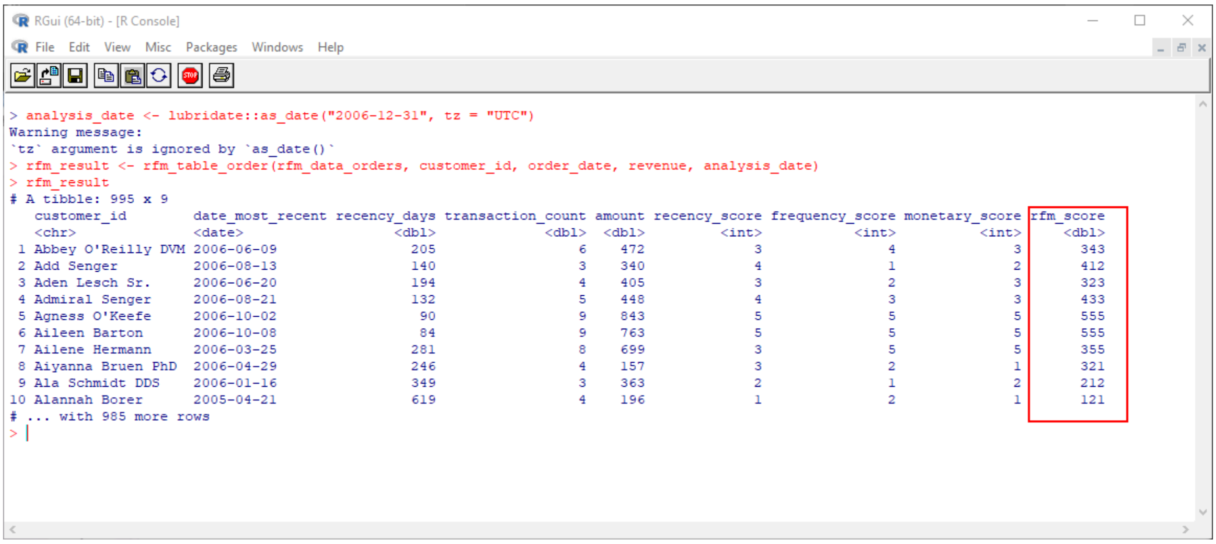
de récence de 4, et ceux avec des dates d'achat dans un passé lointain reçoivent un classement de récence de 1.

- Un classement de fréquence est attribué de la même manière. Les clients dont la fréquence d'achat est élevée se voient attribuer un score plus élevé (4 ou 5) et ceux dont la fréquence d'achat est la plus faible reçoivent un score 1.

- Le score monétaire est attribué sur la base du revenu total généré par le client au cours de la période considérée pour l'analyse. Les clients dont le chiffre d'affaires/le montant de la commande est le plus élevé se voient attribuer un score plus élevé tandis que ceux dont le chiffre d'affaires est le plus faible se voient attribuer un score de 1.

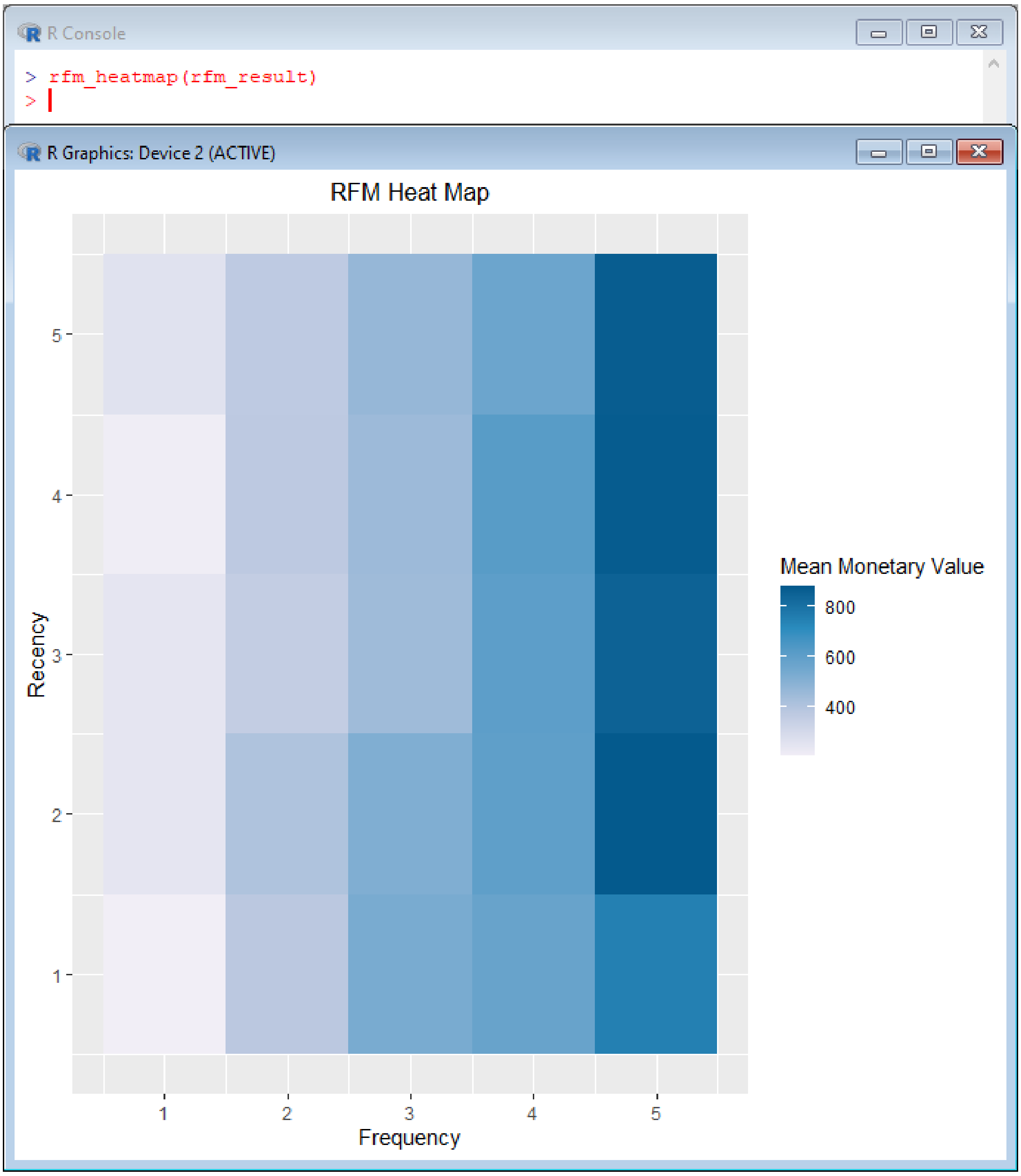
- Un quatrième score, le score RFM est généré, qui est simplement les trois scores individuels concaténés en une seule valeur.

Les clients ayant les scores RFM les plus élevés sont les plus susceptibles de répondre à une offre. Maintenant que nous avons compris comment le score RFM est calculé, il est temps de le mettre en pratique. Nous allons utiliser **rfm_table_order()** pour générer le score de chaque client à partir de l'exemple de jeu de données *rfm_data_orders*

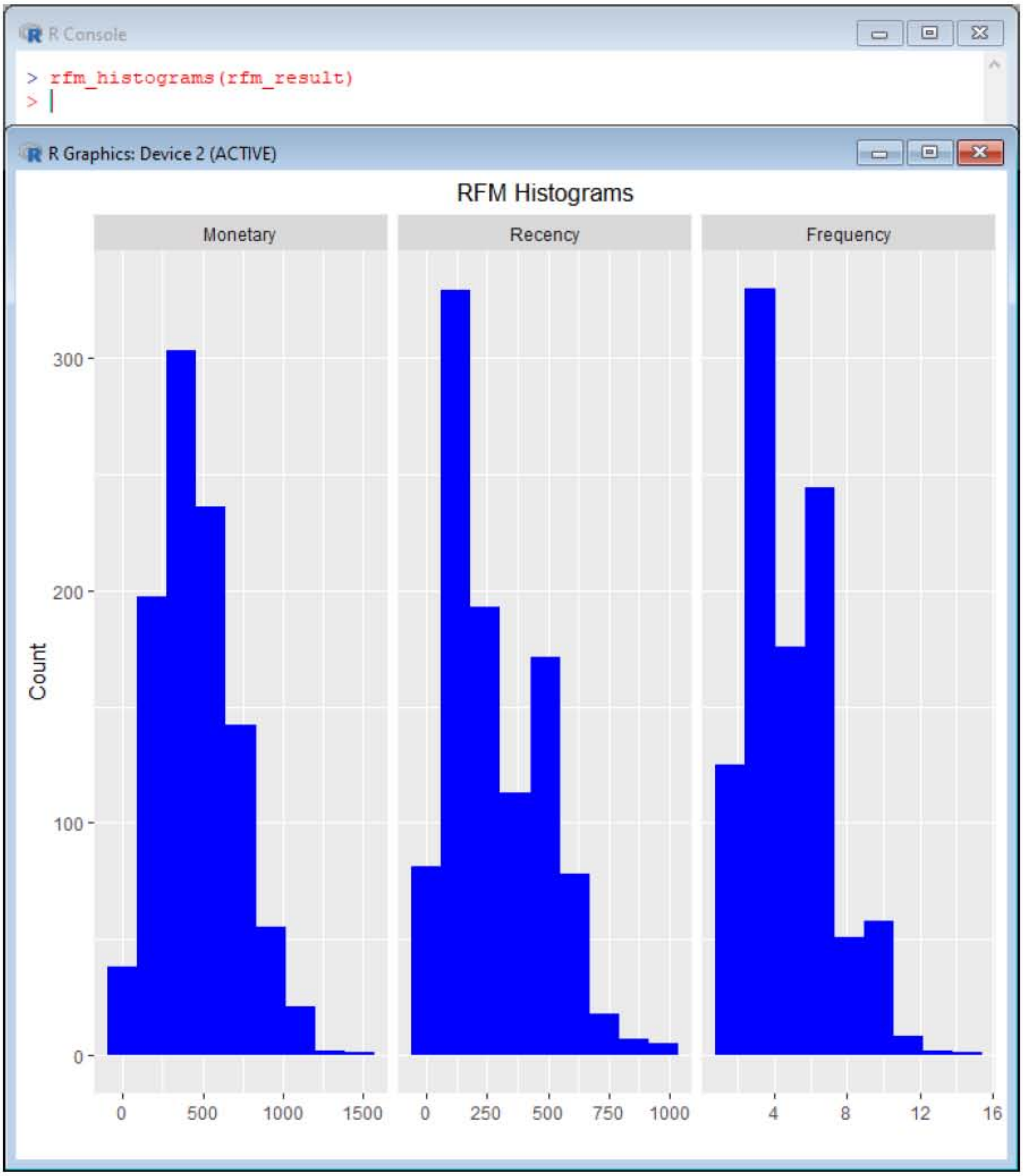



```
> analysis_date <- lubridate::as_date("2006-12-31", tz = "UTC")
Warning message:
`tz` argument is ignored by `as_date()`
> rfm_result <- rfm_table_order(rfm_data_orders, customer_id, order_date, revenue, analysis_date)
> rfm_result
# A tibble: 995 x 9
   customer_id        date_most_recent recency_days transaction_count amount recency_score frequency_score monetary_score rfm_score
   <chr>              <date>                  <dbl>             <dbl>  <dbl>         <int>           <int>          <int>     <dbl>
 1 Abbey O'Reilly DVM 2006-06-09                205                 6    472             3               4              3       343
 2 Add Senger         2006-08-13                140                 3    340             4               1              2       412
 3 Aden Lesch Sr.     2006-06-20                194                 4    405             3               2              3       323
 4 Admiral Senger     2006-08-21                132                 5    448             4               3              3       433
 5 Agness O'Keefe     2006-10-02                 90                 9    843             5               5              5       555
 6 Aileen Barton      2006-10-08                 84                 9    763             5               5              5       555
 7 Ailene Hermann     2006-03-25                281                 8    699             3               5              5       355
 8 Aiyanna Bruen PhD  2006-04-29                246                 4    157             3               2              1       321
 9 Ala Schmidt DDS    2006-01-16                349                 3    363             2               1              2       212
10 Alannah Borer      2005-04-21                619                 4    196             1               2              1       121
# ... with 985 more rows
>
```

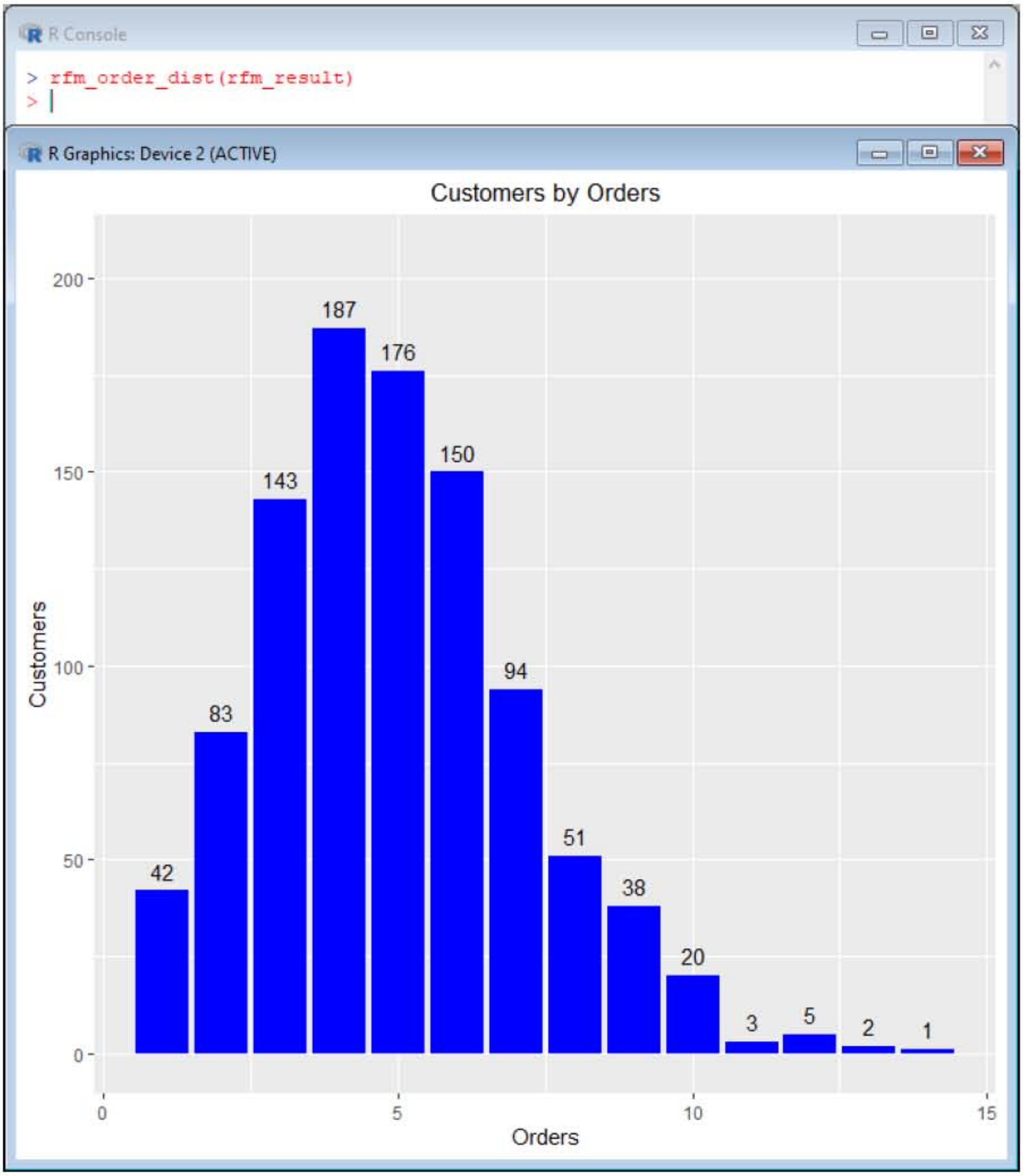
On peut analyser le rfm score avec un heat map:


R Console
> rfm_heatmap(rfm_result)
>
R Graphics: Device 2 (ACTIVE)
RFM Heat Map
Recency
Frequency
Mean Monetary Value
800
600
400
1
2
3
4
5

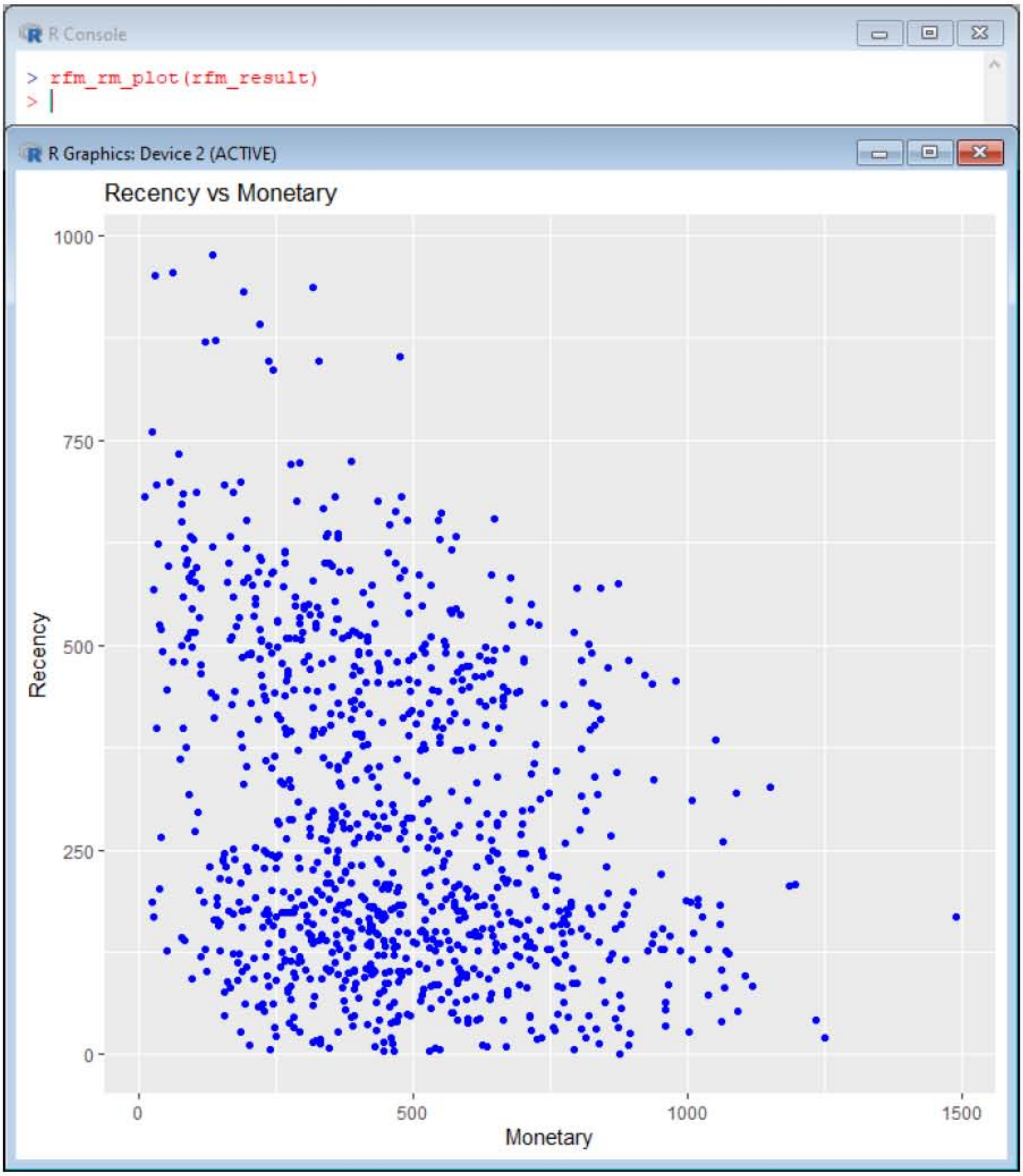
Ou encore:


R Console
> rfm_histograms(rfm_result)
R Graphics: Device 2 (ACTIVE)
RFM Histograms
Monetary
Recency
Frequency
Count
300
200
100
0
0
500
1000
1500
0
250
500
750
1000
4
8
12
16

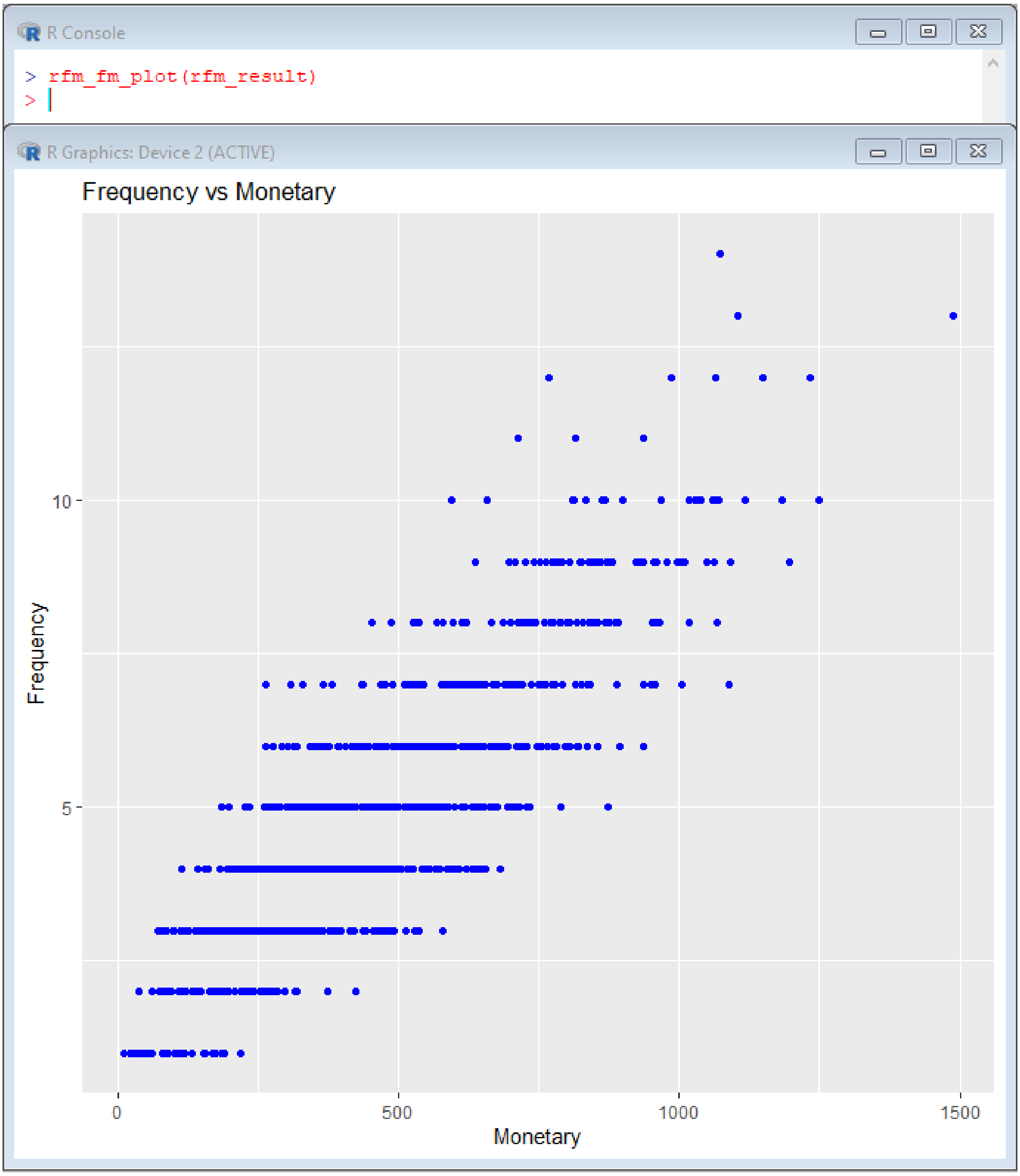
ou encore:


R Console
> rfm_order_dist(rfm_result)
>
R Graphics: Device 2 (ACTIVE)
Customers by Orders
Customers
Orders
200
150
100
50
0
0
5
10
15
42
83
143
187
176
150
94
51
38
20
3
5
2
1

ou encore:


R Console
> rfm_rm_plot(rfm_result)
R Graphics: Device 2 (ACTIVE)
Recency vs Monetary
Recency
Monetary
1000
750
500
250
0
0
500
1000
1500

ou encore:

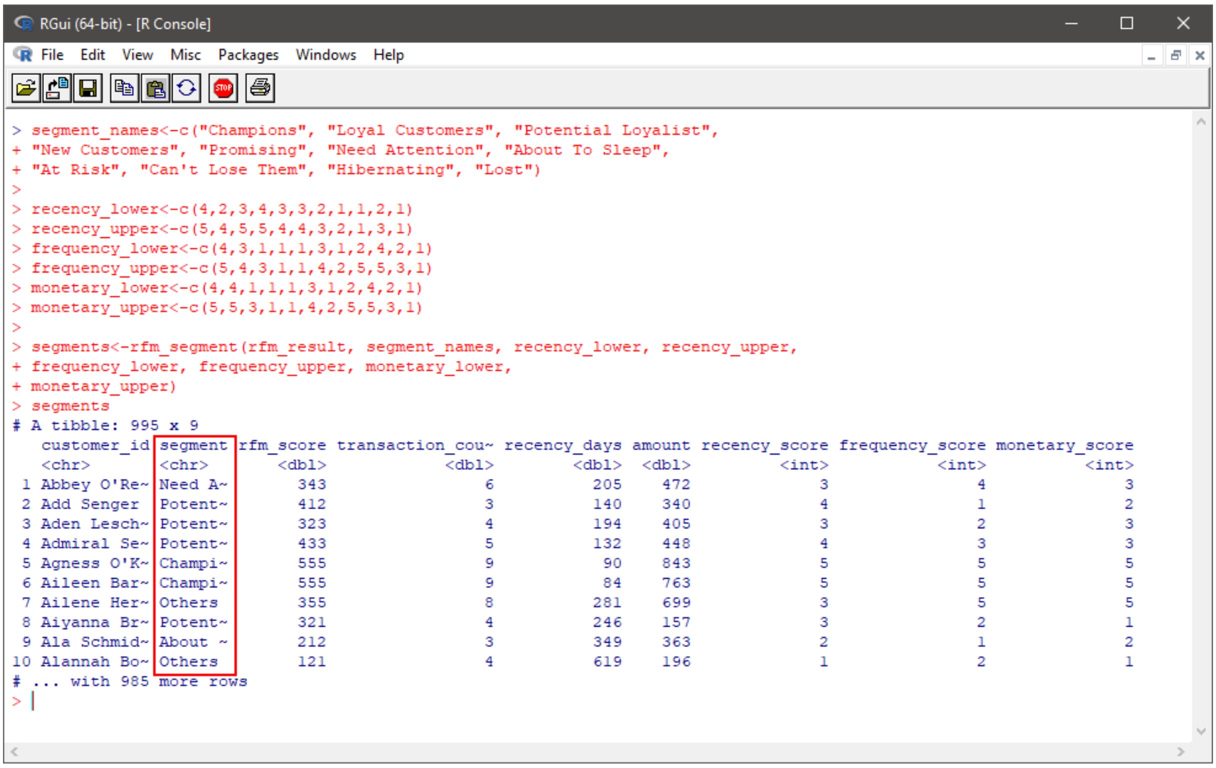


Classifisions maintenant nos clients en fonction de la récence individuelle, de la fréquence et des scores monétaires selon:

Segment	Description	R	F	M
Champions	Bought recently, buy often and spend the most	4 - 5	4 - 5	4 - 5
Loyal Customers	Spend good money. Responsive to promotions	2 - 5	3 - 5	3 - 5
Potential Loyalist	Recent customers, spent good amount, bought more than once	3 - 5	1 - 3	1 - 3
New Customers	Bought more recently, but not often	4 - 5	<= 1	<= 1
Promising	Recent shoppers, but haven't spent much	3 - 4	<= 1	<= 1
Need Attention	Above average recency, frequency & monetary values	2 - 3	2 - 3	2 - 3
About To Sleep	Below average recency, frequency & monetary values	2 - 3	<= 2	<= 2
At Risk	Spent big money, purchased often but long time ago	<= 2	2 - 5	2 - 5
Can't Lose Them	Made big purchases and often, but long time ago	<= 1	4 - 5	4 - 5
Hibernating	Low spenders, low frequency, purchased long time ago	1 - 2	1 - 2	1 - 2

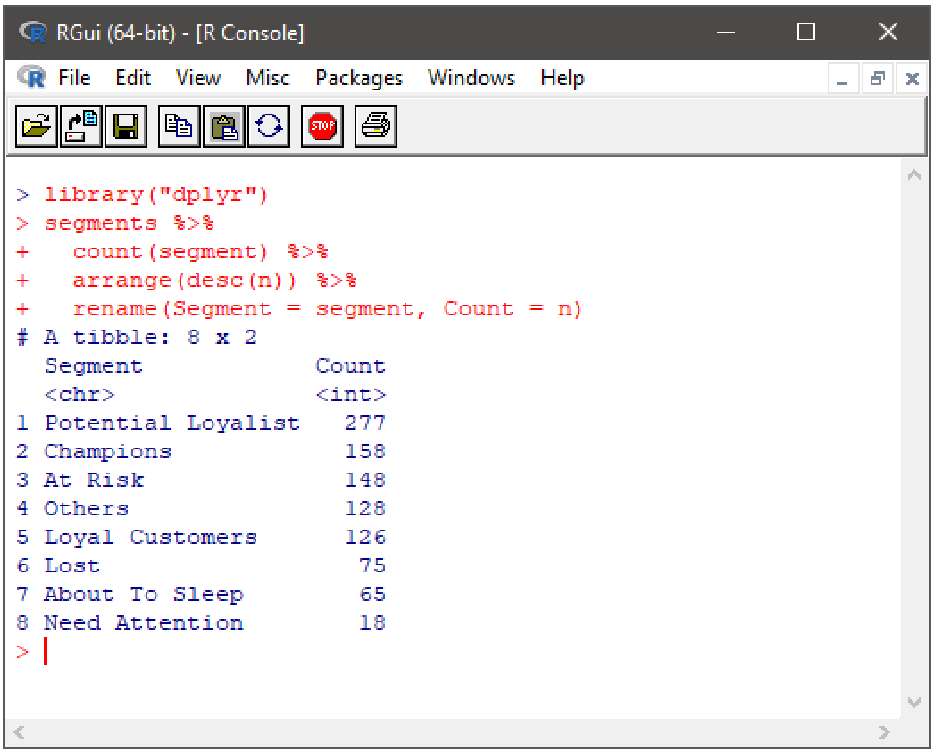
Lost	Lowest recency, frequency & monetary scores	<= 2	<= 2	<= 2

Pour faire cela nous écrirons dans R:



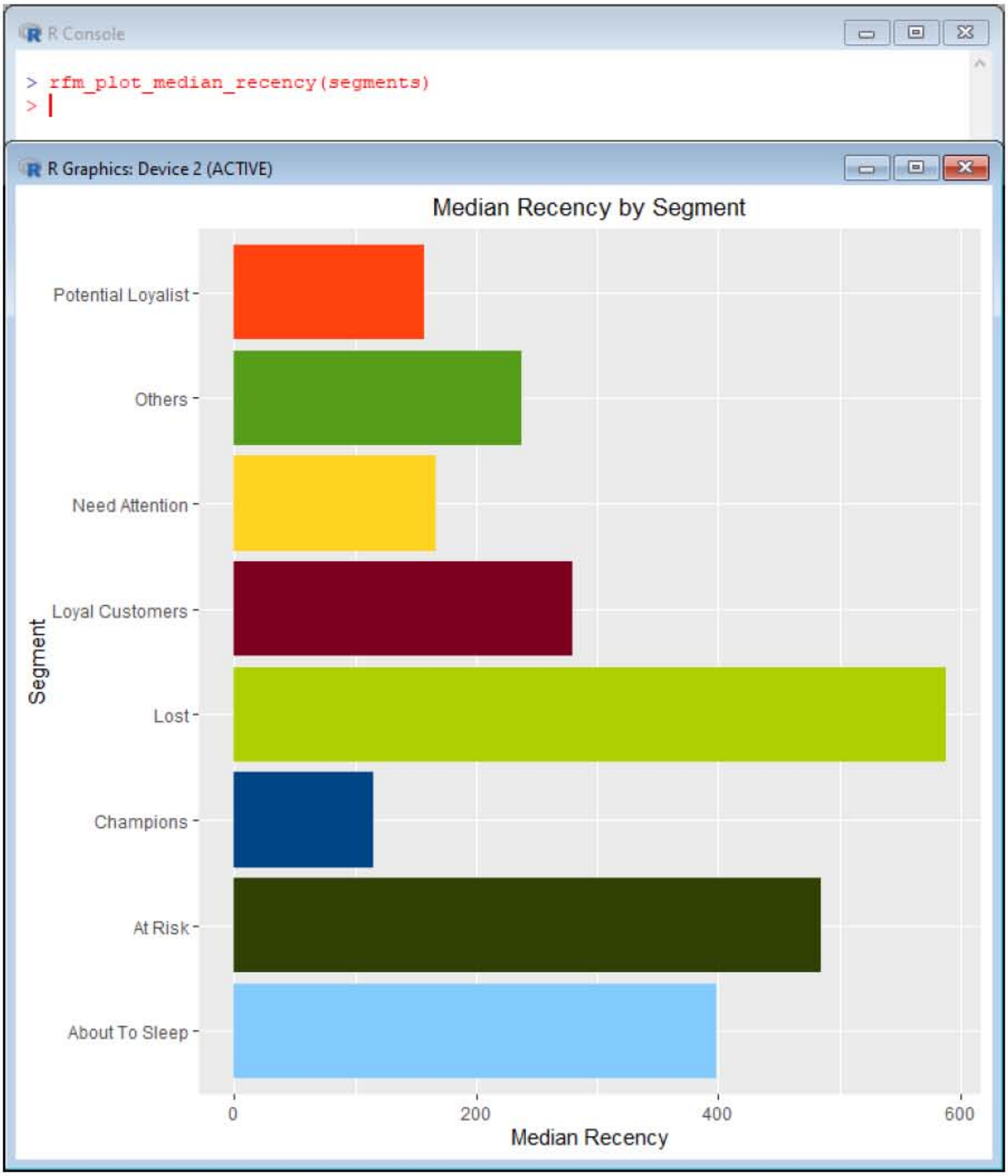
```
> segment_names<-c("Champions", "Loyal Customers", "Potential Loyalist",
+ "New Customers", "Promising", "Need Attention", "About To Sleep",
+ "At Risk", "Can't Lose Them", "Hibernating", "Lost")
>
> recency_lower<-c(4,2,3,4,3,3,2,1,1,2,1)
> recency_upper<-c(5,4,5,5,4,4,3,2,1,3,1)
> frequency_lower<-c(4,3,1,1,1,3,1,2,4,2,1)
> frequency_upper<-c(5,4,3,1,1,4,2,5,5,3,1)
> monetary_lower<-c(4,4,1,1,1,3,1,2,4,2,1)
> monetary_upper<-c(5,5,3,1,1,4,2,5,5,3,1)
>
> segments<-rfm_segment(rfm_result, segment_names, recency_lower, recency_upper,
+ frequency_lower, frequency_upper, monetary_lower,
+ monetary_upper)
> segments
# A tibble: 995 x 9
   customer_id segment rfm_score transaction_cou~ recency_days amount recency_score frequency_score monetary_score
   <chr>       <chr>       <dbl>            <dbl>        <dbl>  <dbl>         <int>           <int>          <int>
 1 Abbey O'Re~ Need A~       343                6          205    472             3               4              3
 2 Add Senger  Potent~       412                3          140    340             4               1              2
 3 Aden Lesch~ Potent~       323                4          194    405             3               2              3
 4 Admiral Se~ Potent~       433                5          132    448             4               3              3
 5 Agness O'K~ Champi~       555                9           90    843             5               5              5
 6 Aileen Bar~ Champi~       555                9           84    763             5               5              5
 7 Ailene Her~ Others        355                8          281    699             3               5              5
 8 Aiyanna Br~ Potent~       321                4          246    157             3               2              1
 9 Ala Schmid~ About ~       212                3          349    363             2               1              2
10 Alannah Bo~ Others        121                4          619    196             1               2              1
# ... with 985 more rows
>
```

On peut réarranger cela avec dplyr:

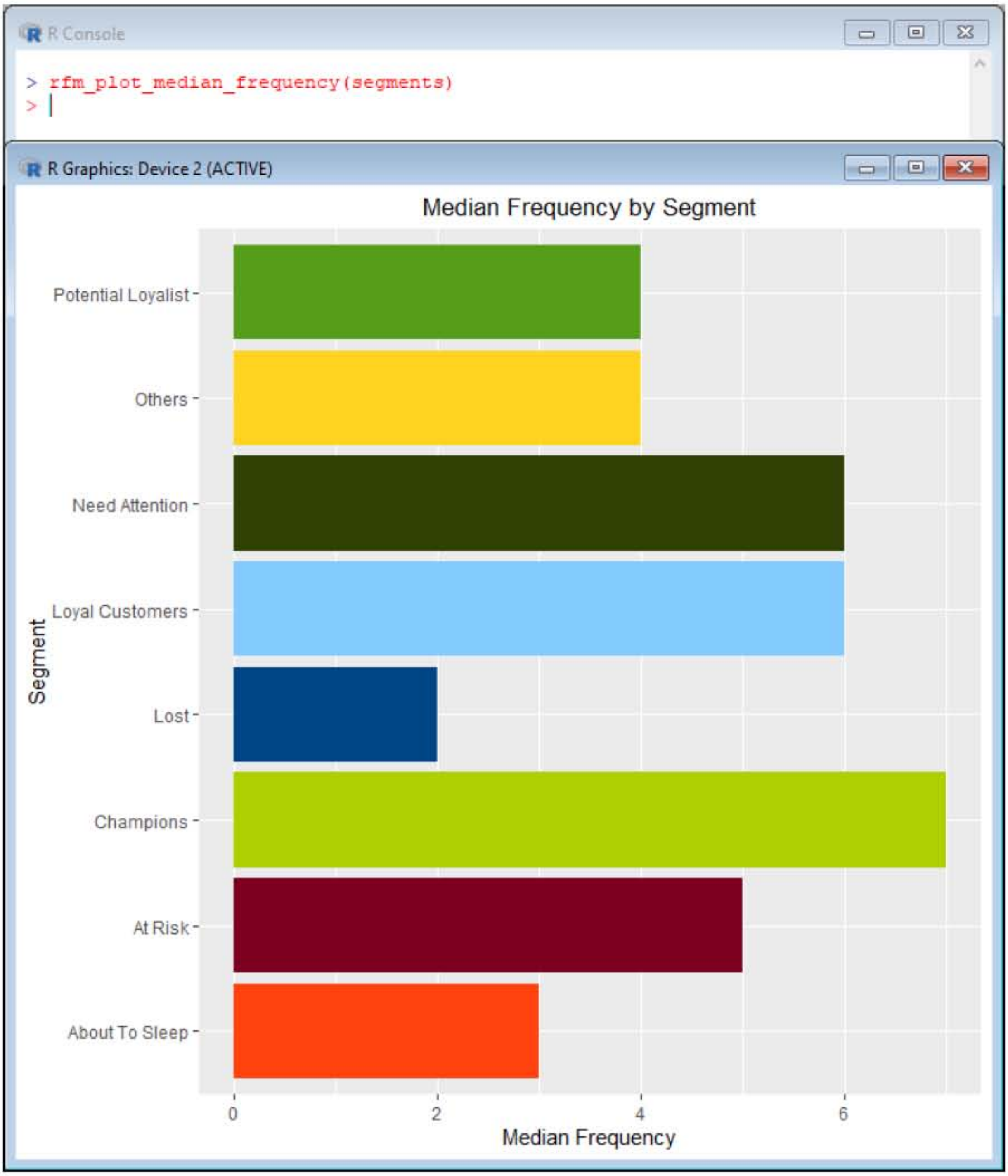


```
> library("dplyr")
> segments %>%
+   count(segment) %>%
+   arrange(desc(n)) %>%
+   rename(Segment = segment, Count = n)
# A tibble: 8 x 2
  Segment            Count
  <chr>              <int>
1 Potential Loyalist   277
2 Champions            158
3 At Risk              148
4 Others               128
5 Loyal Customers      126
6 Lost                  75
7 About To Sleep        65
8 Need Attention        18
>
```

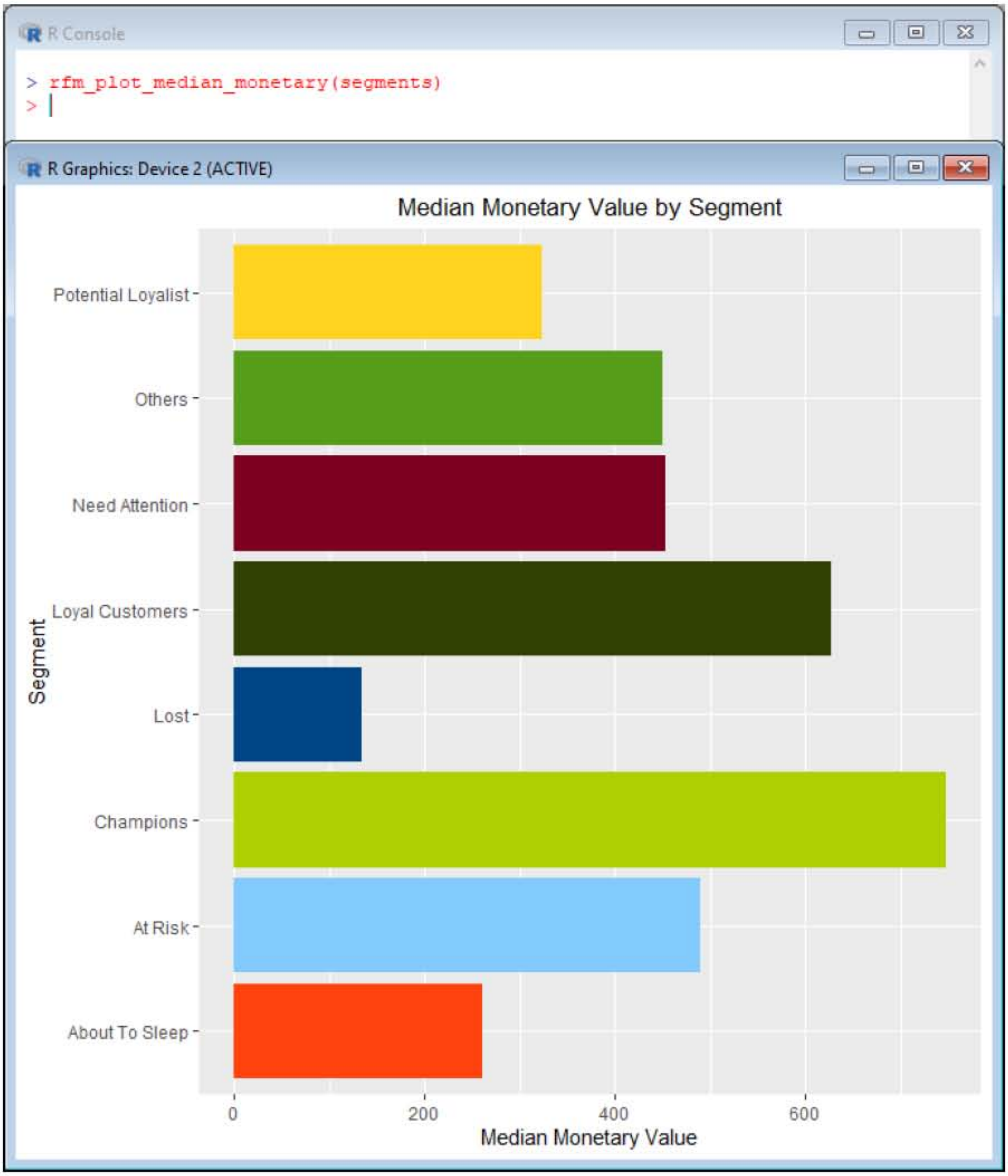
Et ensuite en faire des graphes divers et variés:


R Console
> rfm_plot_median_recency(segments)
>
R Graphics: Device 2 (ACTIVE)
Median Recency by Segment
Potential Loyalist
Others
Need Attention
Loyal Customers
Lost
Champions
At Risk
About To Sleep
Segment
0
200
400
600
Median Recency

ou:


R Console
> rfm_plot_median_frequency(segments)
R Graphics: Device 2 (ACTIVE)
Median Frequency by Segment
Potential Loyalist
Others
Need Attention
Loyal Customers
Lost
Champions
At Risk
About To Sleep
Segment
0
2
4
6
Median Frequency

ou:

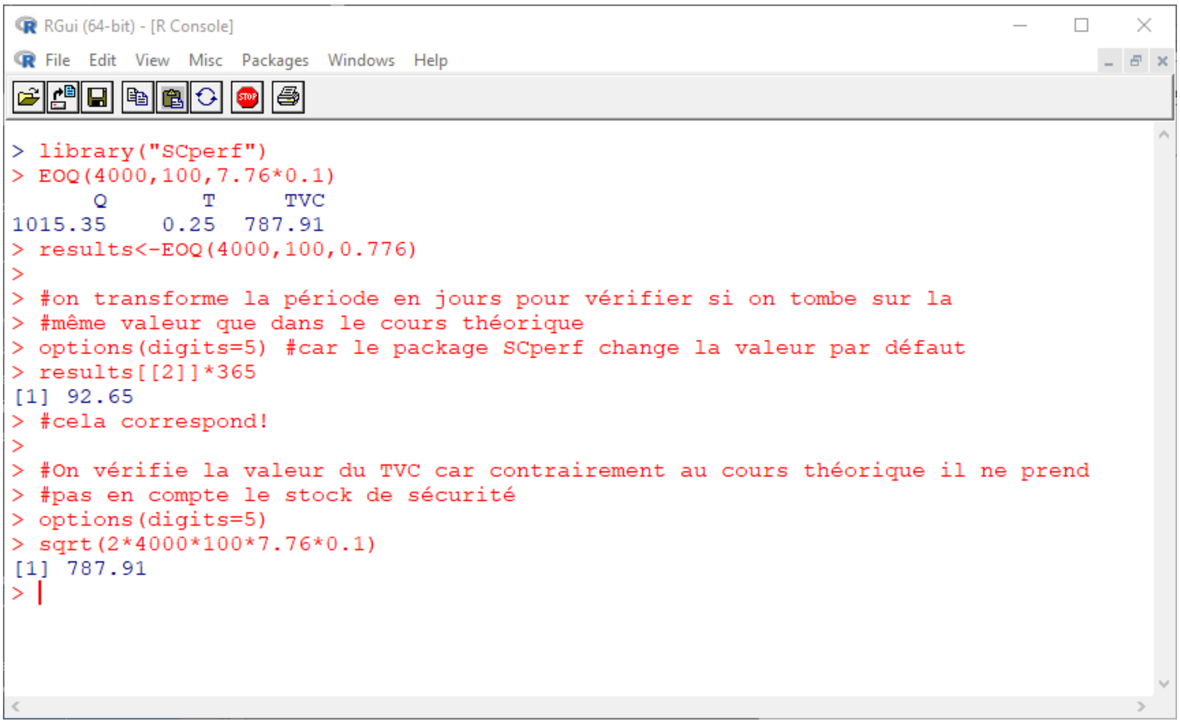

R Console
> rfm_plot_median_monetary(segments)
>
R Graphics: Device 2 (ACTIVE)
Median Monetary Value by Segment
Potential Loyalist
Others
Need Attention
Loyal Customers
Lost
Champions
At Risk
About To Sleep
Segment
0
200
400
600
Median Monetary Value

Exemple 320.: Modèles de Wilson

R 3.5.1

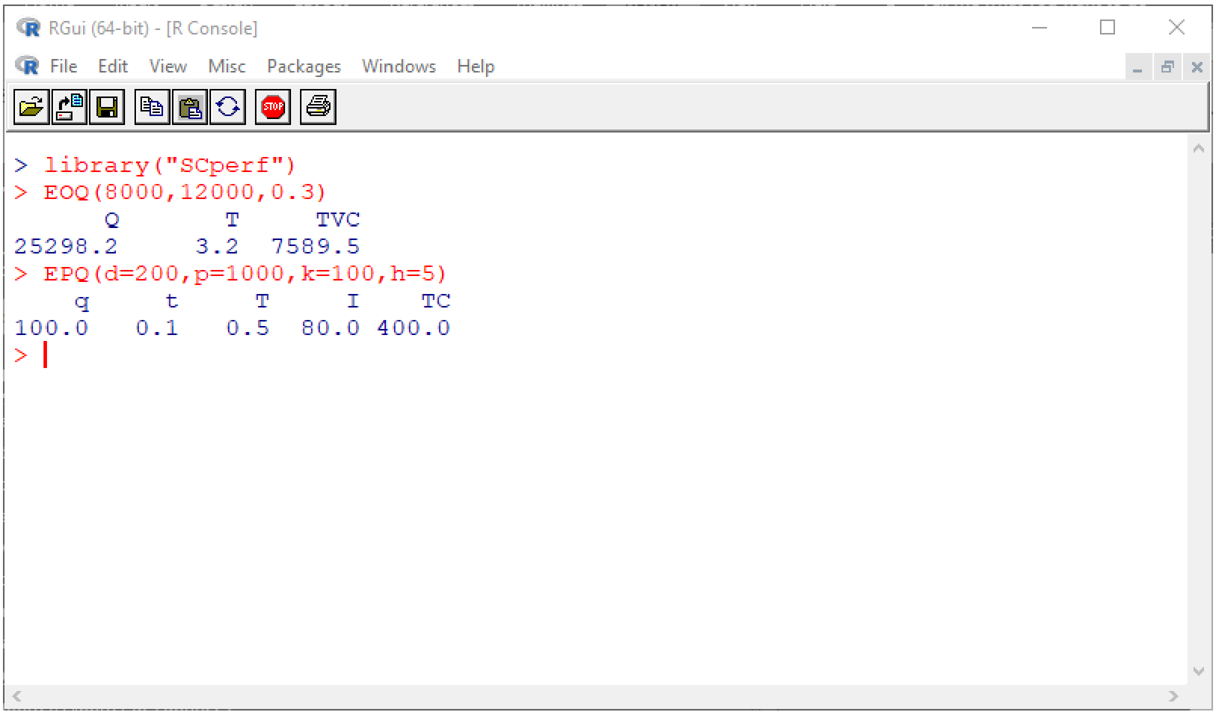
Nous allons avec le package **SCperf** et sa fonction **EOQ( )** faire deux calculs classiques du domaine de la logistique (supply chain).

D'abord les calculs (indicateurs classiques) du Modèle de Wilson avec réapprovisionnement instantanée:



Les résultats correspondent donc bien à ceux du cours théorique tant au niveau de la quantité de commande optimale, quand la période de réapprovisionnement ou le coût de réapprovisionnement optimal!

Maintenant voyons pour le modèle de Wilson avec taux de production et taux de consommation en reprenant les mêmes valeurs que dans l'exemple du cours théorique:



Les résultats correspondent là aussi bien à ce que nous avions calculé à la main dans le cours théorique!

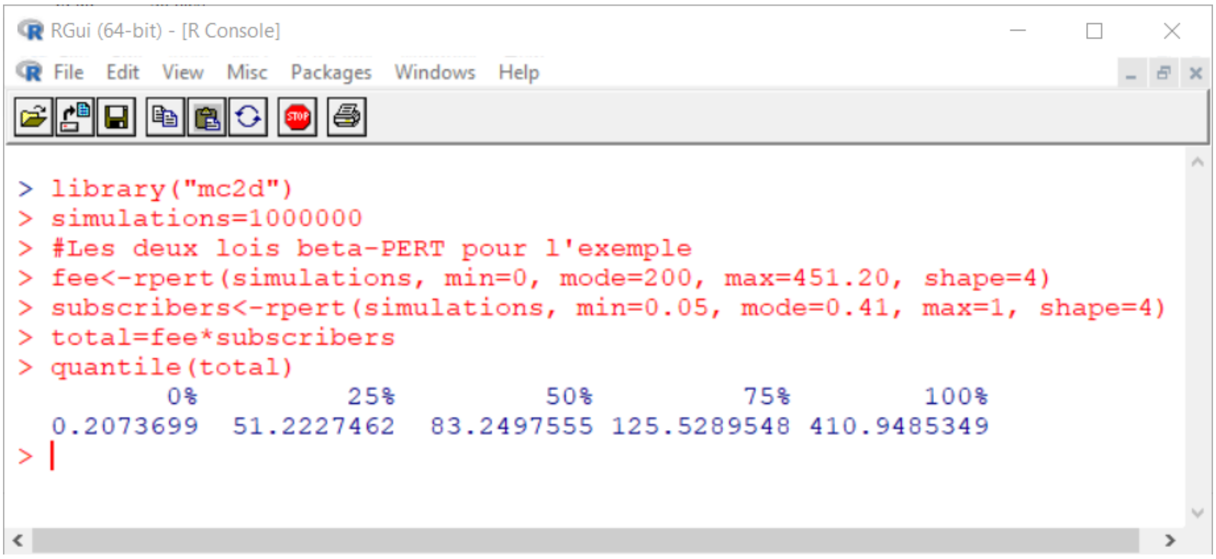
Exemple 321.: Simulations Beta-PERT (three point estimates)

R 3.4.3

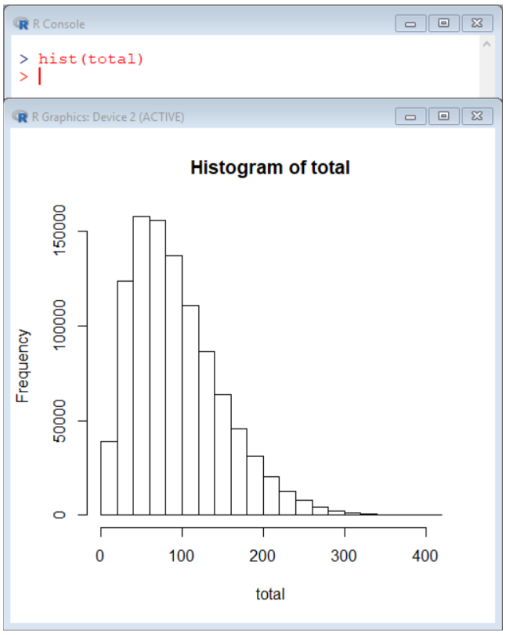
Nous avons vu dans le cours théorique que la distribution Beta-PERT est très utile en gestion de projets. Mais de par le fait qu'elle est bornée et qu'elle s'estime qu'avec trois points, elle est très utile en économie et politique quand les données (enquêtes/sondages) sont très lacunaires et manquent de rigueur scientifique.

Nous allons donc voir ici comment effectuer une simulation Monte Carlo avec la distribution Beta-PERT et aussi comment tracer deux graphiques typiques correspondants.

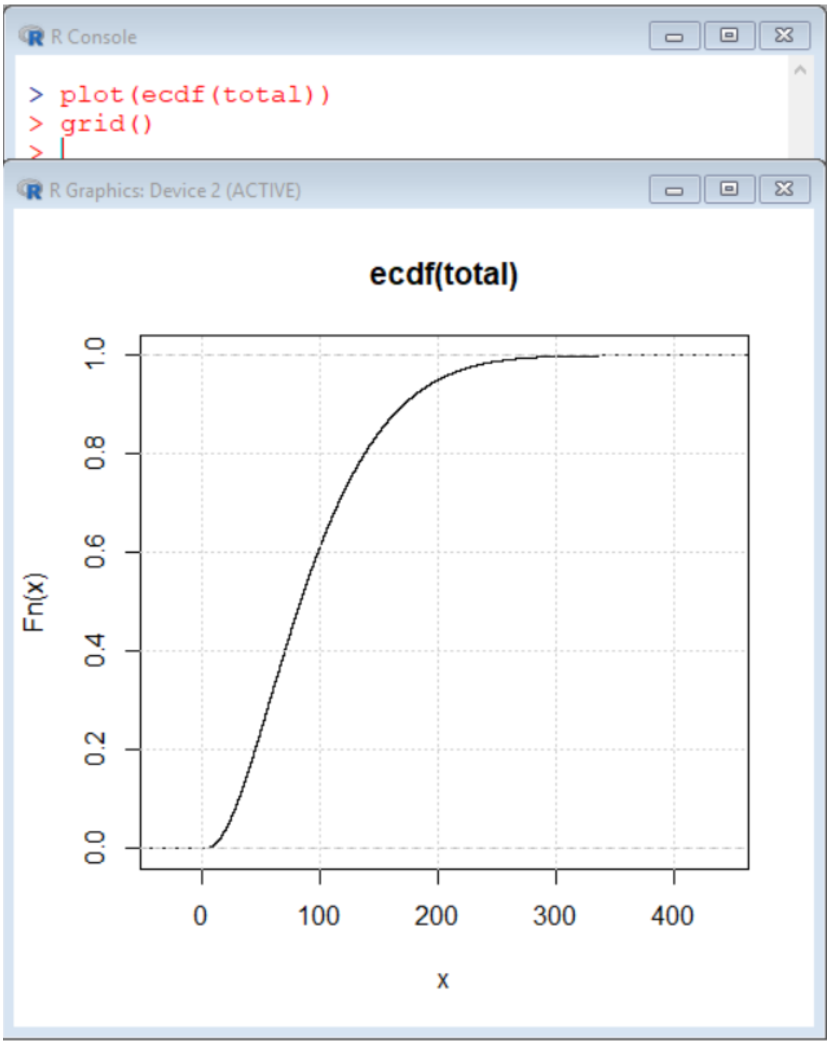
Donc pour les simulations avec par exemple la multiplication de deux lois Beta-PERT, nous utilisons le package **mc2d** et sa fonction **rpert( )** :



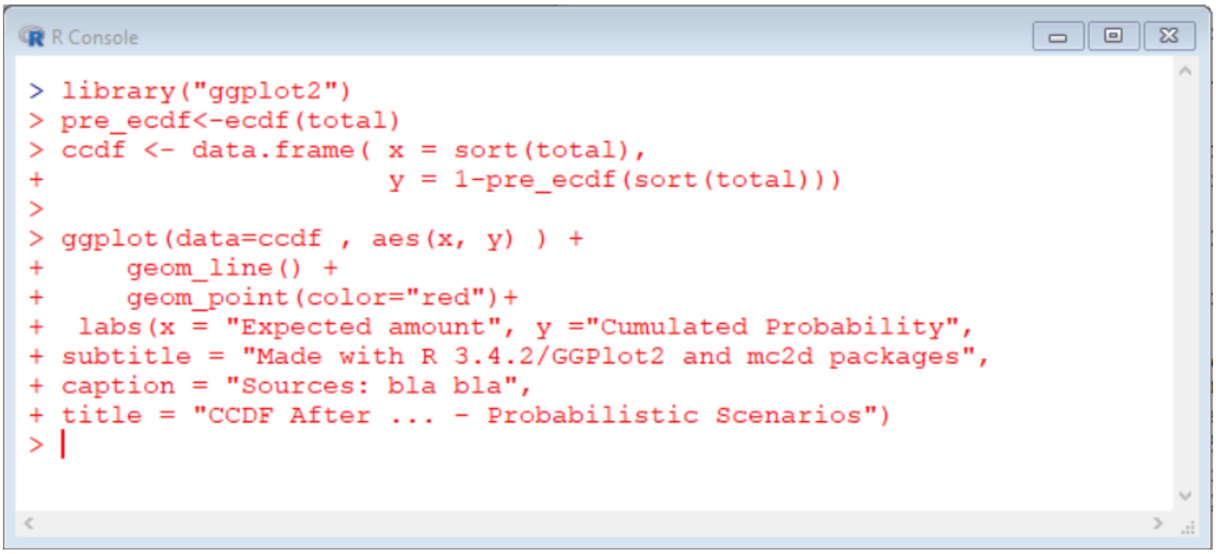
Et enfin on peut faire les quelques graphiques d'usage avec l'histogramme:



Et l'ECDF:



Et enfin pour le confort de certains lecteurs, avec sa CCDF avec ggplot2:



Ce qui donne:

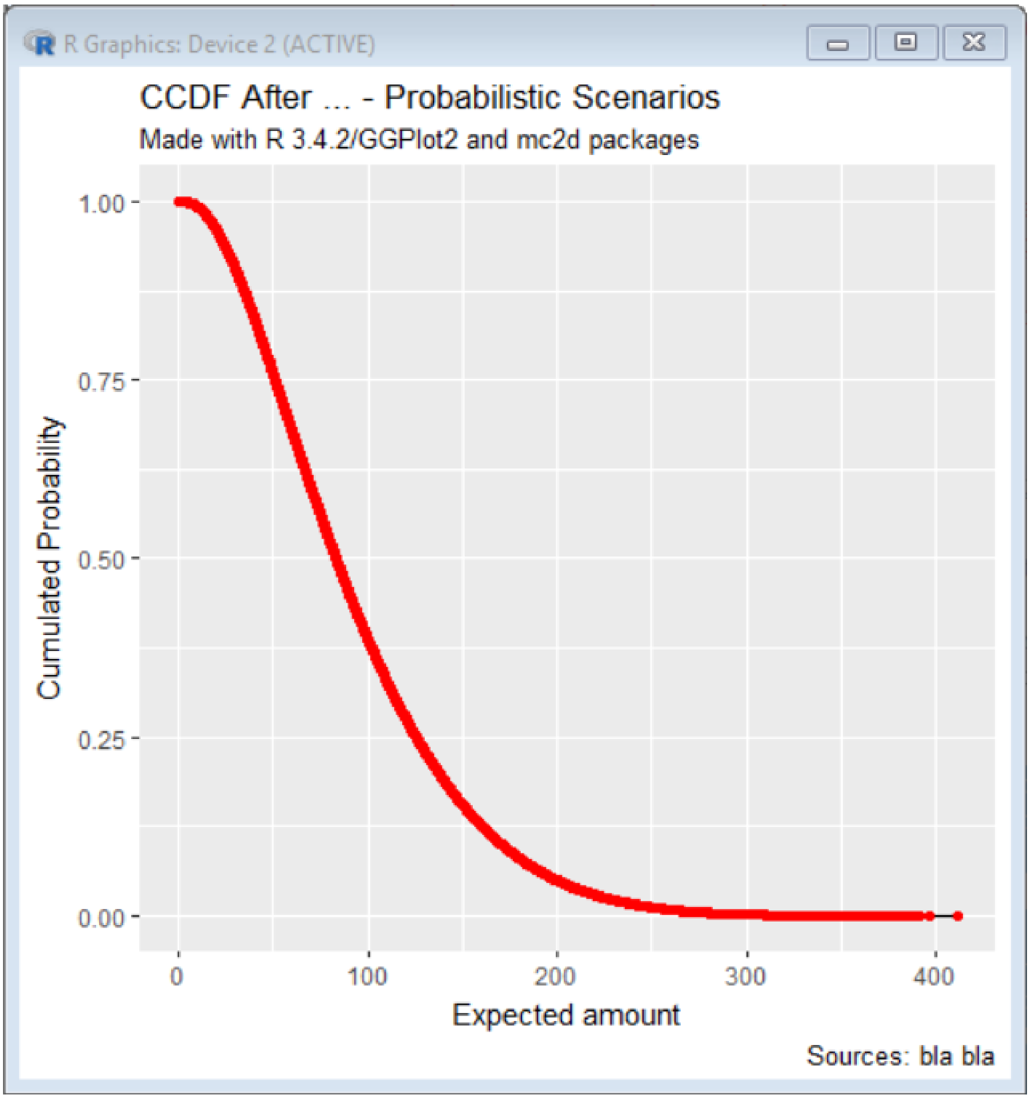


Figure 296 Graph CCDF (ggplot)

Exemple 322.: Offre/Demande et prix optimal

R 3.5.0

Comme promis, nous allons mettre ici en pratique les concepts d'optimisation d'évaluation des prix vue dans le cours théorique. Effectivement, ce travail est très utile car il permet, outre le fait de pratiquer R, il permet répéter des techniques R vues précédemment sur des cas concrets un peu plus complexes que ceux vus précédemment.

L'exemple ci-dessous est entièrement inspiré de l'article du blog suivant:

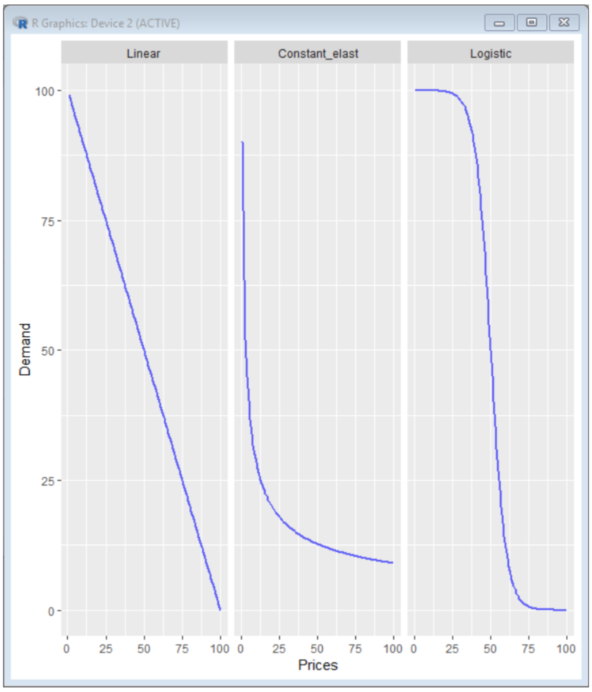
https://insightr.wordpress.com/2018/06/03/different-demand-functions-and-optimal-price-estimation-in-r/

de Yuri Fonseca.

Voyons donc cela en construisant et affichant dans un premier temps les trois modèles à l'aide des package **ggplot2**, **reshape2**, **magrittr** et particulièrement la fonction **melt( )** :

```
> library(ggplot2)
> library(reshape2)
> library(magrittr)
>
> #Modèle linéaire de la demande D(P)
> linear = function(p, alpha, beta){alpha*p + beta}
> #Modèle à élasticité constant de la demande D(P)
> constant_elast = function(p, alpha, beta) exp(alpha*log(p)+beta)
> #Modèle logistique de la demande D(P)
> logistic = function(p, c, alpha, p0) c/(1+exp(-alpha*(p-p0)))
>
> p = seq(1, 100)
> y1 = linear(p, -1, 100)
> y2 = constant_elast(p, -.5, 4.5)
> y3 = logistic(p, 100, -.2, 50)
>
> df = data.frame('Prices' = p, 'Linear' = y1, 'Constant_elast' = y2, 'Logistic' = y3)
> #on réarrange le data.frame en trois colonnes: Prix, Modèle et Demande
> df.plot = melt(df, id = 'Prices') %>% set_colnames(c('Prices', 'Model', 'Demand'))
>
> ggplot(df.plot) + aes(x = Prices, y = Demand) +
+   geom_line(color = 'blue', alpha = .6, lwd = 1) +
+   facet_grid(~Model)
>
```

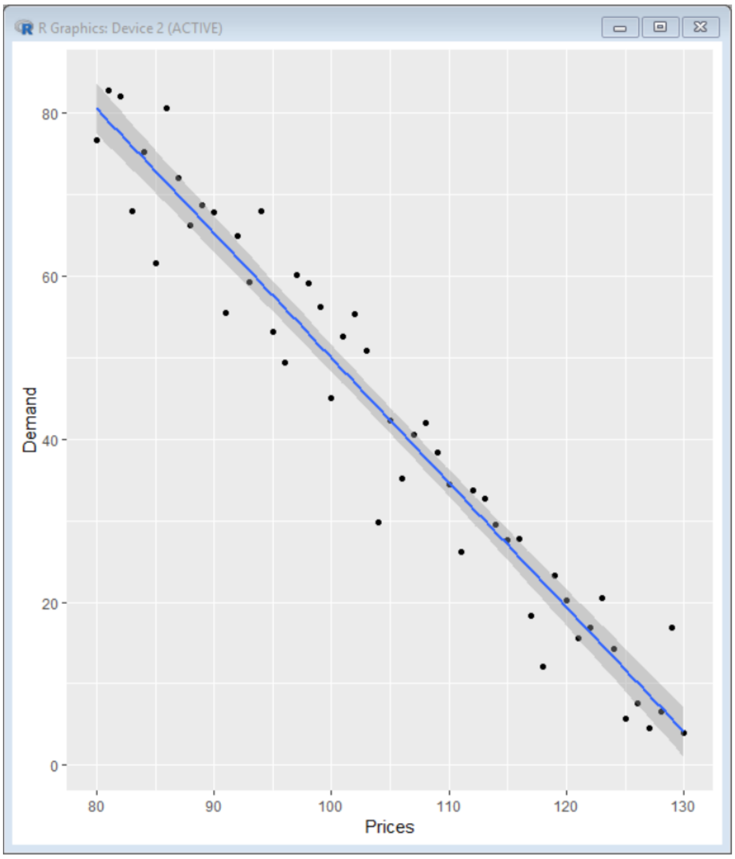
Ce qui donne:



Maintenant déterminons la courbe de demande (de façon statistique) et celle de profit (sur la même idée):

```
> library("tidyverse")
>
> # On reprend la fonction linéaire avec différents paramètres et on ajoute du bruit
> p <- seq(80,130)
> d <- linear(p, alpha = -1.5, beta = 200) + rnorm(sd = 5, length(p))
> c <- 75
> profit <- d*(p-c)
>
> # Régression du modèle
> model1 <- lm(d~p)
> # on calcule le profit sur la base de ce modèle
> profit.fitted = model1$fitted.values*(p - c)
>
> # Optimisation du prix
> alpha <- model1$coefficients[2]
> beta <- model1$coefficients[1]
> p.max.profit <- (alpha*c - beta)/(2*alpha)
> p.max.profit
       p
104.7229
>
> # graphs
> df.linear <- data.frame('Prices' = p, 'Demand' = d,
+                         'Profit.fitted' = profit.fitted, 'Profit' = profit)
>
> #le graph de la demande en fonction du prix (du modèle linéaire)
> ggplot(select(df.linear, Prices, Demand)) + aes(x = Prices, y = Demand) +
+   geom_point() + geom_smooth(method = lm)
>
```

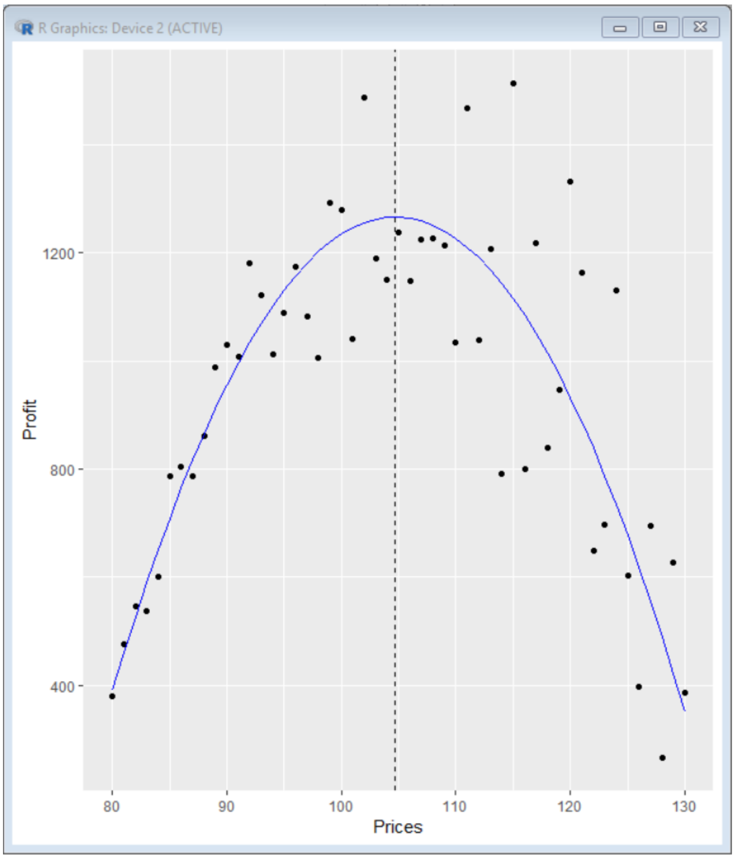
Ce qui donne:



Et pour le Profit en fonction du Prix:

```
> #le graph du profit en fonction du prix (du modèle linéaire)
> ggplot(select(df.linear, Prices, Profit)) + aes(x = Prices, y = Profit) +
+   geom_point() + geom_vline(xintercept = p.max.profit, lty = 2) +
+   geom_line(data = df.linear, aes(x = Prices, y = Profit.fitted), color = 'blue')
>
```

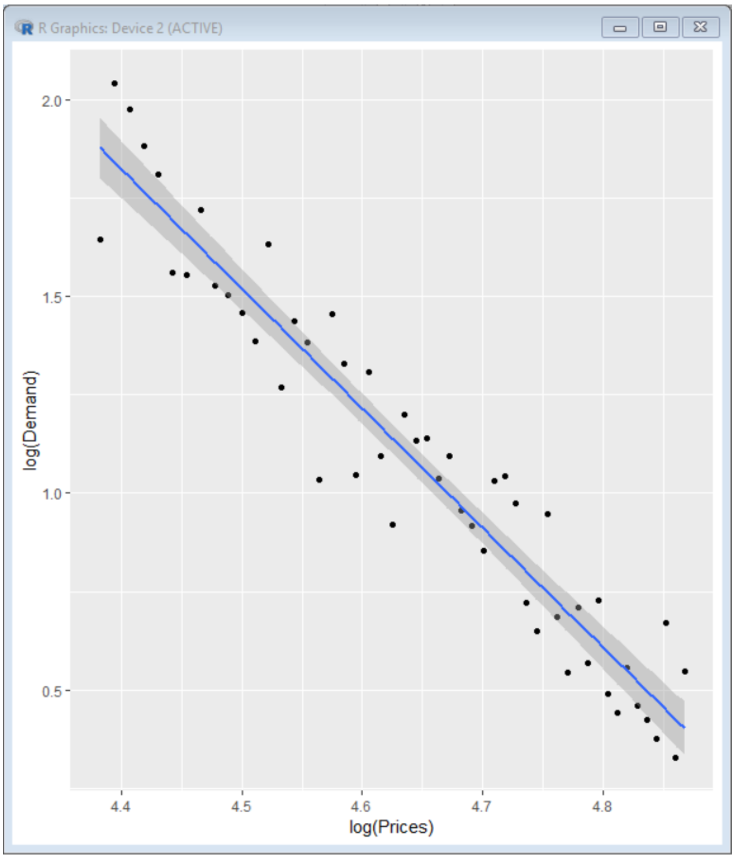
Ce qui donnera:



Maintenant allons-y pour le modèle d'arc-élasticité constante:

```
R Console
> # Modèle avec l'arc-élasticité constante avec du bruit
> p <- seq(80,130)
> d <- constant_elast(p, alpha = -3, beta = 15)*exp(rnorm(sd = .15, length(p)))
> c <- 75
> profit <- d*(p-c)
>
> # Régression du modèle
> model2 = lm(log(d)~log(p))
> profit.fitted <- exp(model2$fitted.values)*(p - c)
>
> # Optimisation du prix
> alpha <- model2$coefficients[2]
> p.max.profit <- c/(1-1/abs(alpha))
> p.max.profit
  log(p)
111.8492
>
> df.const_elast = data.frame('Prices' = p, 'Demand' = d,
+                         'Profit.fitted' = profit.fitted, 'Profit' = profit)
>
> # le graph de la demande en fonction du prix (modèle arc-élasticité constante)
> ggplot(select(df.const_elast, Prices, Demand)) + aes(x = log(Prices), y = log(Dema$
+   geom_point() + geom_smooth(method = lm)
> 
```

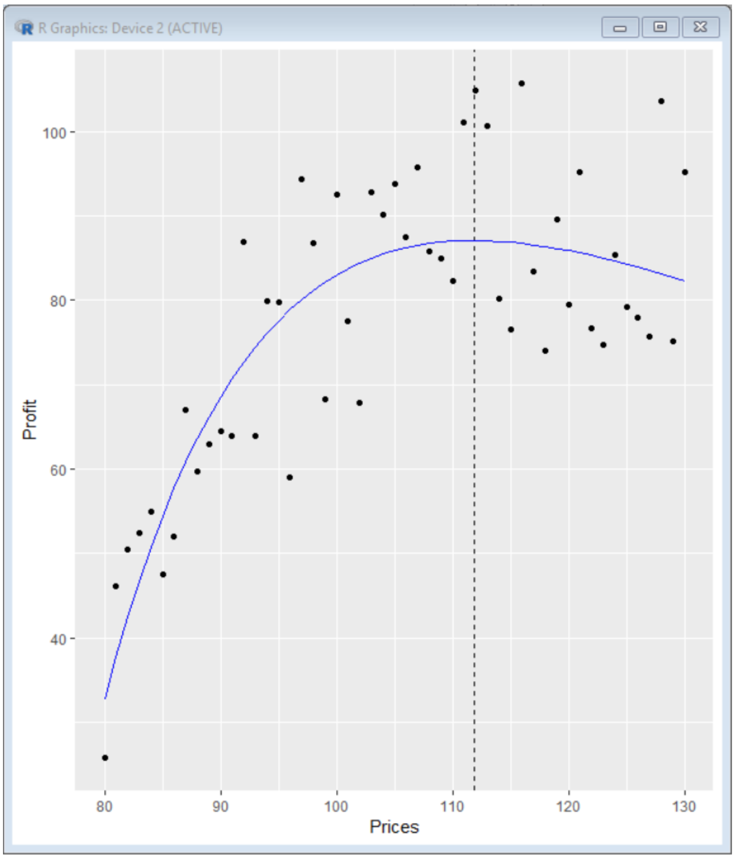
Ce qui donne pour le logarithme de la Demande en fonction du logarithme du Prix:



et en ce qui concerne le Profit en fonction du Prix:

```
> # le graph du profit en fonction du prix (modèle arc-élasticité constante)
> ggplot(select(df.const_elast, Prices, Profit)) + aes(x = Prices, y = Profit) +
+   geom_point() + geom_vline(xintercept = p.max.profit, lty = 2) +
+   geom_line(data = df.const_elast, aes(x = Prices, y = Profit.fitted), color = 'blue')
> 
```

Ce qui donne:

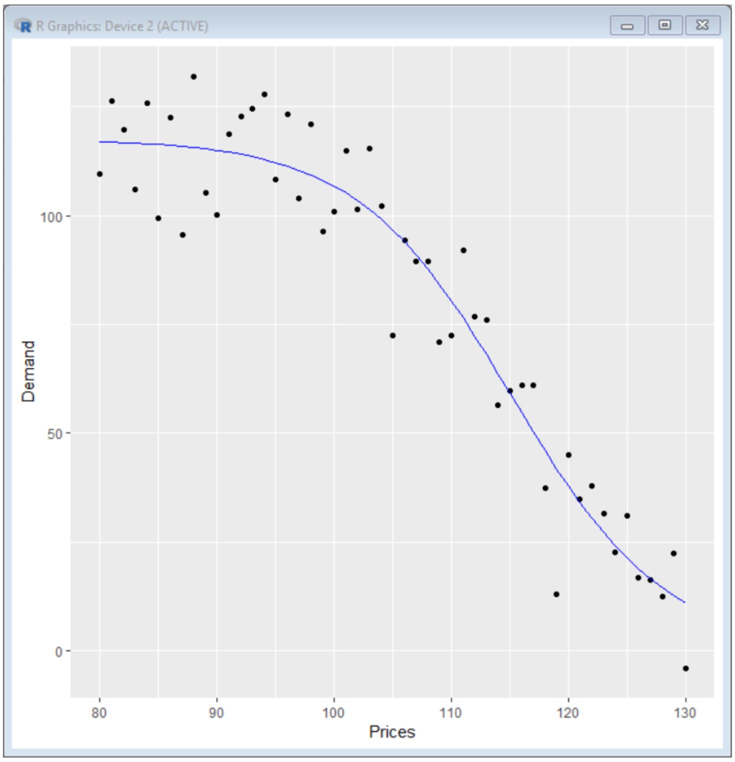


Maintenant allons-y pour le modèle logit:

```
> # Fonctions objectifs pour faire une régression par optimisation quadratique
> demand_objective <- function(par, p, d) sum((d - logistic(p, par[1], par[2], par[3]))^2)
> price_objective <- function(p, alpha, c, p0) (exp(-alpha*(p-p0))*alpha*(p-c) + 1)^2
>
> p <- seq(80,130)
> c <- 75
> d <- logistic(p, 120, -.15, 115) + rnorm(sd = 10, length(p))
> profit <- d*(p-c)
>
> # On régresse par une approche brute avec la fonction optim donc on a besoin de
> par.start <- c(max(d), 0, mean(d)) #valeur estimée des paramètres
>
> demand_fit <- optim(par = par.start, fn = demand_objective, method = 'BFGS',
+                    p = p, d = d)
>
> par <- demand_fit$par # valeurs estimées des paramètres de la fonction logistique
> demand_fit$par
[1] 117.4703199  -0.1521246 115.0904525
> # On crée les valeurs ajustées
> demand.fitted <- logistic(p, c = par[1], alpha = par[2], p0 = par[3])
> # et donc les profits corresondants
> profit.fitted <- demand.fitted*(p - c)
>
> # Optimisation du prix où la encore nous n'avons pas de solution algébrique
> p.start<- mean(p) #valeur estimée du paramètre
> price_fit <- optim(p.start, price_objective, method = 'BFGS',
+ alpha = par[2], c = c, p0 = par[3])
>
> # both results are almost identical
> p.max.profit <- price_fit$par
> p.max.profit
[1] 105.091
> |
```

Ce qui donne pour la Demande en fonction du logarithme du Prix:

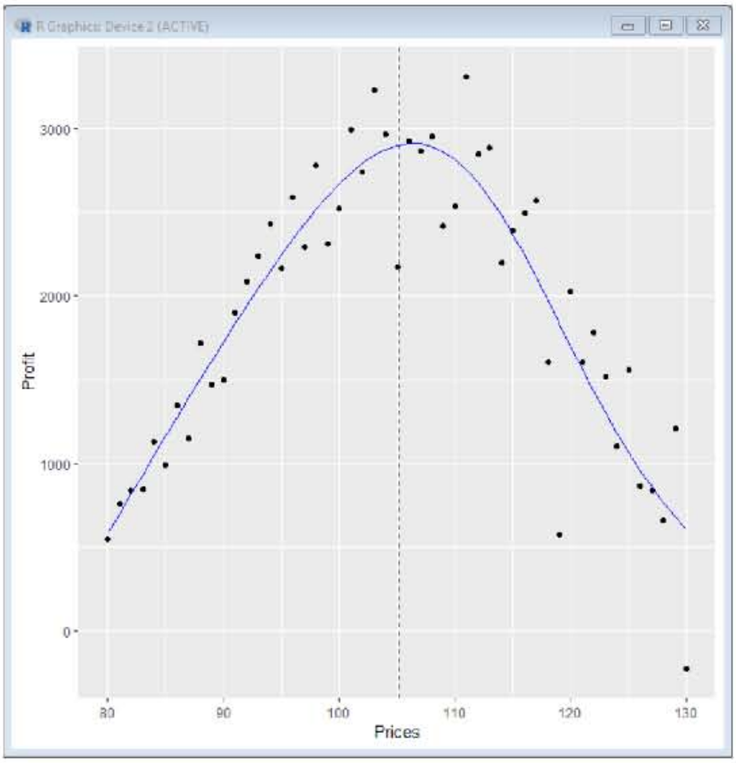
```
> df.logistic <- data.frame("Prices" = p, "Demand" = d, "Demand.fitted" = demand.fitted,
+                           "Profit.fitted" = profit.fitted, "Profit" = profit)
>
> ggplot(select(df.logistic, Prices, Demand)) + aes(x = Prices, y = Demand) +
+   geom_point() +
+   geom_line(data = df.logistic, aes(x = Prices, y = Demand.fitted), color = 'blue')
> |
```



et en ce qui concerne le Profit en fonction du Prix:

```
> ggplot(select(df.logistic, Prices, Profit)) + aes(x = Prices, y = Profit) +
+   geom_point() + geom_vline(xintercept = p.max.profit, lty = 2) +
+   geom_line(data = df.logistic, aes(x = Prices, y = Profit.fitted), color = 'blue')
> |
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
3000
2000
1000
0
Profit
80
90
100
110
120
130
Prices

Exemple 323.: Analyse Procédurale Hiérarchique

R 3.1.1

Bon il s'agit ici d'analyse une matrice de préférence à un niveau qui pour rappel est basé sur trois choix d'un facteur A:

[Choix A] 9 8 7 6 5 4 3 2 1 2 **3** 4 5 6 7 8 9 [Choix B]

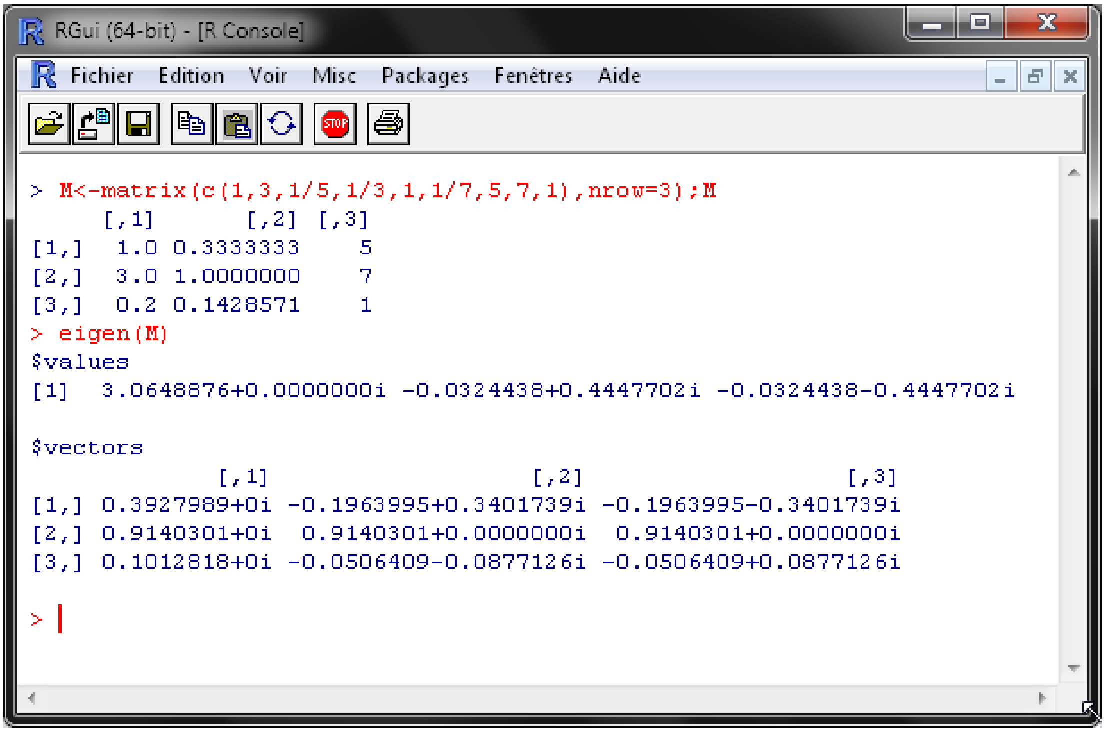
[Choix A] 9 8 7 6 **5** 4 3 2 1 2 3 4 5 6 7 8 9 [Choix C]

[Choix B] 9 8 **7** 6 5 4 3 2 1 2 3 4 5 6 7 8 9 [Choix C]

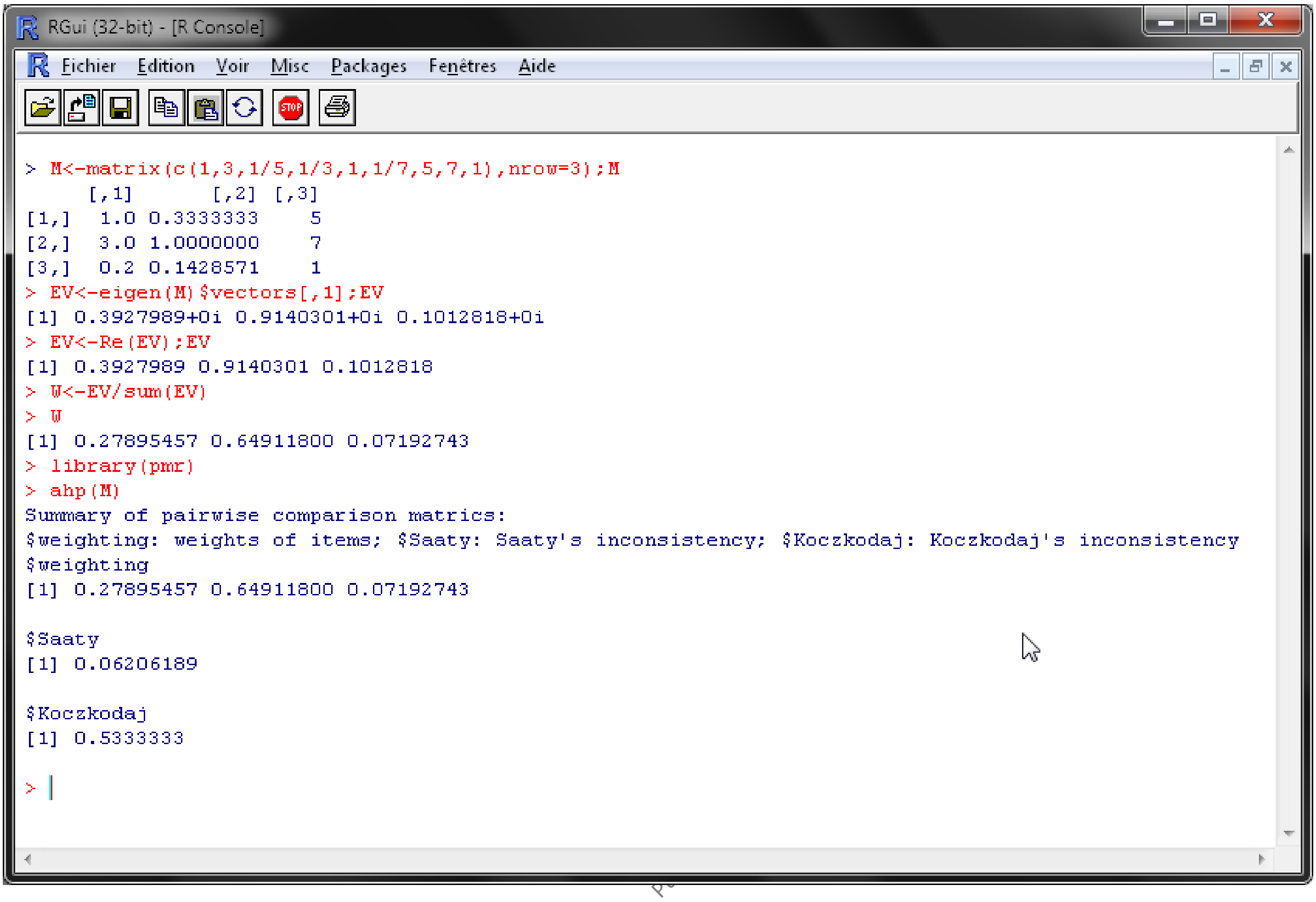
Ce qui donne sous forme de matrice:

Facteur A	Choix A	Choix B	Choix C
Choix A	1	**1/3**	**5**
Choix B	3	1	**7**
Choix C	1/5	1/7	1

D'abord vérifions qu'il y a une faible inconsistance a priori en analysant avec R les trois valeurs propres:



Donc c'est tout jusque là! Poursuivons avec la fonction **ahp( )** du package **pmr**:



Nous retrouvons donc avec R les valeurs calculées approximativement avec un logiciel comme Microsoft Excel excepté pour l'indice de Koczkodaj que nous n'avons pas étudié dans le cours théorique.

Ceux qui me connaissent savent que lorsque je le peux montre l'article original des auteurs comme exemple de base dans mes cours. Donc dans le cas présent prenons le premier exemple de l'article de Gale & Shapley (1962 et prix Nobel en 2012):

QUESTION: *For any pattern of preferences is it possible to find a stable set of marriages?*

Before giving the answer let us look at some examples.

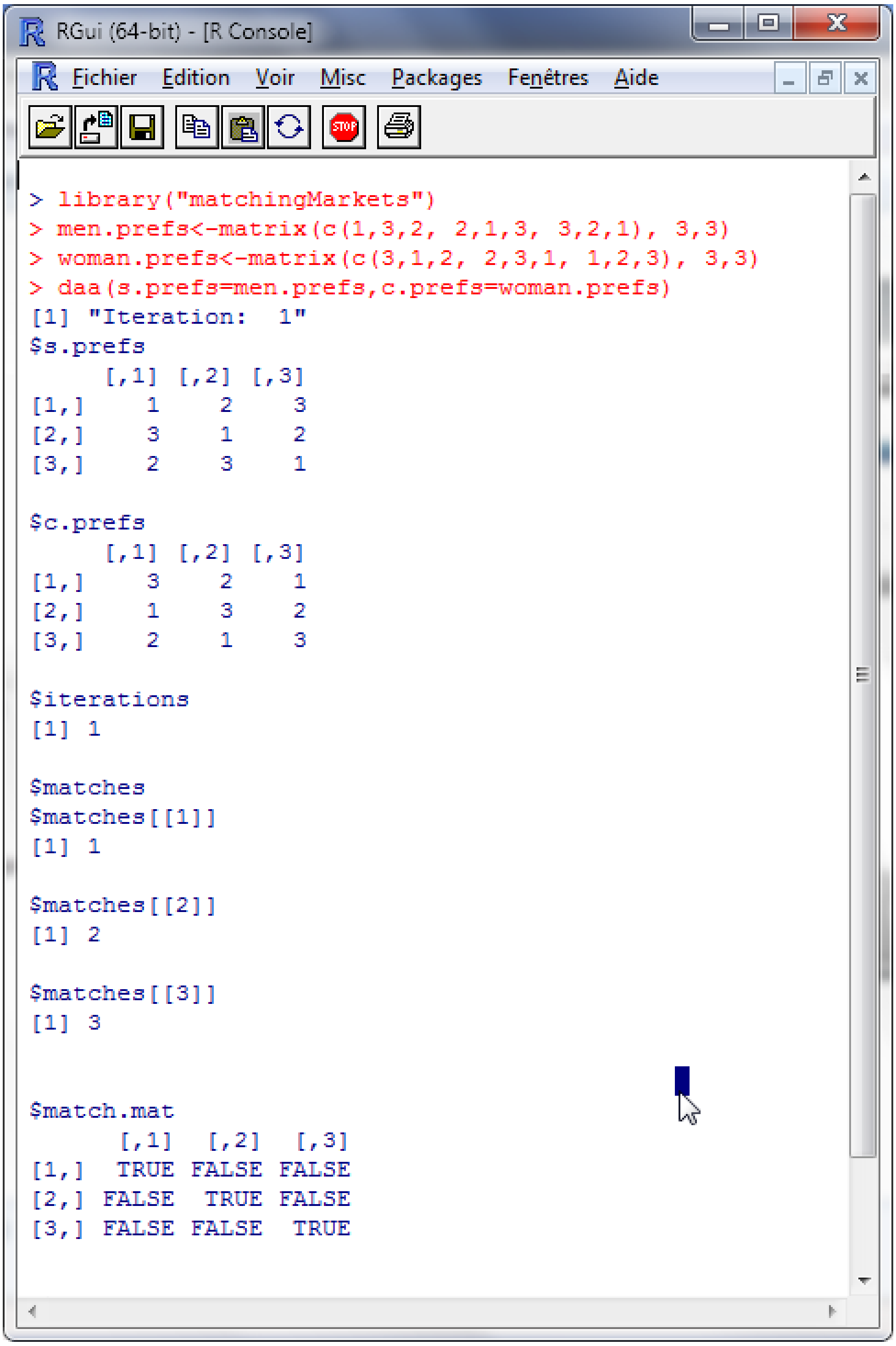
Example 1. The following is the "ranking matrix" of three men, α, β, and γ, and three women, A, B, and C.

	A	B	C
α	1, 3	2, 2	3, 1
β	3, 1	1, 3	2, 2
γ	2, 2	3, 1	1, 3

The first number of each pair in the matrix gives the ranking of women by the men, the second number is the ranking of the men by the women. Thus, α ranks A first, B second, C third, while A ranks β first, γ second, and α third, etc.

There are six possible sets of marriages; of these, three are stable. One ot these is realized by giving each man his first choice, thus α marries A, β marries B, and γ marries C. Note that although each woman gets her last choice, the arrangement is nevertheless stable. Alternatively one may let the women have their first choices and marry α to C, β to A, and γ to B. The third stable arrangement is to give everyone his or her second choice and have α marry B, β marry C, and γ marry A. The reader will easily verify that all other arrangements are unstable.

La solution est ici triviale mais le but est de voir si l'algorithme converge vers la solution triviale (qui est la diagonale). Nous avons alors en utilisant le package **matchingMarkets** la fonction **daa( )** . Nous avons alors (voir la capture unique à la page suivante):



Le résultat est bien conforme!

Exemple 324.: Problèmes des mariages stables (algorithme de satisfaction de Lloy Shaply)

R 3.2.1

Nous allons ici monter une implémentation de l'algorithme des mariages stables dont les applications sont courantes dans l'affectation des étudiantes à des écoles en fonction des préférences des uns et des autres. Ce modèle est un cas académique de l'application de la théorie des jeux est il est très connu car outre le fait qu'il soit très utilisé dans la pratique, il est très simple à appréhender et de plus un de ses fondateurs (Lloy Shaply) est prix Nobel d'Économie pour des travaux relatifs à ce type de problématique.

Ce problème consiste à trouver, étant donné n hommes et n femmes, une façon stable de les mettre en couple. La stabilité signifie que chaque femme et chaque homme préfère rester avec son conjoint présent plutôt que d'être avec quelqu'un d'autre. Un exemple de situation instable serait que Monsieur Dupont préfère Madame Durand à Madame Dupont, et Madame Durand préfère Monsieur Dupont à Monsieur Durand.

Plus formellement, on se donne deux ensembles A et B ayant chacun n éléments. On se donne aussi, pour chaque élément de A et B, une fonction de préférence, qui classe les éléments de l'autre ensemble. On cherche alors à associer de façon bijective les éléments de A avec ceux de B, pour qu'il n'existe pas $a \in A$ et $b \in B$ tels que a préfère b à l'élément qui lui est associé, et b préfère a à l'élément qui lui est associé.

Reste à suivre dans une future version de ce PDF....

Exemple 325.: Chaînes de Markov Discrètes (Discrete Time Markov Chain (DTMC))

R 3.2.1

Nous avons vu dans le cours théorique que les chaînes de Markov discrètes (dans le temps) pouvaient être très utiles en biologie (dynamique des populations) en théorie des graphes associées aux probabilités (probabilité de passage d'un état à l'autre) ainsi qu'en maintenance et météorologie. Bien que leur application soit bien plus vaste considérons l'étude d'un cas trivial inspiré à 99% du blog de Nicole M. Radziwill:

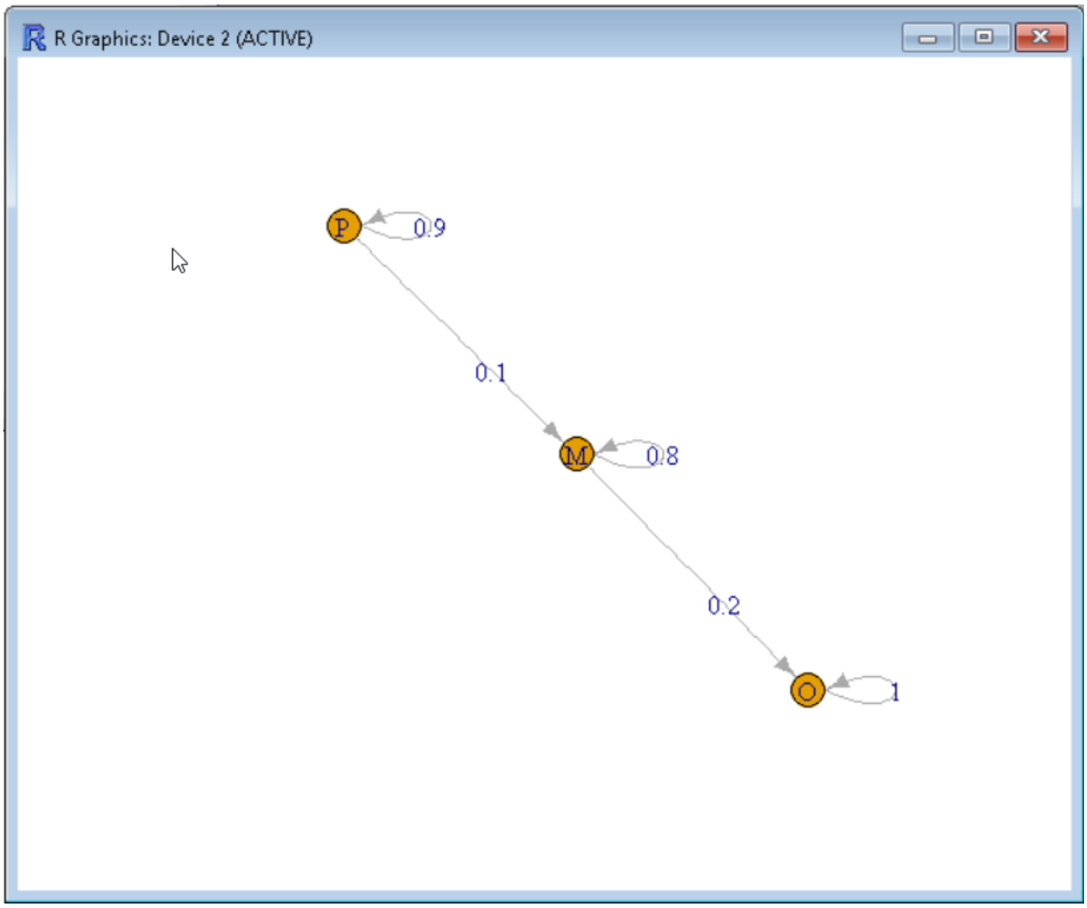
http://qualityandinnovation.com/2015/12/08/a-discrete-time-markov-chain-dtmc-sir-model-in-r/#comments

D'abord nous définissons la matrice de transition et affichons le graphe de Markov correspondant de surveillance d'une machine à trois états {Opérationnelle, Maintenance, Panne}:

```
> library(markovchain)
> mcSIR <- new("markovchain", states=c("Panne","Maintenance","Opérationnel"),
+     transitionMatrix=matrix(data=c(0.9,0.1,0,0,0.8,0.2,0,0,1),
+     byrow=TRUE, nrow=3), name="PMO")
> initialState <- c(99,1,0)
> show(mcPMO)
PMO
 A  3 - dimensional discrete Markov Chain with following states
 P M O
 The transition matrix   (by rows)  is defined as follows
    P   M   O
P 0.9 0.1 0.0
M 0.0 0.8 0.2
O 0.0 0.0 1.0

> plot(mcPMO,package="diagram")
> 
```

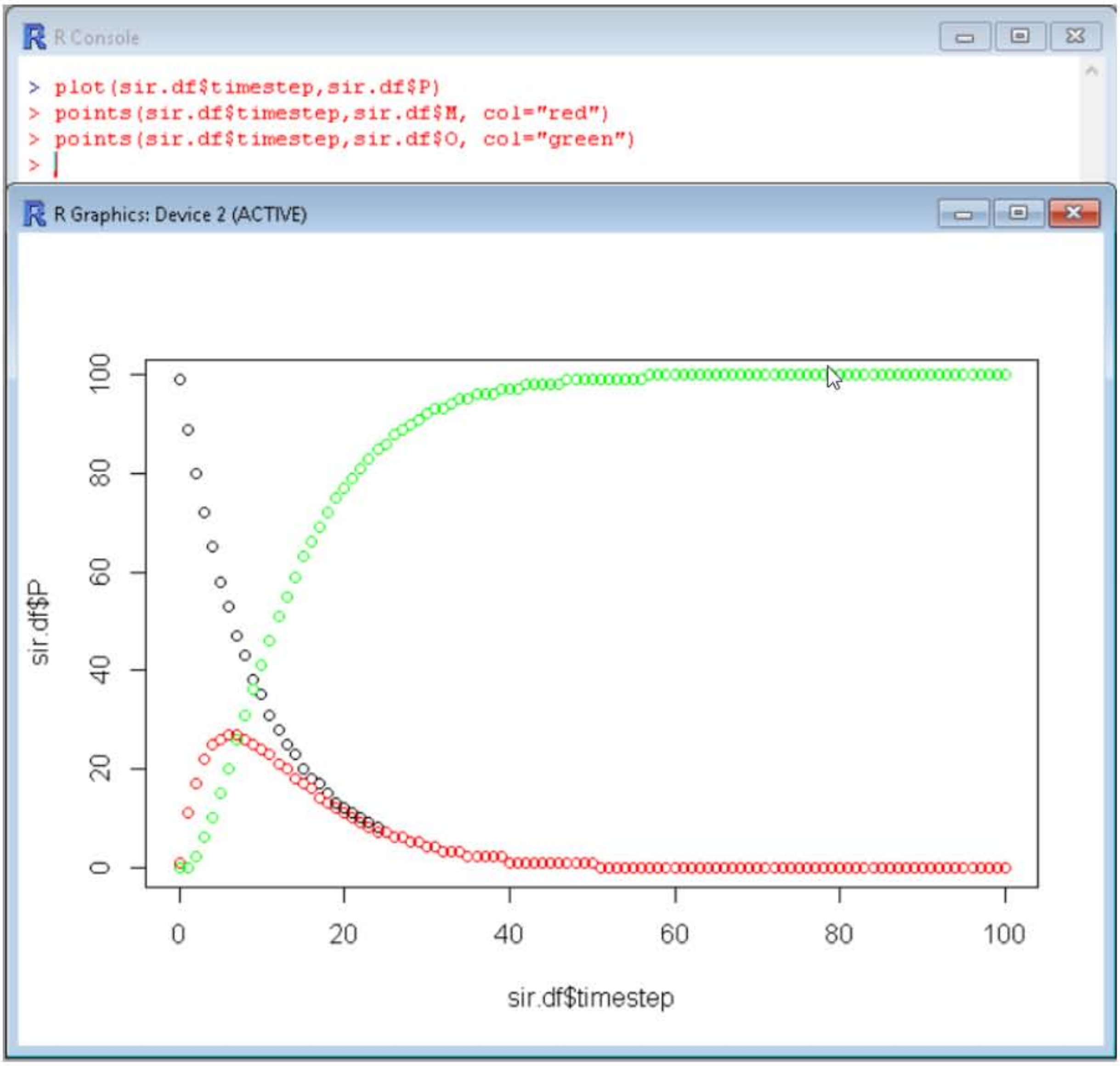
Ce qui donne au niveau du graphe:



Nous procédons maintenant à 100 simulations temporelles t=0...100:


```
> timesteps <- 100
> sir.df <- data.frame( "timestep" = numeric(),
+  "P" = numeric(), "M" = numeric(),
+  "O" = numeric(), stringsAsFactors=FALSE)
>  for (i in 0:timesteps) {
+ newrow <- as.list(c(i,round(as.numeric(initialState * mcPMO ^ i),0)))
+  sir.df[nrow(sir.df) + 1, ] <- newrow
+  }
>
> head(sir.df)
  timestep  P  M  O
1        0 99  1  0
2        1 89 11  0
3        2 80 17  2
4        3 72 22  6
5        4 65 25 10
6        5 58 26 15
>
```

Et le graph correspondant:



Exemple 326.: Analyse TURF (Total Unduplicated Reach and Frequency)

R 3.6.3

Le but ici va être de reproduire l'exemple TURF que nous avons calculé à la main dans le cours théorique:

```
> library("turfR")
> mydata<-read.csv("c:/tmp/TURF.csv",header=T,sep=",")
> colnames(mydata)<-c("Individu","Poids","Chocolat","Vanille","Pistache")
> mydata
   Individu Poids Chocolat Vanille Pistache
1         1     1        1       0        0
2         2     1        1       0        0
3         3     1        1       0        0
4         4     1        0       1        0
5         5     1        0       1        0
6         6     1        0       1        0
7         7     1        1       1        0
8         8     1        1       1        0
9         9     1        0       1        1
10       10     1        1       1        1
> #évalue toutes les combinaisons de 2 éléments choisis parmi 3 éléments
> turf(mydata,3,2)
2 of 3: 0.03500199 sec
total time elapsed: 0.04300213 sec
$turf
$turf[[1]]
  combo rchX frqX 1 2 3
1     1  1.0  1.3 1 1 0
2     3  0.7  0.9 0 1 1
3     2  0.7  0.8 1 0 1


$call
turf(data = mydata, n = 3, k = 2)

> 
```

Ce qui correspond bien aux résultants obtenus dans le cours théorique!

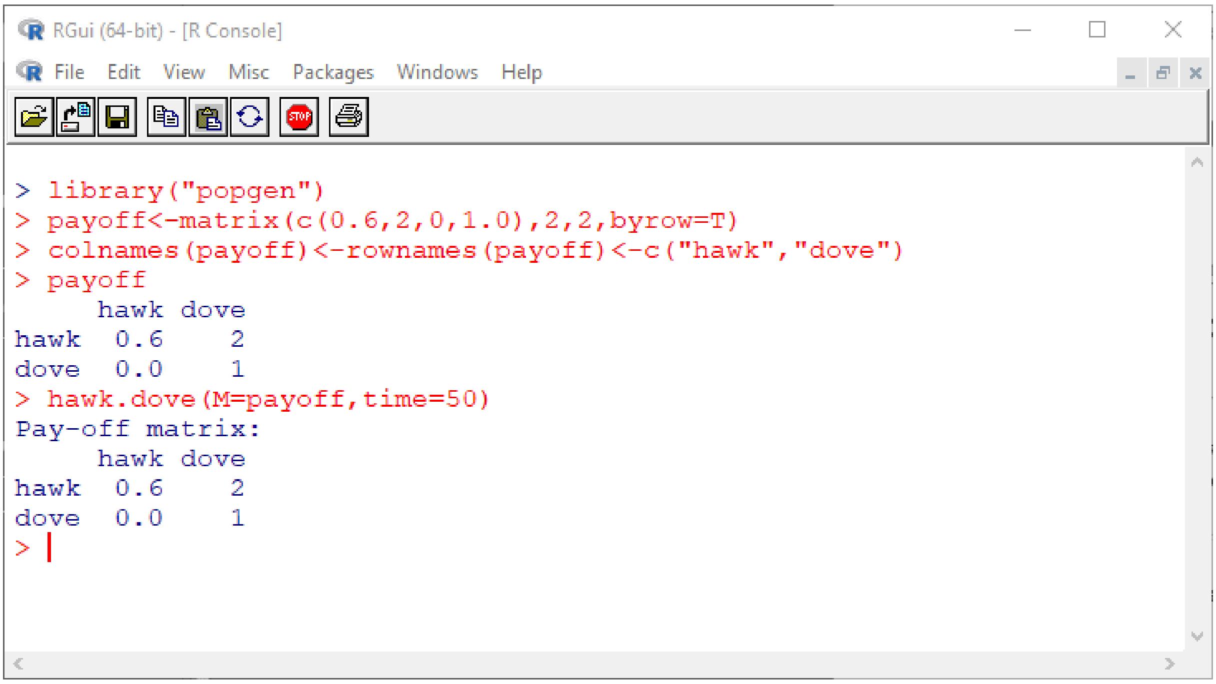
Exemple 327.: Jeux évolutionnaires

R 3.5.2

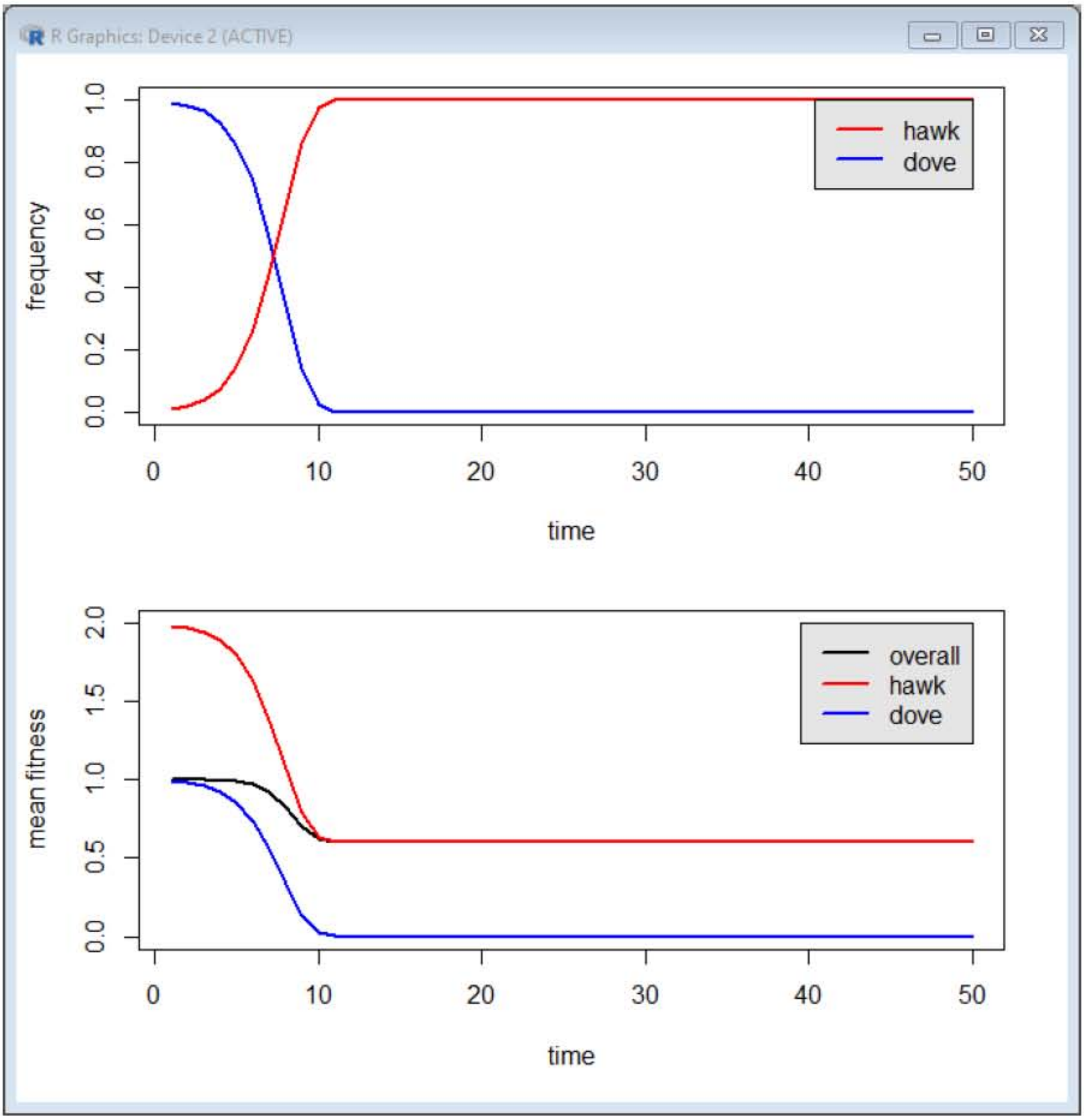
Le but ici va être de voir comment on peut utiliser le package popgen (non-disponible sur le serveur CRAN à ce jour), pour voir l'évolution d'un jeu évolutionnaire.

Nous allosn reprendre l'exemple du cours théorique.

Nous construisons d'abord la matrice des gains:



Soit graphiquement:

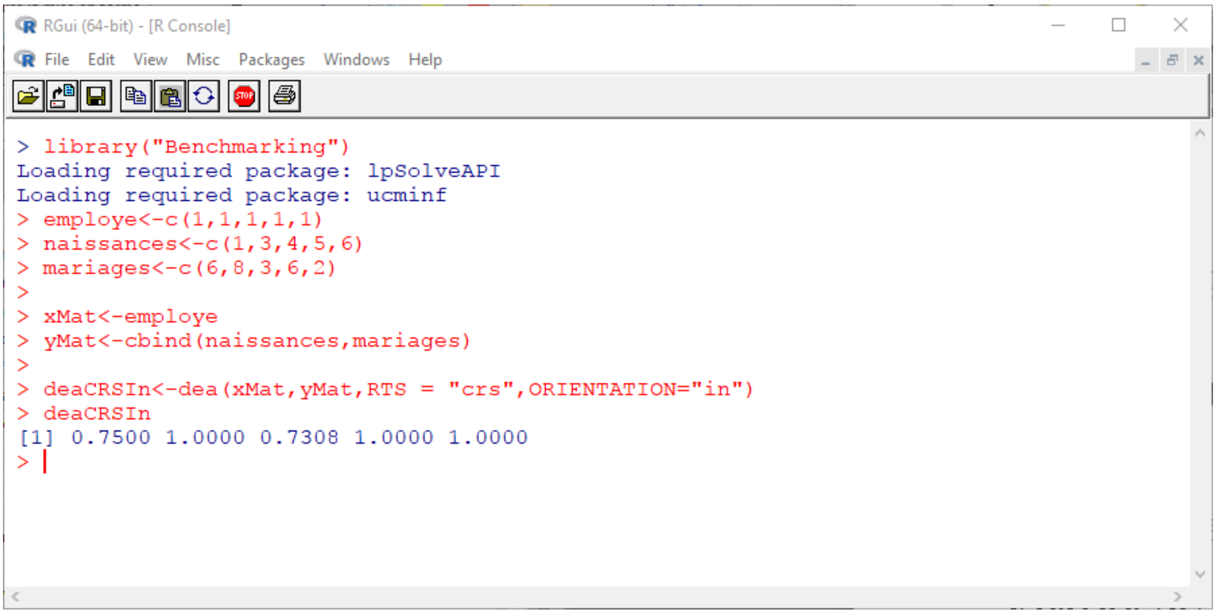

R Graphics: Device 2 (ACTIVE)
frequency
time
hawk
dove
mean fitness
time
overall
hawk
dove

Exemple 328.: Data envelopment analysis (DEA) CRS

R 3.4.3

Le but ici va être d'appliqué l'exemple DEA que nous avons vu dans le cadre des techniques utilisées dans les hauts postes décisionnels de l'administration public et voir si nous retombons sur les mêmes résultats.

Donc nous utilisons la fonction **dea( )** package Benchmarking en nous conduisons une analyse DEA de type CRS (constant returns to scale) avec une orientation de type "input":



Et donc nous retombons bien sur les résultats que nous avons obtenu dans le cours théorique où nous avions utilisé le solveur de Microsoft Excel.

24. Machine Learning (Big Data[34]/B.I. avancée)

Il s'agit ici de présenter l'ensemble des techniques empiriques de Machine Learning (supervisées ou non supervisées) dont les détails mathématiques ont été vus dans le cours théorique.

Attention!!! Rappelons que dans la pratique nous séparons normalement toujours les données initiales en trois jeux:

1) Le jeu d'entraînement
2) Le jeu de test
3) Le tout

Ainsi, L'ensemble des fonctions de Data Mining de R permet de mettre en entrée un jeu d'entraînement et ensuite de faire une prédiction avec un jeu de test pour créer une matrice de confusion et de conclure...

Il convient d'utiliser avec toute la prudence nécessaire et l'esprit très critique ces techniques empiriques car parfois les résultats peuvent mener à des aberrations (et ce d'autant plus que le jeu de données est petit).

Le Data Mining est la technique qui fait appel à la quantité chaque fois que l'on veut congédier la qualité...

Préambule

Suite à l'insistance d'un client qui n'avait pas mis en place de cours théorique dans son organisation voici exceptionnellement quelques petites notions de culture générale reprises du cours théorique:

En analyse de données statistiques, le clustering (data clustering pour les anglophones) décrit des méthodes empiriques de classification de données (méthode de regroupement hiérarchique ou méthode de partitionnement de données).

Il s'agit de techniques permettant typiquement la segmentation de l'ensemble des clients d'une entreprise en fonction de leur démographie ou de leurs habitudes d'achat, de grouper des documents pour des présentations, d'identifier de nouvelles espèces animales ou végétales, de regrouper de l'information ou des individus par intérêts.

Nous considérons dans la pratique deux grandes familles de techniques de clustering (attention! même les spécialistes du domaine n'arrivent pas à se mettre d'accord sur une classification commune...):

1. Les "techniques non hiérarchiques": c'est-à-dire où le nombre de classes (groupes) finales est choisi à l'avance.
2. Les "techniques hiérarchiques": c'est-à-dire où l'on aboutit à un classement par agrégations successives.

[34] Rappelons que le Big Data n'est pas une méthode d'analyse mais un champ d'étude relativement au stockage, à la gestion et l'exploitation des données.

Parmi ces deux familles nous distinguons grossièrement deux-sous familles:

1. Les techniques qui font usage de données dont la propriété nominale de classification est connue à l'avance pour entraîner un modèle de prédiction: "machine learning" ou "apprentissage supervisé" ("supervised learning" en anglais). Nous y retrouvons la technique de régression logistique binaire, les CRT, l'ID3, l'analyse discriminante, les réseaux bayésiens, les listes de décisions, les k-NN, les réseaux de neurones....

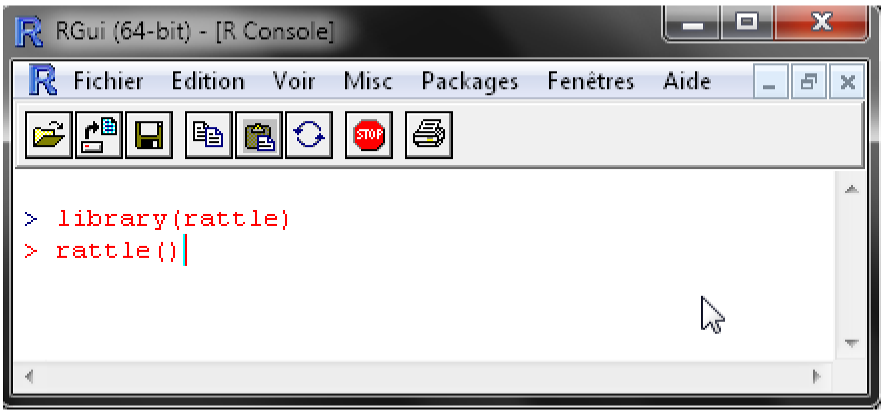
2. Les techniques qui font usage de données dont aucune propriété nominale connue à l'avance ne laisse supposer d'une classification et qui cherchent une classification possible (c'est ce qui au niveau du business est souvent le plus intéressant): "data mining" ou "apprentissage non supervisé" ("unspervised learning" en anglais) ou plus rarement "fouille de données", "forage de données", "prospection de données". Nous y retrouvons les techniques CAH, les k-means, etc.

L'utilisation industrielle ou opérationnelle de ce savoir dans le monde professionnel permet de résoudre des problèmes très divers, allant de la gestion de la relation client à la maintenance préventive, en passant par la détection de fraudes ou encore l'optimisation de sites web, la surveillance des marchés financiers, la prospection pro-active (préférences de consommation des clients), l'optimisation des chemins (analyse des routes à bouchons et à virages à gauche), ou choix des cibles (probabilité d'acquérir un nouveau prospect donné), anticiper l'identifications des terroristes, etc.

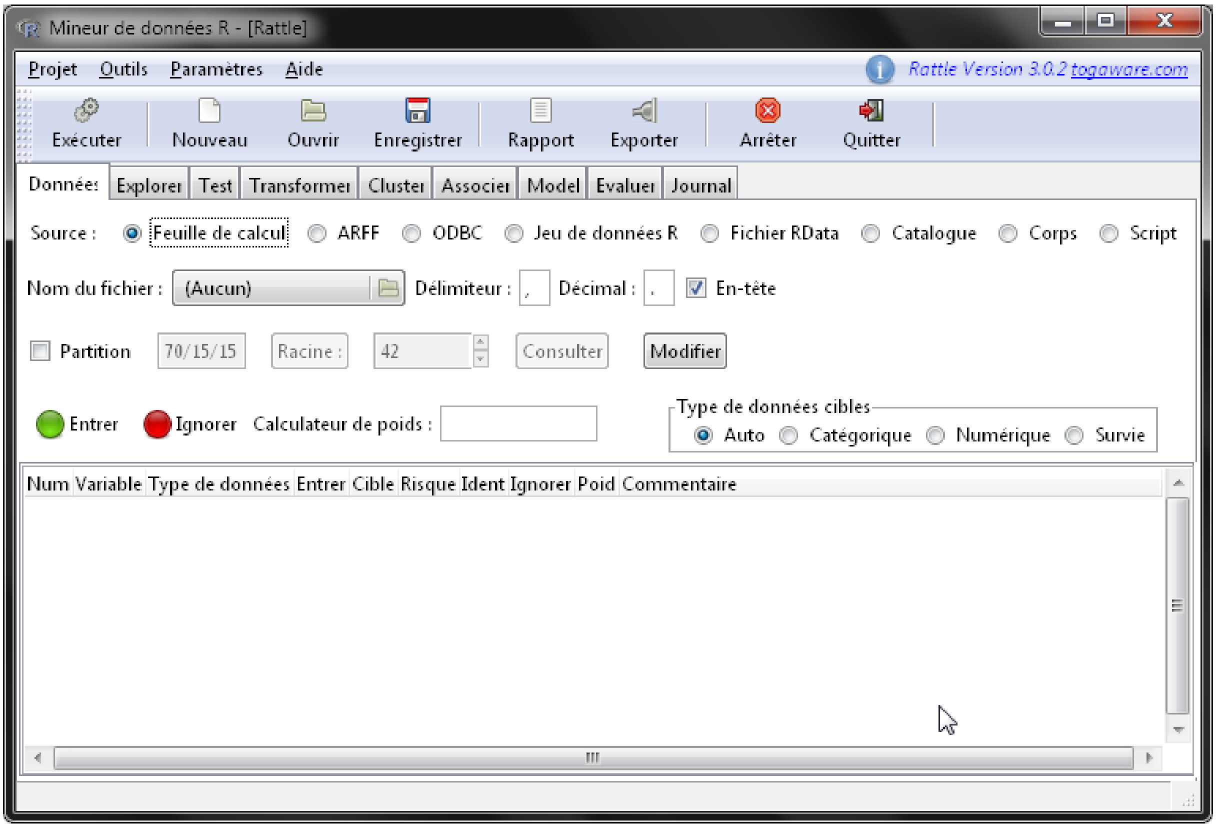
Remarque: Attention à ne pas confondre en toute rigueur les notions de classification, segmentation et association. Bien que les deux premiers soient souvent confondus (classification/segmentation) car de nombreux algorithmes font les deux à la fois, la classification est la prédiction d'une ou plusieurs variables discrètes basée sur la valeur des autres champs du jeu de données alors que la segmentation divise les données en groupes d'éléments ayant des propriétés les plus identiques possible. Quant à l'association il est bien évident que de nombreux algorithmes de segmentation montre qui est associé avec quoi mais l'idée à proprement parler de l'association est de quantifier par un scalaire le degré d'association entre deux jeux de données.

Fin du préambule!

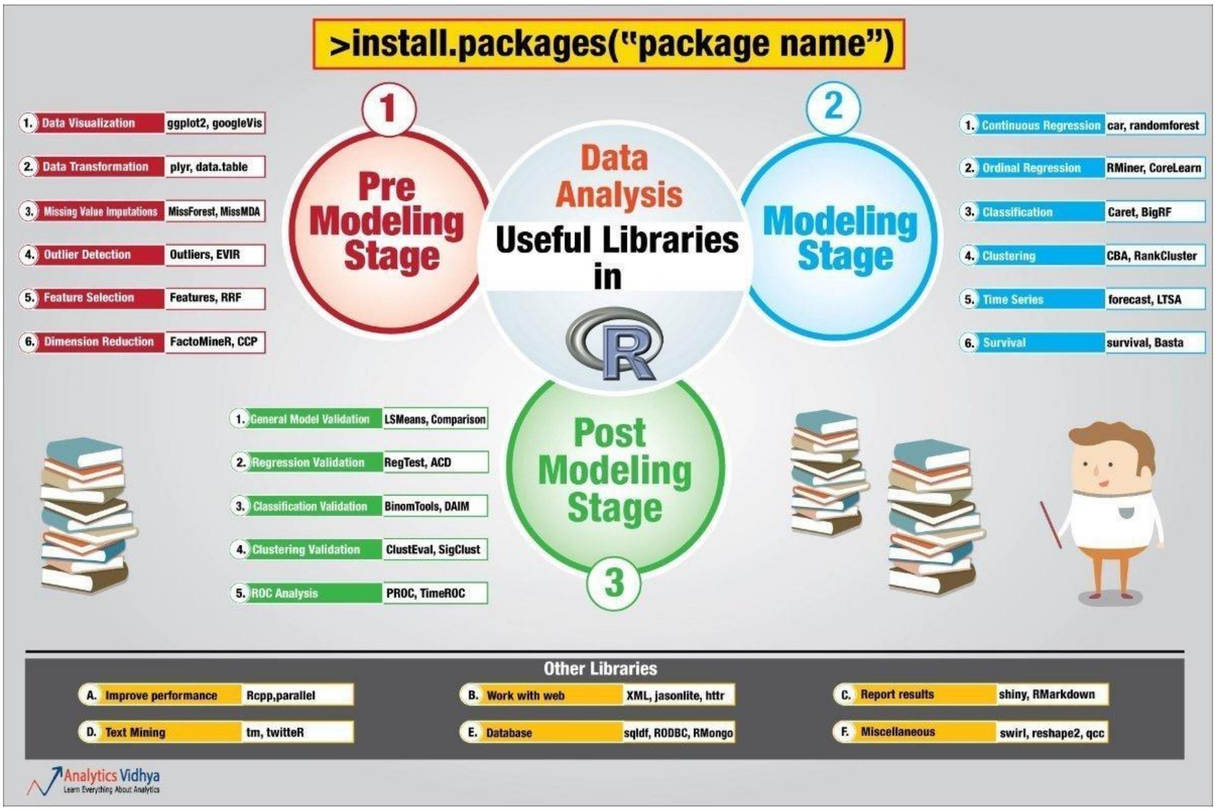
Signalons avec de commencer avec les cas pratiques un petit package visuel pour faire quelques analyses de Data Mining pour ceux qui aiment les interfaces graphiques... Il s'agit du package **rattle**:



qui donne:



Et un petit résumé de l'état de l'art en 2018:



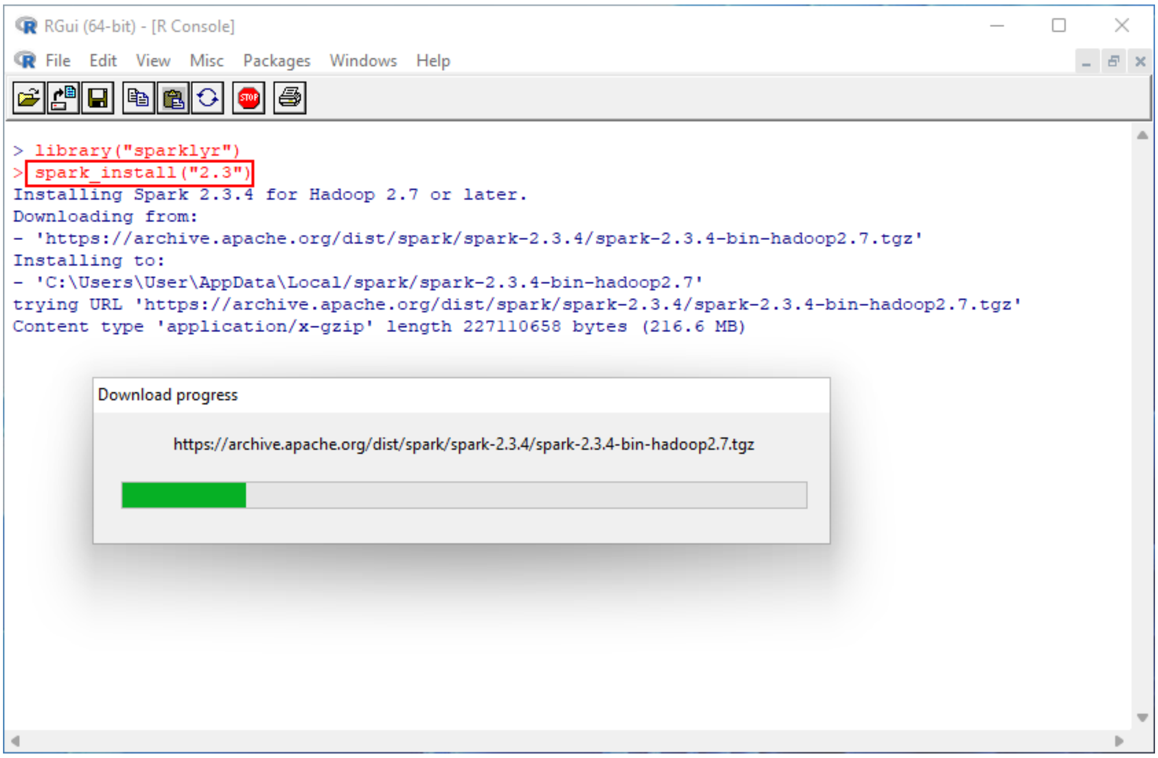
Exemple 329.: Travail avec Apache Spark

R 4.2.0

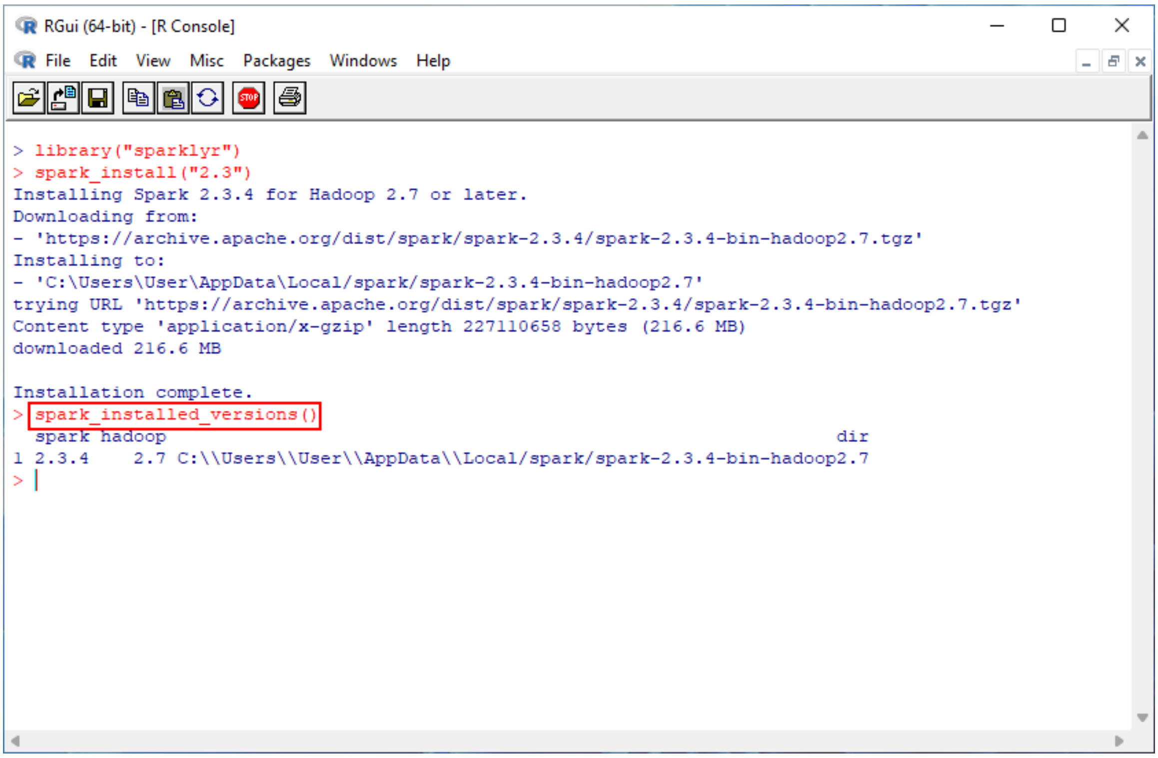
Pour des flux de données en temps réel ou pour le stockage de quantités vraiment massive de données, Apache Spark peut s'avérer un outil adapté.

Voyons donc comment un Data Scientist/Analyst peut installer et utiliser ce dernier simplement pour s'entraîner et faire des tests (évidemment une bonne installation et configuration de Apache Spark est le métier d'un Data Artchitect ou DBA).

Nous installons le package **sparklyr** et avec la fonction spark_install( ), nous installons une des versions de Apache Spark (la version 2.3 dans le cas présent). Ce qui donnera:

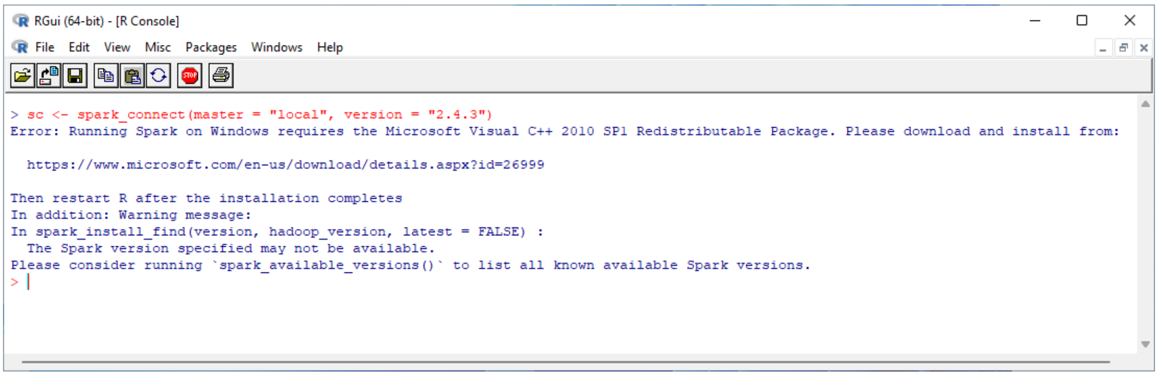


Une fois Apache Spark installé, nous pouvons vérifier qu'il l'est bien avec la fonction spark_installed_version( ):

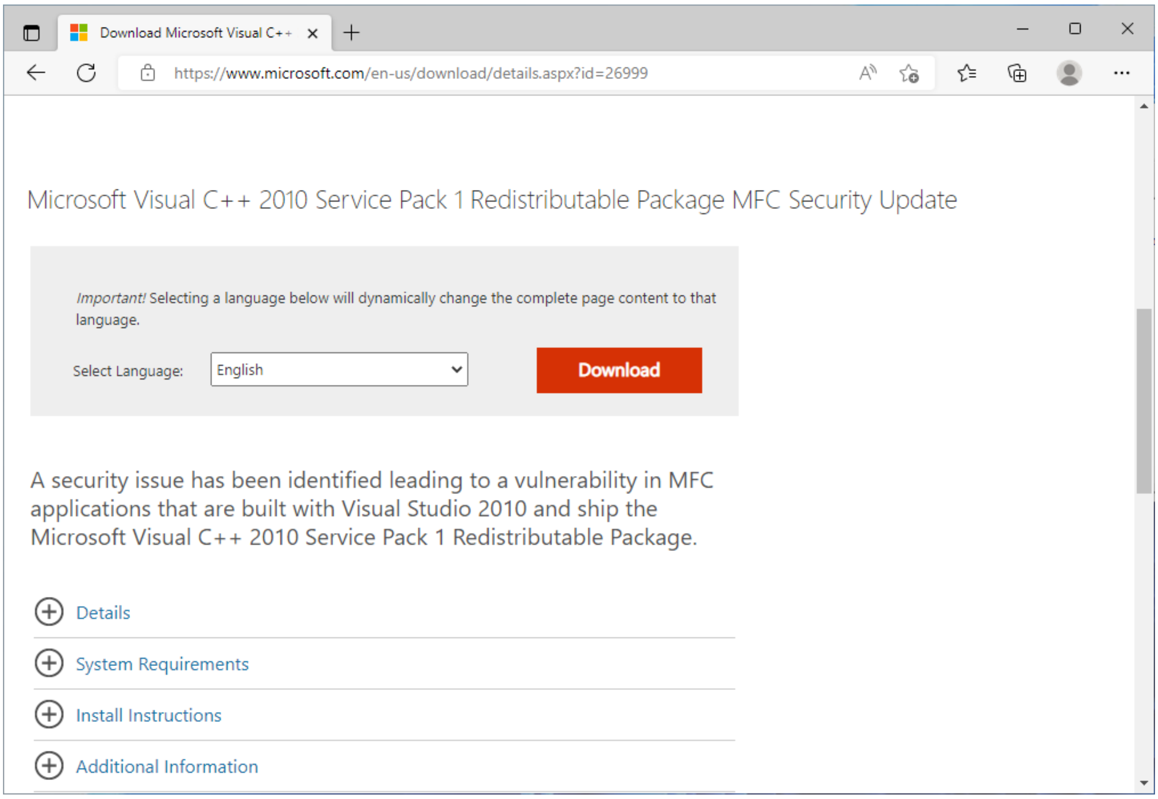


Grâce au numéro de version détaillé, nous pouvons tenter de nous connecter à Apache Spark avec la fonction spark_connect( ), mais nous aurons un message nous indiquant d'aller télécharger Microsoft Visual C++ 2010 SP1 sur le lien suivant:

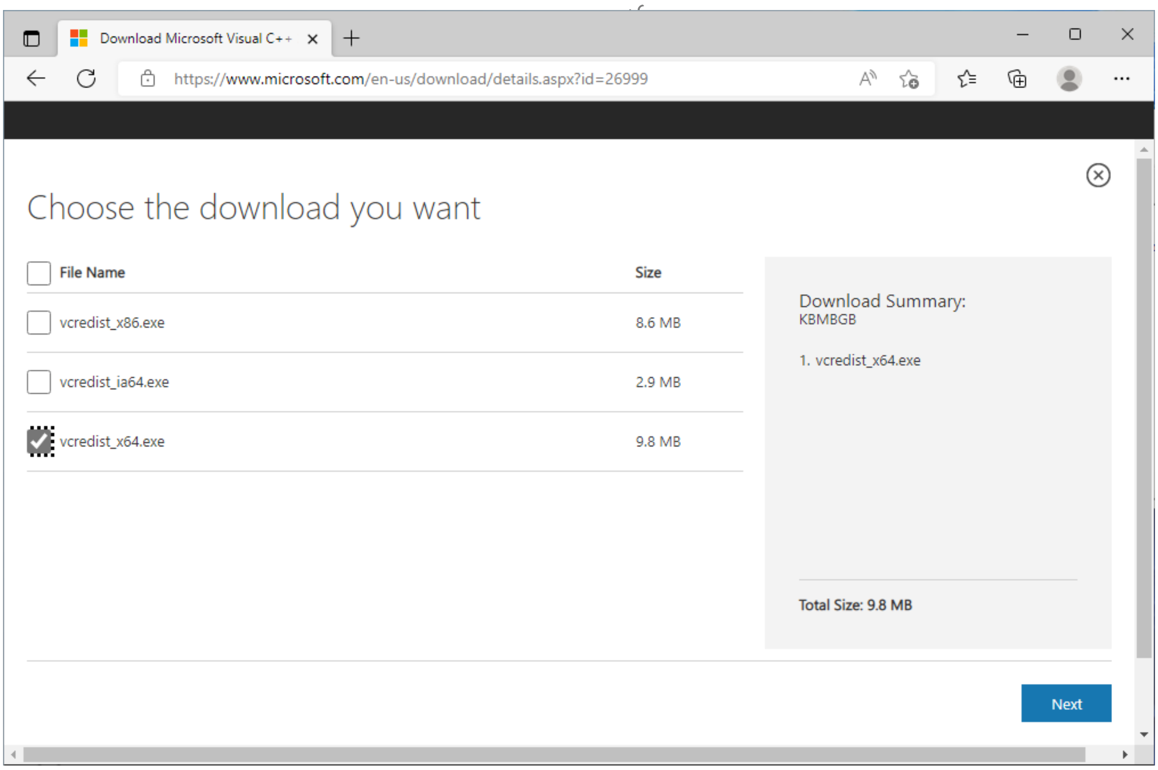
https://www.microsoft.com/en-us/download/details.aspx?id=26999



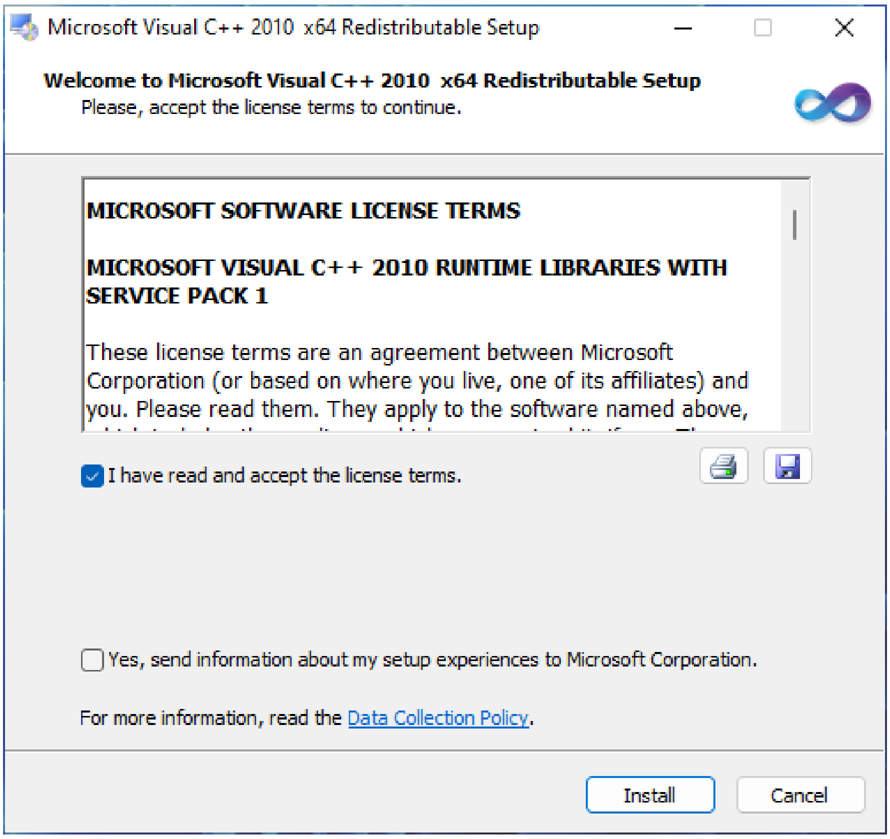
Nous allons donc sur le site web correspondant:



Nous cliquons sur le bout Download et prenons l'exécutable vcredist_x64.exe pour une machine Windows en 64bits tournant sur processeur Intel:

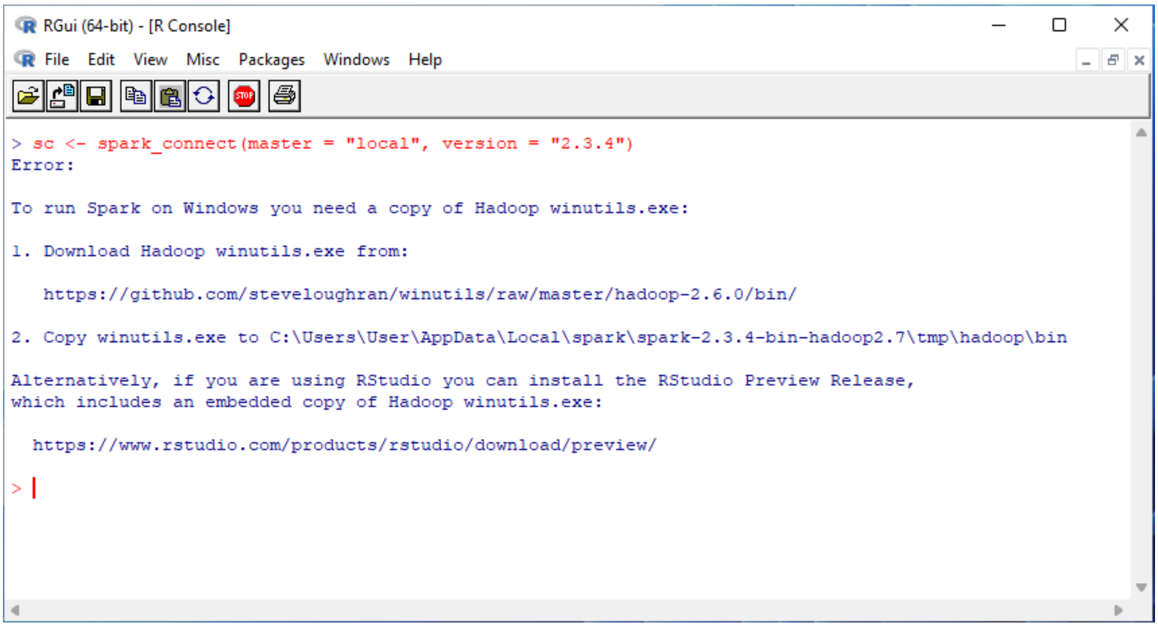


Une fois téléchargé, nous lançons l'installation:

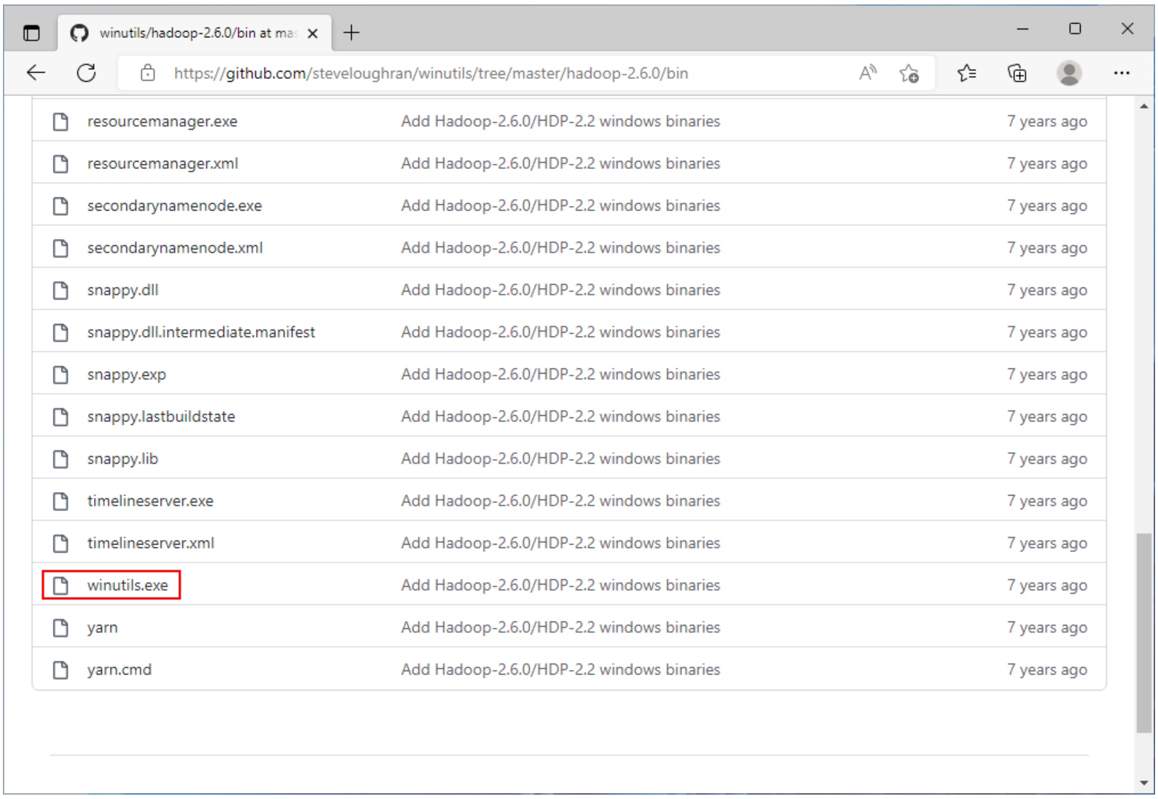


Nous ressayons la connexion et nous avons un deuxième message nous distant qu'un module est manquant et qui peut être téléchargé à l'adresse:

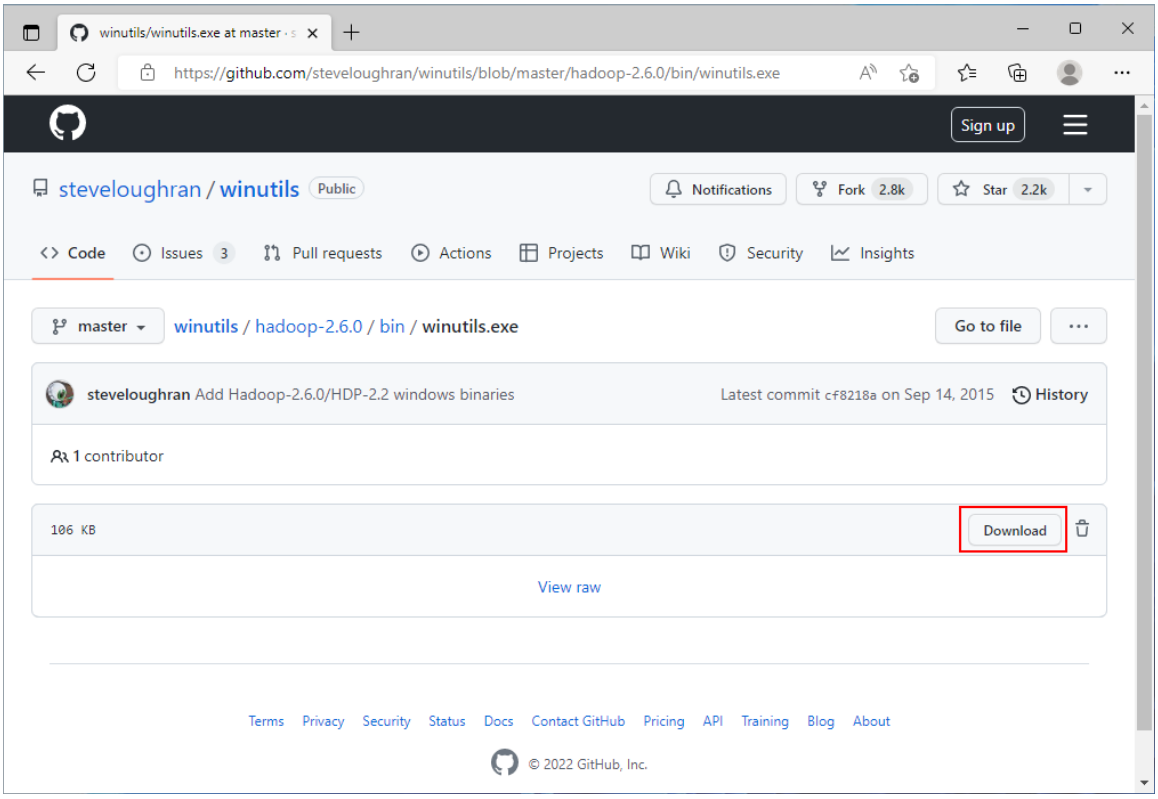
https://github.com/steveloughran/winutils/raw/master/hadoop-2.6.0/bin



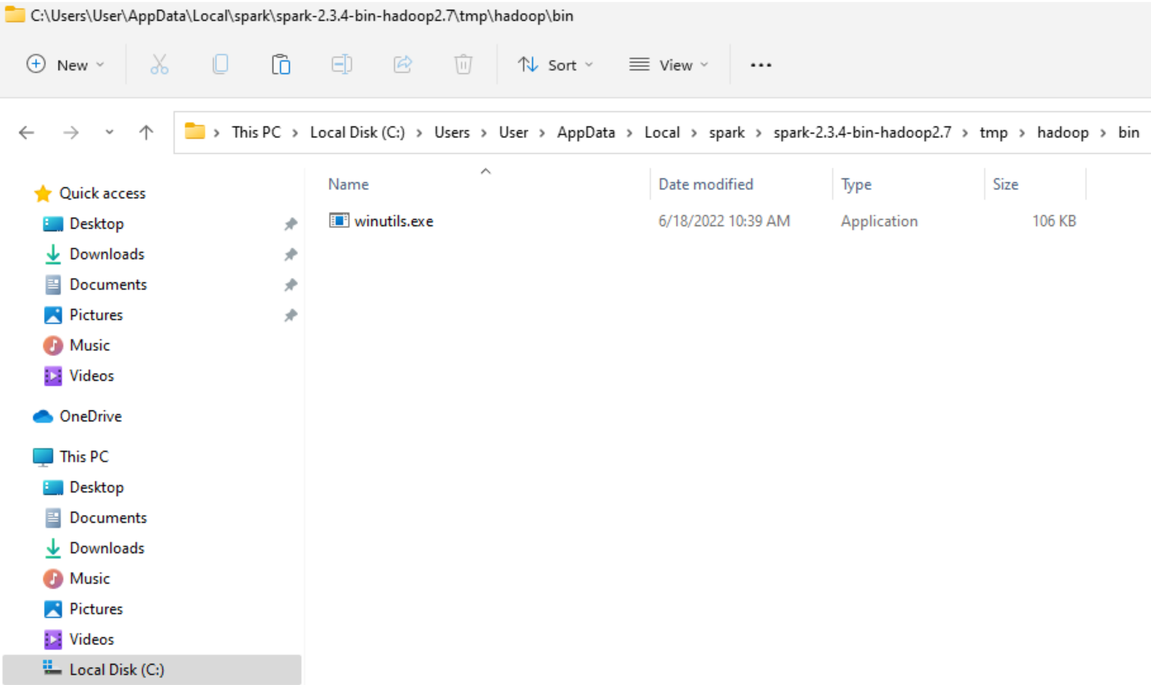
Effectivement, si nous nous rendons à l'adresse mentionnée, nous arrivons sur:



Après avoir cliqué sur winutils.exe, nous arrivons sur:



Nous téléchargons le fichier en cliquant sur **Download** et nous entrepesons le fichier dans le dossier indiqué par le message dans la console R:



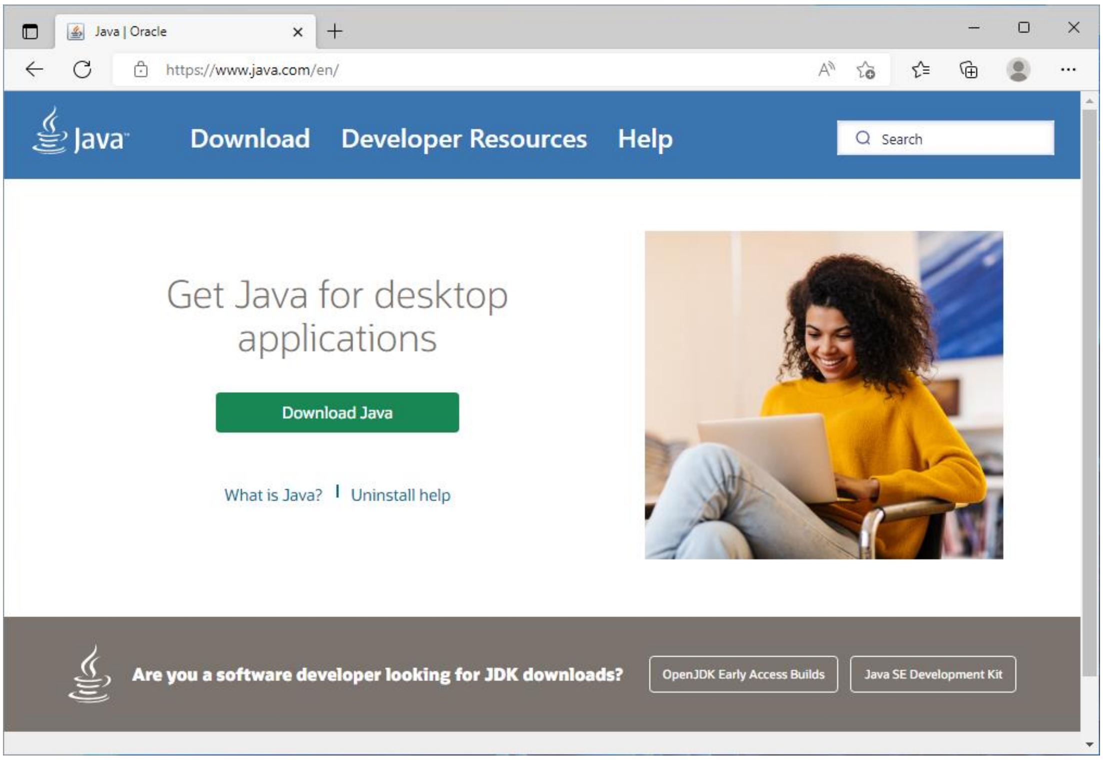
Si nous retentons la connexion, nous obtenons encore un autre message nous disant qu'il faut le runtime Java:



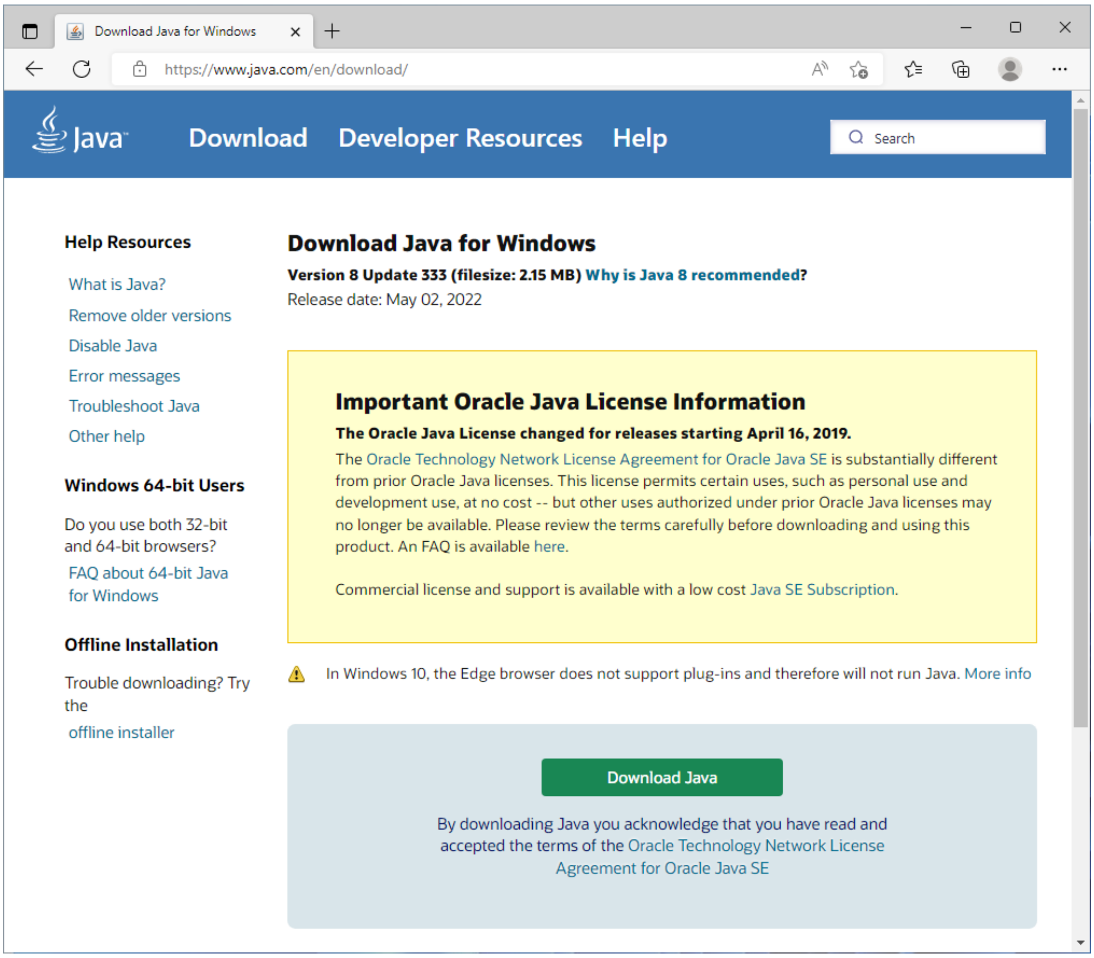
Nous allons alors sur:

https://www.java.com/en

Pour le télécharger:



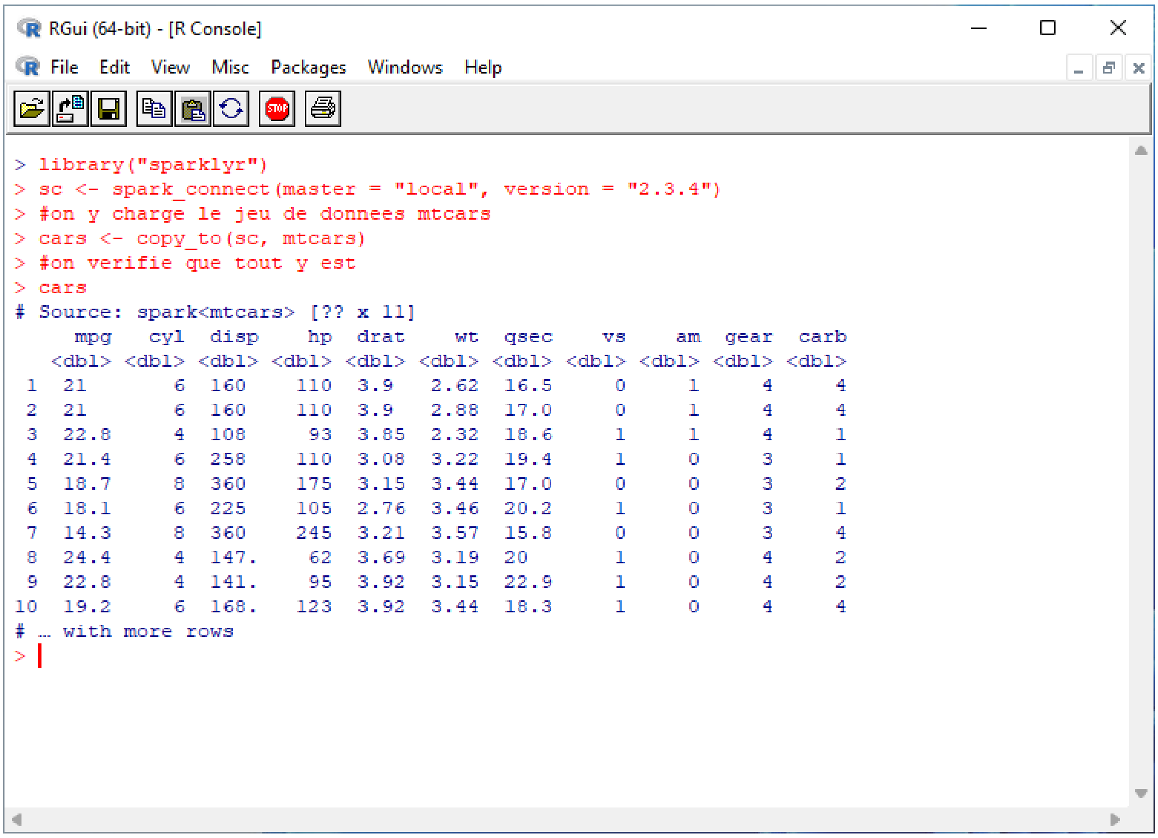
Nous cliquons sur **Download Java** pour arriver sur:



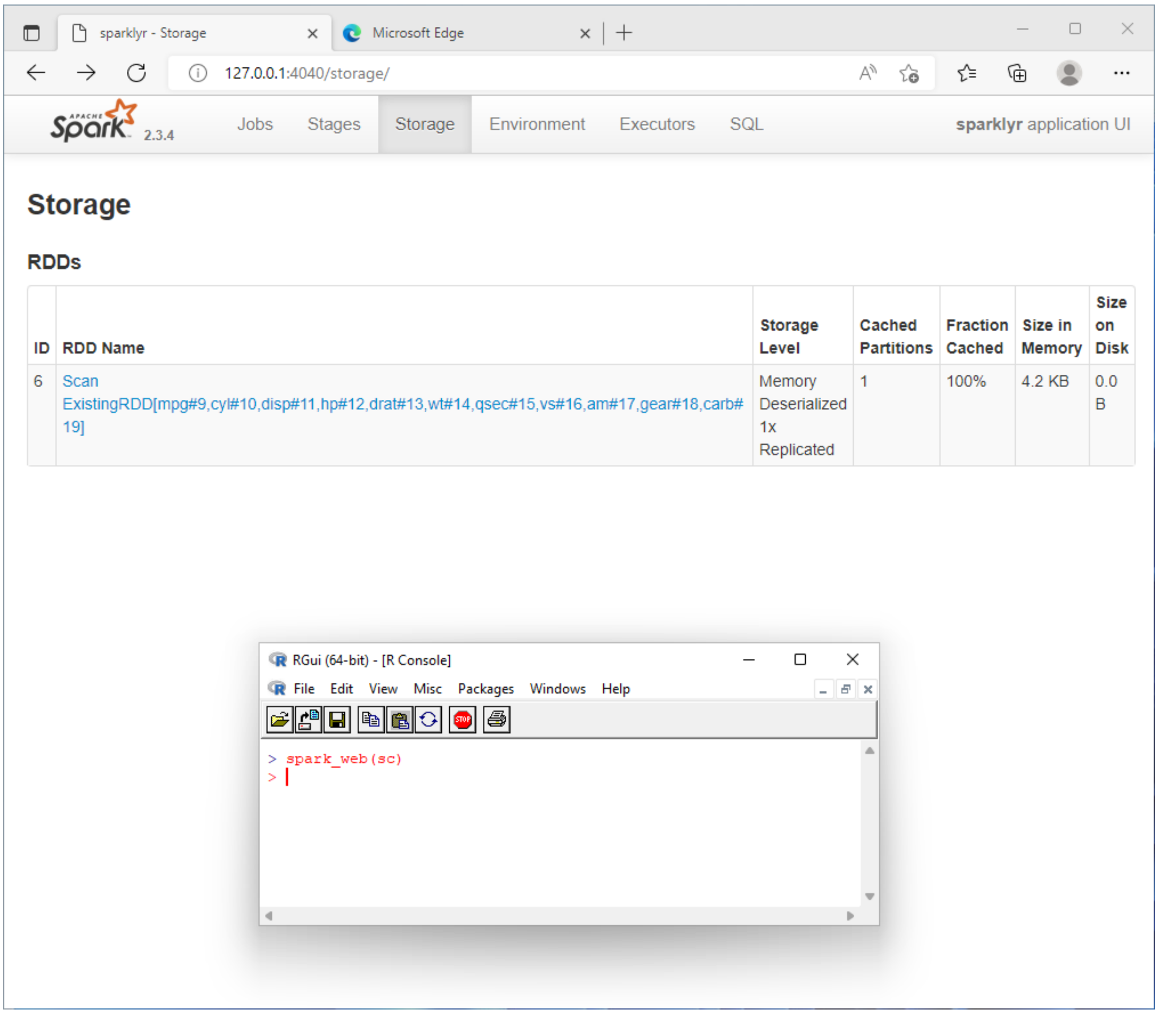
Après le téléchargement, nous installons le runtime Java:



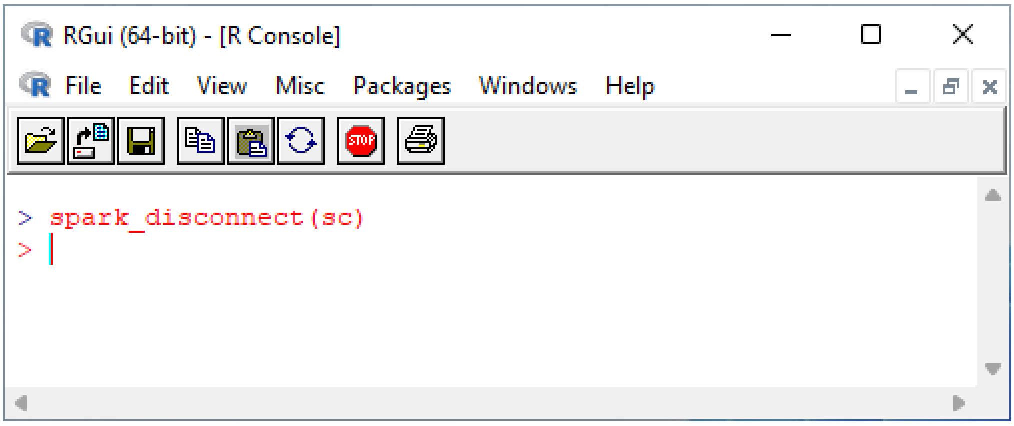
Et il faut redémarrer l'ordinateur avant de poursuivre!!! Une fois l'ordinateur redémarré, nous pouvons poursuivre en chargeant des données dans Apach Spark et en les lisant après coup:



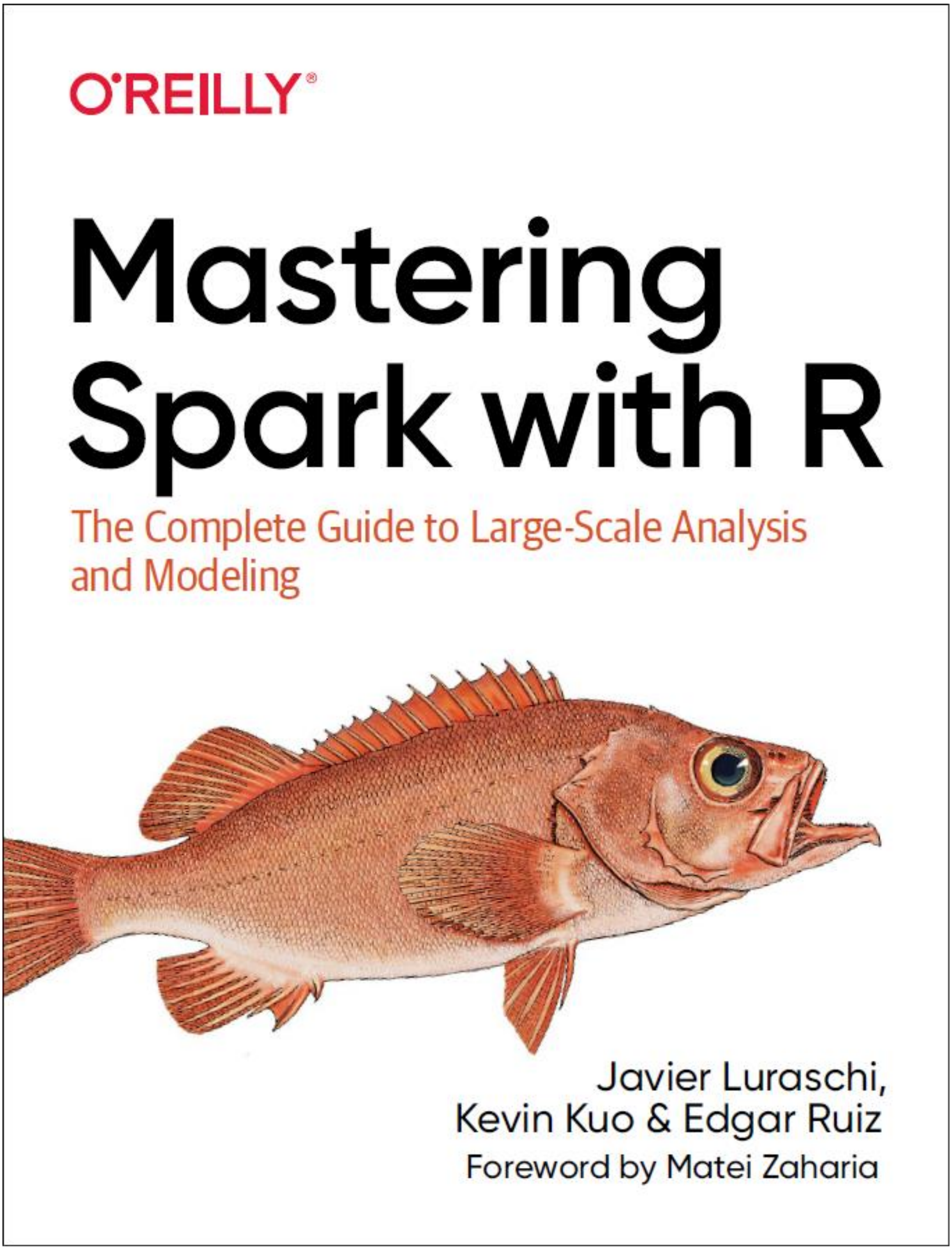
Avec la commande spark_web( ), nous pouvons ouvrir dans le navigateur Internet la sessions Apache Spark correspondante:



Et quand le travail terminé, n'oubliez pas de déconnecter la session avec la commande spark_disconnect( ):



Pour en savoir plus, nous recommandons l'excellent livre suivant:



Si des lecteurs savant comment connecter la session Apach Spark du package sparklyr à Microsoft Power BI, qu'ils n'hésitent pas à nous contacter!

Exemple 330.: Exploration de données

R 3.6.1

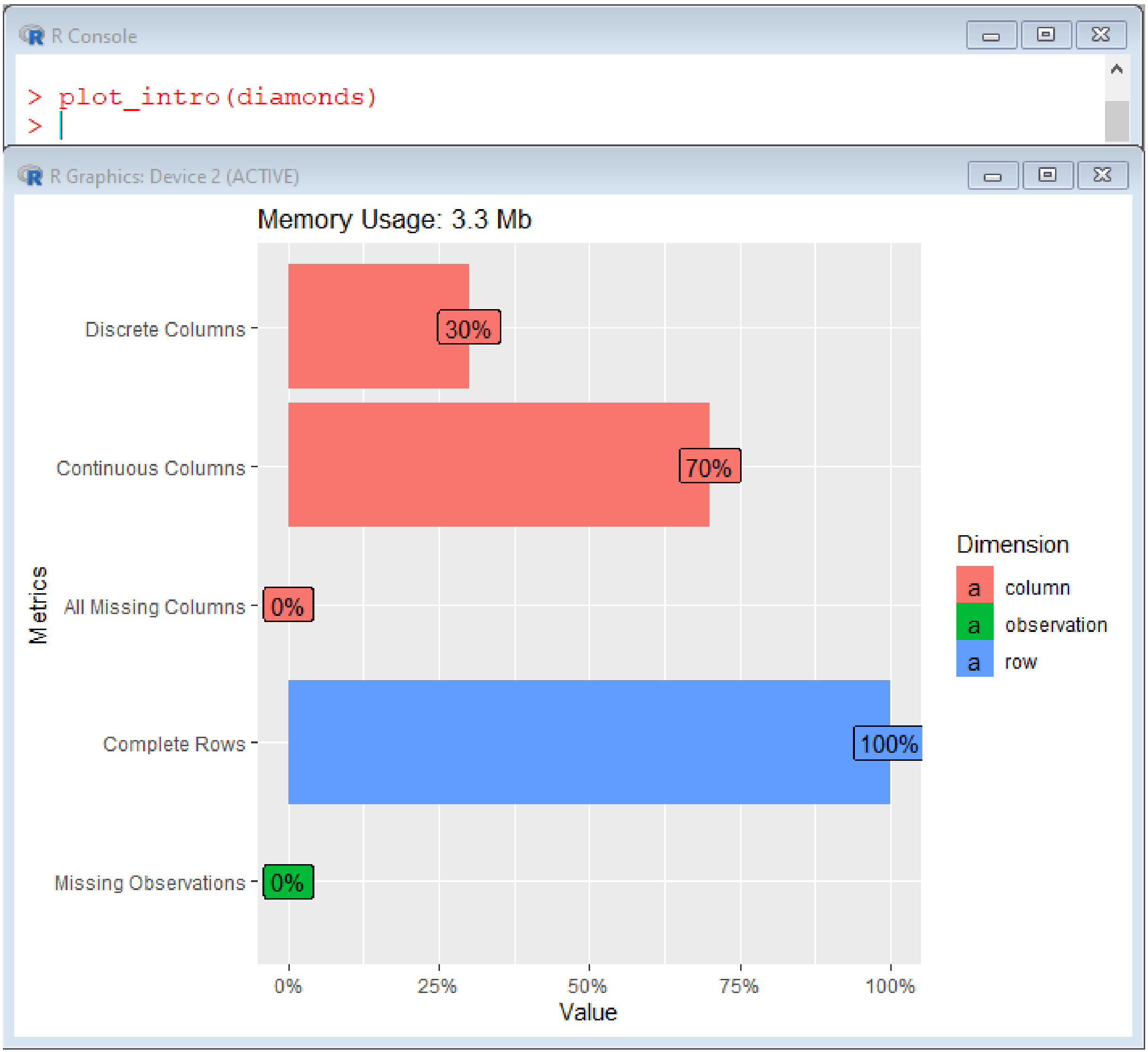
Voyons un petit un package intéressant permettant de se familiariser brièvement avec un jeu de données (nous recommandons toutefois au lecteur de lire la documentation du package pour en apprendre plus!).

Considérons le jeu de données diamonds du package ggplot2:

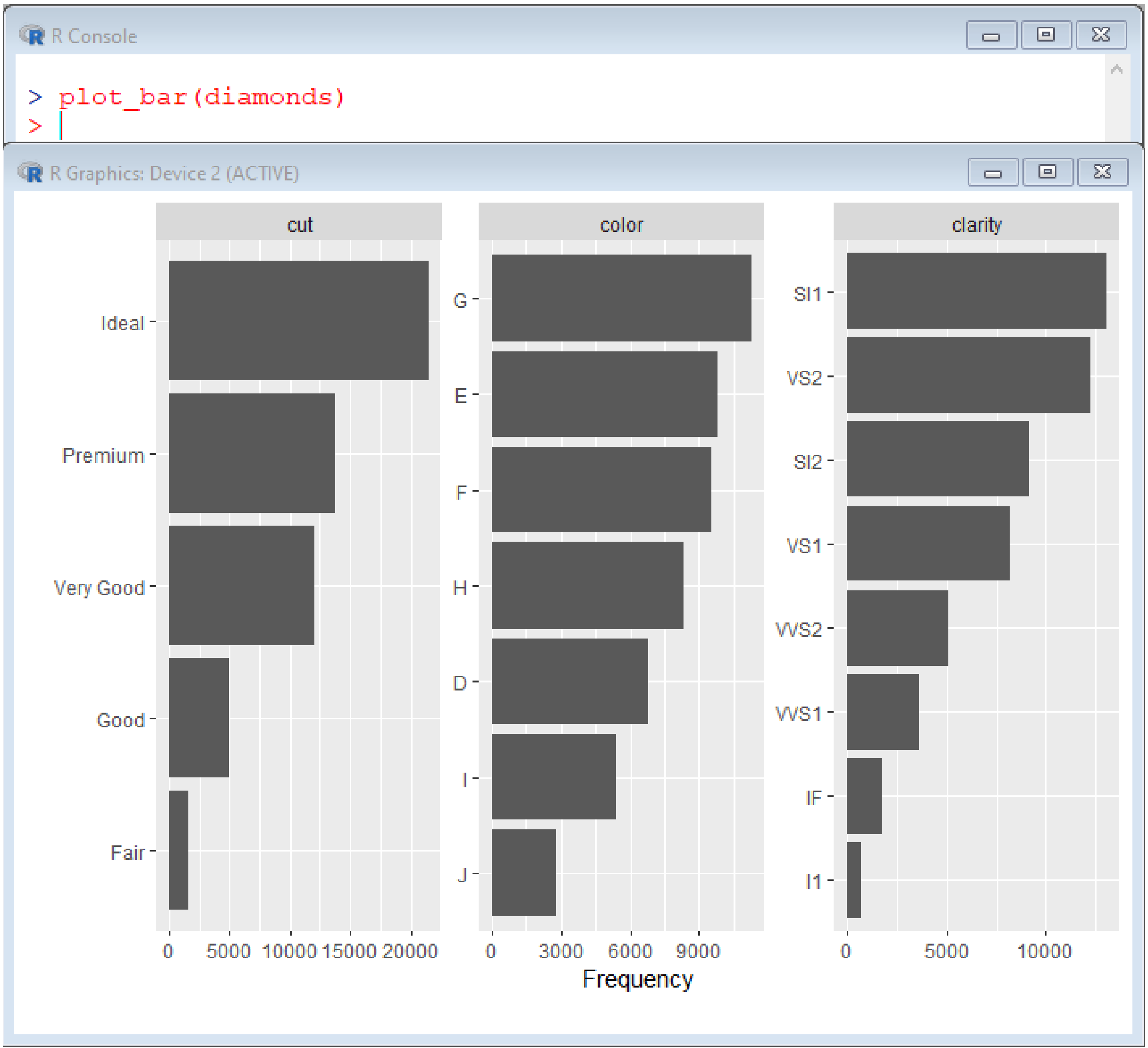
```
> library("ggplot2")
> data("diamonds")
> str(diamonds)
Classes 'tbl_df', 'tbl' and 'data.frame':       53940 obs. of  10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> head(diamonds,20)
# A tibble: 20 x 10
   carat cut    color clarity depth table
   <dbl> <ord>  <ord> <ord>   <dbl> <dbl>
 1 0.23  Ideal  E     SI2      61.5    55
 2 0.21  Premi~ E     SI1      59.8    61
 3 0.23  Good   E     VS1      56.9    65
 4 0.290 Premi~ I     VS2      62.4    58
 5 0.31  Good   J     SI2      63.3    58
 6 0.24  Very ~ J     VVS2     62.8    57
 7 0.24  Very ~ I     VVS1     62.3    57
 8 0.26  Very ~ H     SI1      61.9    55
 9 0.22  Fair   E     VS2      65.1    61
10 0.23  Very ~ H     VS1      59.4    61
11 0.3   Good   J     SI1      64      55
12 0.23  Ideal  J     VS1      62.8    56
13 0.22  Premi~ F     SI1      60.4    61
14 0.31  Ideal  J     SI2      62.2    54
15 0.2   Premi~ E     SI2      60.2    62
16 0.32  Premi~ E     I1       60.9    58
17 0.3   Ideal  I     SI2      62      54
18 0.3   Good   J     SI1      63.4    54
19 0.3   Good   J     SI1      63.8    56
20 0.3   Very ~ J     SI1      62.7    59
# ... with 4 more variables:
#   price <int>, x <dbl>, y <dbl>,
#   z <dbl>
> 
```

Maintenant voyons ce que proprose le package **DataExplorer**!

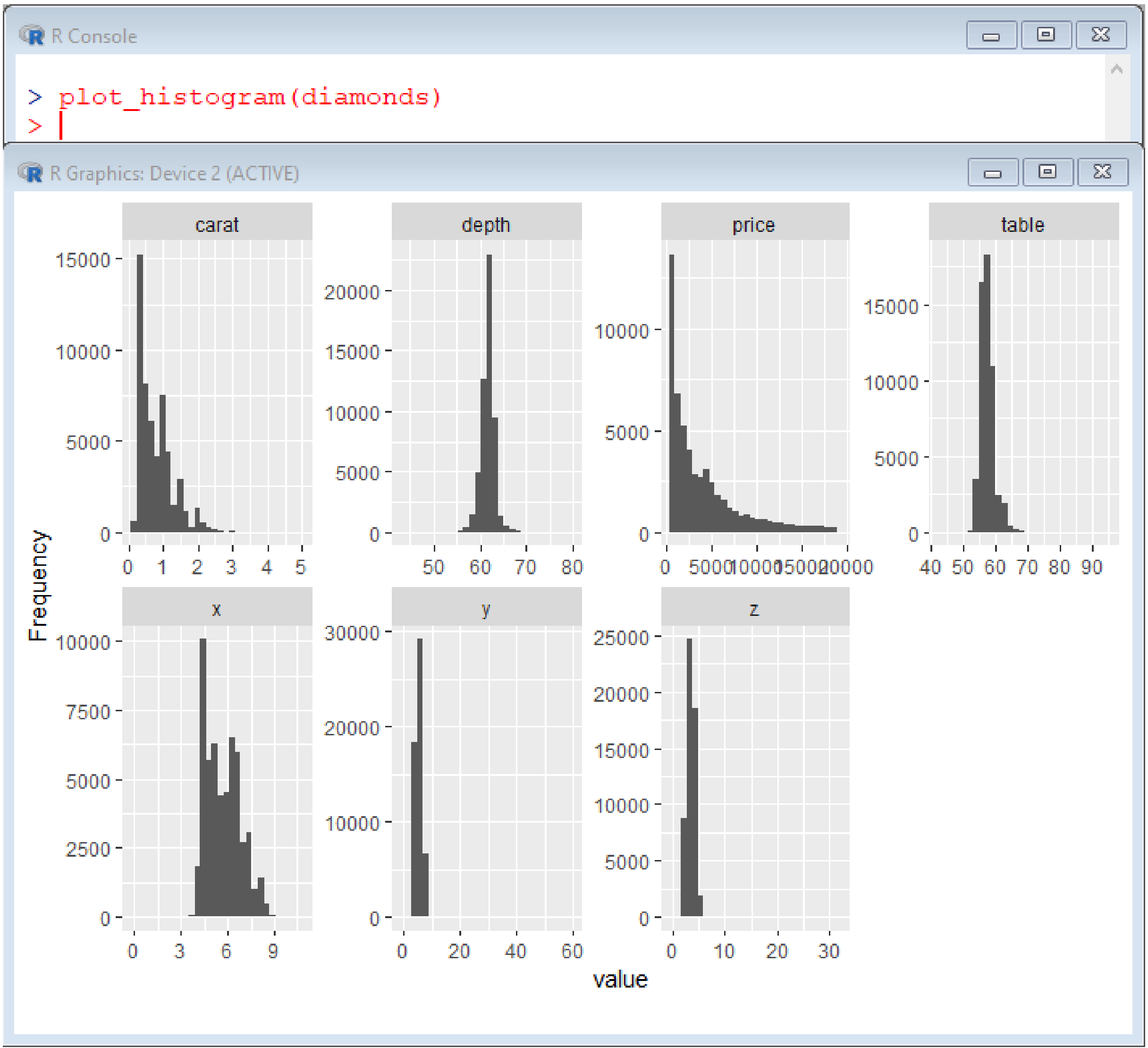
Commençons par la fonction **plot_intro( )**:



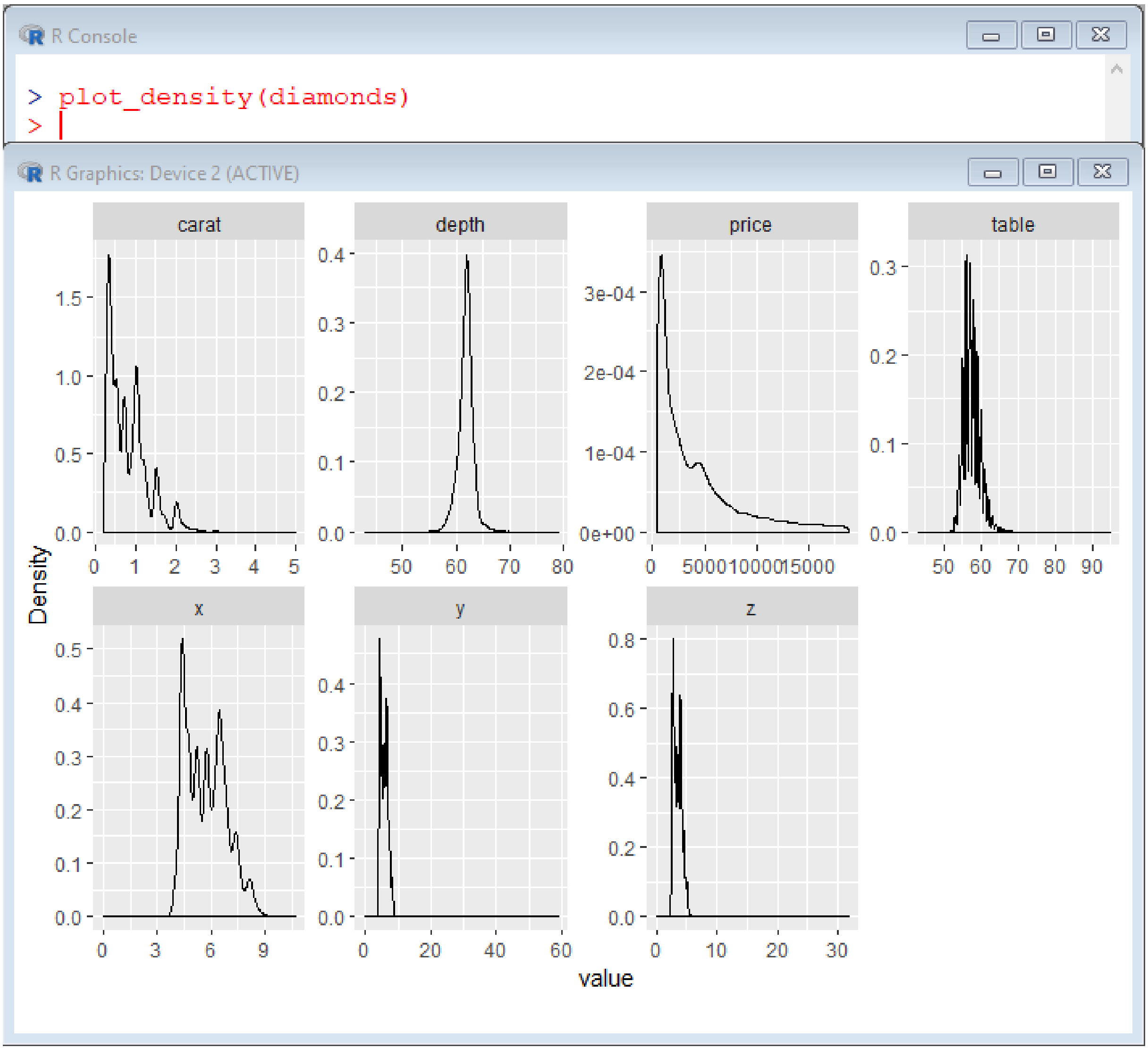
Voyons la fonction **plot_bar( )**:



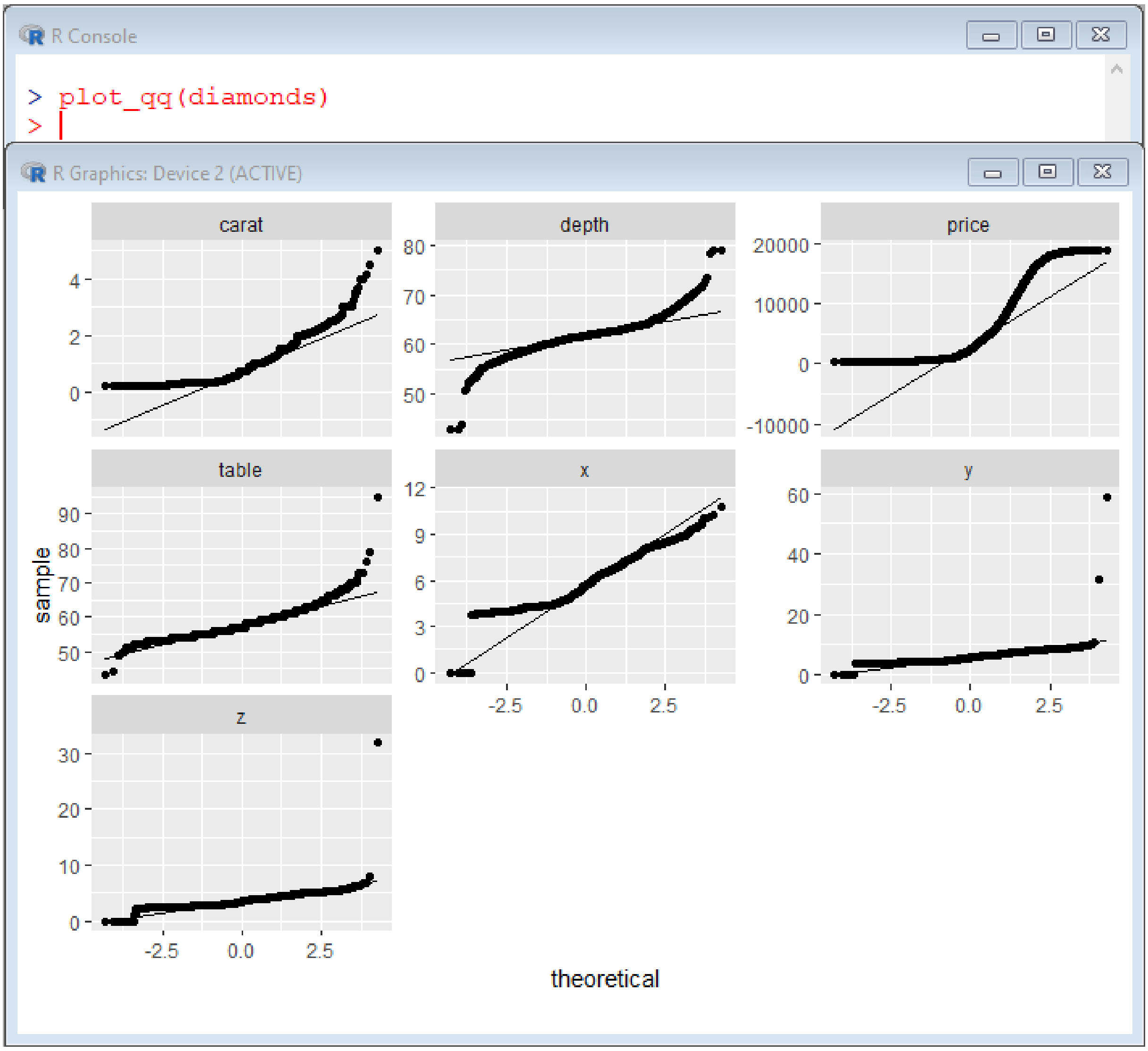
Et **plot_histogram( )**:



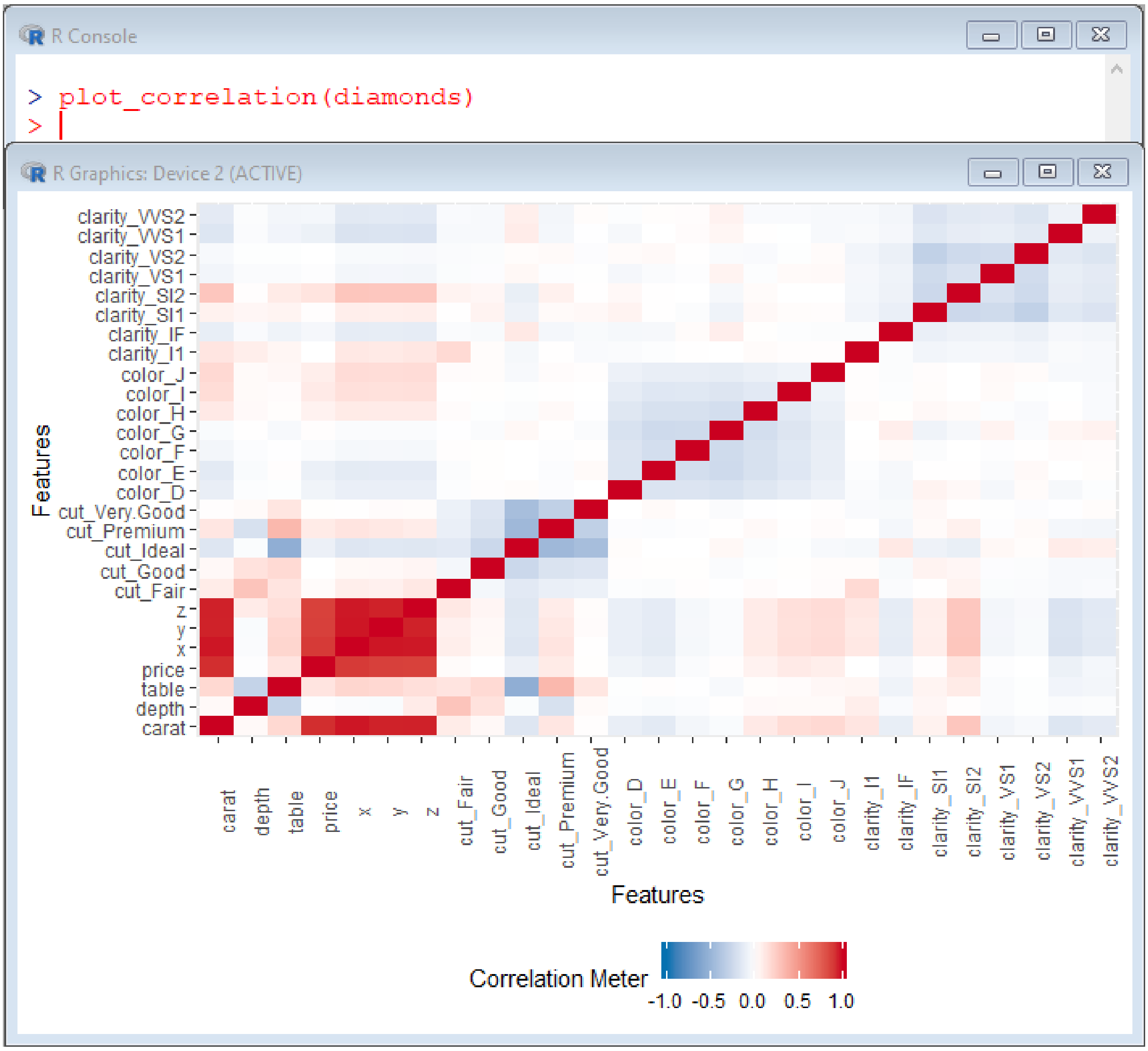
Et **plot_density( )**:


R Console
> plot_density(diamonds)
>
R Graphics: Device 2 (ACTIVE)
carat
depth
price
table
x
y
z
Density
value

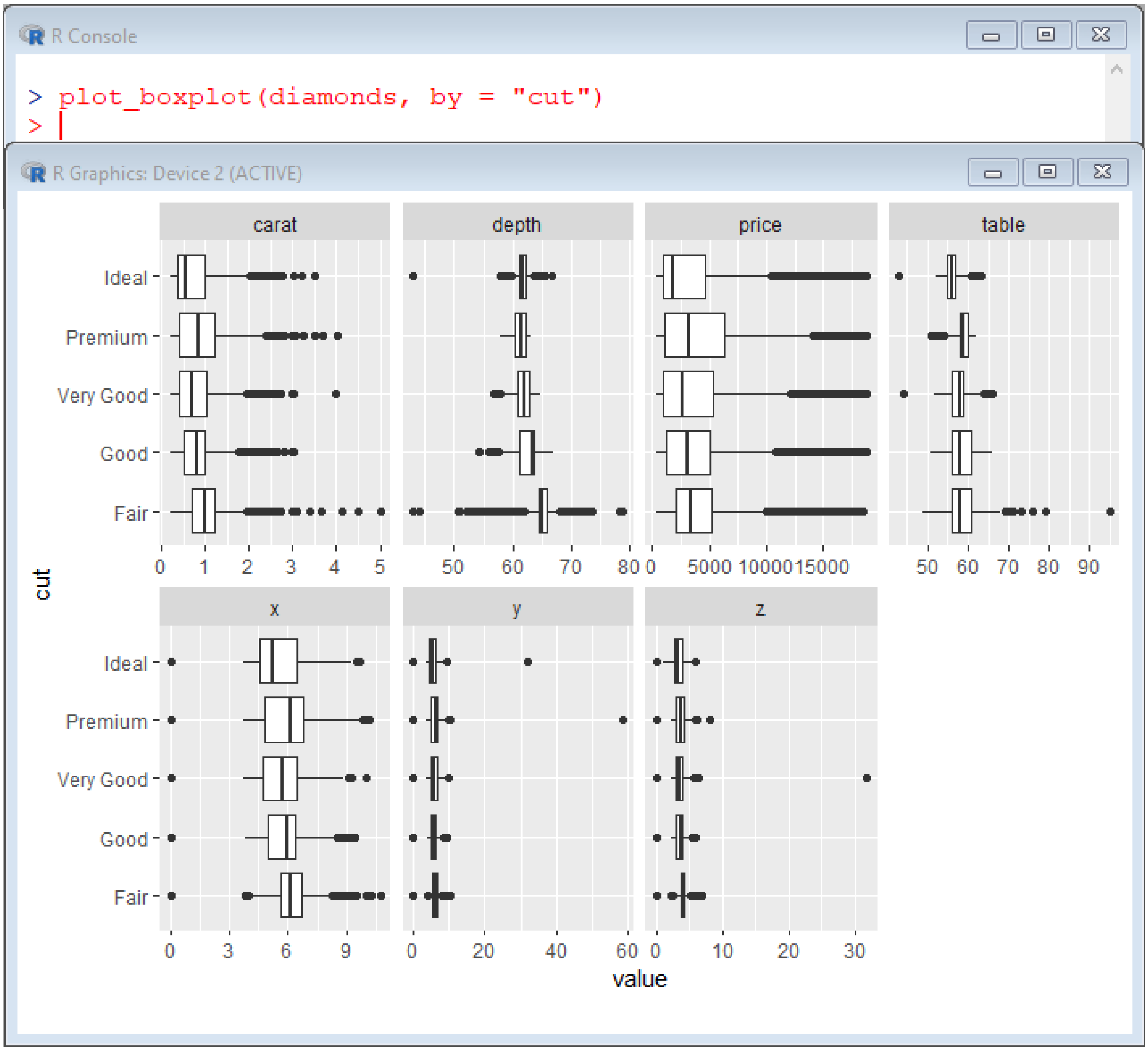
Et **plot_qq( )**:



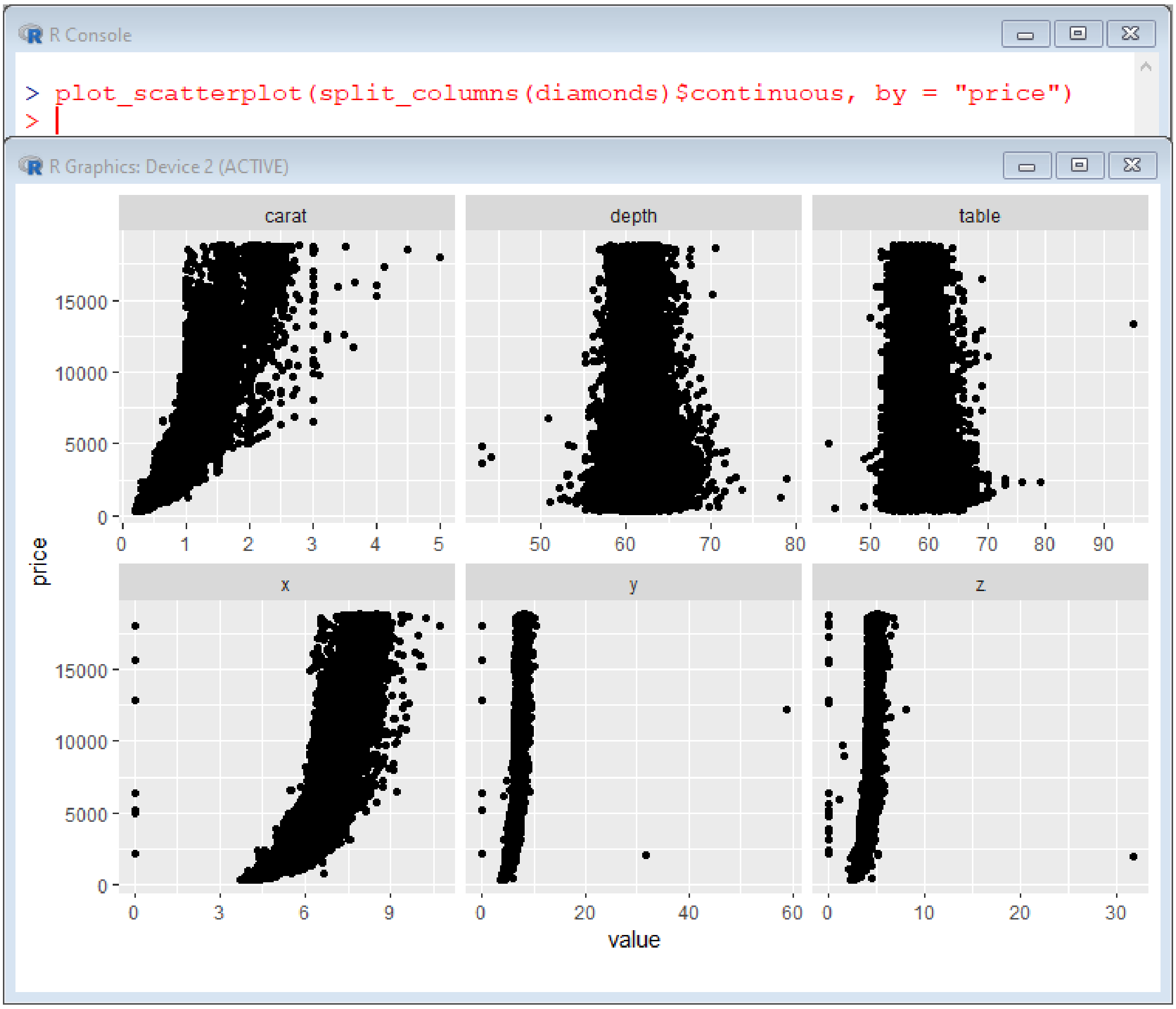
Et **plot_correlation( )**:



Et **plot_boxplot( )**:



Et **plot_scatterplot( )**:



Et **plot_prcomp( )**:



Et **plot_missing( )**:


R Console
> plot_missing(diamonds)
>
R Graphics: Device 2 (ACTIVE)
z
y
x
price
table
depth
clarity
color
cut
carat
Features
0%
-0.050
-0.025
0.000
0.025
0.05
Missing Rows
Band
a
Good

Exemple 331.: Dumification

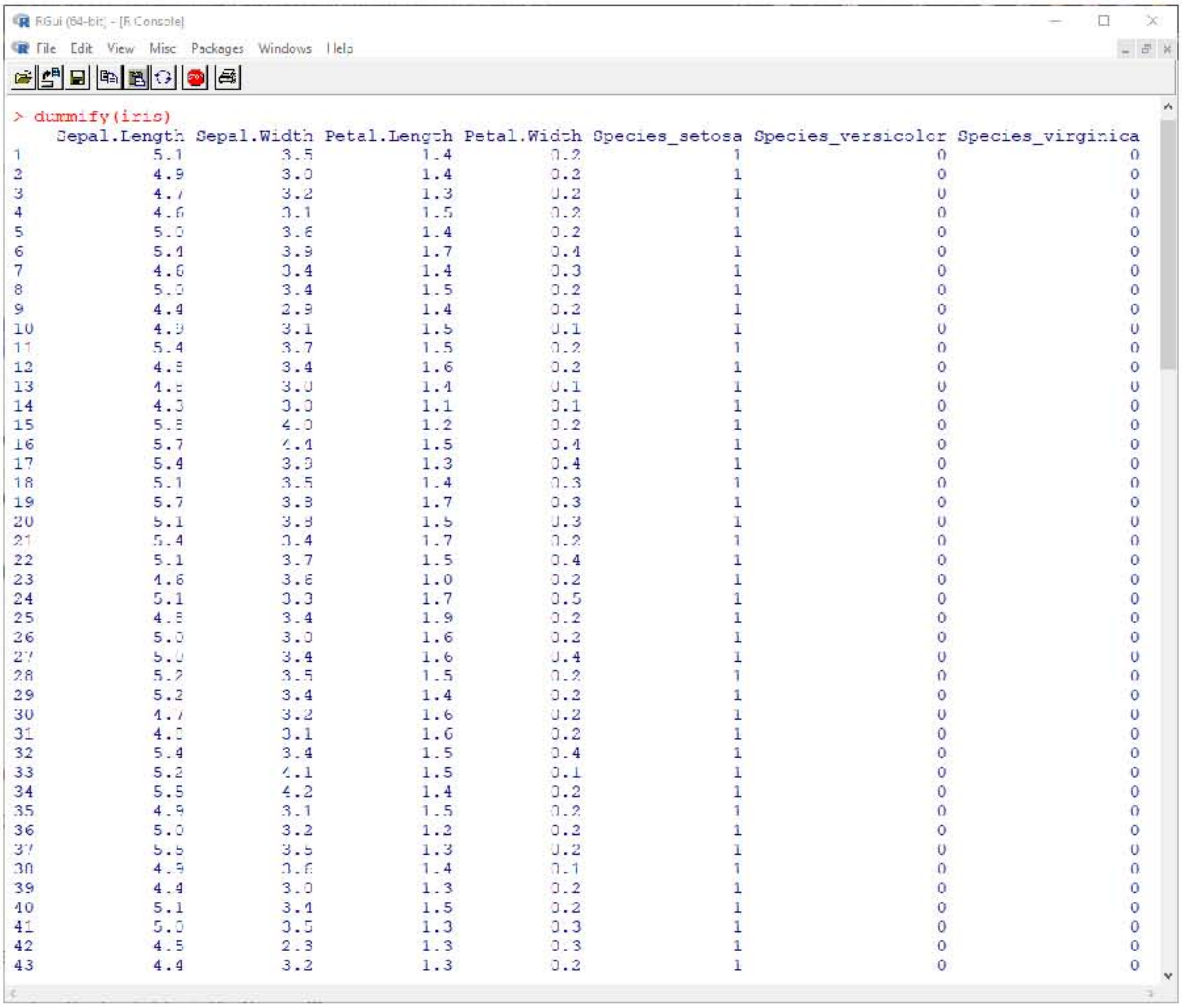
R 3.6.2

La dumification de jeu de donnée est parfois une manipulatione essentielle lors du développement de certains algorithmes particuliers. Voyons comment faire cela rapidement avec la fonction dumify( ) du package DataExplorer (package sur lequel nous reviendrons plus tard!).

Considérons le jeu de données *iris* qui pour rappel est structure de la façon suivante:

```
> data("iris")
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1$
> head(iris,20)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
8           5.0         3.4          1.5         0.2  setosa
9           4.4         2.9          1.4         0.2  setosa
10          4.9         3.1          1.5         0.1  setosa
11          5.4         3.7          1.5         0.2  setosa
12          4.8         3.4          1.6         0.2  setosa
13          4.8         3.0          1.4         0.1  setosa
14          4.3         3.0          1.1         0.1  setosa
15          5.8         4.0          1.2         0.2  setosa
16          5.7         4.4          1.5         0.4  setosa
17          5.4         3.9          1.3         0.4  setosa
18          5.1         3.5          1.4         0.3  setosa
19          5.7         3.8          1.7         0.3  setosa
20          5.1         3.8          1.5         0.3  setosa
> 
```

Allons-y pour découvrir la fonction **dummify( )** du package **DataExplorer**:



```
> dummify(iris)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species_setosa Species_versicolor Species_virginica
1           5.1         3.5          1.4         0.2              1                  0                 0
2           4.9         3.0          1.4         0.2              1                  0                 0
3           4.7         3.2          1.3         0.2              1                  0                 0
4           4.6         3.1          1.5         0.2              1                  0                 0
5           5.0         3.6          1.4         0.2              1                  0                 0
6           5.4         3.9          1.7         0.4              1                  0                 0
7           4.6         3.4          1.4         0.3              1                  0                 0
8           5.0         3.4          1.5         0.2              1                  0                 0
9           4.4         2.9          1.4         0.2              1                  0                 0
10          4.9         3.1          1.5         0.1              1                  0                 0
11          5.4         3.7          1.5         0.2              1                  0                 0
12          4.8         3.4          1.6         0.2              1                  0                 0
13          4.8         3.0          1.4         0.1              1                  0                 0
14          4.3         3.0          1.1         0.1              1                  0                 0
15          5.8         4.0          1.2         0.2              1                  0                 0
16          5.7         4.4          1.5         0.4              1                  0                 0
17          5.4         3.9          1.3         0.4              1                  0                 0
18          5.1         3.5          1.4         0.3              1                  0                 0
19          5.7         3.8          1.7         0.3              1                  0                 0
20          5.1         3.8          1.5         0.3              1                  0                 0
21          5.4         3.4          1.7         0.2              1                  0                 0
22          5.1         3.7          1.5         0.4              1                  0                 0
23          4.6         3.6          1.0         0.2              1                  0                 0
24          5.1         3.3          1.7         0.5              1                  0                 0
25          4.8         3.4          1.9         0.2              1                  0                 0
26          5.0         3.0          1.6         0.2              1                  0                 0
27          5.0         3.4          1.6         0.4              1                  0                 0
28          5.2         3.5          1.5         0.2              1                  0                 0
29          5.2         3.4          1.4         0.2              1                  0                 0
30          4.7         3.2          1.6         0.2              1                  0                 0
31          4.8         3.1          1.6         0.2              1                  0                 0
32          5.4         3.4          1.5         0.4              1                  0                 0
33          5.2         4.1          1.5         0.1              1                  0                 0
34          5.5         4.2          1.4         0.2              1                  0                 0
35          4.9         3.1          1.5         0.2              1                  0                 0
36          5.0         3.2          1.2         0.2              1                  0                 0
37          5.5         3.5          1.3         0.2              1                  0                 0
38          4.9         3.6          1.4         0.1              1                  0                 0
39          4.4         3.0          1.3         0.2              1                  0                 0
40          5.1         3.4          1.5         0.2              1                  0                 0
41          5.0         3.5          1.3         0.3              1                  0                 0
42          4.5         2.3          1.3         0.3              1                  0                 0
43          4.4         3.2          1.3         0.2              1                  0                 0
```

Exemple 332.: Normaliser un data frame

R 3.6.2

Nous avons parlé de nombreuses fois dans le cours théorique de l'importance de normaliser ou recoder certaines variables pour calculer des métriques qui ont du sens ou analyser certains modèles sur des variables catégorielles. Comme ce sujet devrait s'appliquer normalement à presque tous les sujets qui vont suivre (et à certains de ceux que nous avons déjà vu, comme en particulier les régressions!), voyons comme faire cela!

D'abord nous importons le jeu *iris* (c'est un bon petit jeu de données pour le coup!):

```
> mydata<-read.csv("c:/tmp/iris.csv",header=F,sep=",",stringsAsFactors=TRUE)
> colnames(mydata)<-c("sepal length","sepal width","petal length","petal width","species")
> head(mydata,10)
  sepal length sepal width petal length petal width     species
1          5.1         3.5          1.4         0.2 Iris-setosa
2          4.9         3.0          1.4         0.2 Iris-setosa
3          4.7         3.2          1.3         0.2 Iris-setosa
4          4.6         3.1          1.5         0.2 Iris-setosa
5          5.0         3.6          1.4         0.2 Iris-setosa
6          5.4         3.9          1.7         0.4 Iris-setosa
7          4.6         3.4          1.4         0.3 Iris-setosa
8          5.0         3.4          1.5         0.2 Iris-setosa
9          4.4         2.9          1.4         0.2 Iris-setosa
10         4.9         3.1          1.5         0.1 Iris-setosa
> 
```

Et on y va pour utiliser la fonction **normalize( )** du package **BBmisc** et **dummy_cols( )** du package **fastDummies**:

```
> #on standardise les variables numériques
> library("BBmisc")
> std_mydata<-normalize(mydata,method="standardize")
> head(std_mydata,10)
  sepal length sepal width petal length petal width     species
1   -0.8976739   1.0286113    -1.336794   -1.308593 Iris-setosa
2   -1.1392005  -0.1245404    -1.336794   -1.308593 Iris-setosa
3   -1.3807271   0.3367203    -1.393470   -1.308593 Iris-setosa
4   -1.5014904   0.1060900    -1.280118   -1.308593 Iris-setosa
5   -1.0184372   1.2592416    -1.336794   -1.308593 Iris-setosa
6   -0.5353840   1.9511326    -1.166767   -1.046525 Iris-setosa
7   -1.5014904   0.7979809    -1.336794   -1.177559 Iris-setosa
8   -1.0184372   0.7979809    -1.280118   -1.308593 Iris-setosa
9   -1.7430170  -0.3551707    -1.336794   -1.308593 Iris-setosa
10  -1.1392005   0.1060900    -1.280118   -1.439627 Iris-setosa
>
> #on dummy code les variables factorielles (catégorielles)
> library("fastDummies")
> std_mydata<-dummy_cols(std_mydata)
> head(std_mydata,10)
  sepal length sepal width petal length petal width     species species_Iris-setosa species_Iris-versicolor species_Iris-virginica
1   -0.8976739   1.0286113    -1.336794   -1.308593 Iris-setosa                   1                       0                      0
2   -1.1392005  -0.1245404    -1.336794   -1.308593 Iris-setosa                   1                       0                      0
3   -1.3807271   0.3367203    -1.393470   -1.308593 Iris-setosa                   1                       0                      0
4   -1.5014904   0.1060900    -1.280118   -1.308593 Iris-setosa                   1                       0                      0
5   -1.0184372   1.2592416    -1.336794   -1.308593 Iris-setosa                   1                       0                      0
6   -0.5353840   1.9511326    -1.166767   -1.046525 Iris-setosa                   1                       0                      0
7   -1.5014904   0.7979809    -1.336794   -1.177559 Iris-setosa                   1                       0                      0
8   -1.0184372   0.7979809    -1.280118   -1.308593 Iris-setosa                   1                       0                      0
9   -1.7430170  -0.3551707    -1.336794   -1.308593 Iris-setosa                   1                       0                      0
10  -1.1392005   0.1060900    -1.280118   -1.439627 Iris-setosa                   1                       0                      0
> 
```

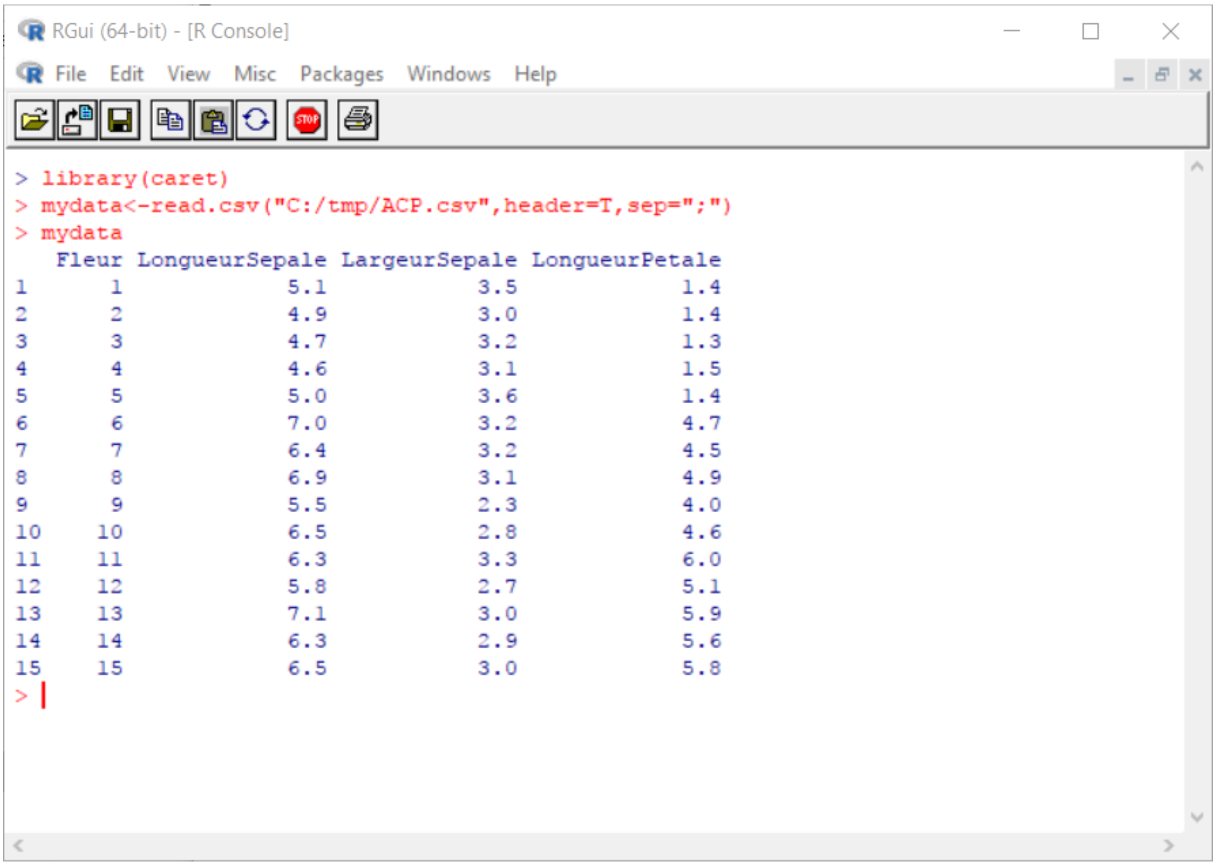
Exemple 333.: caret (classification and regression training)

R 3.0.2

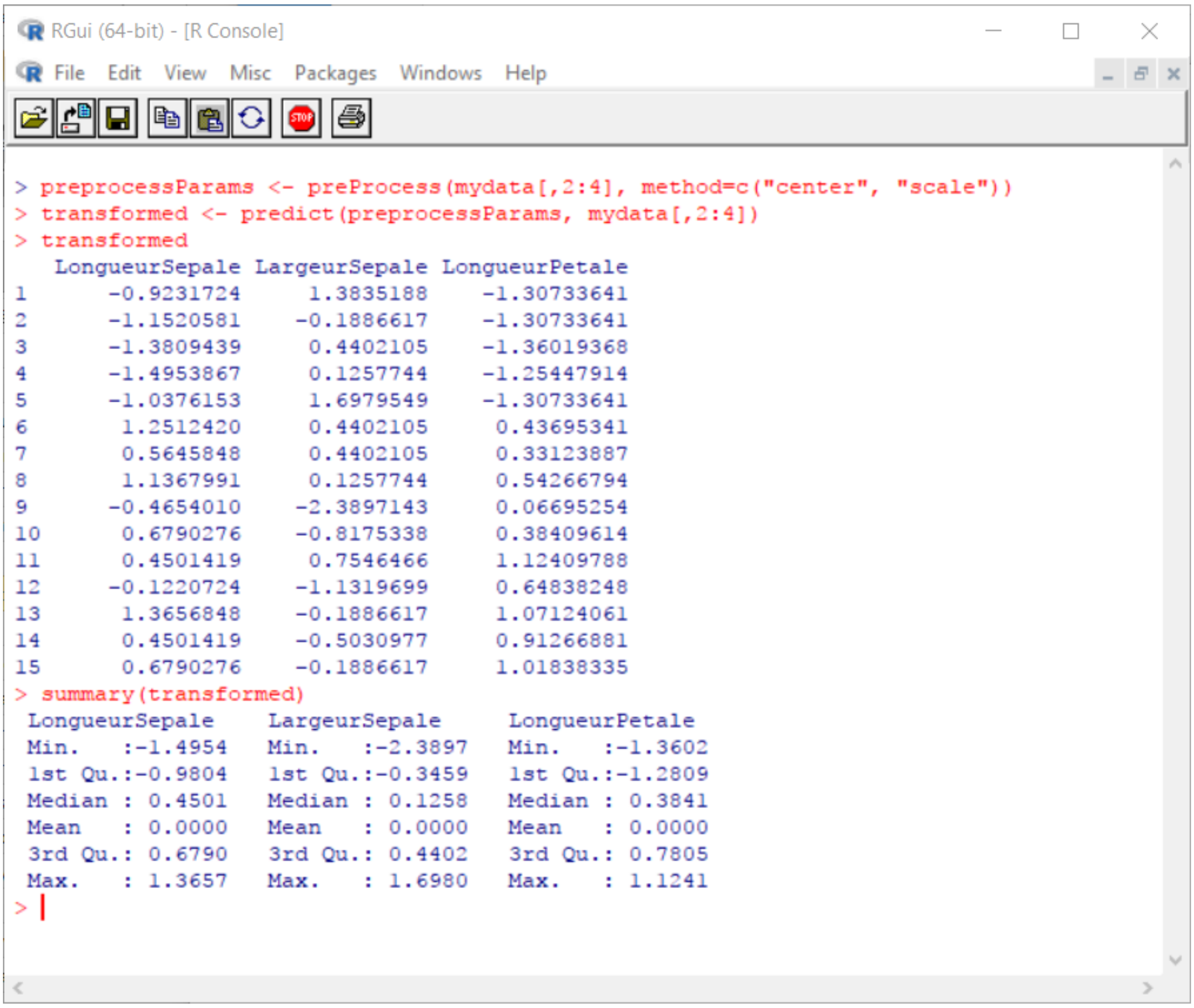
Le package **caret** est très utile pour préparer les données dans le cadre du Data Mining/Machine Learning (gain de temps considérable) et particulièrement dans le domaine de la transformation de variables et réduction dimensionnelle. Ce n'est d'ailleurs pas la première fois que nous l'utilisons dans ce livre.

Voyons quelques exemples.

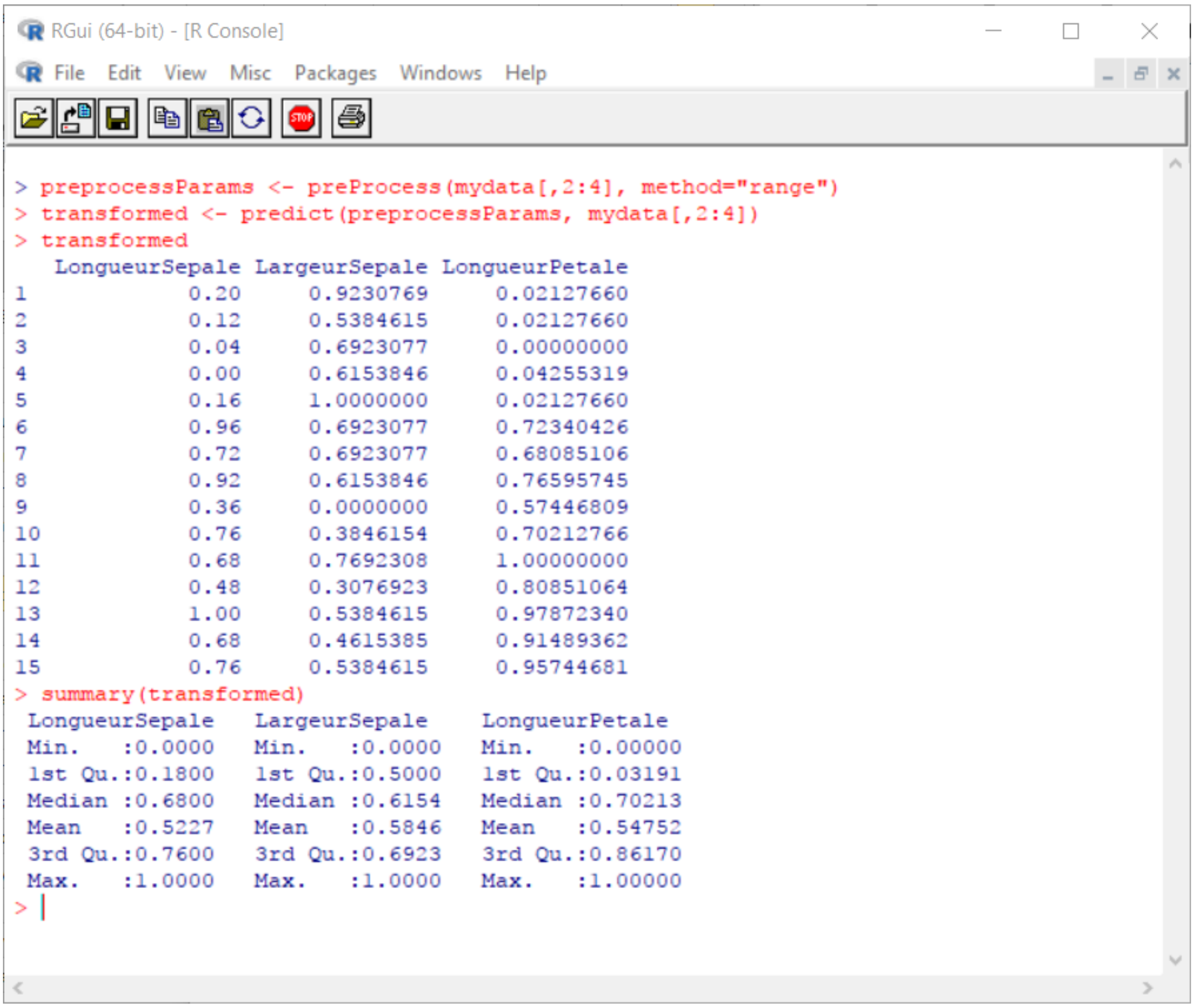
Nous chargeons donc d'abord le package **caret** avec le jeu de données qui va nous accompagner tout du long:



On commence le classique consistant à centrer réduire les données avec la fonction **preProcess( )** :



Ensuite la transformation qui normalise tout dans une plage Min-Max, toujours avec **preProcess( )** :



```
> preprocessParams <- preProcess(mydata[,2:4], method="range")
> transformed <- predict(preprocessParams, mydata[,2:4])
> transformed
   LongueurSepale LargeurSepale LongueurPetale
1            0.20     0.9230769     0.02127660
2            0.12     0.5384615     0.02127660
3            0.04     0.6923077     0.00000000
4            0.00     0.6153846     0.04255319
5            0.16     1.0000000     0.02127660
6            0.96     0.6923077     0.72340426
7            0.72     0.6923077     0.68085106
8            0.92     0.6153846     0.76595745
9            0.36     0.0000000     0.57446809
10           0.76     0.3846154     0.70212766
11           0.68     0.7692308     1.00000000
12           0.48     0.3076923     0.80851064
13           1.00     0.5384615     0.97872340
14           0.68     0.4615385     0.91489362
15           0.76     0.5384615     0.95744681
> summary(transformed)
 LongueurSepale   LargeurSepale    LongueurPetale
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.1800   1st Qu.:0.5000   1st Qu.:0.03191
 Median :0.6800   Median :0.6154   Median :0.70213
 Mean   :0.5227   Mean   :0.5846   Mean   :0.54752
 3rd Qu.:0.7600   3rd Qu.:0.6923   3rd Qu.:0.86170
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
>
```

Et pour normaliser, la non-moins fameuse transformation de Box-Cox, toujours avec la fonction **preProcess( )** :

```
> preprocessParams <- preProcess(mydata[,2:4], method="BoxCox")
> transformed <- predict(preprocessParams, mydata[,2:4])
> preprocessParams
Created from 15 samples and 3 variables

Pre-processing:
  - Box-Cox transformation (3)
  - ignored (0)

Lambda estimates for Box-Cox transformation:
1.6, 2, 1.1
> transformed
   LongueurSepale LargeurSepale LongueurPetale
1        7.847124         5.625            1.4
2        7.321827         4.000            1.4
3        6.809240         4.620            1.3
4        6.557780         4.305            1.5
5        7.582899         5.980            1.4
6       13.436669         4.620            4.7
7       11.558385         4.620            4.5
8       13.116640         4.305            4.9
9        8.935053         2.145            4.0
10      11.864395         3.420            4.6
11      11.255231         4.945            6.0
12       9.782941         3.145            5.1
13      13.759454         4.000            5.9
14      11.255231         3.705            5.6
15      11.864395         4.000            5.8
> 
```

et la transformation de Yeo-Johnson, toujours avec la fonction **preProcess( )** :

```
> preprocessParams <- preProcess(mydata[,2:4], method="YeoJohnson")
> preprocessParams
Created from 15 samples and 3 variables

Pre-processing:
  - ignored (0)
  - Yeo-Johnson transformation (3)

Lambda estimates for Yeo-Johnson transformation:
1.79, 3.2, 1.42
> transformed <- predict(preprocessParams, mydata[,2:4])
> transformed
   LongueurSepale LargeurSepale LongueurPetale
1        13.65738      38.34347       1.738598
2        12.83397      26.19231       1.738598
3        12.03233      30.67702       1.595220
4        11.63971      28.37441       1.884517
5        13.24297      41.16377       1.738598
6        22.53787      30.67702       7.649392
7        19.52982      30.67702       7.235818
8        22.02370      28.37441       8.069132
9        15.36885      13.99774       6.229724
10       20.01828      22.17555       7.441826
11       19.04654      33.10368      10.482786
12       16.70847      20.33388       8.494917
13       23.05715      26.19231      10.256278
14       19.04654      24.12720       9.585081
15       20.01828      26.19231      10.031150
> 
```

Et pour faire une analyse en composantes principales et automatiquement extraire les axes principaux qui "expliquent" 95% de la variance, nous avons là encore très facilement et toujours avec la fonction **preProcess( )** :

```
> preprocessParams <- preProcess(mydata[,2:4], method=c("center", "scale", "pca"))
> preprocessParams
Created from 15 samples and 3 variables

Pre-processing:
  - centered (3)
  - ignored (0)
  - principal component signal extraction (3)
  - scaled (3)

PCA needed 2 components to capture 95 percent of the variance
> transformed <- predict(preprocessParams, mydata[,2:4])
> transformed
          PC1        PC2
1  -1.9709735 -0.7650496
2  -1.5631021  0.7643386
3  -1.9676850  0.2806385
4  -1.8580624  0.6002476
5  -2.1552491 -1.0099953
6   0.9446253 -0.9308860
7   0.4324134 -0.6573665
8   1.0542478 -0.6112769
9   0.5902901  2.3615116
10  0.9854731  0.4467935
11  0.7886114 -0.9951791
12  0.7630417  1.0089665
13  1.6721973 -0.4717654
14  1.0881551  0.1834595
15  1.1960170 -0.2044370
>
```

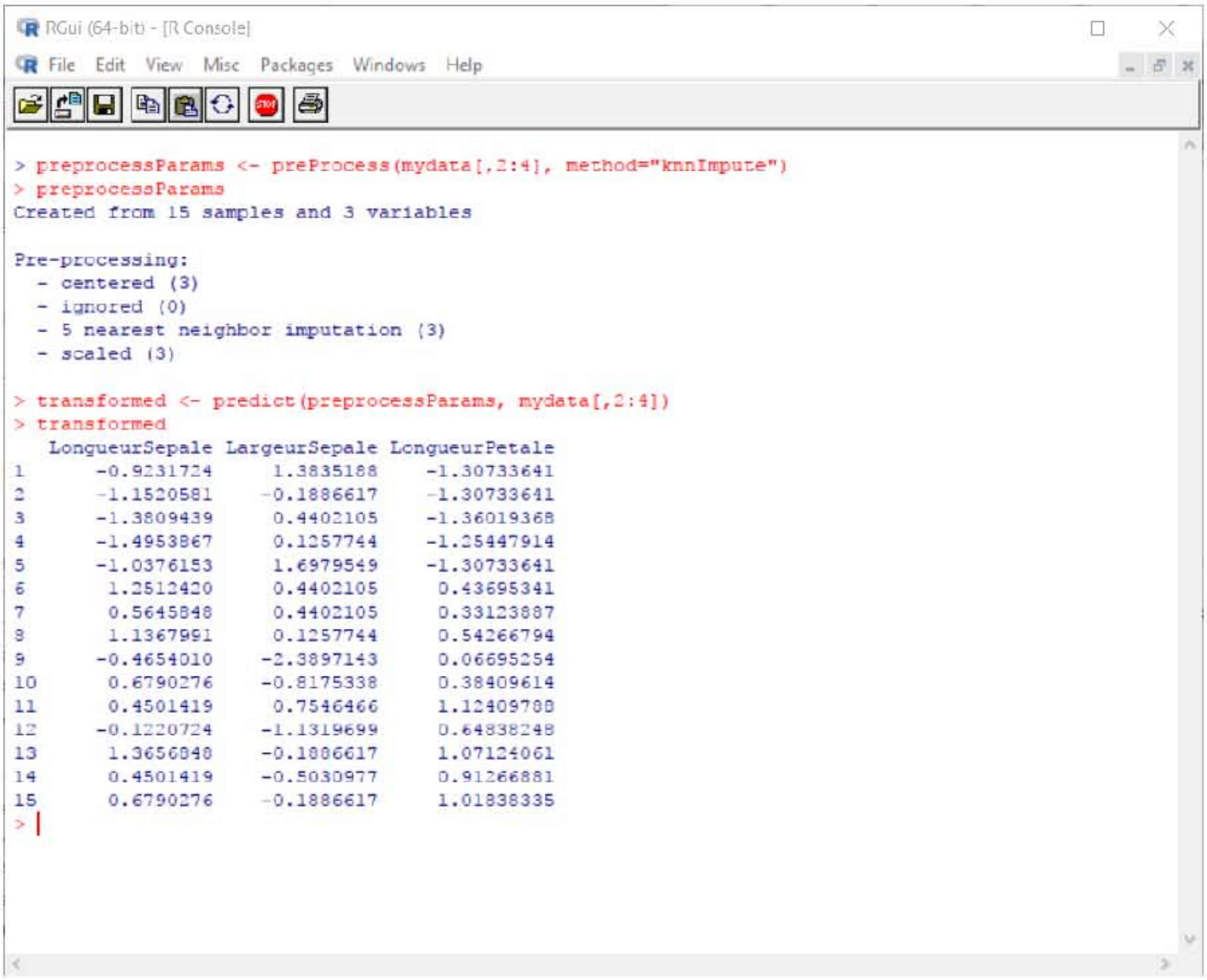
Nous pouvons aussi faire une analyse en composantes indépendantes (ICA), toujours avec la fonction **preProcess( )** :

```
> preprocessParams <- preProcess(mydata[,2:4], method=c("center", "scale", "ica"), n.comp=2)
> preprocessParams
Created from 15 samples and 3 variables

Pre-processing:
  - centered (3)
  - independent component signal extraction (3)
  - ignored (0)
  - scaled (3)

ICA used 2 components
> transformed <- predict(preprocessParams, mydata[,2:4])
> transformed
          ICA1       ICA2
1  -1.62263495 -0.3552151
2  -0.81780701  1.1525071
3  -1.26263553  0.7378897
4  -1.07760573  1.0474644
5  -1.83366871 -0.5698301
6   0.33378712 -1.1870655
7   0.07368723 -0.7852856
8   0.51881692 -0.8774908
9   1.21545241  2.3369767
10  0.83329433  0.2448392
11  0.20406851 -1.2190898
12  0.87202926  0.8831307
13  0.99322369 -0.8710944
14  0.81410762 -0.0538095
15  0.75588481 -0.4839271
> 
```

Et nous pouvons compléter les valeurs manquantes numériques avec les k-NN, encore et toujours avec la fonction **preProcess( )** :


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> preprocessParams <- preProcess(mydata[,2:4], method="knnImpute")
> preprocessParams
Created from 15 samples and 3 variables
Pre-processing:
- centered (3)
- ignored (0)
- 5 nearest neighbor imputation (3)
- scaled (3)
> transformed <- predict(preprocessParams, mydata[,2:4])
> transformed
LongueurSepale LargeurSepale LongueurPetale
1 -0.9231724 1.3835188 -1.30733641
2 -1.1520581 -0.1886617 -1.30733641
3 -1.3809439 0.4402105 -1.36019368
4 -1.4953867 0.1257744 -1.25447914
5 -1.0376153 1.6979549 -1.30733641
6 1.2512420 0.4402105 0.43695341
7 0.5645848 0.4402105 0.33123887
8 1.1367991 0.1257744 0.54266794
9 -0.4654010 -2.3897143 0.06695254
10 0.6790276 -0.8175338 0.38409614
11 0.4501419 0.7546466 1.12409788
12 -0.1220724 -1.1319699 0.64838248
13 1.3656848 -0.1886617 1.07124061
14 0.4501419 -0.5030977 0.91266881
15 0.6790276 -0.1886617 1.01838335
>

Exemple 334.: Balancement (équilibrage) des données

R 3.6.3

Le texte qui suite est une adaptation en français du blog suivant (2 Avril 2017, Shirin Elsinghorst):

https://shiring.github.io/machine_learning/2017/04/02/unbalanced

Le but est de mettre en pratique des techniques de balancement de données que nous avons brièvement parcourues dans le cours théorique.

Commençons par importer et préparer un peu les données:

```
> bc_data <- read.table("c:/tmp/breast-cancer-wisconsin.data", header = F, sep = ",")
> #on renomme les colonnes
> colnames(bc_data) <- c("sample_code_number", "clump_thickness", "uniformity_of_cell_size",
+                        "uniformity_of_cell_shape", "marginal_adhesion",
+                        "single_epithelial_cell_size", "bare_nuclei", "bland_chromatin",
+                        "normal_nucleoli", "mitosis", "classes")
>
> #on renomme les valeurs de la colonne Classes
> bc_data$classes <- ifelse(bc_data$classes == "2", "benign",
+                           ifelse(bc_data$classes == "4", "malignant", NA))
>
> #on transforme les valeurs de cette colonnes en facteurs
> bc_data$classes<- as.factor(bc_data$classes)
>
> #un petit résumé et aperçu pour aider à la compréhension
> str(bc_data)
'data.frame':   699 obs. of  11 variables:
 $ sample_code_number         : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 ...
 $ clump_thickness            : int  5 5 3 6 4 8 1 2 2 4 ...
 $ uniformity_of_cell_size    : int  1 4 1 8 1 10 1 1 1 2 ...
 $ uniformity_of_cell_shape   : int  1 4 1 8 1 10 1 2 1 1 ...
 $ marginal_adhesion          : int  1 5 1 1 3 8 1 1 1 1 ...
 $ single_epithelial_cell_size: int  2 7 2 3 2 7 2 2 2 2 ...
 $ bare_nuclei                : Factor w/ 11 levels "?","1","10","2",..: 2 3 4 6 2 3 3 2 2 2 ...
 $ bland_chromatin            : int  3 3 3 3 3 9 3 3 1 2 ...
 $ normal_nucleoli            : int  1 2 1 7 1 7 1 1 1 1 ...
 $ mitosis                    : int  1 1 1 1 1 1 1 1 5 1 ...
 $ classes                    : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
>
```

Voyons un peut le contenu du jeu de données:

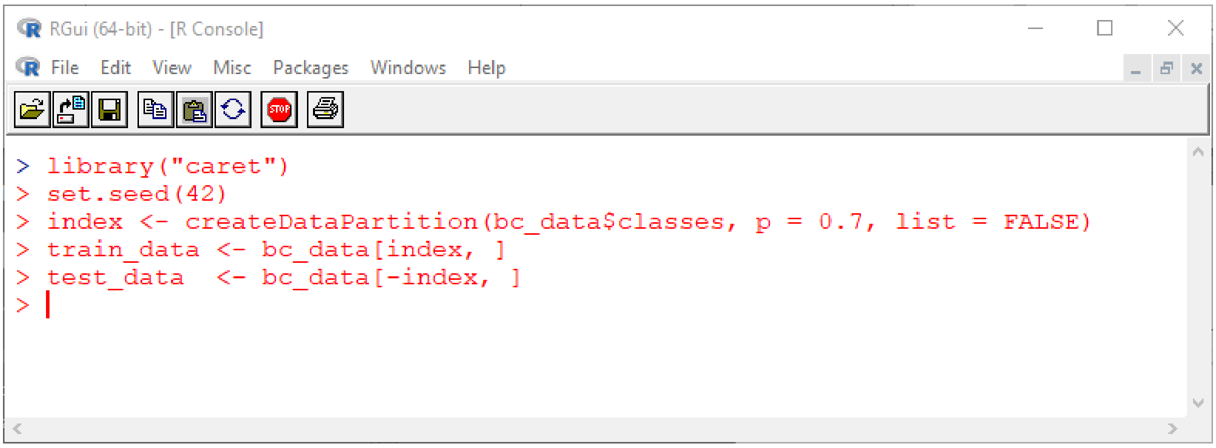
```
> #un petit aperçu pour ceux ne connaissant pas le contenu du jeu de données
> head(bc_data,10)
   sample_code_number clump_thickness uniformity_of_cell_size uniformity_of_cell_shape
1             1000025               5                       1                        1
2             1002945               5                       4                        4
3             1015425               3                       1                        1
4             1016277               6                       8                        8
5             1017023               4                       1                        1
6             1017122               8                      10                       10
7             1018099               1                       1                        1
8             1018561               2                       1                        2
9             1033078               2                       1                        1
10            1033078               4                       2                        1
   marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin
1                  1                           2           1               3
2                  5                           7          10               3
3                  1                           2           2               3
4                  1                           3           4               3
5                  3                           2           1               3
6                  8                           7          10               9
7                  1                           2          10               3
8                  1                           2           1               3
9                  1                           2           1               1
10                 1                           2           1               2
   normal_nucleoli mitosis   classes
1                1       1    benign
2                2       1    benign
3                1       1    benign
4                7       1    benign
5                1       1    benign
6                7       1 malignant
7                1       1    benign
8                1       1    benign
9                1       5    benign
10               1       1    benign
>
```


Voyons si les classes sont équilibrées:

```
> #voyons si les classes sont équilibrées
> summary(bc_data$classes)
   benign malignant
      458       241
>
>
```

Pas vraiment…

Pour la suite nous allons créer un jeu de données d'entraînement et de test:



Maintenant voyons les différent scénarios:

- Random Forest sur données d'origine (non balancées)
- Random Forest sur données sous-balancées
- Random Forest sur données sur-balancées
- Random Forest sur données balancées avec ROSE (Random Over-Sampling Examples)
- Random Forest sur données balancées avec SMOTe (Synthetic Minority Over-sampling Technique)

Random Forest sur données d'origine (non balancées)

Avec le package **caret**, exécutons un random forest et construisons sa matrice de confusion:

```
> library("caret")
> set.seed(42)
> model_rf <- train(classes ~ .,data = train_data,method = "rf",
+                          preProcess = c("scale", "center"),
+                          trControl = trainControl(method = "repeatedcv",
+                                                   number = 10, repeats = 10,
+                                                   verboseIter = FALSE))
> final <- data.frame(actual = test_data$classes,
+                      predict(model_rf, newdata = test_data, type = "prob"))
> final$predict <- ifelse(final$benign > 0.5, "benign", "malignant")
> (cm_original <- confusionMatrix(as.factor(final$predict), test_data$classes))
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign       134         0
  malignant      3        72

               Accuracy : 0.9856
                 95% CI : (0.9586, 0.997)
    No Information Rate : 0.6555
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9685

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9781
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9600
             Prevalence : 0.6555
         Detection Rate : 0.6411
   Detection Prevalence : 0.6411
      Balanced Accuracy : 0.9891

       'Positive' Class : benign

>
```

Random Forest sur données sous-balancées

Toujours avec le package **caret**, exécutons un random forest sur les données sous-balancées et construisons sa matrice de confusion:

```
> library("caret")
> set.seed(42)
> ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
+ verboseIter = FALSE, sampling = "down")
>
> model_rf_under <- train(classes ~ ., data = train_data,
+                          method = "rf",preProcess = c("scale", "center"),
+                          trControl = ctrl)
> final_under <- data.frame(actual = test_data$classes,
+                     predict(model_rf_under, newdata = test_data, type = "prob"))
> final_under$predict <- ifelse(final_under$benign > 0.5, "benign", "malignant")
> (cm_under <- confusionMatrix(as.factor(final_under$predict), test_data$classes))
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign       134         0
  malignant      3        72

               Accuracy : 0.9856
                 95% CI : (0.9586, 0.997)
    No Information Rate : 0.6555
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9685

 Mcnemar's Test P-Value : 0.2482

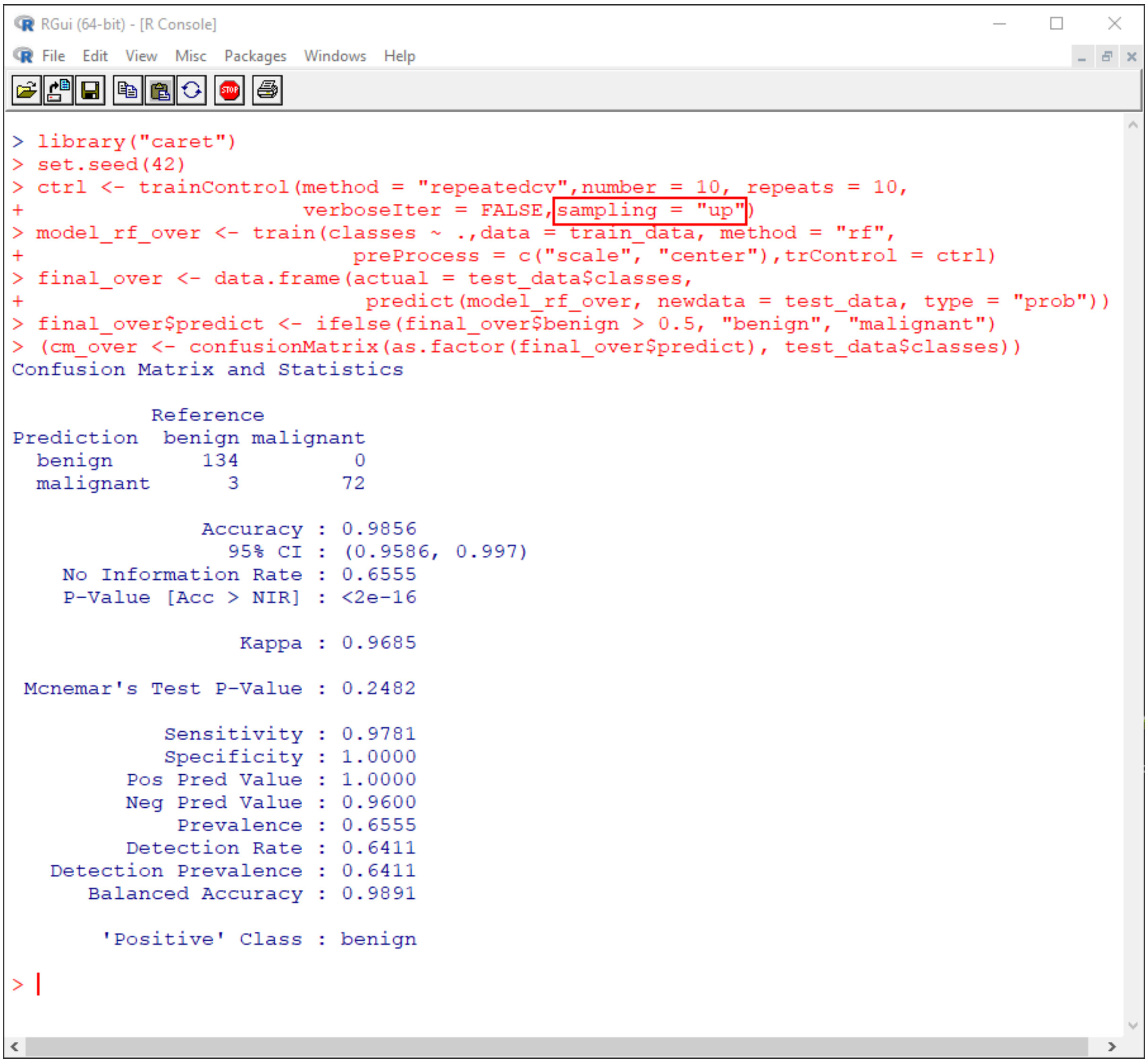
            Sensitivity : 0.9781
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9600
             Prevalence : 0.6555
         Detection Rate : 0.6411
   Detection Prevalence : 0.6411
      Balanced Accuracy : 0.9891

       'Positive' Class : benign

>
```

Random Forest sur données sur-balancées

Toujours avec le package **caret**, exécutons un random forest sur les données sur-balancées et construisons sa matrice de confusion:



Random Forest sur données balancées avec ROSE (Random Over-Sampling Examples)

Toujours avec le package **caret**, exécutons un random forest sur les données balancées avec la méthode ROSE et construisons sa matrice de confusion:

```
> library("caret")
> library("ROSE")
> set.seed(42)
> ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
+                      verboseIter = FALSE, sampling = "rose")
> model_rf_rose <- caret::train(classes ~ .,data = train_data,method = "rf",
+                               preProcess = c("scale", "center"),trControl = ctrl)
> final_rose <- data.frame(actual = test_data$classes,
+                          predict(model_rf_rose, newdata = test_data, type = "prob"))
> final_rose$predict <- ifelse(final_rose$benign > 0.5, "benign", "malignant")
> (cm_rose <- confusionMatrix(as.factor(final_rose$predict), test_data$classes))
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign       137         5
  malignant      0        67

               Accuracy : 0.9761
                 95% CI : (0.9451, 0.9922)
    No Information Rate : 0.6555
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9461

 Mcnemar's Test P-Value : 0.07364

            Sensitivity : 1.0000
            Specificity : 0.9306
         Pos Pred Value : 0.9648
         Neg Pred Value : 1.0000
             Prevalence : 0.6555
         Detection Rate : 0.6555
   Detection Prevalence : 0.6794
      Balanced Accuracy : 0.9653

       'Positive' Class : benign

> 
```

Random Forest sur données balancées avec SMOTe (Synthetic Minority Over-sampling Technique)

Toujours avec le package **caret**, exécutons un random forest sur les données balancées avec la méthode SMOTe et construisons sa matrice de confusion:

```
> library("caret")
> library("ROSE")
> set.seed(42)
> ctrl <- trainControl(method = "repeatedcv",number = 10, repeats = 10,
+                       verboseIter = FALSE,sampling = "smote")
> model_rf_smote <- caret::train(classes ~ ., data = train_data,
+                                method = "rf", preProcess = c("scale", "center"),
+                                trControl = ctrl)
Loading required package: grid
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
> final_smote <- data.frame(actual = test_data$classes,
+                           predict(model_rf_smote, newdata = test_data, type = "prob"$
> final_smote$predict <- ifelse(final_smote$benign > 0.5, "benign", "malignant")
> (cm_smote <- confusionMatrix(as.factor(final_smote$predict), test_data$classes))
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign       134         0
  malignant      3        72

               Accuracy : 0.9856
                 95% CI : (0.9586, 0.997)
    No Information Rate : 0.6555
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9685

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9781
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9600
             Prevalence : 0.6555
         Detection Rate : 0.6411
   Detection Prevalence : 0.6411
      Balanced Accuracy : 0.9891

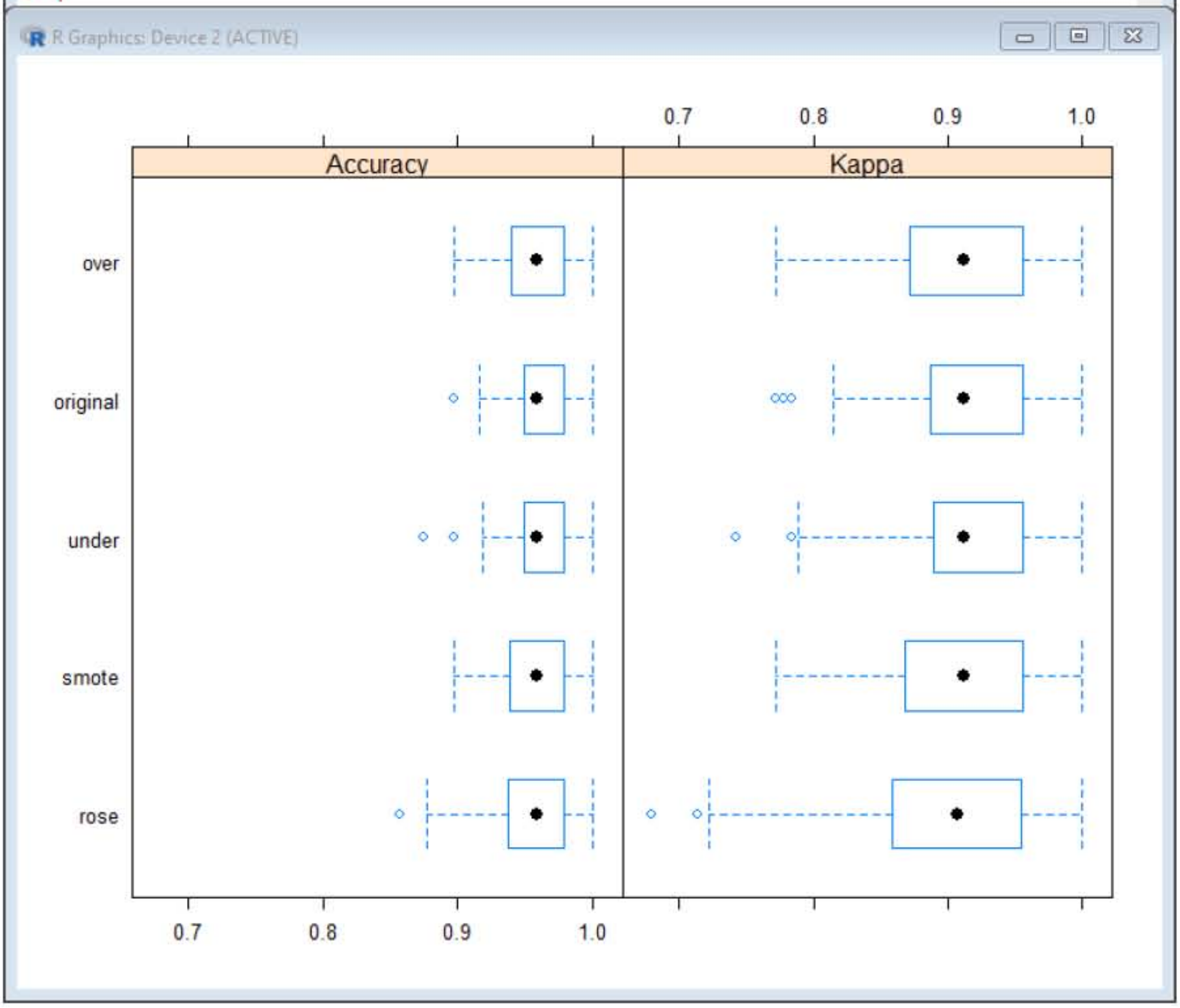
       'Positive' Class : benign

>
```

Comparaison des résultats

Maintenant comparons tous les résultats d'abord sous forme graphique:

```
> models <- list(original = model_rf,
+                        under = model_rf_under,
+                        over = model_rf_over,
+                        smote = model_rf_smote,
+                        rose = model_rf_rose)
>
> resampling <- resamples(models)
> bwplot(resampling)
>
```

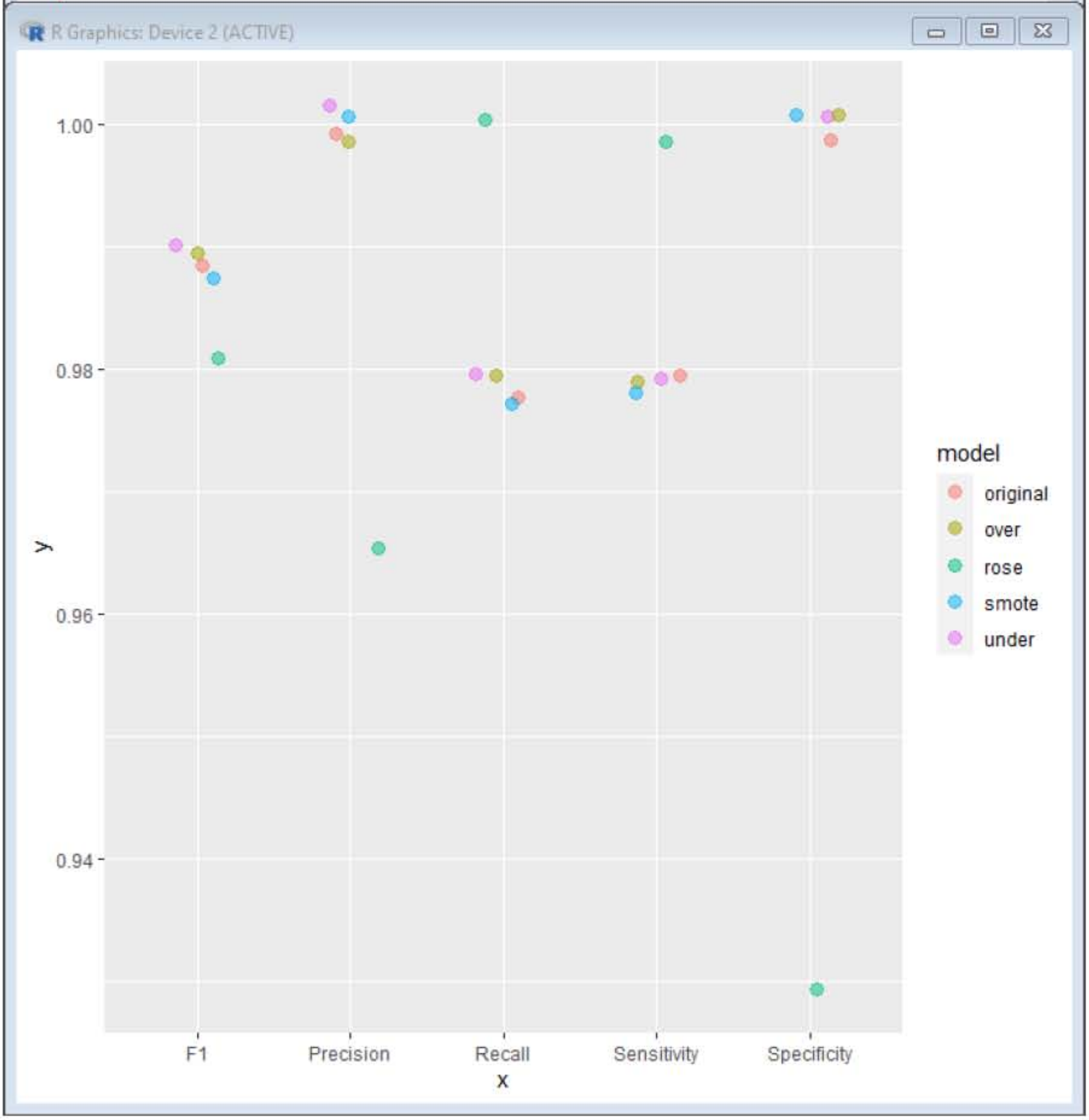


Ou une comparaison sous forme de tableau:

```
> (comparison <- data.frame(model = names(models),
+                           Sensitivity = rep(NA, length(models)),
+                           Specificity = rep(NA, length(models)),
+                           Precision = rep(NA, length(models)),
+                           Recall = rep(NA, length(models)),
+                           F1 = rep(NA, length(models))))
     model Sensitivity Specificity Precision Recall F1
1 original          NA          NA        NA     NA NA
2    under          NA          NA        NA     NA NA
3     over          NA          NA        NA     NA NA
4    smote          NA          NA        NA     NA NA
5     rose          NA          NA        NA     NA NA
>
> (comparison <- data.frame(model = names(models)))
     model
1 original
2    under
3     over
4    smote
5     rose
>
> for (name in names(models)) {
+ model <- get(paste0("cm_", name))
+
+ comparison[comparison$model == name, "Sensitivity"] <- model$byClass[["Sensitivity"]]
+ comparison[comparison$model == name, "Specificity"] <- model$byClass[["Specificity"]]
+ comparison[comparison$model == name, "Precision"] <- model$byClass[["Precision"]]
+ comparison[comparison$model == name, "Recall"] <- model$byClass[["Recall"]]
+ comparison[comparison$model == name, "F1"] <- model$byClass[["F1"]]
+ }
>
> comparison
     model Sensitivity Specificity Precision    Recall        F1
1 original   0.9781022   1.0000000 1.0000000 0.9781022 0.9889299
2    under   0.9781022   1.0000000 1.0000000 0.9781022 0.9889299
3     over   0.9781022   1.0000000 1.0000000 0.9781022 0.9889299
4    smote   0.9781022   1.0000000 1.0000000 0.9781022 0.9889299
5     rose   1.0000000   0.9305556 0.9647887 1.0000000 0.9820789
>
```

Ou en mettant ce tableau sous forme de graphique:

```
> library(tidyr)
> comparison %>%
+   gather(x, y, Sensitivity:F1) %>%
+   ggplot(aes(x = x, y = y, color = model)) +
+     geom_jitter(width = 0.2, alpha = 0.5, size = 3)
>
```



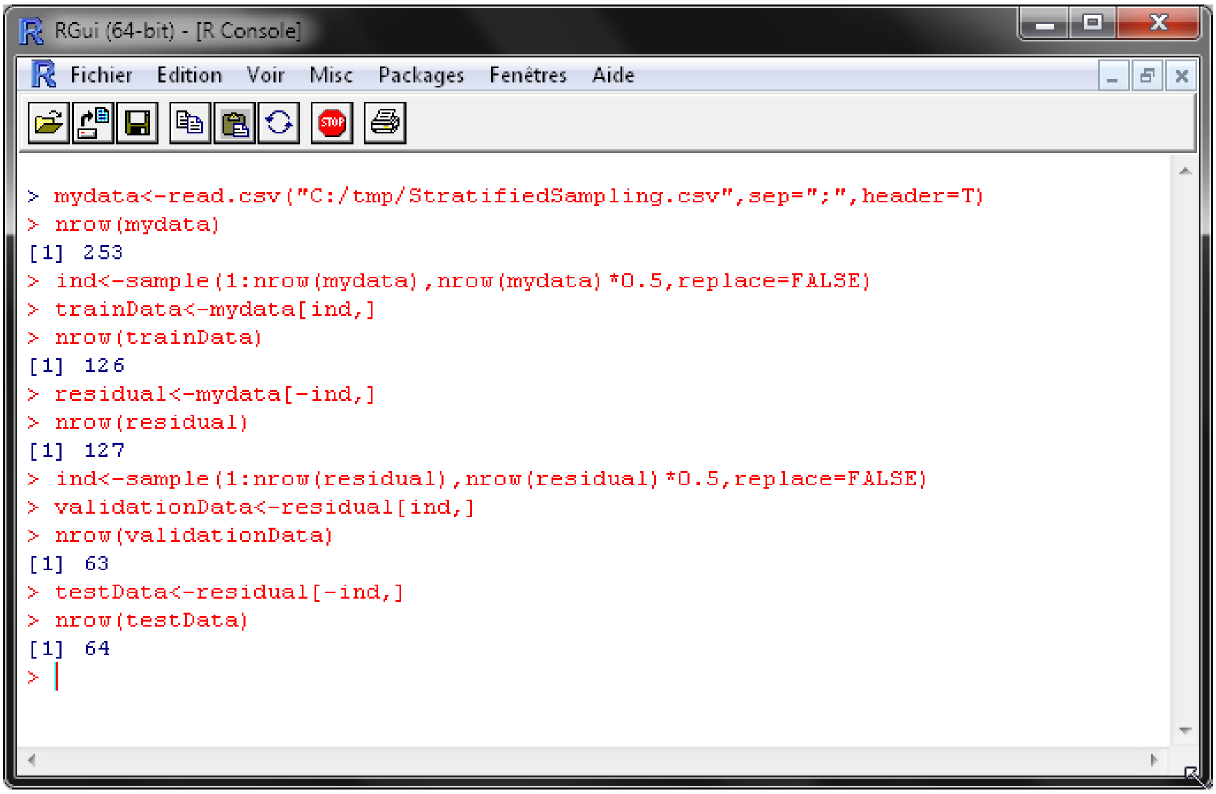
Exemple 335.: Partitionnement

R 3.0.2

La base élémentaire rigoureuse du data mining est de séparer le jeu de données en 3 ensembles qui sont dans l'ordre:

1. Entraînement (calibrage) des modèles pour déterminer les paramètres
2. Validation pour le choix du modèle qui performe le mieux ou pour affiner un modèle
3. Test du modèle choisi pour vérifier sa précision

Une manière simple de procéder et d'utiliser la fonction **sample( )** que nous avons déjà étudié:



Nous reviendrons plus loin sur des techniques plus élaborées lors de notre études des techniques exhaustives ou non exhaustives de cross-over à la page 2255.

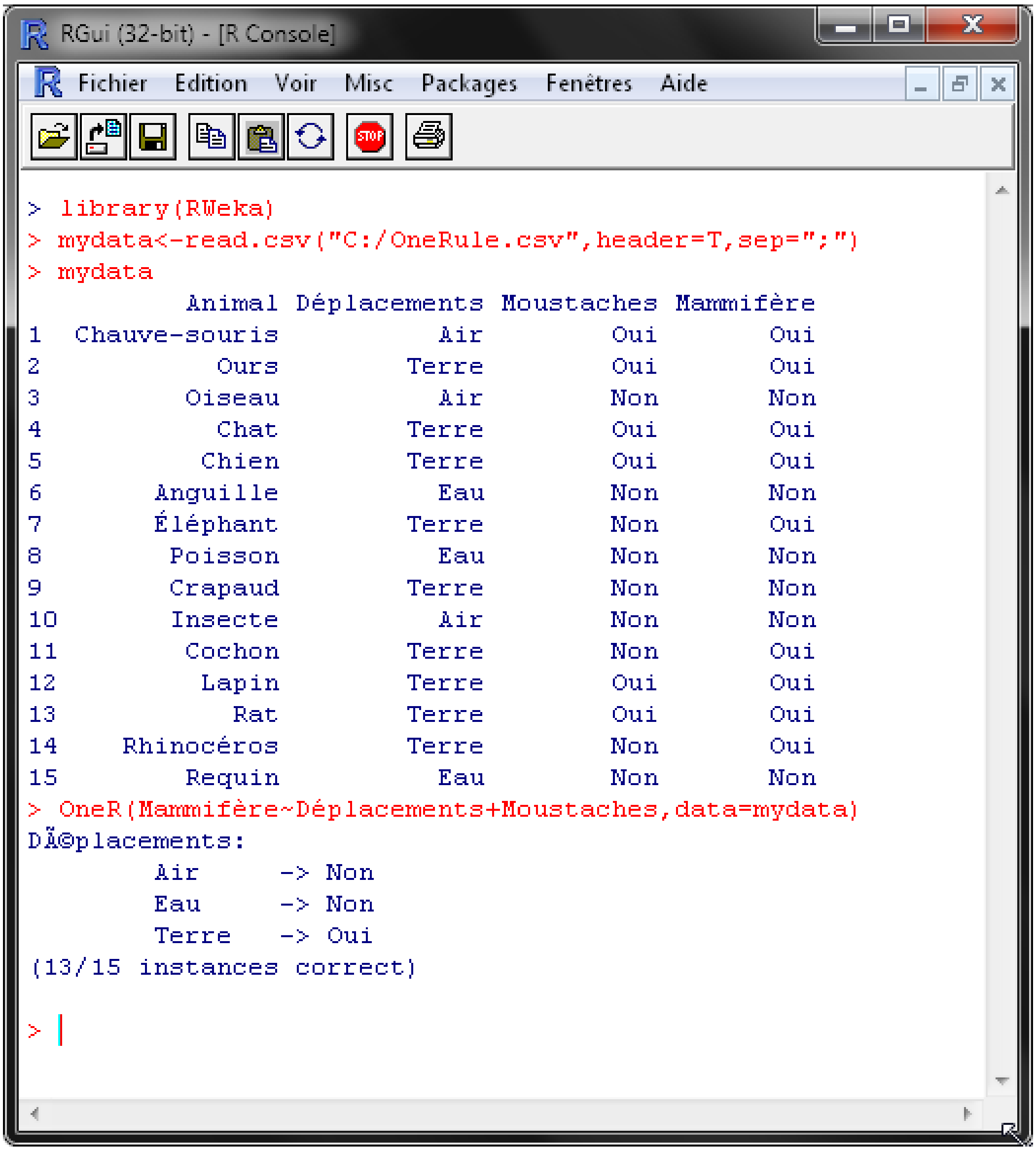
Exemple 336.: OneR (règle unique de classification)

R 3.0.2

Nous allons maintenant appliquer l'algorithme élémentaire *One Rule* rappelons que cet algorithme fonctionne avec des variables (qualitatives) nominales et choisit parmi l'ensemble des prédicteurs proposés seulement l'unique prédicteur qui en proportion explique le mieux la variable d'intérêt.

Attention l'exemple qui suit ne marche qu'avec la version 32 bits de R!!!

Reprenons l'exemple vu dans le cours théorique en utilisant la commande **OneR( )** du package **RWeka**:



Comme nous pouvons le constater (c'est tellement simple que même un enfant pourrait mettre cet algorithme dans n'importe quel tableur) entre *Déplacements* et *Moustaches*, l'unique prédicateur qui donne le meilleur résultat est *Déplacements*. La fonction nous aussi déjà les sorties potentielles pour un type de *Déplacements* donné.

Nous pouvons avoir un résumé de la performance du modèle aussi:

```
> modeleOneR<-OneR(Mammifère~Déplacements+Moustaches,data=mydata)
> summary(modeleOneR)

=== Summary ===

Correctly Classified Instances          13               86.6667 %
Incorrectly Classified Instances         2               13.3333 %
Kappa statistic                          0.7222
Mean absolute error                      0.1333
Root mean squared error                  0.3651
Relative absolute error                 27.6423 %
Root relative squared error             74.5141 %
Total Number of Instances               15

=== Confusion Matrix ===

 a b   <-- classified as
 5 1 | a = Non
 1 8 | b = Oui
> 
```

Bref c'est sympa et ludique mais tellement trivial que nous n'irons pas plus loin.

Exemple 337.: A priori

R 3.4.1

L'algorithme d'A priori est assez simple mais assez gourmand algorithmiquement parlant. Voici un exemple d'application:

```
> library("arules");options(digits=3)
> mydata<-read.csv("C:/tmp/Apriori.csv",header=T,sep=";");mydata
   Volé Couleur   Type    Origine
1   Oui   Rouge Sports Domestique
2   Non   Rouge Sports Domestique
3   Oui   Rouge Sports Domestique
4   Non   Jaune Sports Domestique
5   Oui   Jaune Sports    Importé
6   Non   Jaune    SUV    Importé
7   Oui   Jaune    SUV    Importé
8   Non   Jaune    SUV Domestique
9   Non   Rouge    SUV    Importé
10  Oui   Rouge Sports    Importé
11  Oui   Rouge    SUV Domestique
> rules<-apriori(mydata)
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 1

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[8 item(s), 11 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(rules)
    lhs                                          rhs                     support confidence lift
[1] {Couleur=Jaune,Origine=Domestique}        => {Volé=Non}              0.182   1          2.20
[2] {Volé=Non,Origine=Importé}                => {Type=SUV}              0.182   1          2.20
[3] {Volé=Non,Type=Sports}                    => {Origine=Domestique}    0.182   1          1.83
[4] {Volé=Oui,Couleur=Jaune}                  => {Origine=Importé}       0.182   1          2.20
[5] {Type=Sports,Origine=Importé}             => {Volé=Oui}              0.182   1          1.83
[6] {Volé=Oui,Origine=Domestique}             => {Couleur=Rouge}         0.273   1          1.83
[7] {Volé=Oui,Type=Sports,Origine=Domestique} => {Couleur=Rouge}         0.182   1          1.83
>
```

Exemple 338.: ECLAT (Equivalence CLAss Transformation)

R 3.4.1

L'algorithme d'A priori est assez simple mais assez gourmand algorithmiquement parlant. Nous avons mentionné dans le cours théorique que l'algorithme ECLAT est considéré comme plus rapide et plus efficace sur les petits jeux de données.

Voici un exemple d'application, toujours avec le package **arules** et le même je de données qu'avant mais avec la fonction **eclat( )**:

```
> library("arules")
> mydata<-read.csv("c:/tmp/Apriori.csv",header=T,sep=";")
> mydata
   Volé Couleur   Type    Origine
1   Oui   Rouge Sports Domestique
2   Non   Rouge Sports Domestique
3   Oui   Rouge Sports Domestique
4   Non   Jaune Sports Domestique
5   Oui   Jaune Sports    Importé
6   Non   Jaune    SUV    Importé
7   Oui   Jaune    SUV    Importé
8   Non   Jaune    SUV Domestique
9   Non   Rouge    SUV    Importé
10  Oui   Rouge Sports    Importé
11  Oui   Rouge    SUV Domestique
> rules<-eclat(mydata)
Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE     0.1      1     10 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 1

Warning in eclat(mydata) :
  You chose a very low absolute support count of 1. You might run out of memory! Increase minimum support.

create itemset ...
set transactions ...[8 item(s), 11 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating bit matrix ... [8 row(s), 11 column(s)] done [0.00s].
writing  ... [44 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].
> 
```

On peut afficher un résumé de l'analyse:

```
> summary(rules)
set of 44 itemsets

most frequent items:
          Volé=Oui        Type=Sports Origine=Domestique           Volé=Non
                13                 13                 13                 11
     Couleur=Jaune            (Other)
                11                 32

element (itemset/transaction) length distribution:sizes
 1  2  3  4
 8 24 11  1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   2.114   3.000   4.000

summary of quality measures:
    support           count
 Min.   :0.1818   Min.   :2.000
 1st Qu.:0.1818   1st Qu.:2.000
 Median :0.2727   Median :3.000
 Mean   :0.2831   Mean   :3.114
 3rd Qu.:0.3636   3rd Qu.:4.000
 Max.   :0.5455   Max.   :6.000

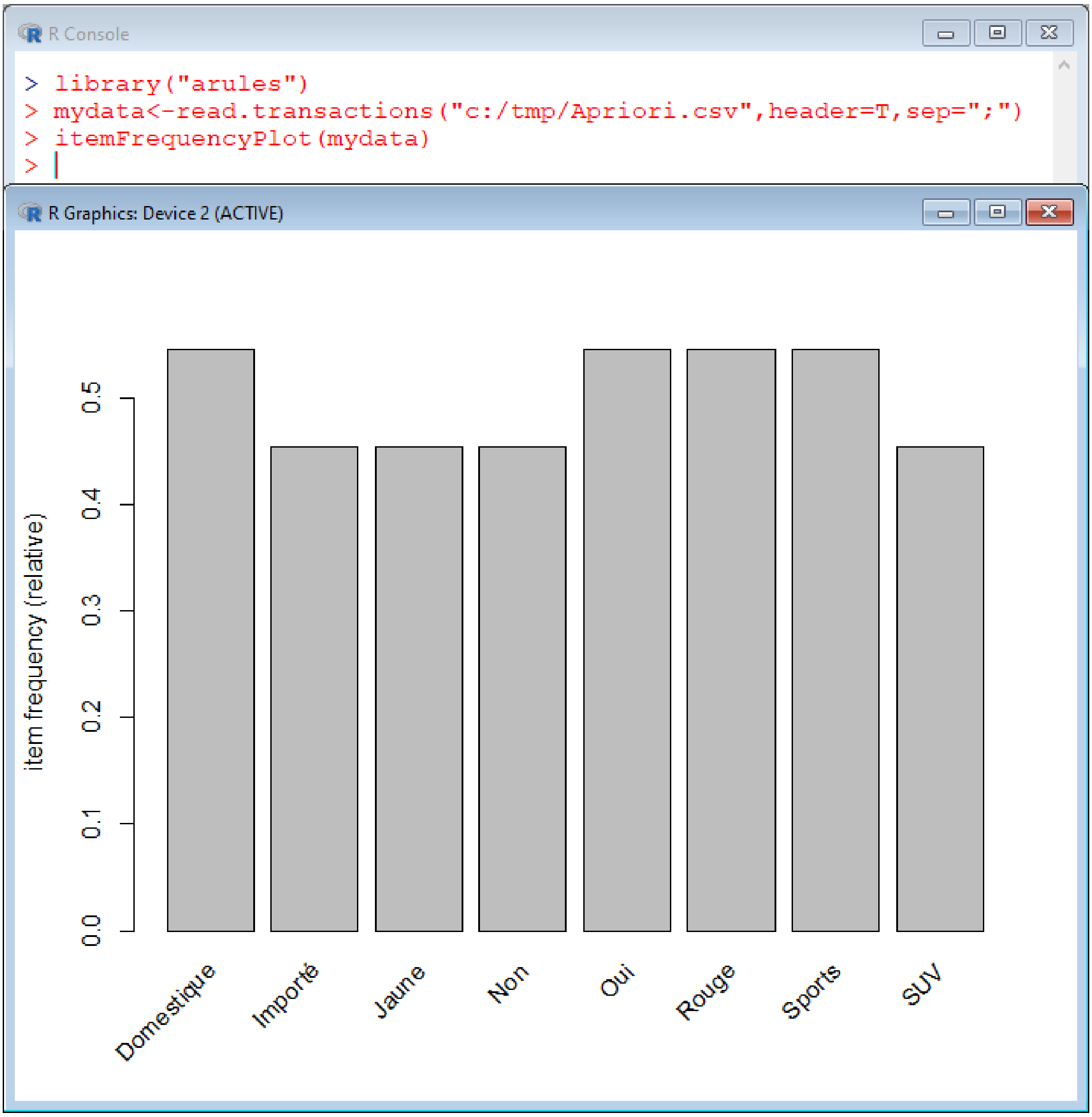
includes transaction ID lists: FALSE

mining info:
   data ntransactions support
 mydata            11     0.1
>
```

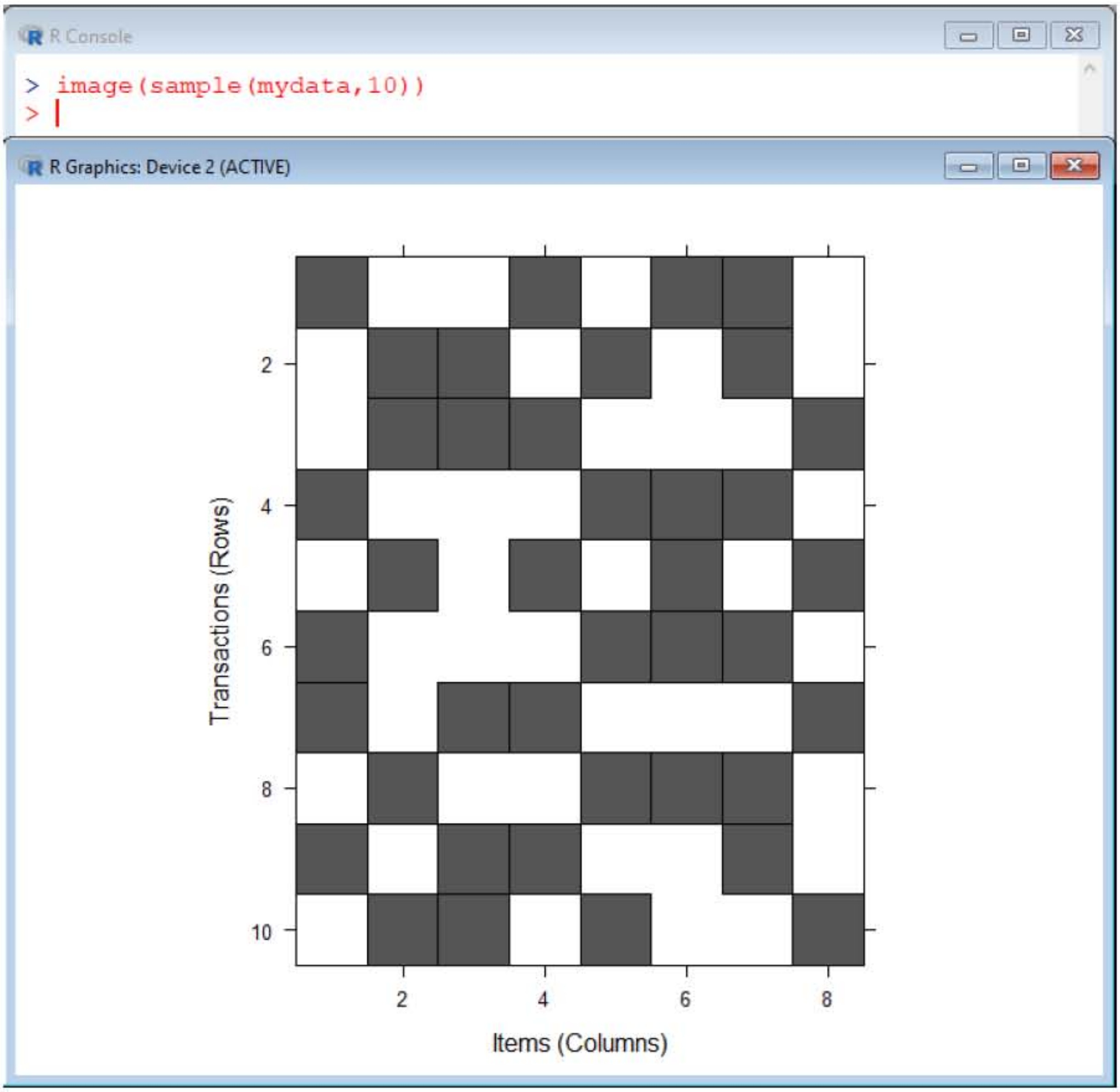
Et afficher l'inspection des règles comme nous l'avons fait avant avec l'algorithme A priori:

```
> inspect(sort(rules, by = "support"))
     items                                                support   count
[1]  {Volé=Oui}                                           0.5454545 6
[2]  {Couleur=Rouge}                                      0.5454545 6
[3]  {Type=Sports}                                        0.5454545 6
[4]  {Origine=Domestique}                                 0.5454545 6
[5]  {Volé=Non}                                           0.4545455 5
[6]  {Couleur=Jaune}                                      0.4545455 5
[7]  {Origine=Importé}                                    0.4545455 5
[8]  {Type=SUV}                                           0.4545455 5
[9]  {Couleur=Rouge,Origine=Domestique}                   0.3636364 4
[10] {Type=Sports,Origine=Domestique}                     0.3636364 4
[11] {Volé=Oui,Type=Sports}                               0.3636364 4
[12] {Couleur=Rouge,Type=Sports}                          0.3636364 4
[13] {Volé=Oui,Couleur=Rouge}                             0.3636364 4
[14] {Volé=Non,Type=SUV}                                  0.2727273 3
[15] {Couleur=Jaune,Type=SUV}                             0.2727273 3
[16] {Type=SUV,Origine=Importé}                           0.2727273 3
[17] {Volé=Oui,Origine=Importé}                           0.2727273 3
[18] {Couleur=Jaune,Origine=Importé}                      0.2727273 3
[19] {Volé=Non,Couleur=Jaune}                             0.2727273 3
[20] {Volé=Non,Origine=Domestique}                        0.2727273 3
[21] {Couleur=Rouge,Type=Sports,Origine=Domestique}       0.2727273 3
[22] {Volé=Oui,Couleur=Rouge,Origine=Domestique}          0.2727273 3
[23] {Volé=Oui,Origine=Domestique}                        0.2727273 3
[24] {Volé=Oui,Couleur=Rouge,Type=Sports}                 0.2727273 3
[25] {Volé=Non,Type=SUV,Origine=Importé}                  0.1818182 2
[26] {Couleur=Jaune,Type=SUV,Origine=Importé}             0.1818182 2
[27] {Volé=Non,Couleur=Jaune,Type=SUV}                    0.1818182 2
[28] {Volé=Oui,Type=SUV}                                  0.1818182 2
[29] {Couleur=Rouge,Type=SUV}                             0.1818182 2
[30] {Type=SUV,Origine=Domestique}                        0.1818182 2
[31] {Volé=Oui,Couleur=Jaune,Origine=Importé}             0.1818182 2
[32] {Volé=Oui,Type=Sports,Origine=Importé}               0.1818182 2
[33] {Couleur=Rouge,Origine=Importé}                      0.1818182 2
[34] {Type=Sports,Origine=Importé}                        0.1818182 2
[35] {Volé=Non,Origine=Importé}                           0.1818182 2
[36] {Volé=Non,Couleur=Jaune,Origine=Domestique}          0.1818182 2
[37] {Volé=Oui,Couleur=Jaune}                             0.1818182 2
[38] {Couleur=Jaune,Type=Sports}                          0.1818182 2
[39] {Couleur=Jaune,Origine=Domestique}                   0.1818182 2
[40] {Volé=Non,Type=Sports,Origine=Domestique}            0.1818182 2
[41] {Volé=Non,Couleur=Rouge}                             0.1818182 2
[42] {Volé=Non,Type=Sports}                               0.1818182 2
[43] {Volé=Oui,Couleur=Rouge,Type=Sports,Origine=Domestique} 0.1818182 2
[44] {Volé=Oui,Type=Sports,Origine=Domestique}            0.1818182 2
> 
```

On peut aussi rapidement faire un plot de toutes les fréquences relatives à l'aide de la fonction **itemFrequencyPlot( )**:



ainsi que la rareté des données:

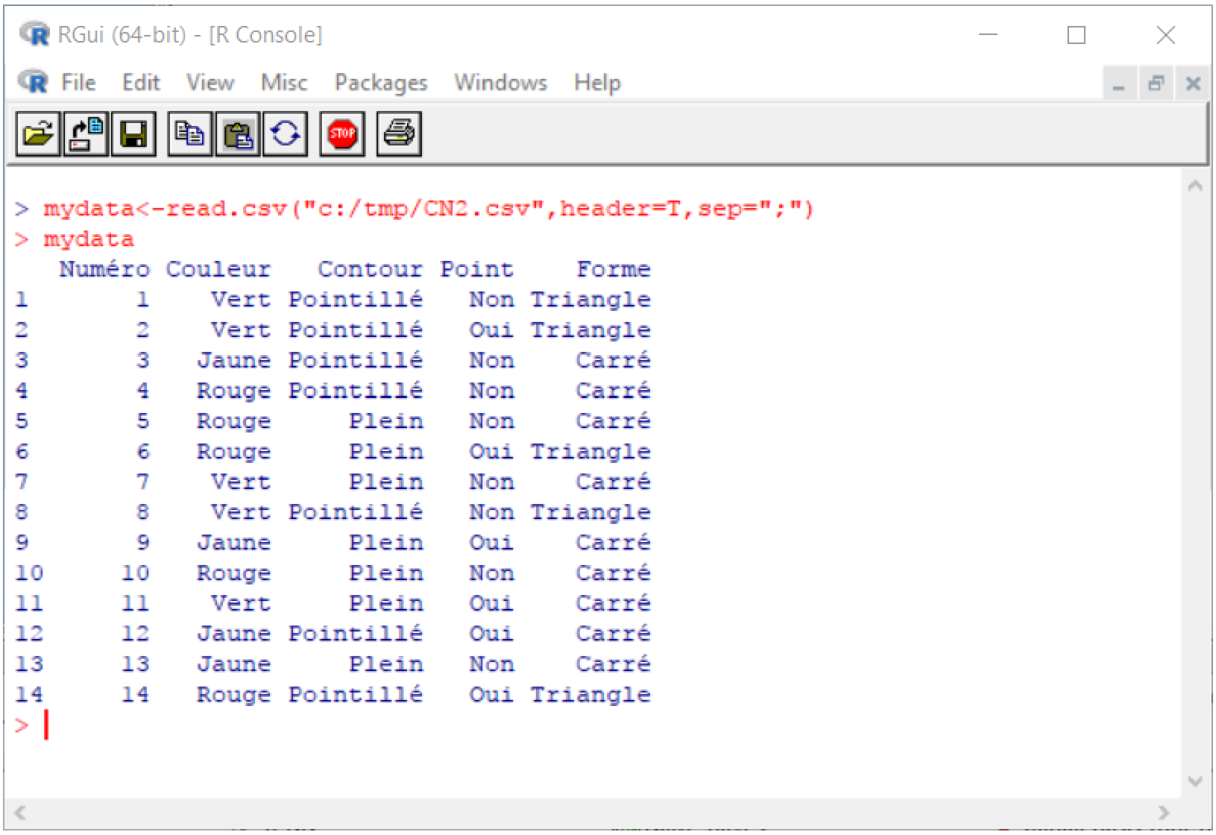

R Console
> image(sample(mydata,10))
>
R Graphics: Device 2 (ACTIVE)
Transactions (Rows)
Items (Columns)
2
4
6
8
10

Exemple 339.: Classification par induction CN2

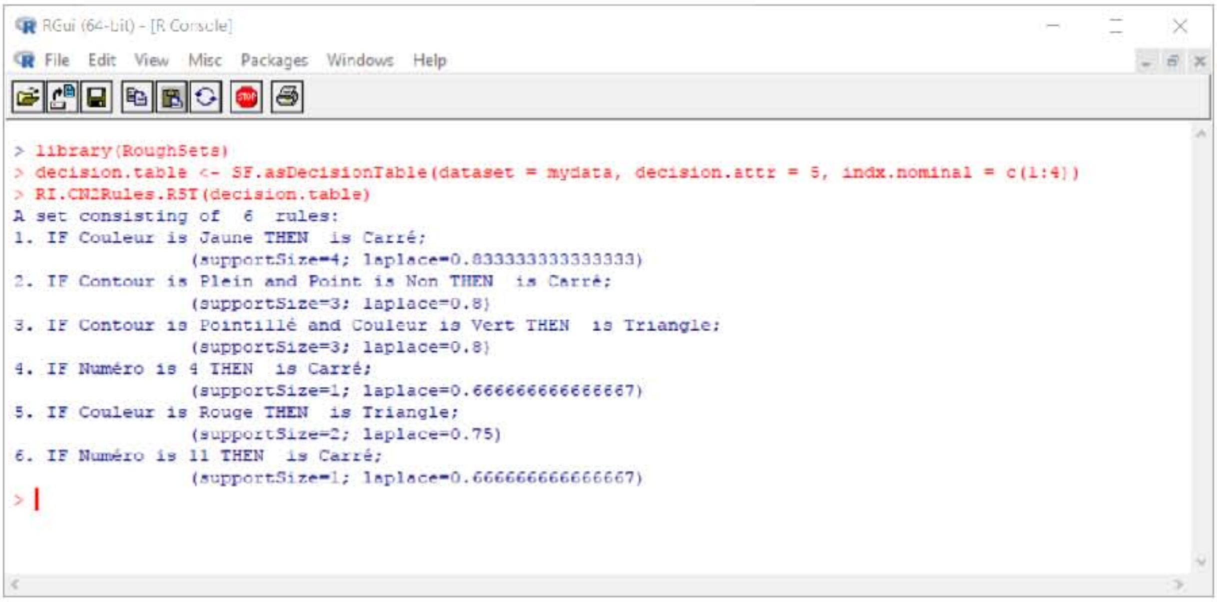
R 3.4.1

Dans les règles de classification par induction, celle qui suit immédiatement la OneR (unique règle basée sur la fréquence de l'attribut cible pour rappel) est la classification par induction avec attributs et règles logiques non hiérarchiques.

Voyons cela avec le jeu de données suivant:



Ensuite, nous utilisons les fonctions **SF.asDecisionTable( )** et **RI.CN2Rules.RST( )** du package **RoughSets:**

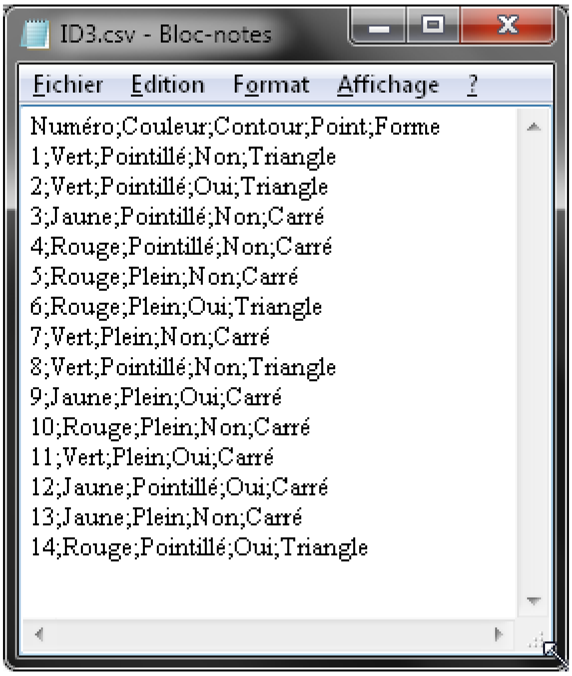

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(RoughSets)
> decision.table <- SF.asDecisionTable(dataset = mydata, decision.attr = 5, indx.nominal = c(1:4))
> RI.CN2Rules.RST(decision.table)
A set consisting of 6 rules:
1. IF Couleur is Jaune THEN is Carré;
(supportSize=4; laplace=0.833333333333333)
2. IF Contour is Plein and Point is Non THEN is Carré;
(supportSize=3; laplace=0.8)
3. IF Contour is Pointillé and Couleur is Vert THEN is Triangle;
(supportSize=3; laplace=0.8)
4. IF Numéro is 4 THEN is Carré;
(supportSize=1; laplace=0.666666666666667)
5. IF Couleur is Rouge THEN is Triangle;
(supportSize=2; laplace=0.75)
6. IF Numéro is 11 THEN is Carré;
(supportSize=1; laplace=0.666666666666667)
>

Exemple 340.: Classification par induction ID-3 et PRISM

R 3.1.2 **i386**

Nous allons ici vérifier la technique de clustering ID-3 et PRISM que nous avons étudié dans le cours théorique de Méthodes Numériques et calculé à la main.

Nous allons donc travailler avec le fichier suivant et donc avec les mêmes données que dans le cours théorique:


ID3.csv - Bloc-notes

Fichier Edition Format Affichage ?

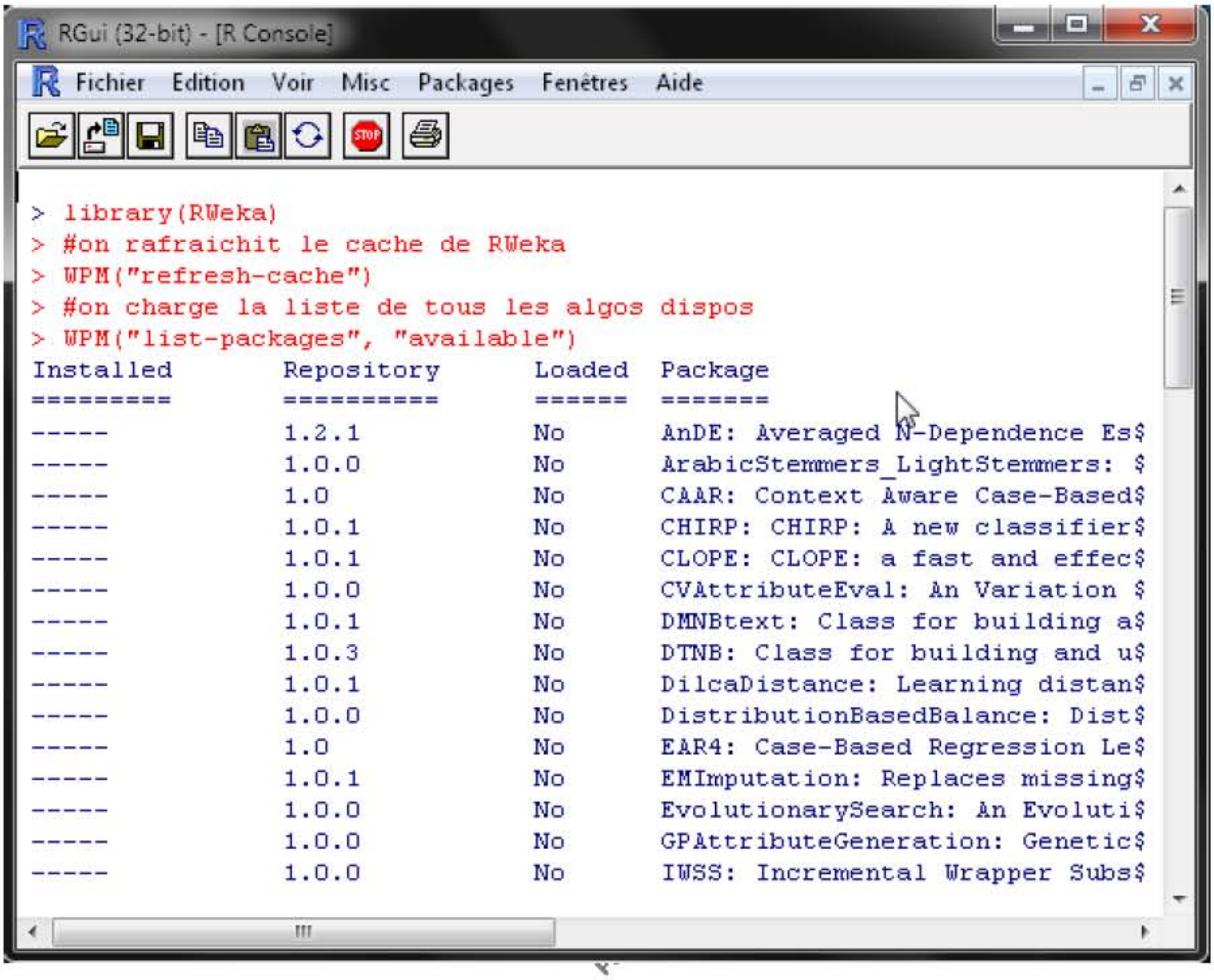
```
Numéro;Couleur;Contour;Point;Forme
1;Vert;Pointillé;Non;Triangle
2;Vert;Pointillé;Oui;Triangle
3;Jaune;Pointillé;Non;Carré
4;Rouge;Pointillé;Non;Carré
5;Rouge;Plein;Non;Carré
6;Rouge;Plein;Oui;Triangle
7;Vert;Plein;Non;Carré
8;Vert;Pointillé;Non;Triangle
9;Jaune;Plein;Oui;Carré
10;Rouge;Plein;Non;Carré
11;Vert;Plein;Oui;Carré
12;Jaune;Pointillé;Oui;Carré
13;Jaune;Plein;Non;Carré
14;Rouge;Pointillé;Oui;Triangle
```

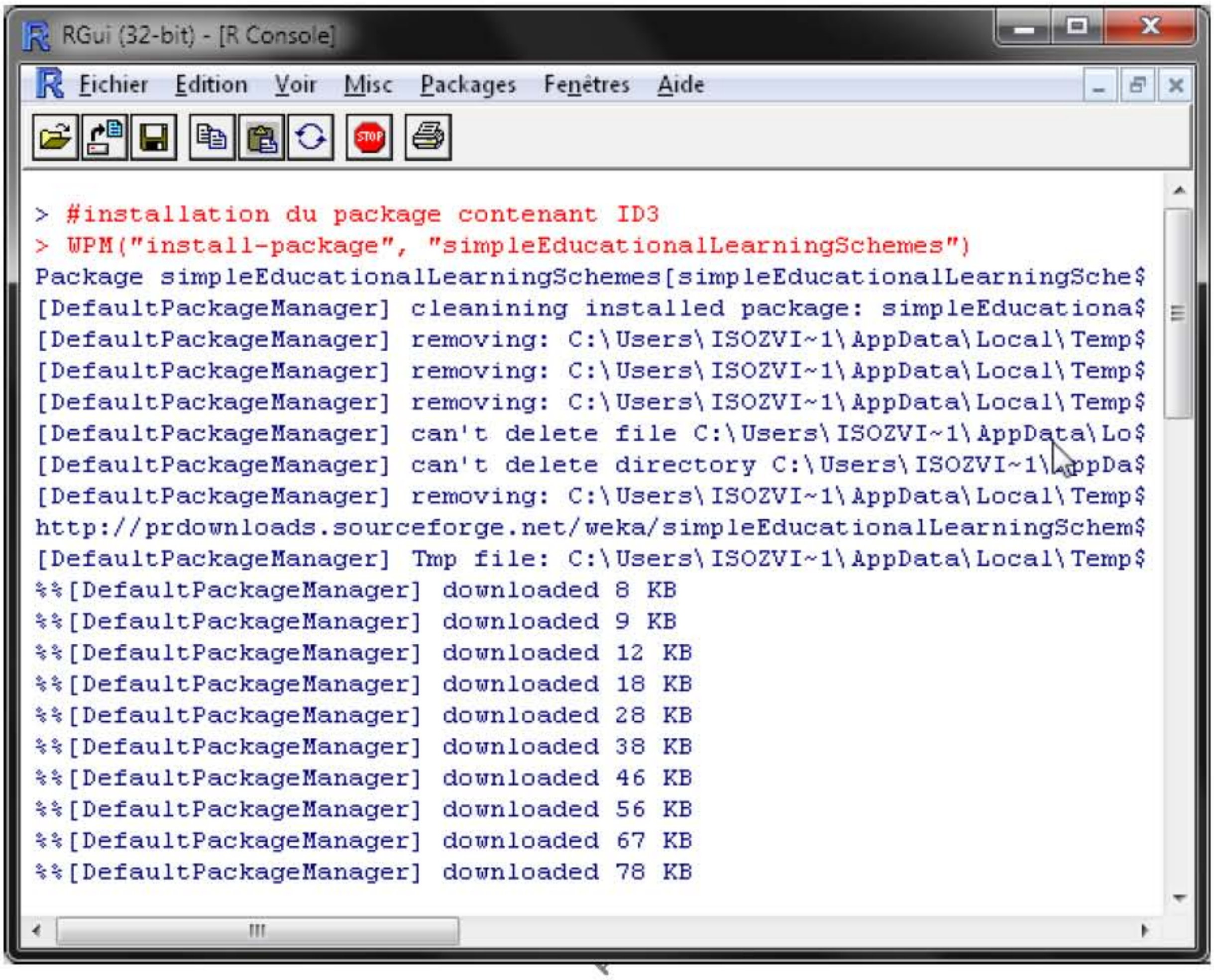
et là au jour où j'écris ces lignes il faut savoir que:

1. Je n'ai pas trouvé de méthodes autres que d'utiliser le package **RWeka**…

2. Cela ne marche que sur la version 32 bits de R

Les étapes sont directement commentées dans les captures d'écran:


RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(RWeka)
> #on rafraichit le cache de RWeka
> WPM("refresh-cache")
> #on charge la liste de tous les algos dispos
> WPM("list-packages", "available")
Installed Repository Loaded Package
========= ========== ====== =======
----- 1.2.1 No AnDE: Averaged N-Dependence Es$
----- 1.0.0 No ArabicStemmers_LightStemmers: $
----- 1.0 No CAAR: Context Aware Case-Based$
----- 1.0.1 No CHIRP: CHIRP: A new classifier$
----- 1.0.1 No CLOPE: CLOPE: a fast and effec$
----- 1.0.0 No CVAttributeEval: An Variation $
----- 1.0.1 No DMNBtext: Class for building a$
----- 1.0.3 No DTNB: Class for building and u$
----- 1.0.1 No DilcaDistance: Learning distan$
----- 1.0.0 No DistributionBasedBalance: Dist$
----- 1.0 No EAR4: Case-Based Regression Le$
----- 1.0.1 No EMImputation: Replaces missing$
----- 1.0.0 No EvolutionarySearch: An Evoluti$
----- 1.0.0 No GPAttributeGeneration: Genetic$
----- 1.0.0 No IWSS: Incremental Wrapper Subs$


RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> #installation du package contenant ID3
> WPM("install-package", "simpleEducationalLearningSchemes")
Package simpleEducationalLearningSchemes[simpleEducationalLearningSche$
[DefaultPackageManager] cleanining installed package: simpleEducationa$
[DefaultPackageManager] removing: C:\Users\ISOZVI~1\AppData\Local\Temp$
[DefaultPackageManager] removing: C:\Users\ISOZVI~1\AppData\Local\Temp$
[DefaultPackageManager] removing: C:\Users\ISOZVI~1\AppData\Local\Temp$
[DefaultPackageManager] can't delete file C:\Users\ISOZVI~1\AppData\Lo$
[DefaultPackageManager] can't delete directory C:\Users\ISOZVI~1\AppDa$
[DefaultPackageManager] removing: C:\Users\ISOZVI~1\AppData\Local\Temp$
http://prdownloads.sourceforge.net/weka/simpleEducationalLearningSchem$
[DefaultPackageManager] Tmp file: C:\Users\ISOZVI~1\AppData\Local\Temp$
%%[DefaultPackageManager] downloaded 8 KB
%%[DefaultPackageManager] downloaded 9 KB
%%[DefaultPackageManager] downloaded 12 KB
%%[DefaultPackageManager] downloaded 18 KB
%%[DefaultPackageManager] downloaded 28 KB
%%[DefaultPackageManager] downloaded 38 KB
%%[DefaultPackageManager] downloaded 46 KB
%%[DefaultPackageManager] downloaded 56 KB
%%[DefaultPackageManager] downloaded 67 KB
%%[DefaultPackageManager] downloaded 78 KB

```
> #chargement du package
> WPM("load-package", "simpleEducationalLearningSchemes")
> #préparation du classificateur
> ID3 <- make_Weka_classifier("weka/classifiers/trees/Id3")
> #et on l'utilise...
> mydata<-read.csv("C:/tmp/ID3.csv",header=T,sep=";")
> #on enlève la première colonne qui dérange et qui ne sert de toute f$
> #à rien
> mydata<-mydata[,-1]
> ID3(mydata$Forme~., data = mydata)
Id3


Couleur = Jaune: CarrÃ©
Couleur = Rouge
|  Point = Non: CarrÃ©
|  Point = Oui: Triangle
Couleur = Vert
|  Contour = Plein: CarrÃ©
|  Contour = PointillÃ©: Triangle
>
```

Nous n'avons donc pas a priori le niveau de finisse de paramétrage de Tanagra mais bon…

À comparer avec Tanagra (les résultats sont les mêmes):

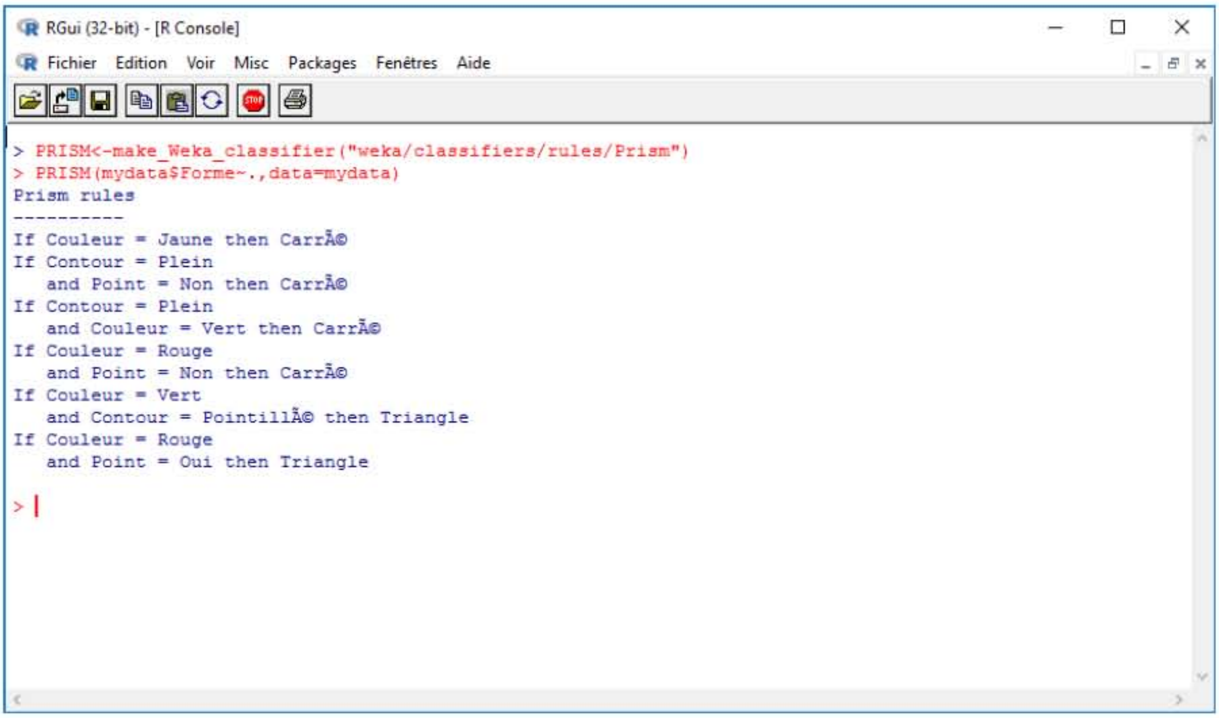
Decision tree

- Couleur in [Vert]
  - Contour in [Pointillé] then Forme = **Triangle** (100,00 % of 3 examples)
  - Contour in [Plein] then Forme = **Carré** (100,00 % of 2 examples)
- Couleur in [Jaune] then Forme = **Carré** (100,00 % of 4 examples)
- Couleur in [Rouge]
  - Point in [Non] then Forme = **Carré** (100,00 % of 3 examples)
  - Point in [Oui] then Forme = **Triangle** (100,00 % of 2 examples)

et cela est conforme aux résultats obtenus dans le cours théorique.

Dommage (au même titre que pour Tanagra) qu'il n'y ait pas de diagramme cependant... cela aiderait à la compréhension.

Et pour PRISM il s'agit d'utiliser:


RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> PRISM<-make_Weka_classifier("weka/classifiers/rules/Prism")
> PRISM(mydata$Forme~.,data=mydata)
Prism rules

If Couleur = Jaune then CarrÃ©
If Contour = Plein
and Point = Non then CarrÃ©
If Contour = Plein
and Couleur = Vert then CarrÃ©
If Couleur = Rouge
and Point = Non then CarrÃ©
If Couleur = Vert
and Contour = PointillÃ© then Triangle
If Couleur = Rouge
and Point = Oui then Triangle
>

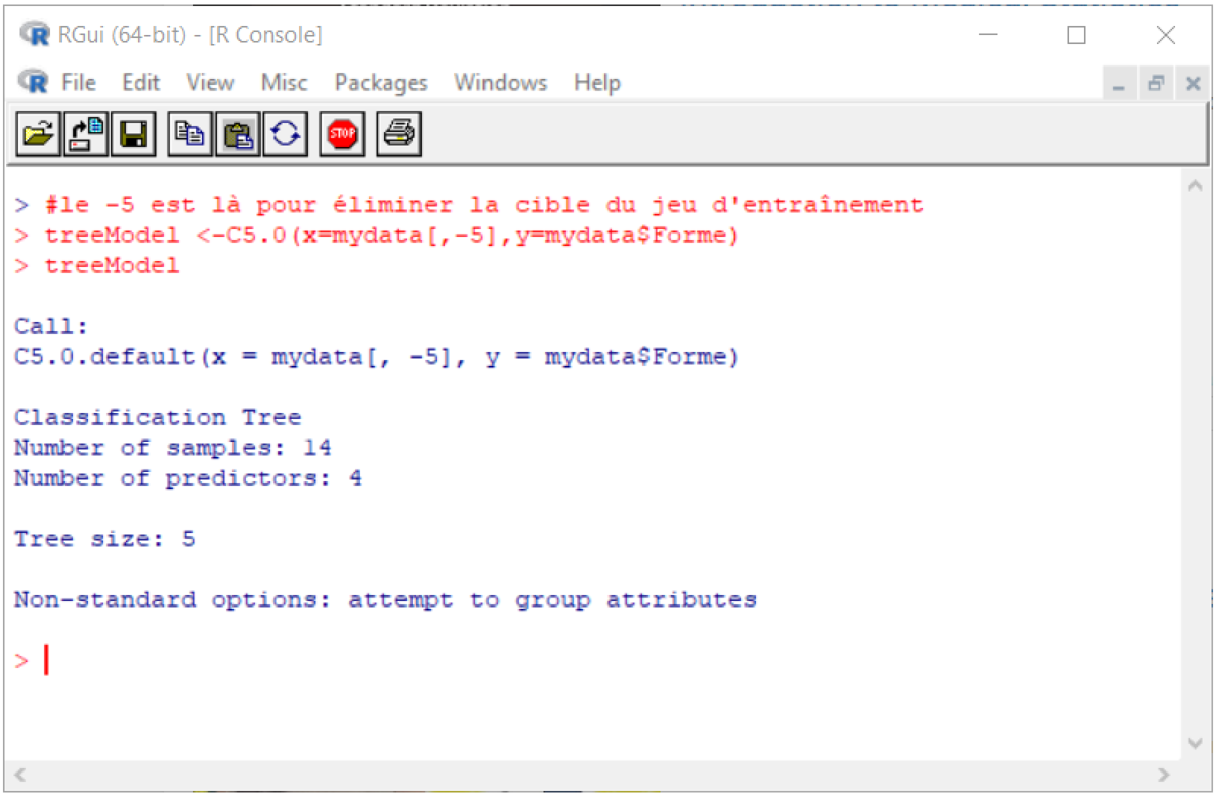
Exemple 341.: Classification par induction C5

R 3.4.1

Nous avons parlé de cet algorithme dans le cours théorique. Nous allons voir à quel niveau il donne des résultats similaires ou différents de ID3 vu juste précédemment. Pour cela nous chargeons le package **C50** et le même jeu de données qu'avant:

```
> library(C50)
> mydata<-read.csv("c:/tmp/C5.csv",header=T,sep=";")
> mydata
   Numéro Couleur   Contour Point    Forme
1       1    Vert Pointillé   Non Triangle
2       2    Vert Pointillé   Oui Triangle
3       3   Jaune Pointillé   Non    Carré
4       4   Rouge Pointillé   Non    Carré
5       5   Rouge     Plein   Non    Carré
6       6   Rouge     Plein   Oui Triangle
7       7    Vert     Plein   Non    Carré
8       8    Vert Pointillé   Non Triangle
9       9   Jaune     Plein   Oui    Carré
10     10   Rouge     Plein   Non    Carré
11     11    Vert     Plein   Oui    Carré
12     12   Jaune Pointillé   Oui    Carré
13     13   Jaune     Plein   Non    Carré
14     14   Rouge Pointillé   Oui Triangle
>
```

Nous entraînons l'algorithme avec la fonction **C5.0( )** :



Nous affichons le résumé (à comparer avec le résultat de l'algorithme ID3):

```
> summary(treeModel)

Call:
C5.0.default(x = mydata[, -5], y = mydata$Forme)


C5.0 [Release 2.07 GPL Edition]         Tue Jul 18 14:39:40 2017
-------------------------------

Class specified by attribute `outcome'

Read 14 cases (5 attributes) from undefined.data

Decision tree:

Couleur = Jaune: Carré (4)
Couleur = Rouge:
:...Point = Non: Carré (3)
:   Point = Oui: Triangle (2)
Couleur = Vert:
:...Contour = Plein: Carré (2)
    Contour = Pointillé: Triangle (3)


Evaluation on training data (14 cases):

            Decision Tree
          ----------------
          Size      Errors

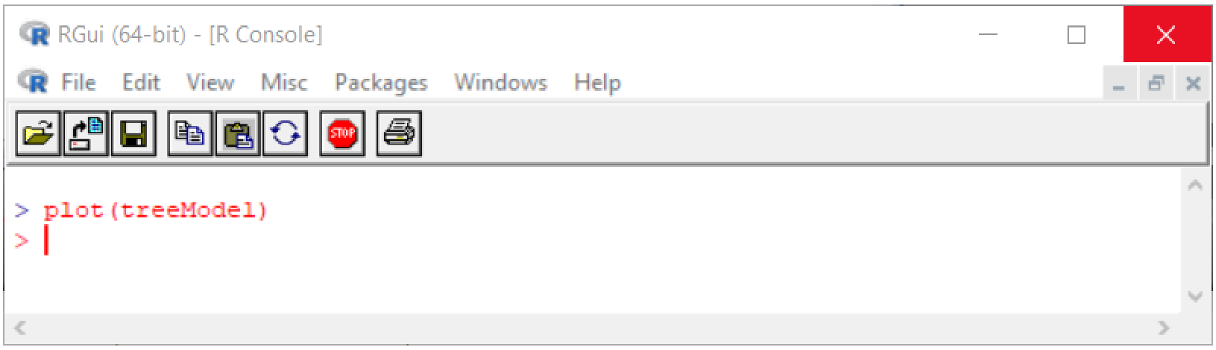
             5    0( 0.0%)   <<


           (a)   (b)    <-classified as
          ----  ----
             9          (a): class Carré
                   5    (b): class Triangle


        Attribute usage:

        100.00% Couleur
         35.71% Contour
         35.71% Point
```

et nous pouvons aussi faire un graphique:



Ce qui donne:

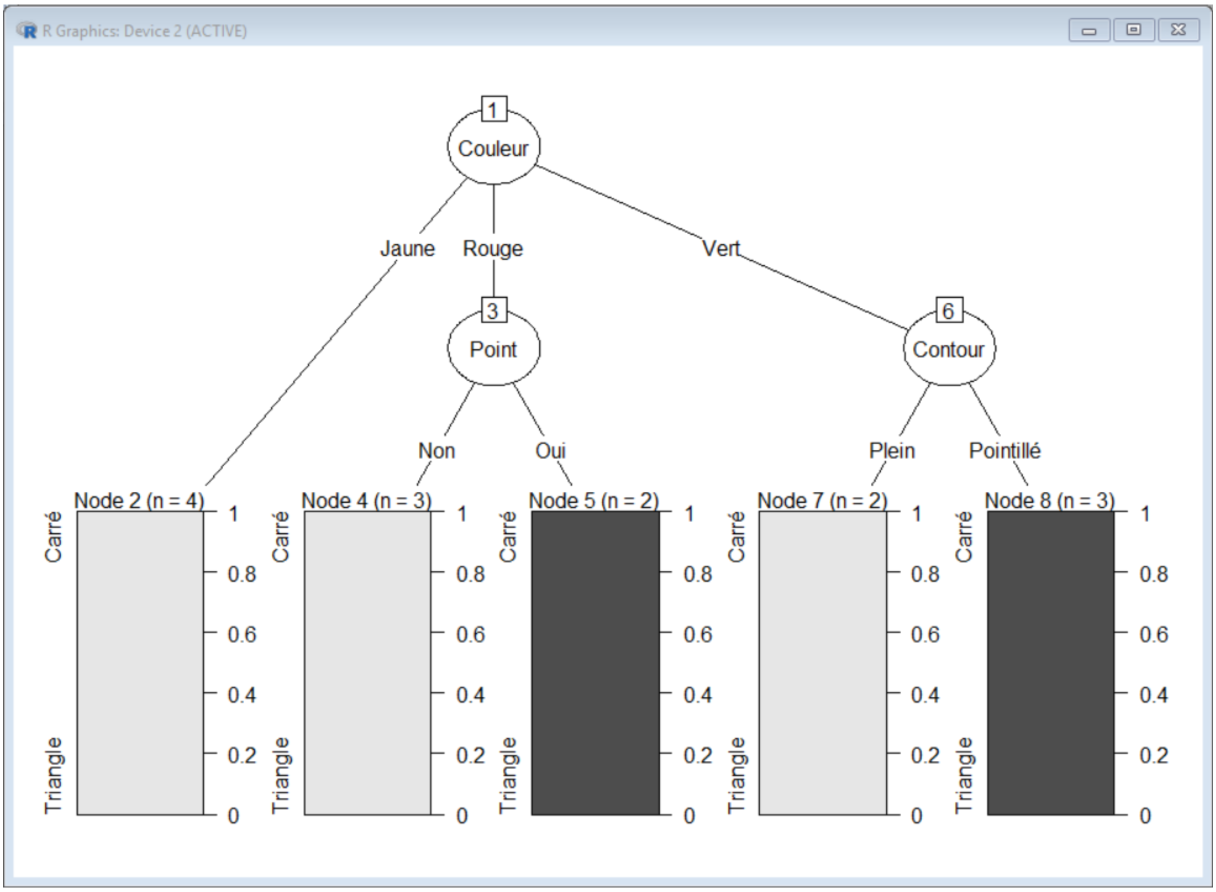


Figure 297 Graph de classification par induction C5

Exemple 342.: Classification Ascendante Hiérarchique (CAH/Dendrogramme)

R 3.0.2

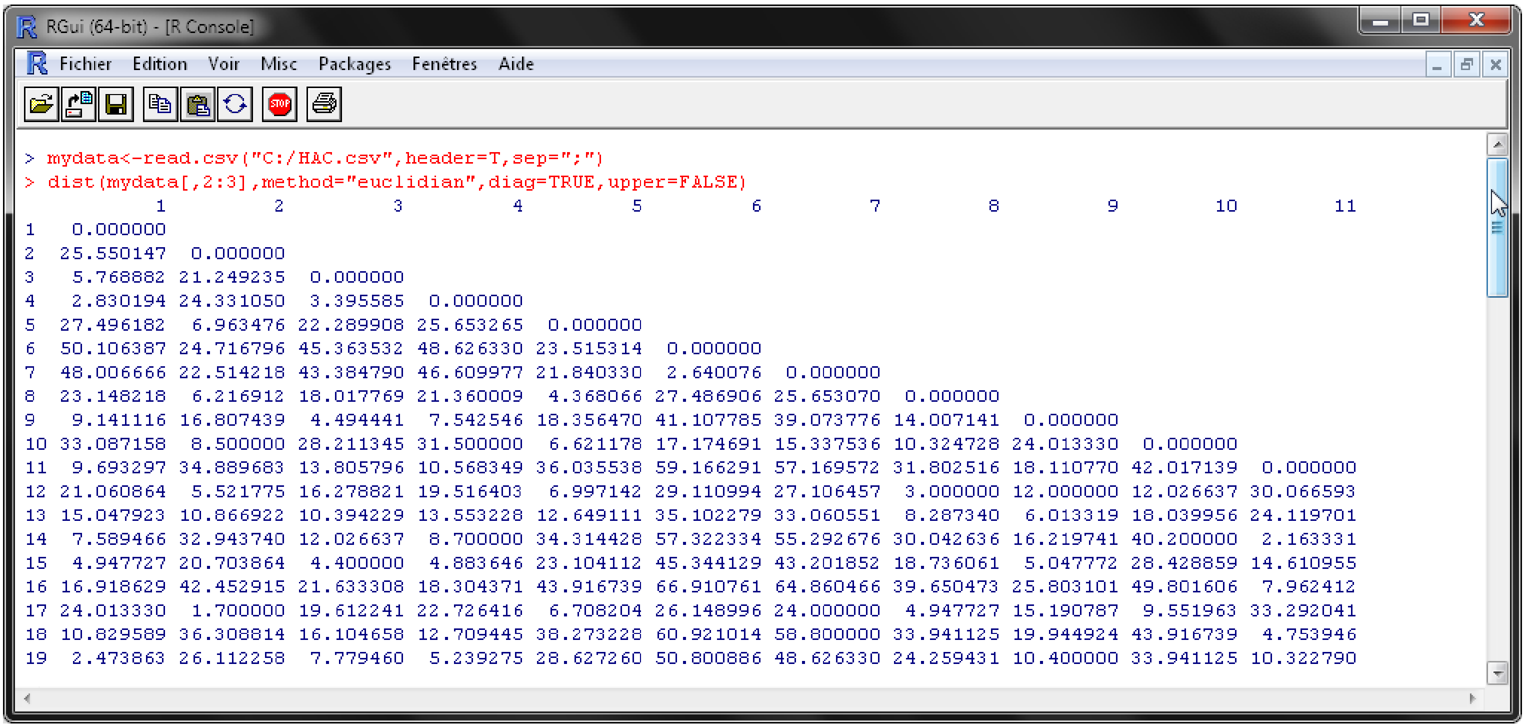
Nous allons donc vérifier ici si nous obtenons le même dendrogramme que dans le cours théorique (où nous nous étions follement amusé à calculer à la main....) et par la même occasion nous pourrons vérifier si Minitab utiliser avec l'OBS une agglomération naïve ou non. Nous utilisons donc les mêmes données que dans le cours théorique:

ArbreRegressionCART.csv - Bloc-...

Fichier Edition Format Affichage ?

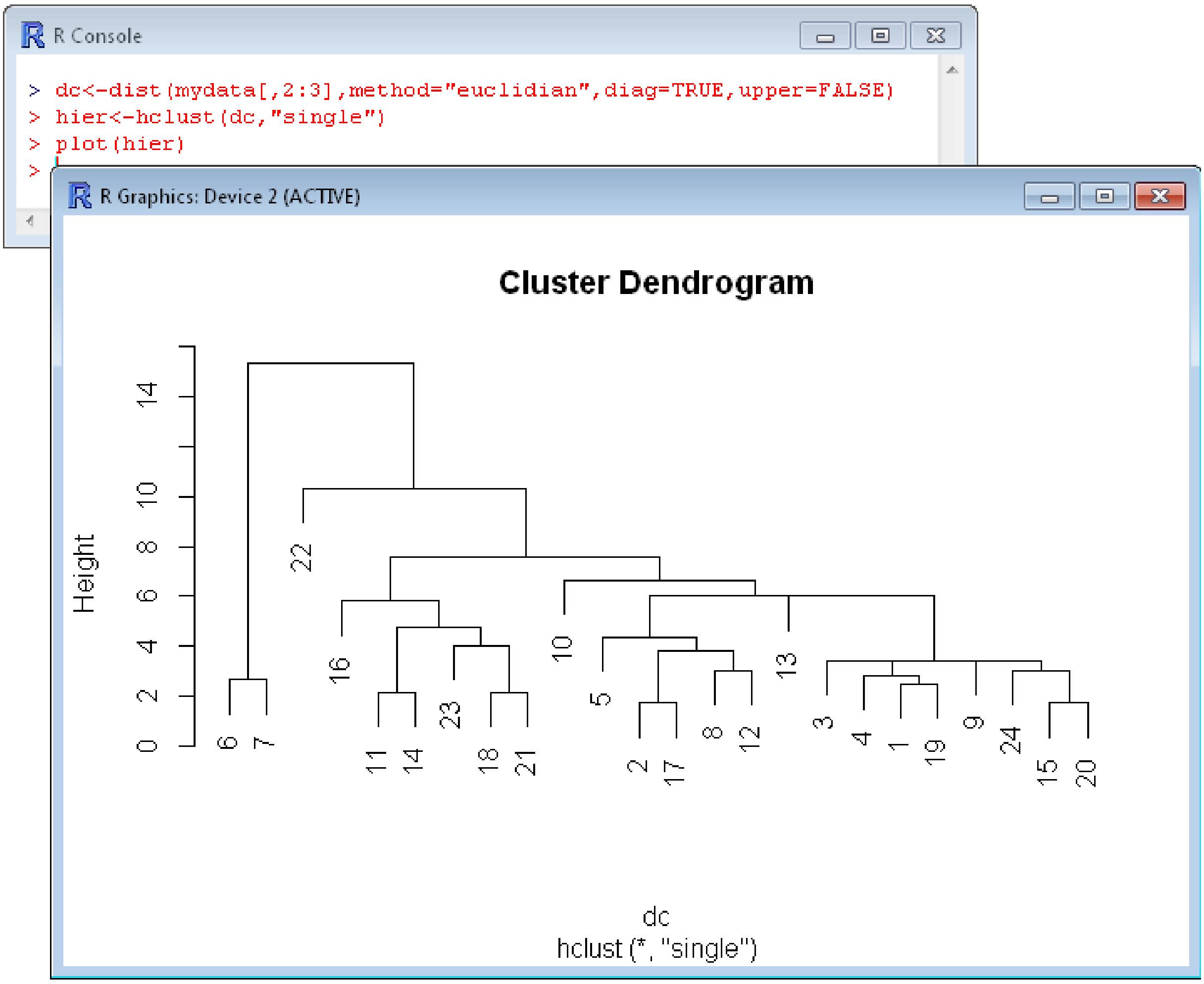
```
Revenus;Surface;Propriétaire
60;18.4;1
85.5;16.8;1
64.8;21.6;1
61.5;20.8;1
87;23.6;1
110.1;19.2;1
108;17.6;1
82.8;22.4;1
69;20;1
93;20.8;1
51;22;1
81;20;1
75;19.6;2
52.8;20.8;2
64.8;17.2;2
43.2;20.4;2
84;17.6;2
49.2;17.6;2
59.4;16;2
66;18.4;2
47.4;16.4;2
33;18.8;2
51;14;2
63;14.8;2
```

Nous pouvons d'abord avec la commande native **dist( )** afficher la matrice symétrique des distances euclidiennes:



```
> mydata<-read.csv("C:/HAC.csv",header=T,sep=";")
> dist(mydata[,2:3],method="euclidian",diag=TRUE,upper=FALSE)
           1          2          3          4          5          6          7          8          9         10         11
1   0.000000
2  25.550147  0.000000
3   5.768882 21.249235  0.000000
4   2.830194 24.331050  3.395585  0.000000
5  27.496182  6.963476 22.289908 25.653265  0.000000
6  50.106387 24.716796 45.363532 48.626330 23.515314  0.000000
7  48.006666 22.514218 43.384790 46.609977 21.840330  2.640076  0.000000
8  23.148218  6.216912 18.017769 21.360009  4.368066 27.486906 25.653070  0.000000
9   9.141116 16.807439  4.494441  7.542546 18.356470 41.107785 39.073776 14.007141  0.000000
10 33.087158  8.500000 28.211345 31.500000  6.621178 17.174691 15.337536 10.324728 24.013330  0.000000
11  9.693297 34.889683 13.805796 10.568349 36.035538 59.166291 57.169572 31.802516 18.110770 42.017139  0.000000
12 21.060864  5.521775 16.278821 19.516403  6.997142 29.110994 27.106457  3.000000 12.000000 12.026637 30.066593
13 15.047923 10.866922 10.394229 13.553228 12.649111 35.102279 33.060551  8.287340  6.013319 18.039956 24.119701
14  7.589466 32.943740 12.026637  8.700000 34.314428 57.322334 55.292676 30.042636 16.219741 40.200000  2.163331
15  4.947727 20.703864  4.400000  4.883646 23.104112 45.344129 43.201852 18.736061  5.047772 28.428859 14.610955
16 16.918629 42.452915 21.633308 18.304371 43.916739 66.910761 64.860466 39.650473 25.803101 49.801606  7.962412
17 24.013330  1.700000 19.612241 22.726416  6.708204 26.148996 24.000000  4.947727 15.190787  9.551963 33.292041
18 10.829589 36.308814 16.104658 12.709445 38.273228 60.921014 58.800000 33.941125 19.944924 43.916739  4.753946
19  2.473863 26.112258  7.779460  5.239275 28.627260 50.800886 48.626330 24.259431 10.400000 33.941125 10.322790
```

Bon nous n'y voyons pas toute la matrice (pas assez de place sur la capture d'écran) mais les valeurs sont bien les mêmes que celles calculées à la main. Vérifions maintenant que nous avons bien le même dendrogramme avec la commande **hclust( )** :



et tout est donc bien conforme :-)

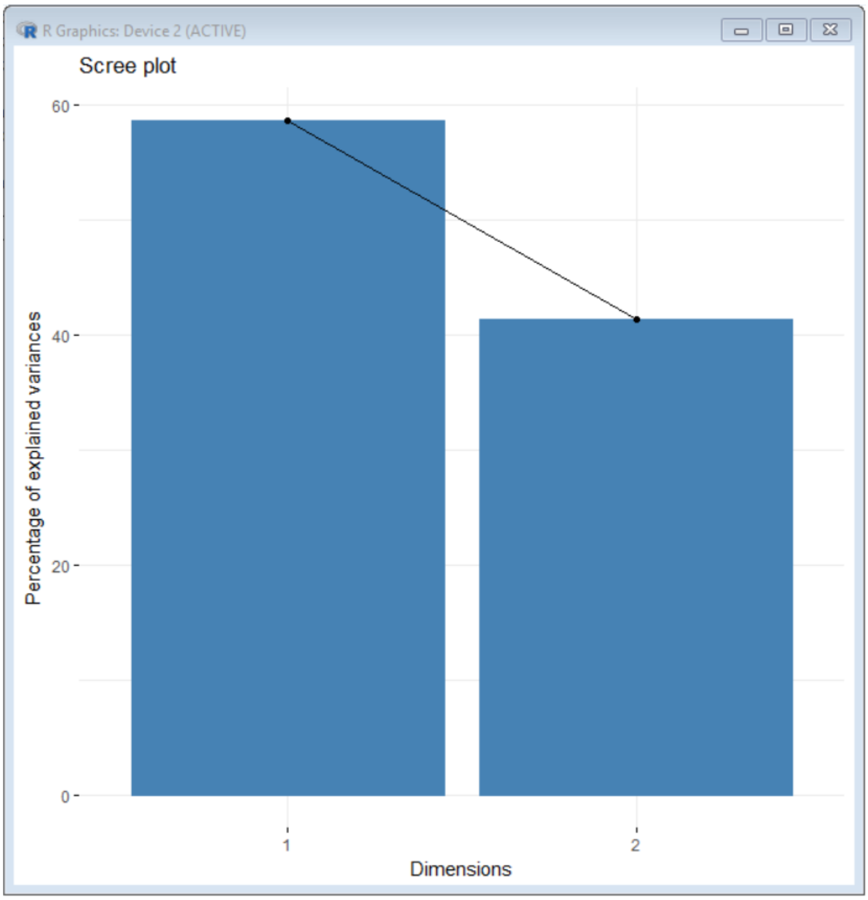
CAH avec ACP

Le but ici est de voir quand peut, comme mentionné dans le cours théorique, bien évidemment exécuter une ACP avant la CAH.

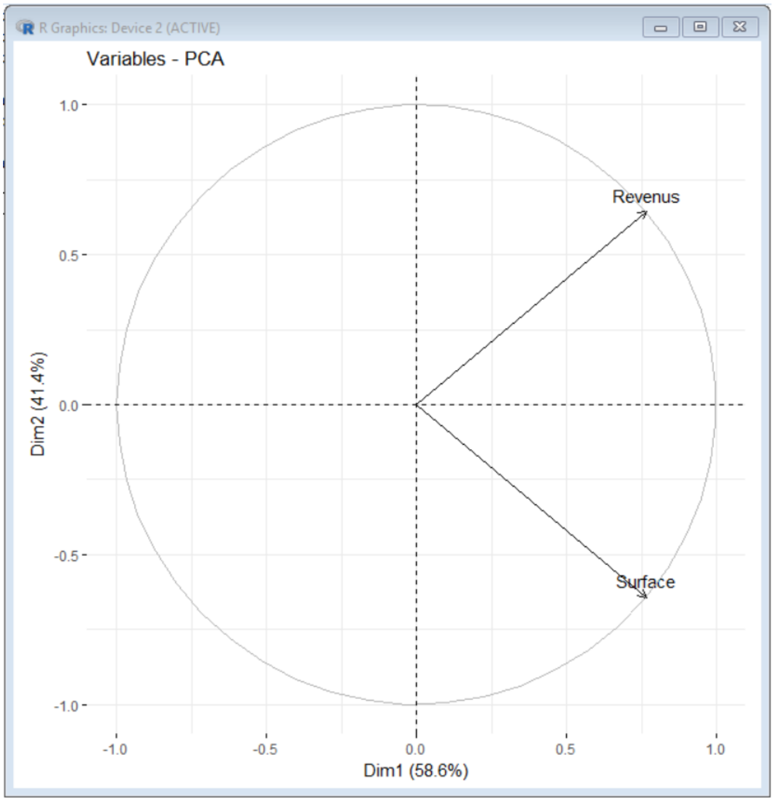
D'abord chargeons les données avec une petite ACP classique:

```
> mydata<-read.csv("c:/tmp/HAC.csv",header=T,sep=";")
> mydata<-mydata[,2:3]
> mydata
   Revenus Surface
1     60.0    18.4
2     85.5    16.8
3     64.8    21.6
4     61.5    20.8
5     87.0    23.6
6    110.1    19.2
7    108.0    17.6
8     82.8    22.4
9     69.0    20.0
10    93.0    20.8
11    51.0    22.0
12    81.0    20.0
13    75.0    19.6
14    52.8    20.8
15    64.8    17.2
16    43.2    20.4
17    84.0    17.6
18    49.2    17.6
19    59.4    16.0
20    66.0    18.4
21    47.4    16.4
22    33.0    18.8
23    51.0    14.0
24    63.0    14.8
>
> library(FactoMineR)
> res.pca <- PCA(mydata, ncp = 2, graph=F)
>
> library(factoextra)
> fviz_eig(res.pca)
> fviz_pca_var(res.pca)
> fviz_pca_ind(res.pca)
>
```

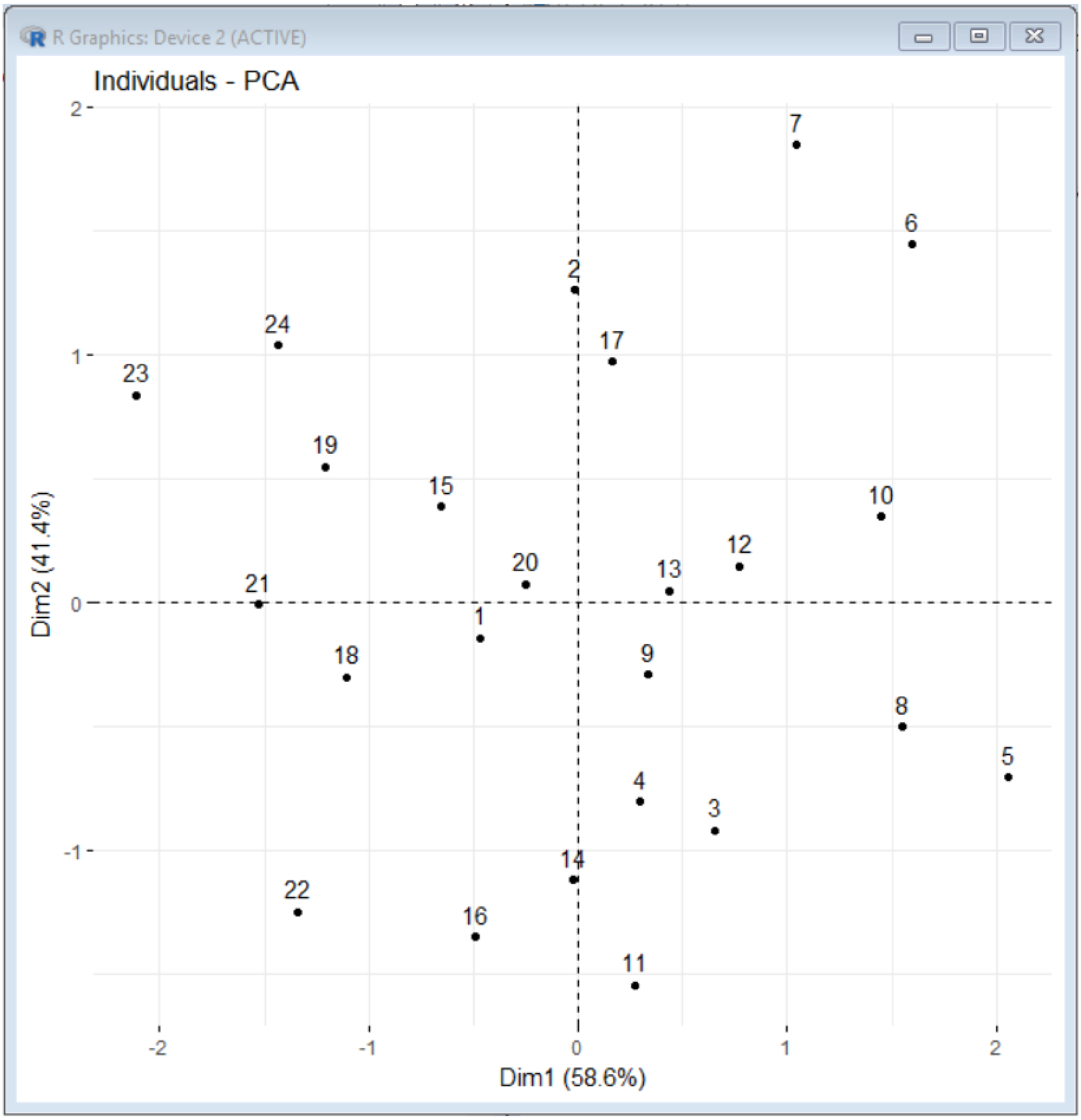
Ce qui donne pour la commande **fviz_eig( )** du package **factoextra**:



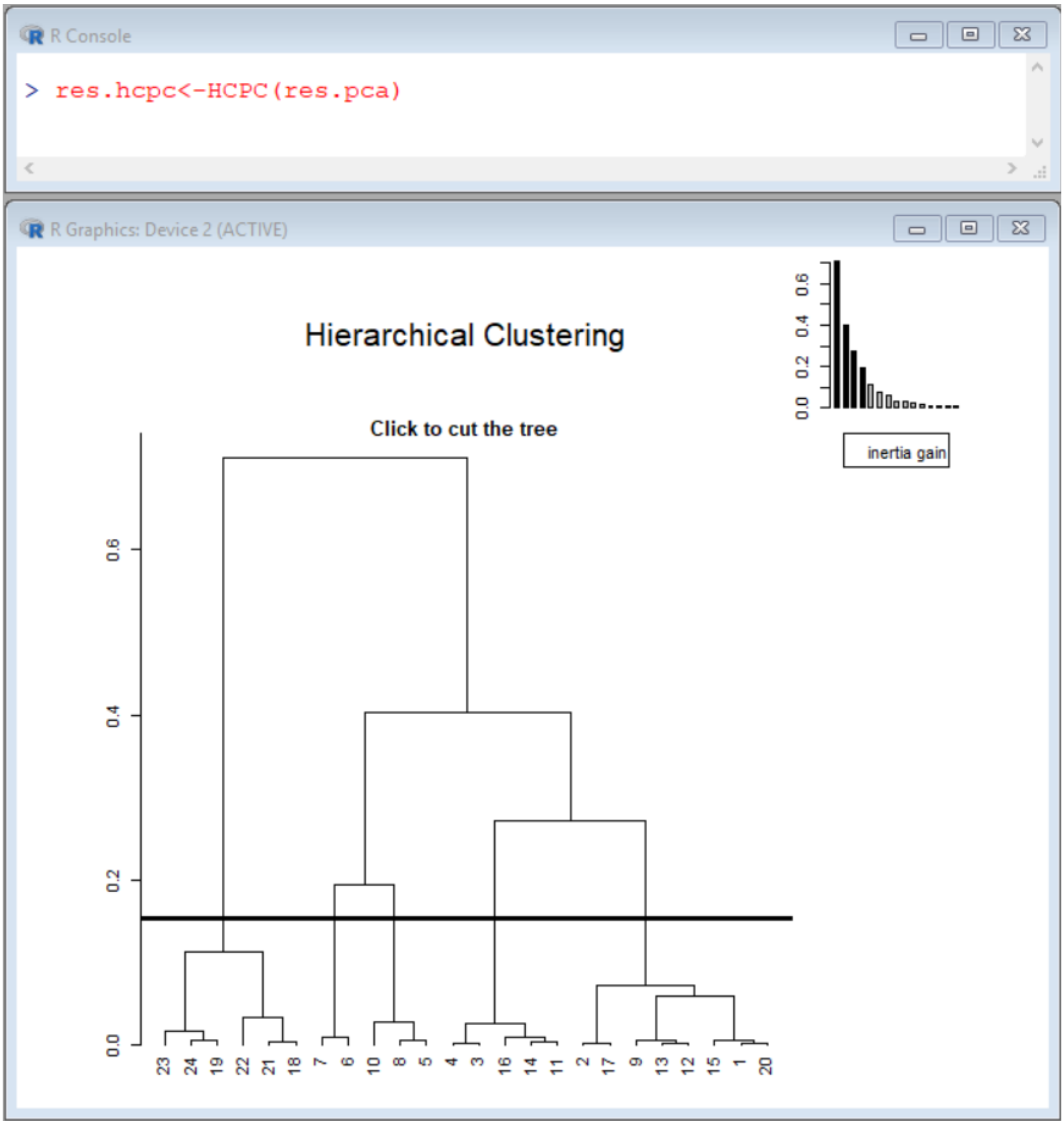
Et pour la commande **fviz_pca_var( )** du package **factoextra**:



Et pour la commande **fviz_pca_ind( )** toujours du package **factoextra**:



Maintenant nous calculons la CAH sur l'ACP avec la fonction **HCPC( )** :



Comme indiqué, nous cliquons là où nous voulons couper l'arbre (je clique environ à 0.6 pour cet exemple) ce qui donne comme premier graph:

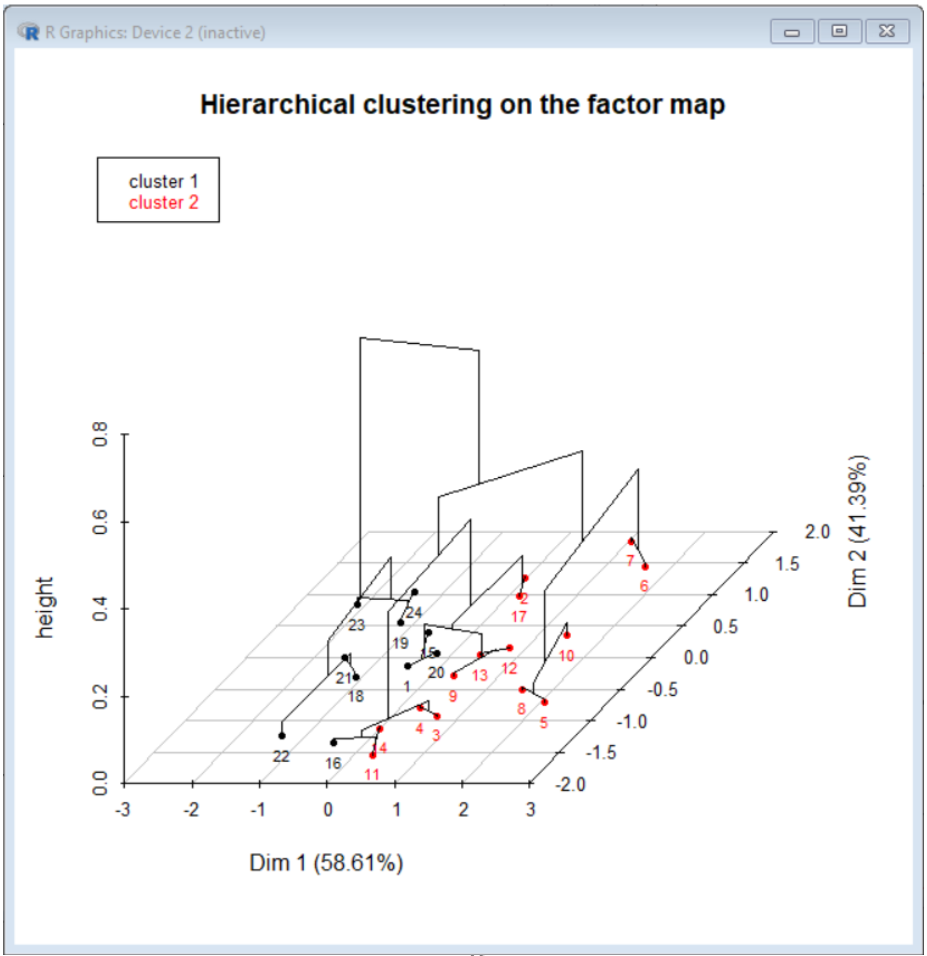
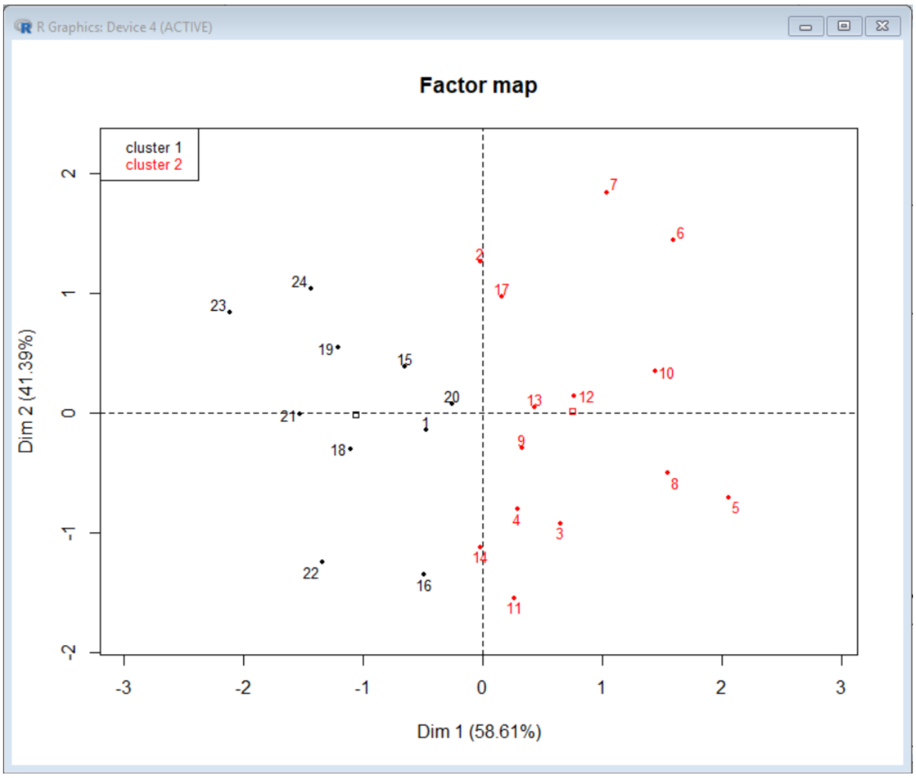


Figure 298 Graph 3D de classification hiérarchique ascendante (CAH) sur une carte factorielle

Et pour le deuxième graph (on y voit les centres des deux clusters par rapport au fait que j'ai cliqué à la hauteur de 0.6):

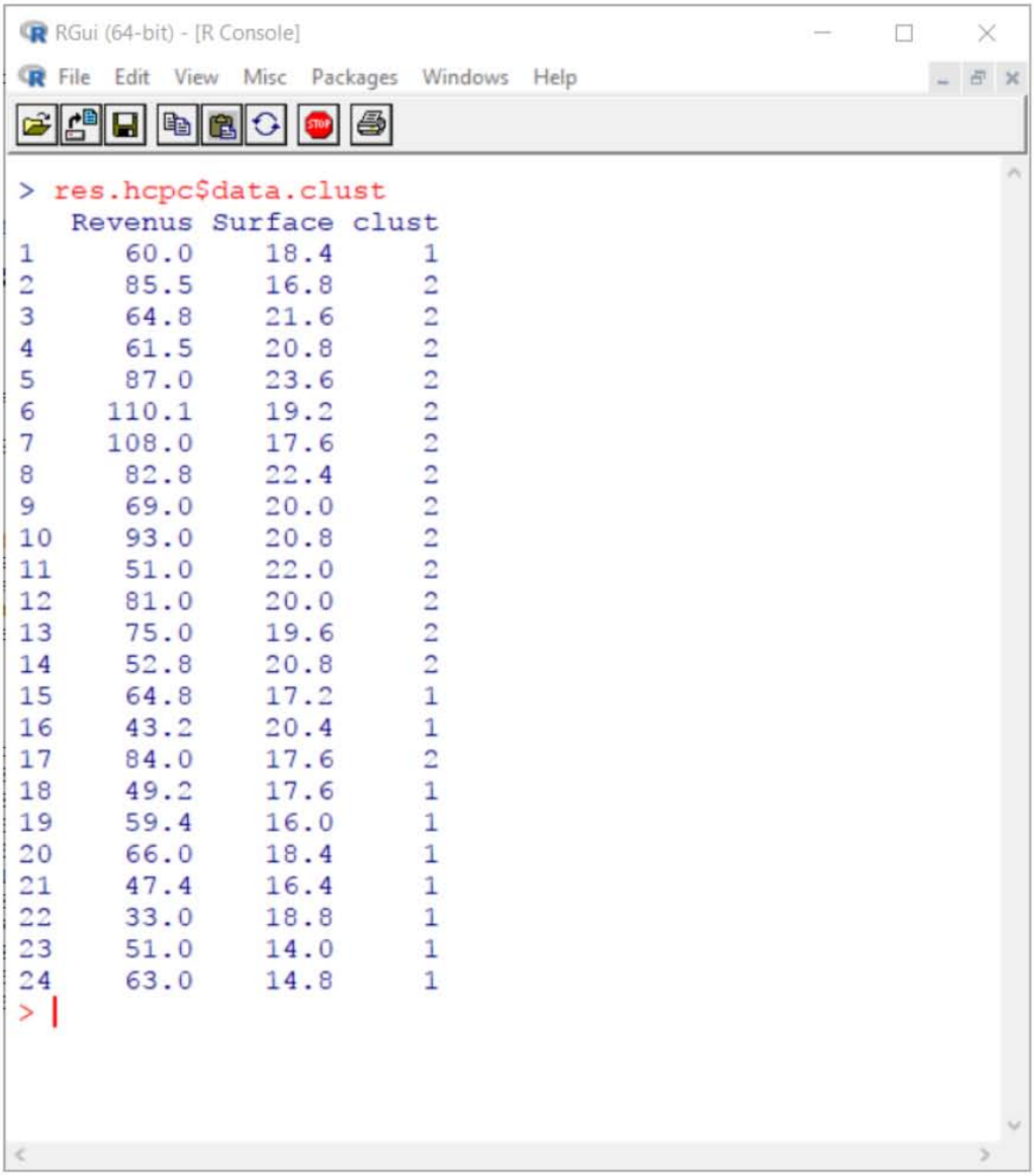


Ou avec **factoextra** et sa fonction **fviz_cluster( )** :



Figure 299 Graph de polygones de classification hiérarchique ascendante (CAH)

On peut évidemment sortir toutes les data avec leurs clusters respectifs:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> res.hcpc$data.clust
Revenus Surface clust
1 60.0 18.4 1
2 85.5 16.8 2
3 64.8 21.6 2
4 61.5 20.8 2
5 87.0 23.6 2
6 110.1 19.2 2
7 108.0 17.6 2
8 82.8 22.4 2
9 69.0 20.0 2
10 93.0 20.8 2
11 51.0 22.0 2
12 81.0 20.0 2
13 75.0 19.6 2
14 52.8 20.8 2
15 64.8 17.2 1
16 43.2 20.4 1
17 84.0 17.6 2
18 49.2 17.6 1
19 59.4 16.0 1
20 66.0 18.4 1
21 47.4 16.4 1
22 33.0 18.8 1
23 51.0 14.0 1
24 63.0 14.8 1
>

CAH avec bagging (bootstrap aggregating)

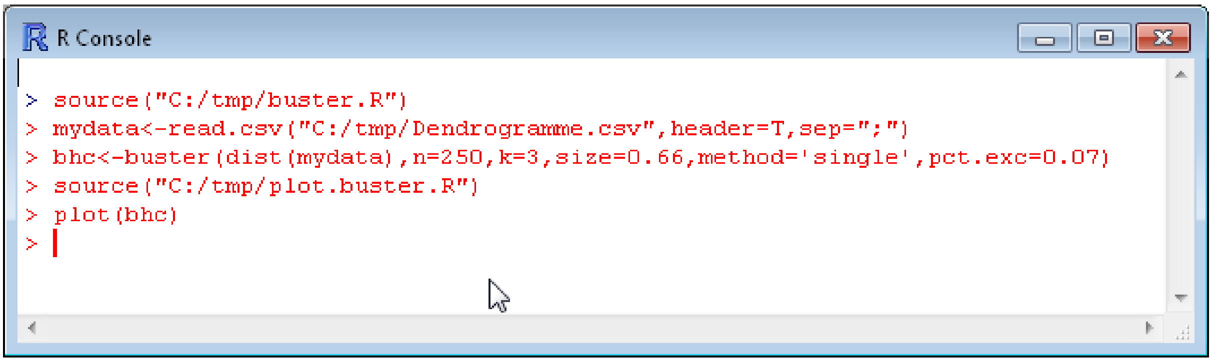
Un algorithme d'apprentissage est dit instable si de petits changements dans le jeu d'entraînement conduisent à des modèles significativement différents et donc des conclusions relativement différentes aussi.

Des méthodes d'analyses instables peuvent voir leur pouvoir prédictif amélioré par perturbations et combinaisons des différentes résultats et en incluant des corrections successives via des pondérations et différentes approches empiriques de corrections. L'idée du bagging (ou "bootstrap aggrégatif") est donc similaire au boostrapping à la différence que des paramètres ajoutés au modèle sont changés à chaque itération.

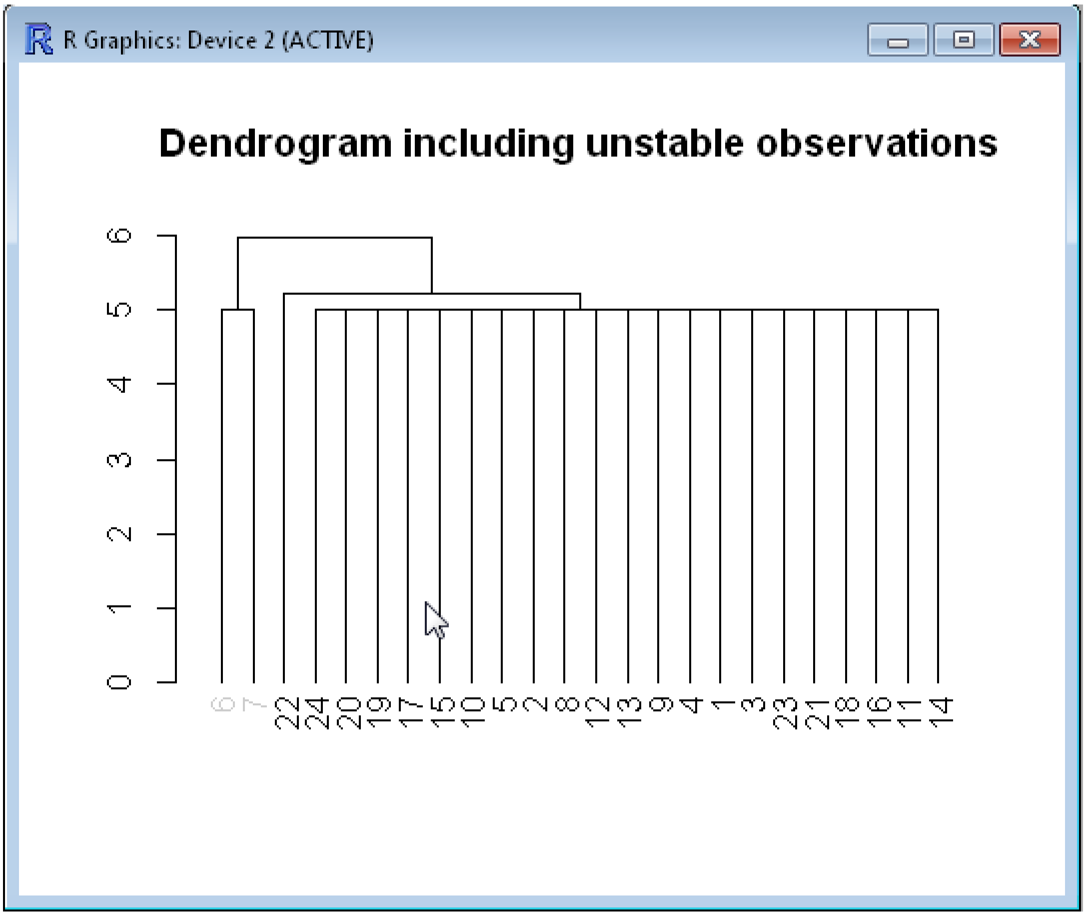
Effectuons un bagging du CAH précédent en utilisant une méthode proposée par Simon Raper sur le lien suivant:

https://github.com/SimonRaper/buster

Ce qui donne:



Avec:



Les éléments instables sont donc les éléments 6 et 7.

Exemple 343.: Clustering K-means (k moyennes en groupe)

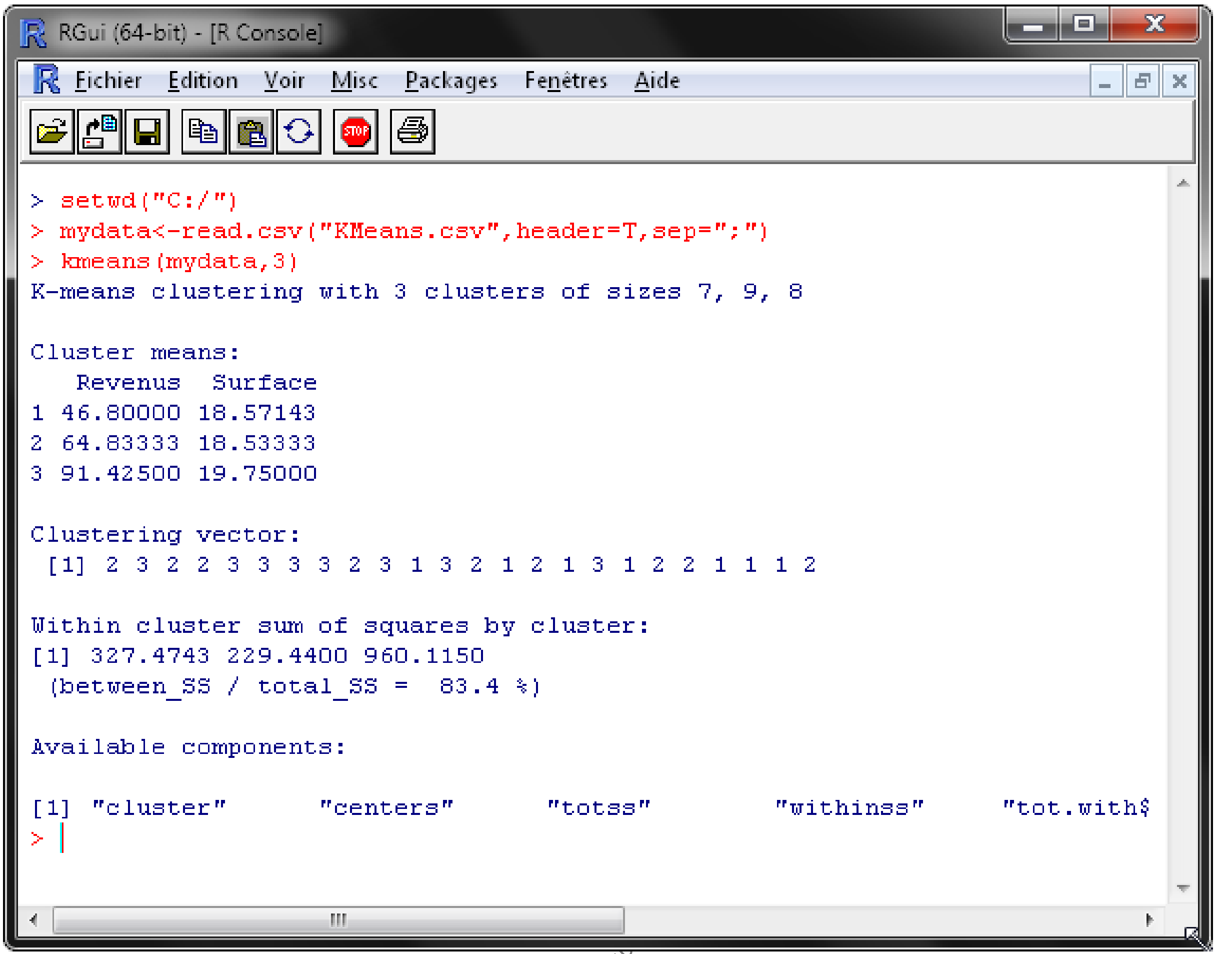
R 3.0.2

Nous reprenons les données utilisées dans le cours théorique (dont nous avons enlevé l'appartenance aux groupes catégoriels connus a priori pour des raisons évidentes de simplicité):

KMeans.csv - Bloc-notes

```
Revenus;Surface
60;18.4
85.5;16.8
64.8;21.6
61.5;20.8
87;23.6
110.1;19.2
108;17.6
82.8;22.4
69;20
93;20.8
51;22
81;20
75;19.6
52.8;20.8
64.8;17.2
43.2;20.4
84;17.6
49.2;17.6
59.4;16
66;18.4
47.4;16.4
33;18.8
51;14
63;14.8
```

Que nous souhaitons associer en trois groupes distincts. Nous utilisons alors la commande **kmeans( )** :



Nous retrouvons donc les mêmes résultats qu'avec Minitab (et a peu près les même que ceux calculés à la main dans le cours théorique).

Voyons si nous pouvons faire mieux avec un visuel:

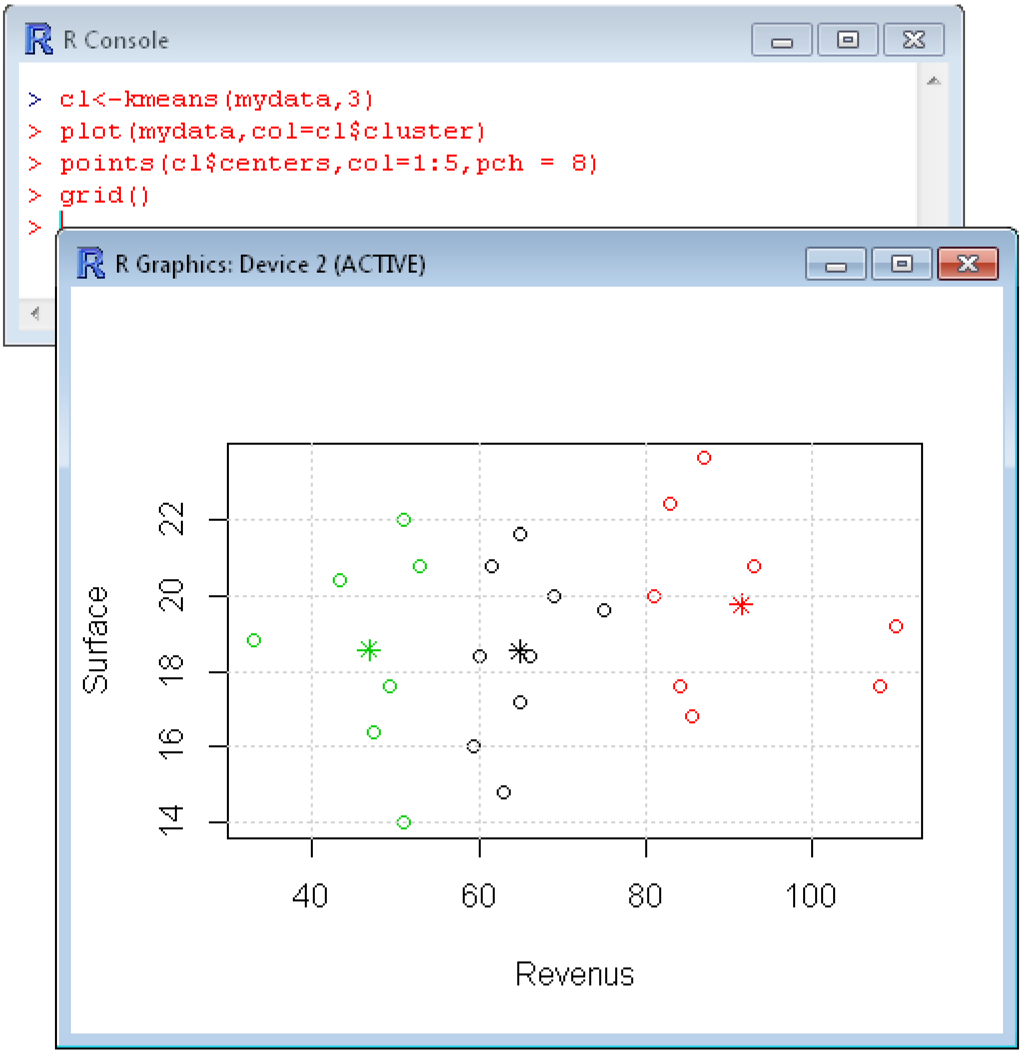
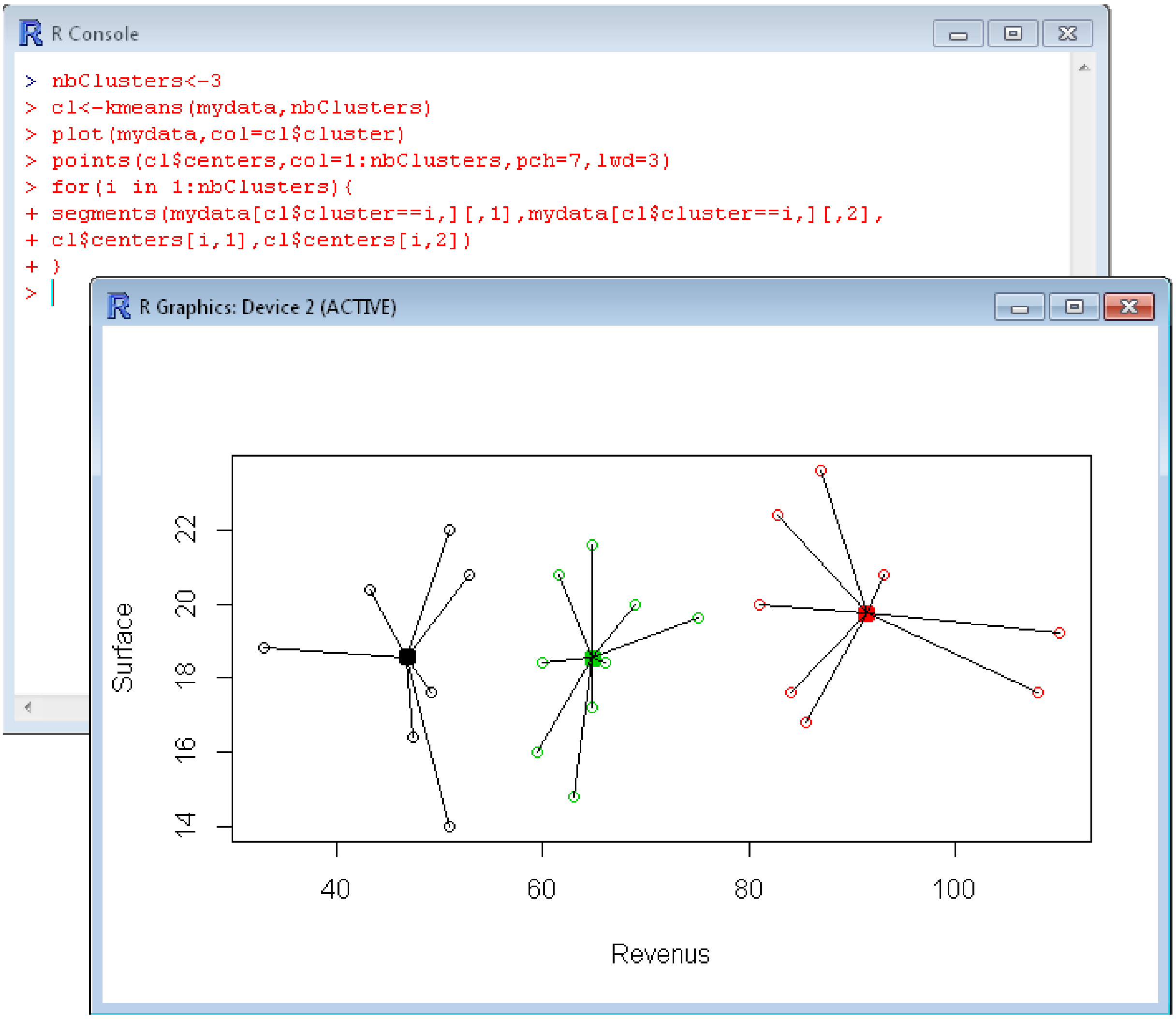


Figure 300 Graph de classification K-means

Ou toujours sans packages:



Et avec un autre visuel un peu plus technique (je dois encore déterminer ce que les nombres tout à droite signifient...) en utilisant les commandes **daisy( )** et **silhouette( )** des packages respectifs **cluster** et **HSAUR**:

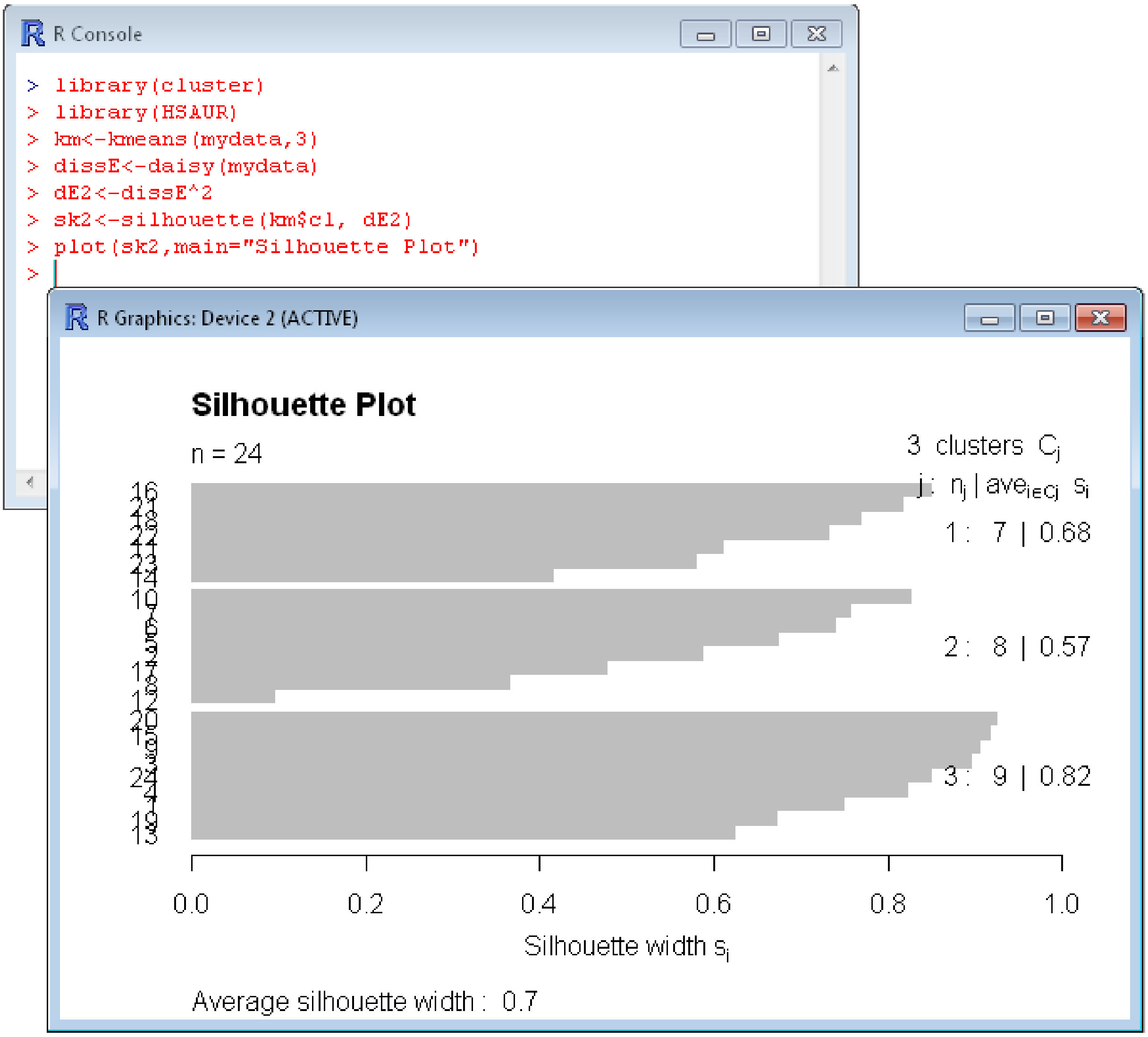


Figure 301 Graph de silhouette

ou encore mieux en utilisant la commande **clusplot( )** du package **fpc**:

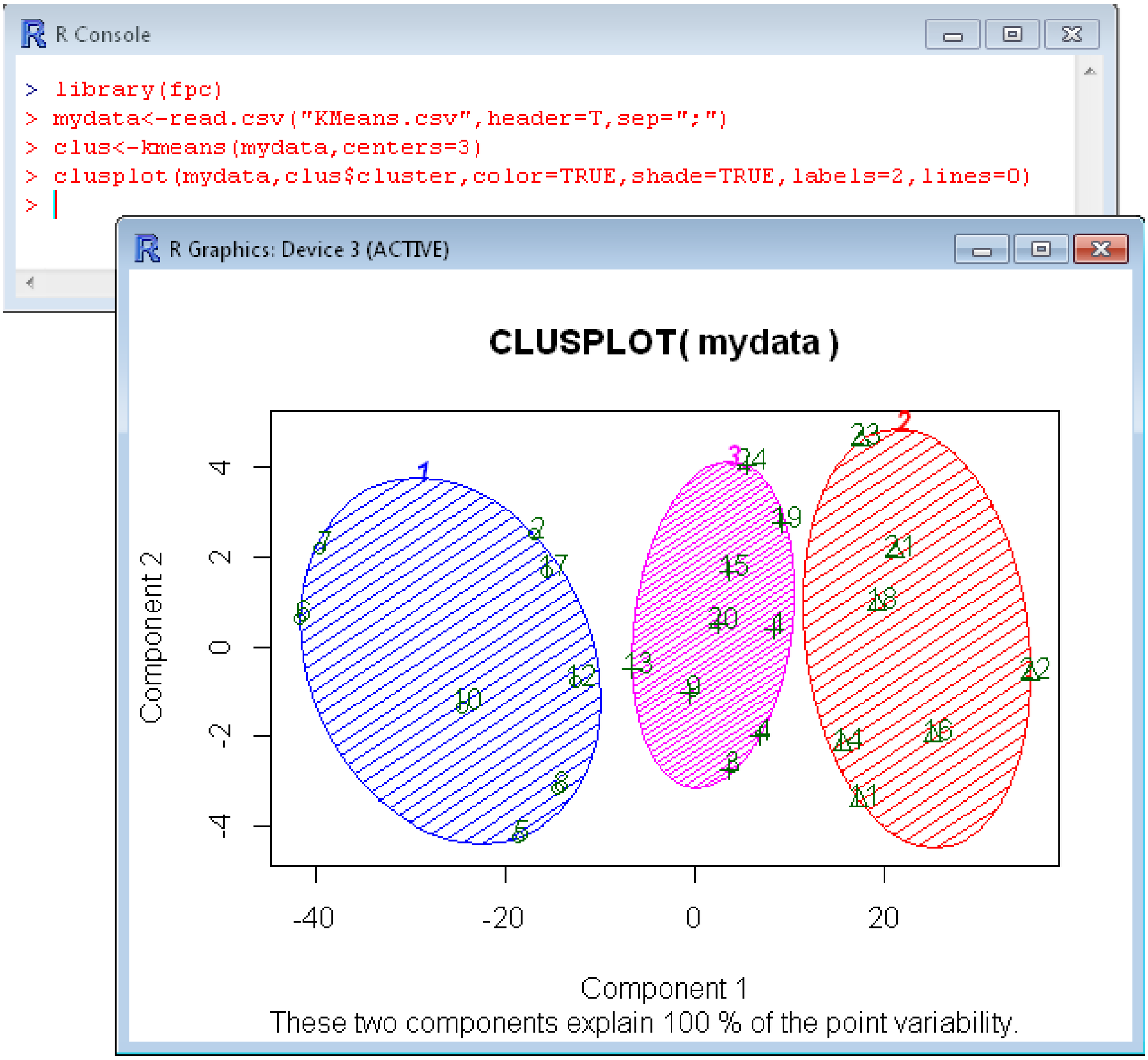
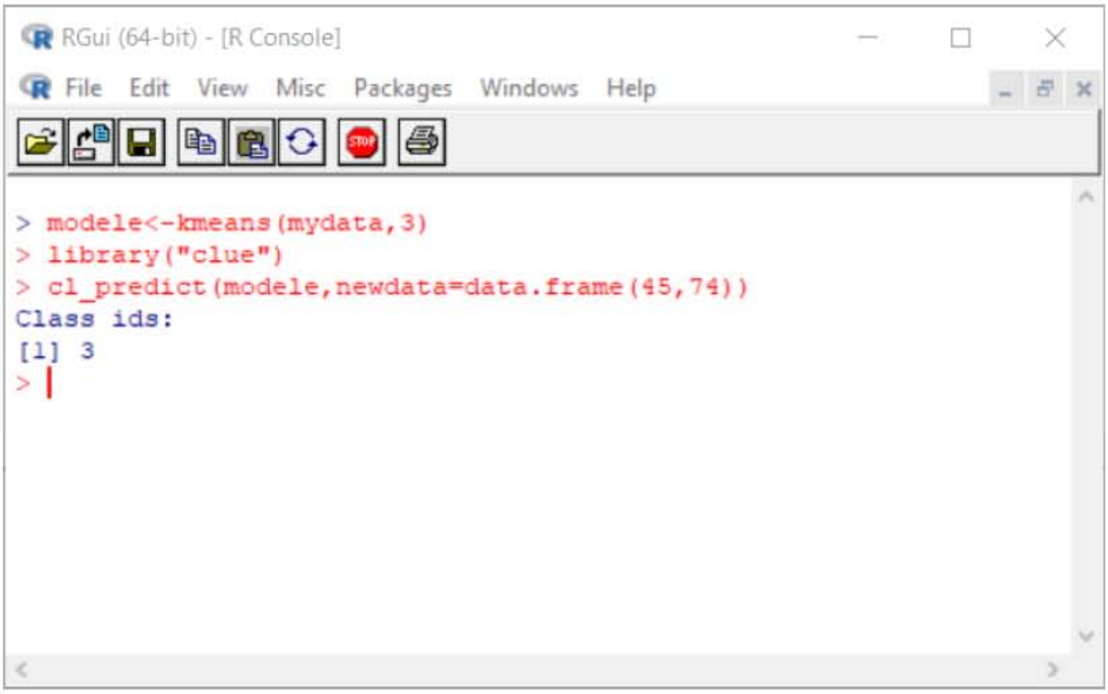


Figure 302 Graph ellipses de clustering par K-Means

Sinon, chose très importante, il est possible d'assigner un nouvel élément à l'aide de la fonction **cl_predict( )** du package **clue**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> modele<-kmeans(mydata,3)
> library("clue")
> cl_predict(modele,newdata=data.frame(45,74))
Class ids:
[1] 3
>

Exemple 344.: Clustering K-medoides

R 3.3.1

Donc ici le but est de voir comment utiliser l'algorithme que nous avons mentionné dans le cours théorique. Nous utilisons donc la commande **pam( )** du package **cluster**:

```
R Console
> library(cluster)
> mydata<-read.csv("c:/tmp/KMeans.csv",header=T,sep=";")
> cl <- kmeans(mydata, 3)
> kclus <- pam(mydata,3)
>
> par(mfrow=c(1,2))
> plot(mydata, col = kclus$clustering, main="Kmedoids Cluster")
> points(kclus$medoids, col = 1:3, pch = 10, cex = 4)
> plot(mydata, col = cl$cluster, main="Kmeans Cluster")
> points(cl$centers, col = 1:3, pch = 10, cex = 4)
>
```

Ce qui donne:

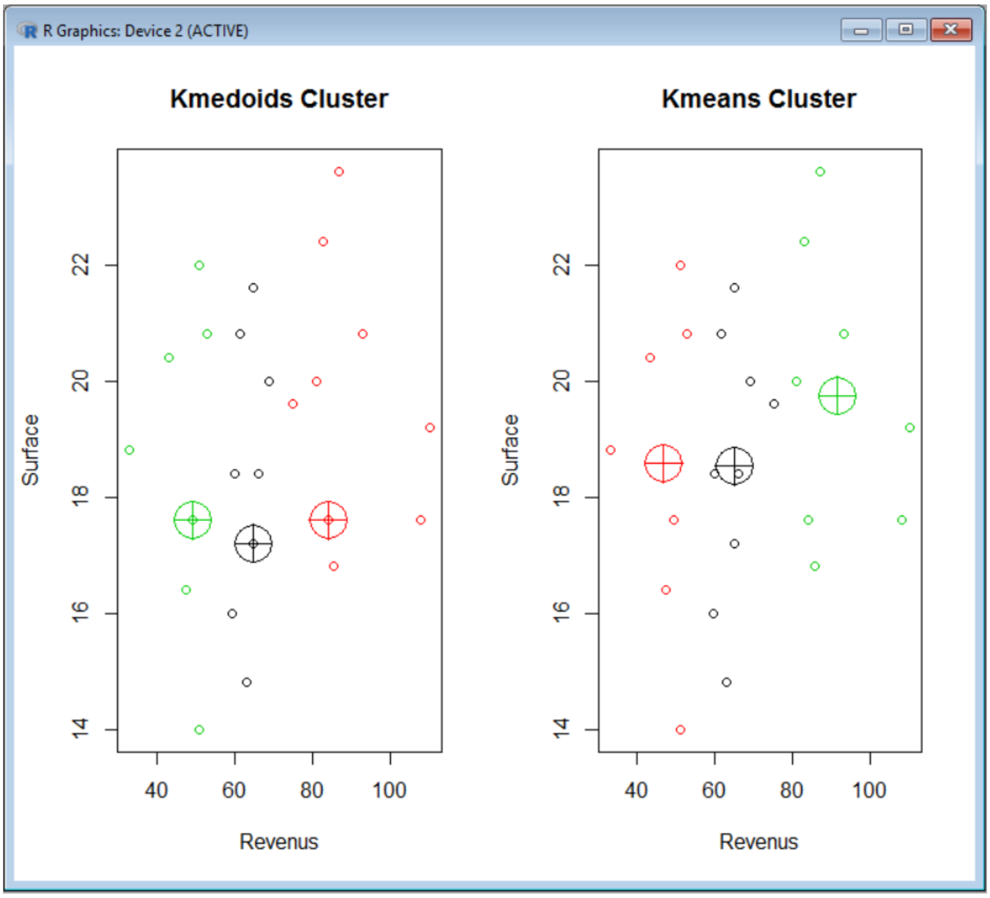
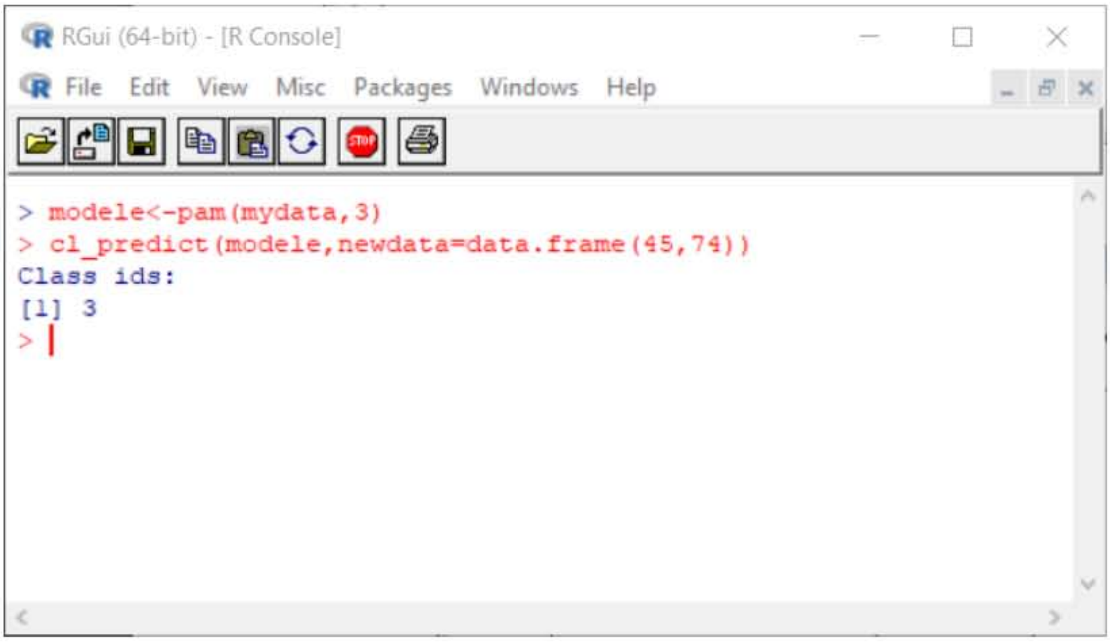


Figure 303 Graph de classification K-medoides

Sinon il est possible de classer un nouvel élément rapidement avec la fonction **cl_predict( )** du package **clue**:



Public

Exemple 345.: Clustering K-means (k moyennes en groupe) hiérarchisée

R 3.0.2

K-means représente l'un des algorithmes de clustering les plus populaires. Cependant, il a certaines limites : il oblige l'utilisateur à spécifier le nombre de clusters à l'avance et sélectionne aléatoirement les centroïdes initiaux. La solution finale de clustering k-means est très sensible à cette sélection aléatoire initiale des centres de cluster. Le résultat peut être (légèrement) différent à chaque fois que vous calculez k-means.

L'algorithme se résume comme suit :

1. Calculer le clustering hiérarchique et découper l'arbre en k-clusters
2. Calculer le centre (c'est-à-dire la moyenne) de chaque cluster
3. Calculer k-means en utilisant l'ensemble des centres de cluster (défini à l'étape 2) comme centres de cluster initiaux

La fonction **hkmeans( )** du package **factoextra**, fournit une solution simple pour calculer le clustering hiérarchique k-means. Le format du résultat est similaire à celui fourni par la fonction standard **kmeans( )** elle-même!

Voyons donc cela:

```
> library("factoextra")
> mydata<-read.csv("c:/tmp/KMeans.csv",header=T,sep=";")
> hkmeans(mydata,3)
Hierarchical K-means clustering with 3 clusters of sizes 9, 8, 7

Cluster means:
   Revenus  Surface
1 64.83333 18.53333
2 91.42500 19.75000
3 46.80000 18.57143

Clustering vector:
 [1] 1 2 1 1 2 2 2 2 1 2 3 2 1 3 1 3 2 3 1 1 3 3 3 1

Within cluster sum of squares by cluster:
[1] 229.4400 960.1150 327.4743
 (between_SS / total_SS =  83.4 %)

Available components:

 [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
 [6] "betweenss"    "size"         "iter"         "ifault"       "data"
[11] "hclust"
> #à comparer avec
> kmeans(mydata,3)
K-means clustering with 3 clusters of sizes 9, 7, 8

Cluster means:
   Revenus  Surface
1 64.83333 18.53333
2 46.80000 18.57143
3 91.42500 19.75000

Clustering vector:
 [1] 1 3 1 1 3 3 3 3 1 3 2 3 1 2 1 2 3 2 1 1 2 2 2 1

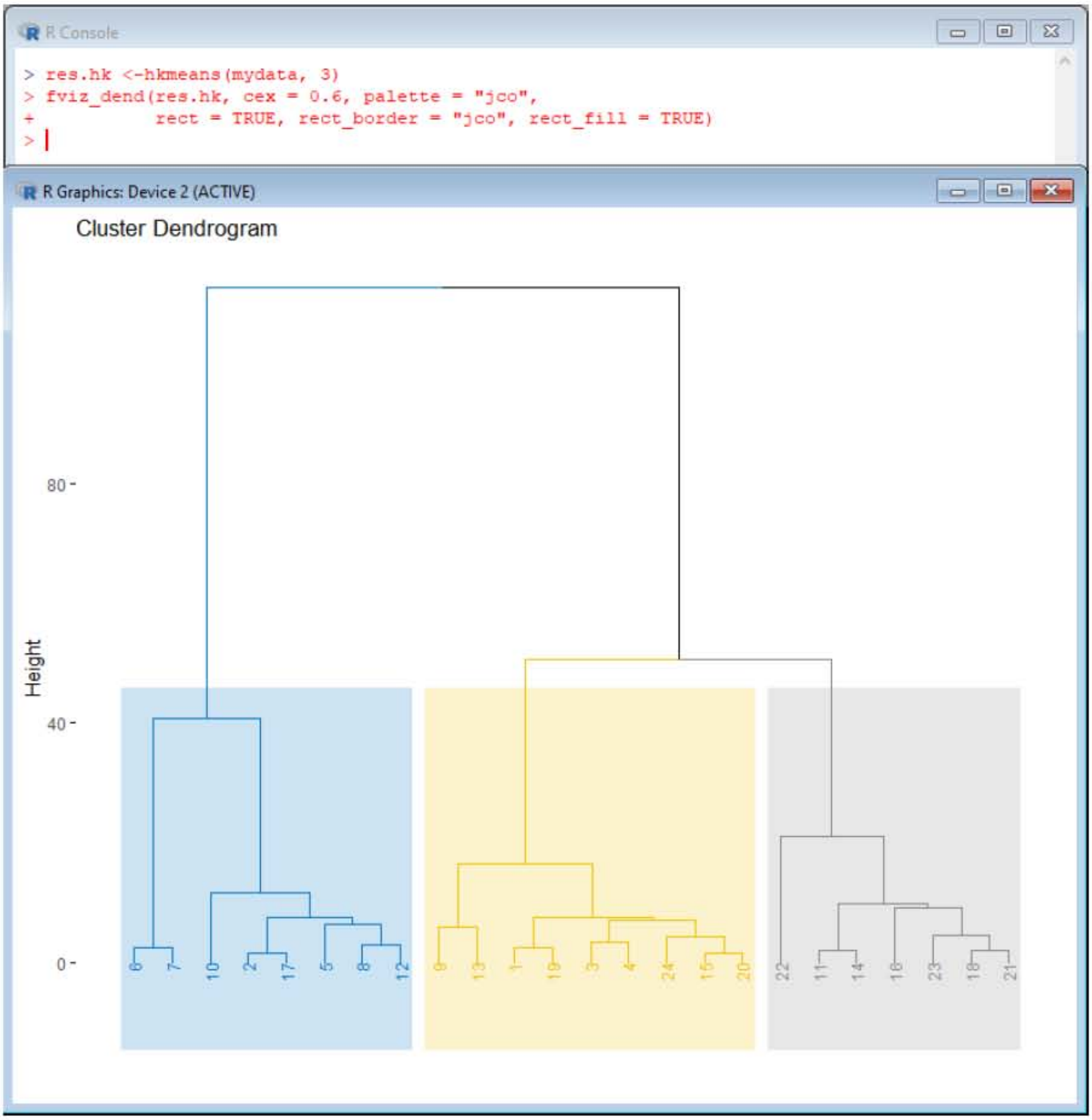
Within cluster sum of squares by cluster:
[1] 229.4400 327.4743 960.1150
 (between_SS / total_SS =  83.4 %)

Available components:

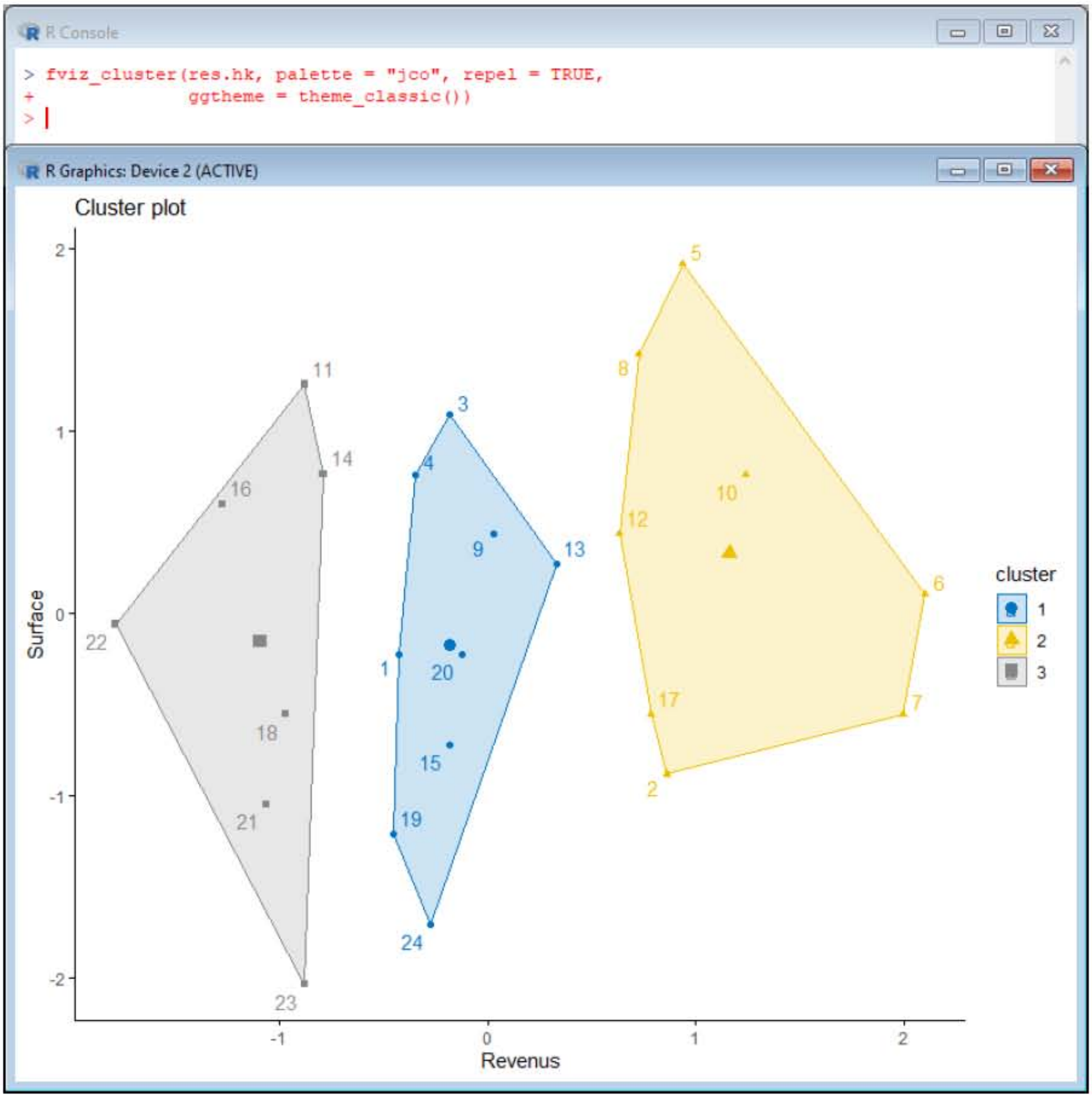
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> 
```

Dans ce cas simpliste, les deux méthodes donnent le même résultat.

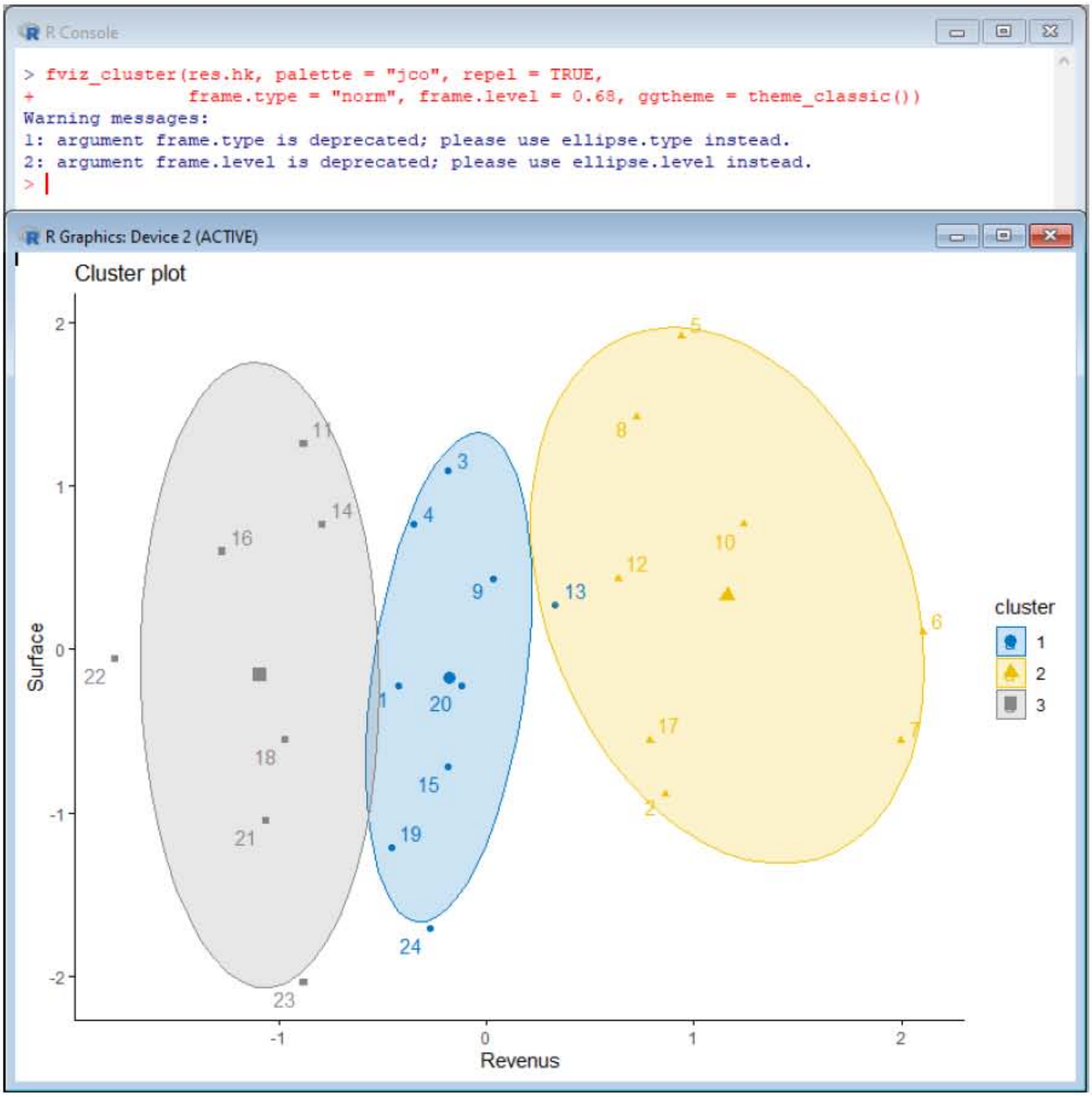
On peut afficher l'arbre visuellement:


R Console
> res.hk <-hkmeans(mydata, 3)
> fviz_dend(res.hk, cex = 0.6, palette = "jco",
+ rect = TRUE, rect_border = "jco", rect_fill = TRUE)
R Graphics: Device 2 (ACTIVE)
Cluster Dendrogram
Height
80
40
0

Et:


R Console
> fviz_cluster(res.hk, palette = "jco", repel = TRUE,
+ ggtheme = theme_classic())
R Graphics: Device 2 (ACTIVE)
Cluster plot
Surface
Revenus
cluster
1
2
3

Ou:

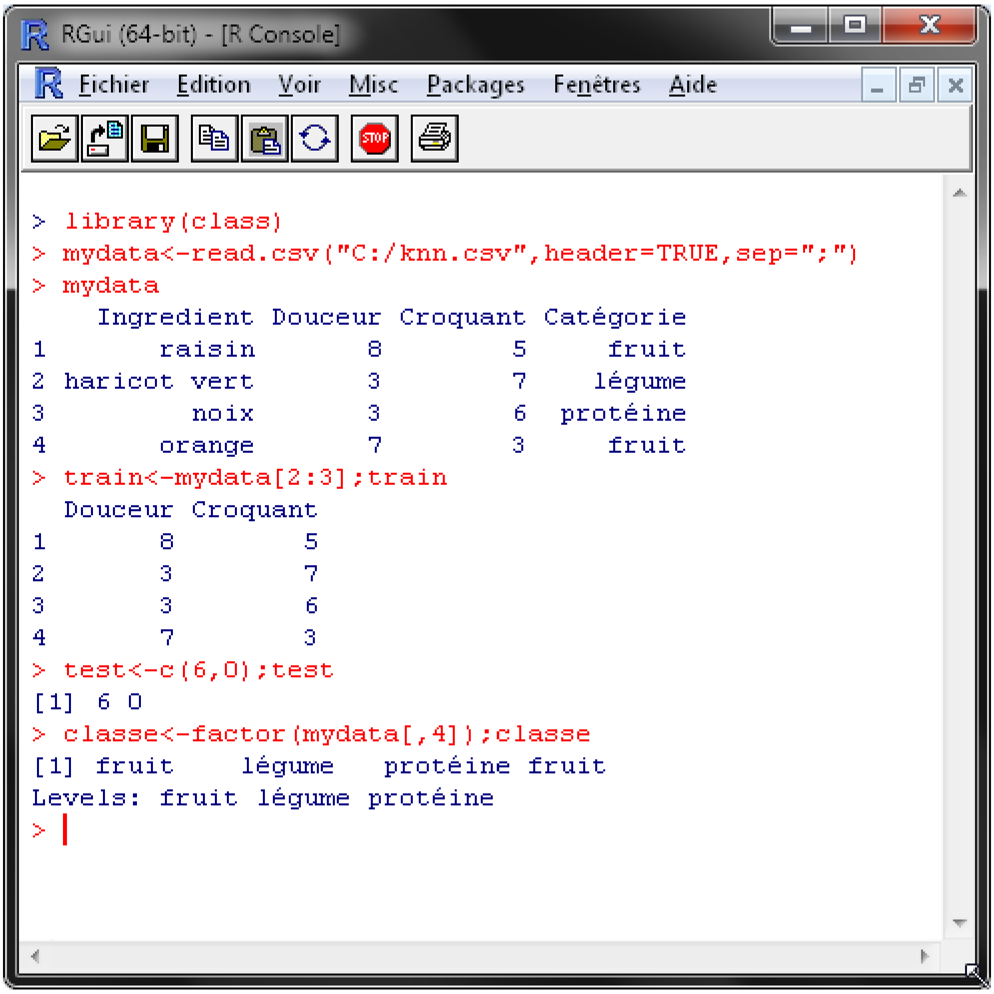

R Console
> fviz_cluster(res.hk, palette = "jco", repel = TRUE,
+ frame.type = "norm", frame.level = 0.68, ggtheme = theme_classic())
Warning messages:
1: argument frame.type is deprecated; please use ellipse.type instead.
2: argument frame.level is deprecated; please use ellipse.level instead.
R Graphics: Device 2 (ACTIVE)
Cluster plot
Surface
Revenus
cluster
1
2
3

Exemple 346.: Clustering K plus proches voisins (kNN: k nearest neighbours)

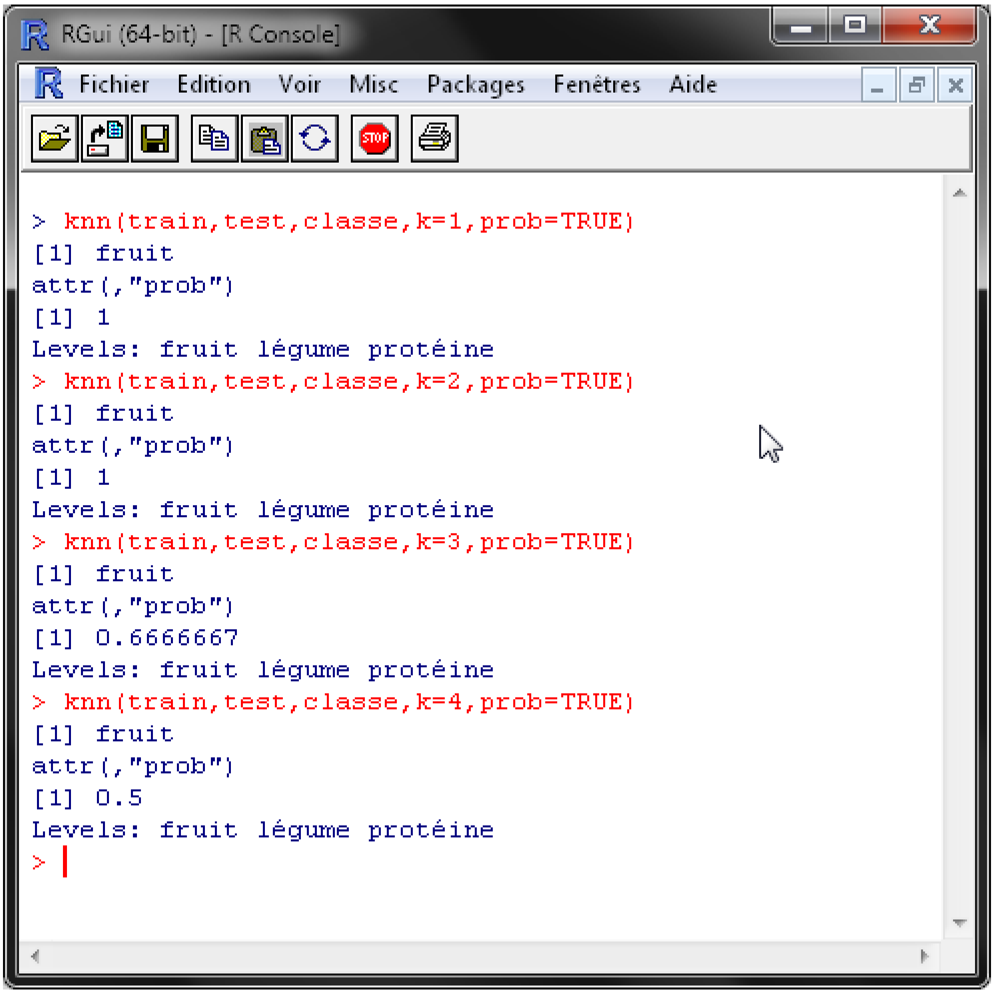
R 3.0.2

Rappelons que ce modèle fonctionne particulièrement bien avec un très grand nombre de données et surtout si les données utilisées pour la distance sont numériques et continues afin de pouvoir calculer la distance euclidienne comme c'est le cas ci-dessous. Dans le cas de valeurs nominales ou ordinales, il faut avec la plus grande prudence les convertir en facteurs numériques.

Nous reprenons les données utilisées dans le cours théorique:



Et nous souhaiterions savoir comment selon la distance euclidienne, une tomate de douceur 6 et de croquant 0 sera donc classifiée... Nous obtenons alors:



Nous voyons que dans tous les cas la tomate est classifiée en tant que fruit (ouf!) que son plus proche voisin (modèle 1-nn) fait partie forcément que d'une seule des classes (d'où le 1=100%). Avec les deux plus proches voisins (modèle 2-nn) nous avons encore une fois deux plus proches voisins faisant partie de la même classe (donc 2/2=1=100%). Avec les trois plus proches voisins (modèle 3-nn) nous avons 2 éléments de la même classe et 1 de classe opposée (donc 2/3=0.66=66%). Avec les 4 plus proches voisions (modèle 4-nn) nous avons 2 éléments de la même classe et deux éléments de deux autres classes (donc 2/4=0.5=50%).

Exemple 347.: Clustering Modèles de mélanges gaussiens (gaussian mixtures models)

R 3.5.1

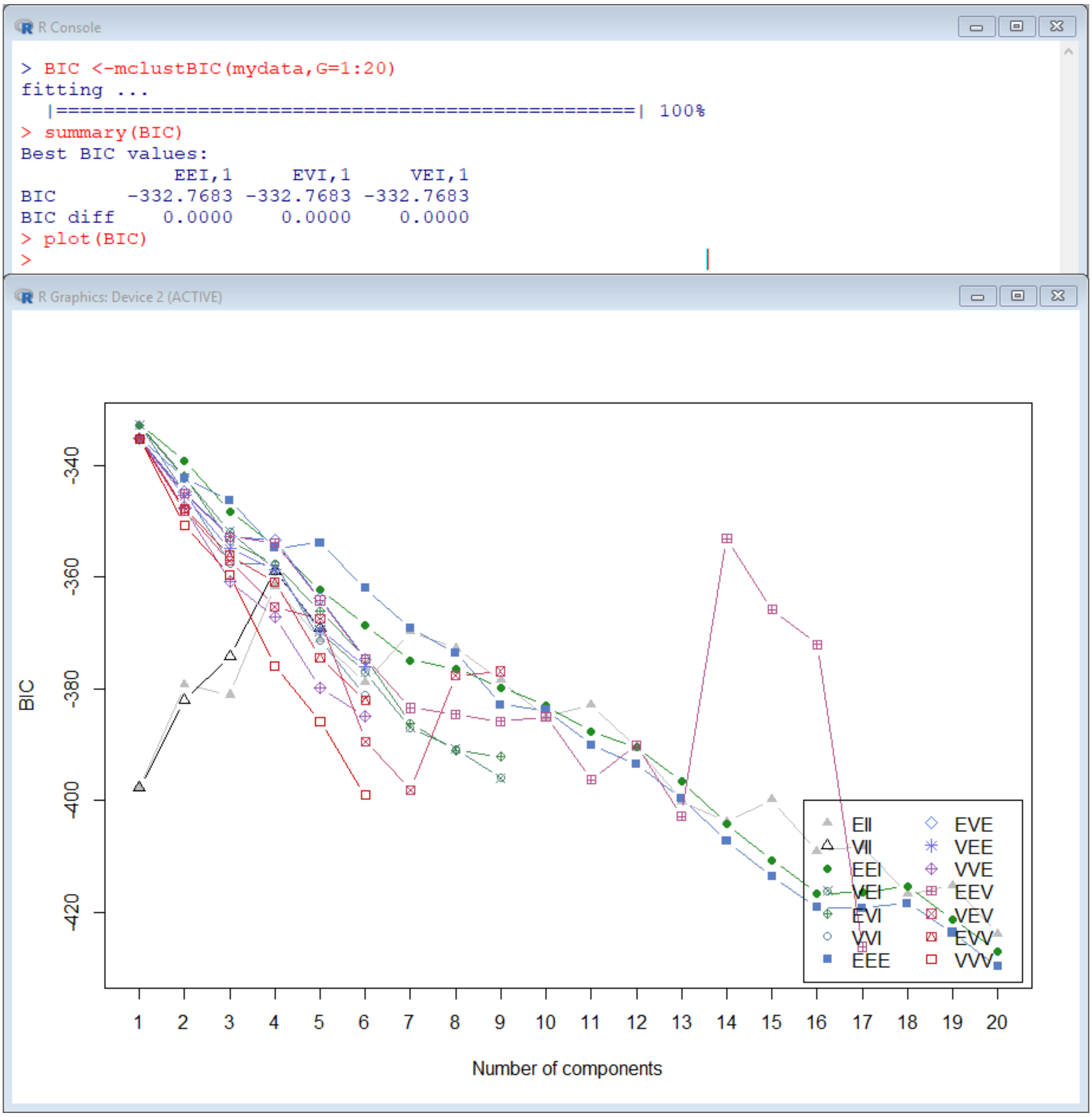
Nous n'avons pas étudié en détails ce modèle "lisse" de classification dans le cours théorique car c'est principalement un agglomérat d'une succession d'outils statistiques et algorithmiques. Cependant nos connaissances des mécanismes sous-jacents sont largement suffisantes pour que nous en voyons un cas pratique!

D'abord nous chargeons le package **mclust** ainsi que le même jeu de données que celui utilisé pour les k-means (nous allons donc faire une classification fuzzy non-supervisée[35]!):

```
> library("mclust")
> mydata<-read.csv("c:/tmp/Kmeans.csv",header=T,sep=";")
> mydata
   Revenus Surface
1     60.0    18.4
2     85.5    16.8
3     64.8    21.6
4     61.5    20.8
5     87.0    23.6
6    110.1    19.2
7    108.0    17.6
8     82.8    22.4
9     69.0    20.0
10    93.0    20.8
11    51.0    22.0
12    81.0    20.0
13    75.0    19.6
14    52.8    20.8
15    64.8    17.2
16    43.2    20.4
17    84.0    17.6
18    49.2    17.6
19    59.4    16.0
20    66.0    18.4
21    47.4    16.4
22    33.0    18.8
23    51.0    14.0
24    63.0    14.8
>
```

[35] Le lecteur doit prendre notre qu'il existe aussi une autre package très puissant et complet appelé Rmixmod qui permette de faire de la classification fuzzy gaussienne aussi bien supervisée que non-supervisée!

D'abord nous lançons une analyse de type BIC avec la fonction **mclustBIC( )** qui va nous montrer en fonction du nombre de clusters et des 14 structures classiques de la matrice de covariance, les valeurs du BIC:



Nous voyons donc que le choix optimal serait 1 seul et unique cluster et la structure de matrice de covariance de type EEI.

Cependant forçons une classification avec 2 clusters à l'aide de la fonction **Mclust( )**:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> model <- Mclust(mydata,G=2, x = BIC)
> summary(model, parameters = TRUE)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EEI (diagonal, equal volume and shape) model with 2 components:

 log.likelihood  n df       BIC       ICL
      -158.4601 24  7 -339.1666 -343.2481

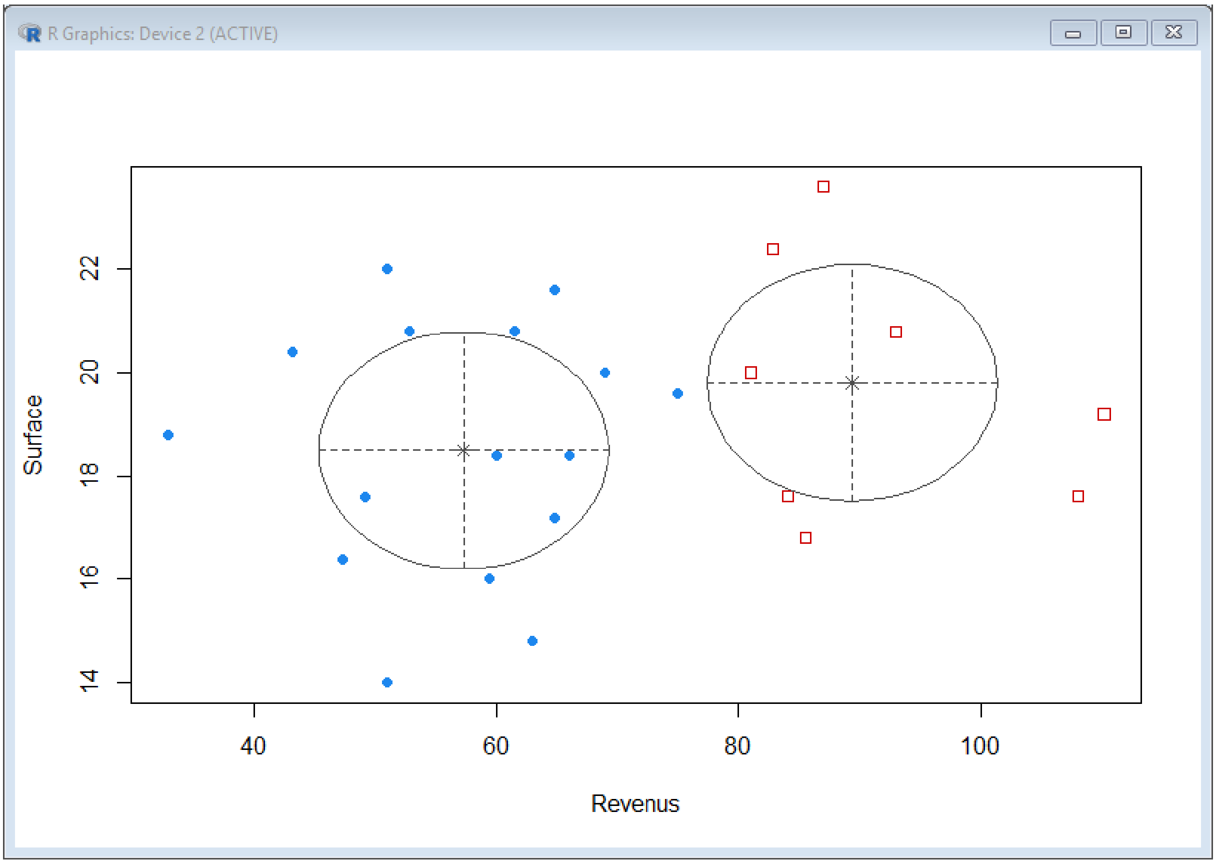
Clustering table:
 1  2
16  8

Mixing probabilities:
        1         2
0.6528833 0.3471167

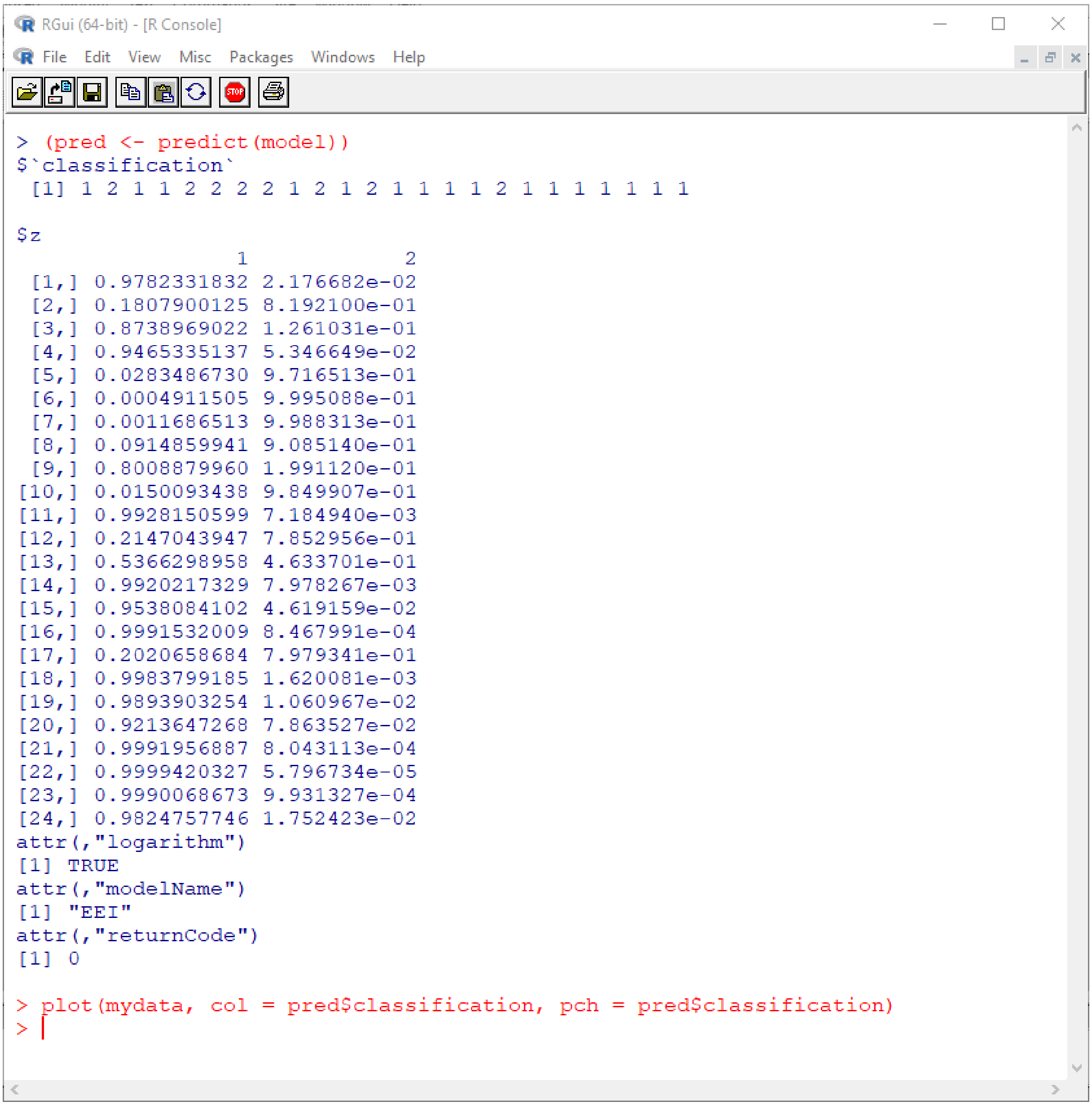
Means:
            [,1]     [,2]
Revenus 57.31958 89.34892
Surface 18.49659 19.80282

Variances:
[,,1]
         Revenus  Surface
Revenus 142.9534 0.000000
Surface   0.0000 5.264154
[,,2]
         Revenus  Surface
Revenus 142.9534 0.000000
Surface   0.0000 5.264154
> plot(model, what = "classification")
> 
```

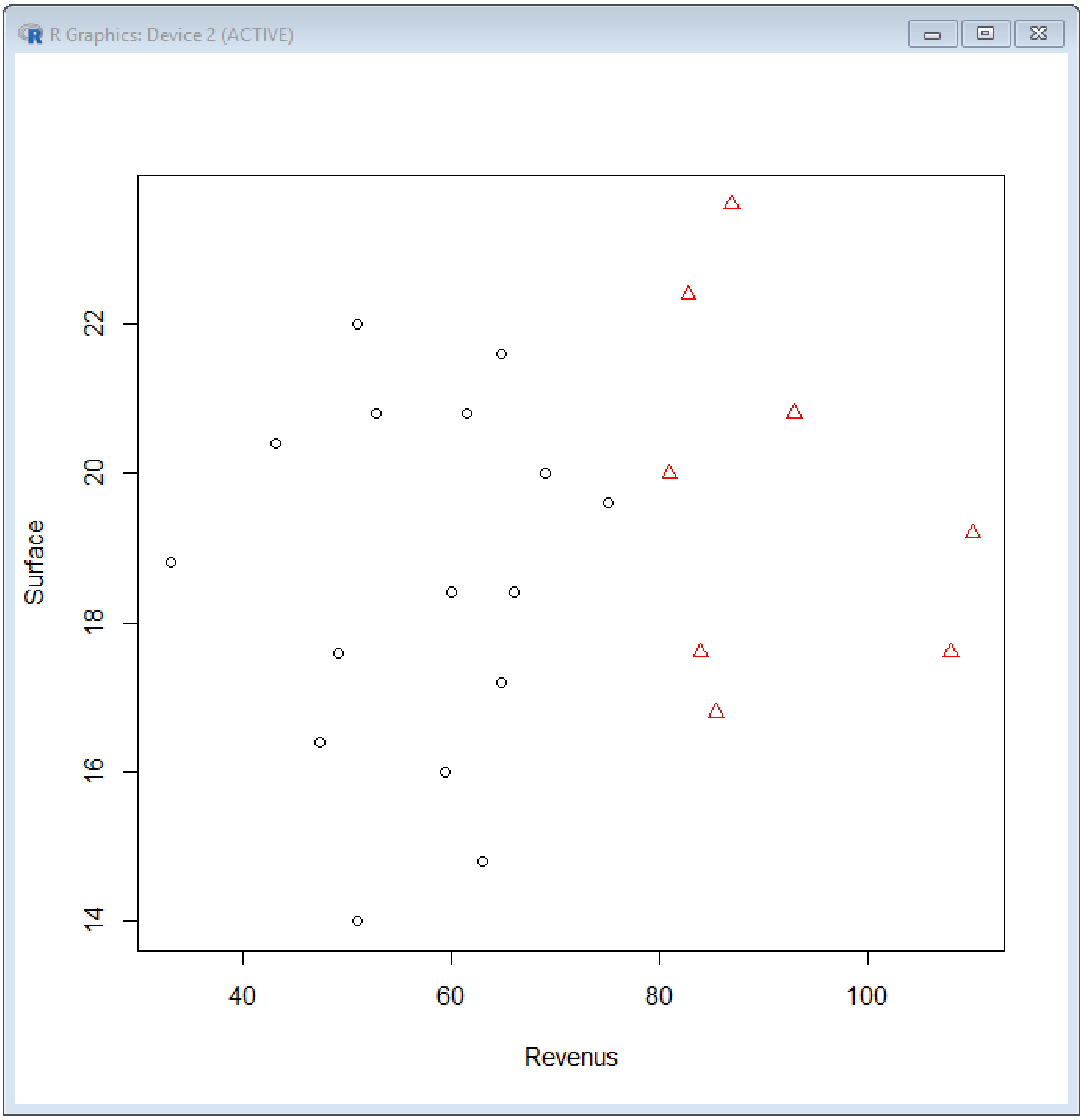
Et au niveau du plot:



Nous pouvons aussi faire des prédictions:



Et aussi un graphique des classes prédites:



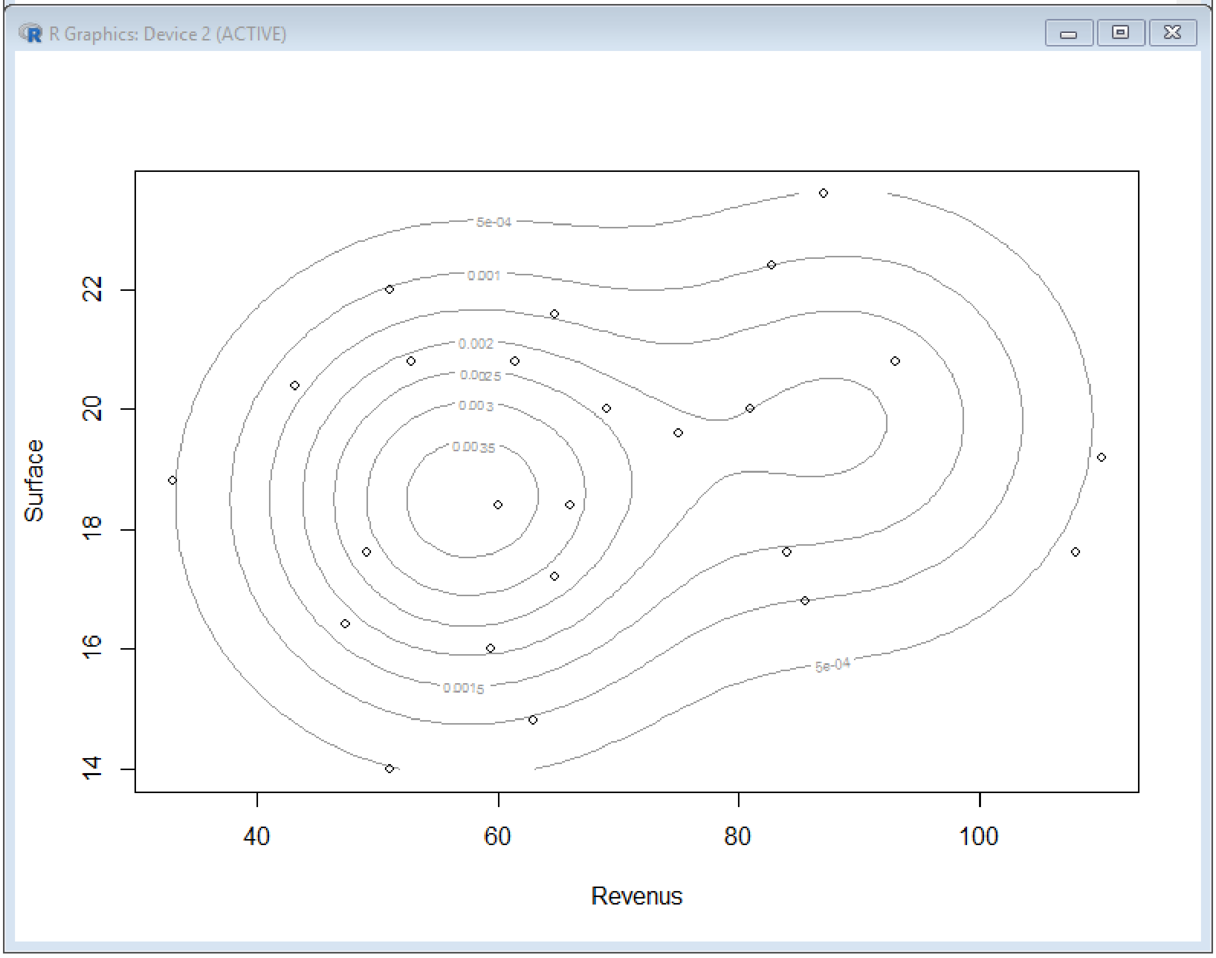
Nous pouvons aussi faire une analyse de densité (bien évidemment elle est bivariée dans le cas présent):

```
> model <- densityMclust(mydata,G=2, x = BIC)
> summary(model)
-------------------------------------------------------
Density estimation via Gaussian finite mixture modeling
-------------------------------------------------------

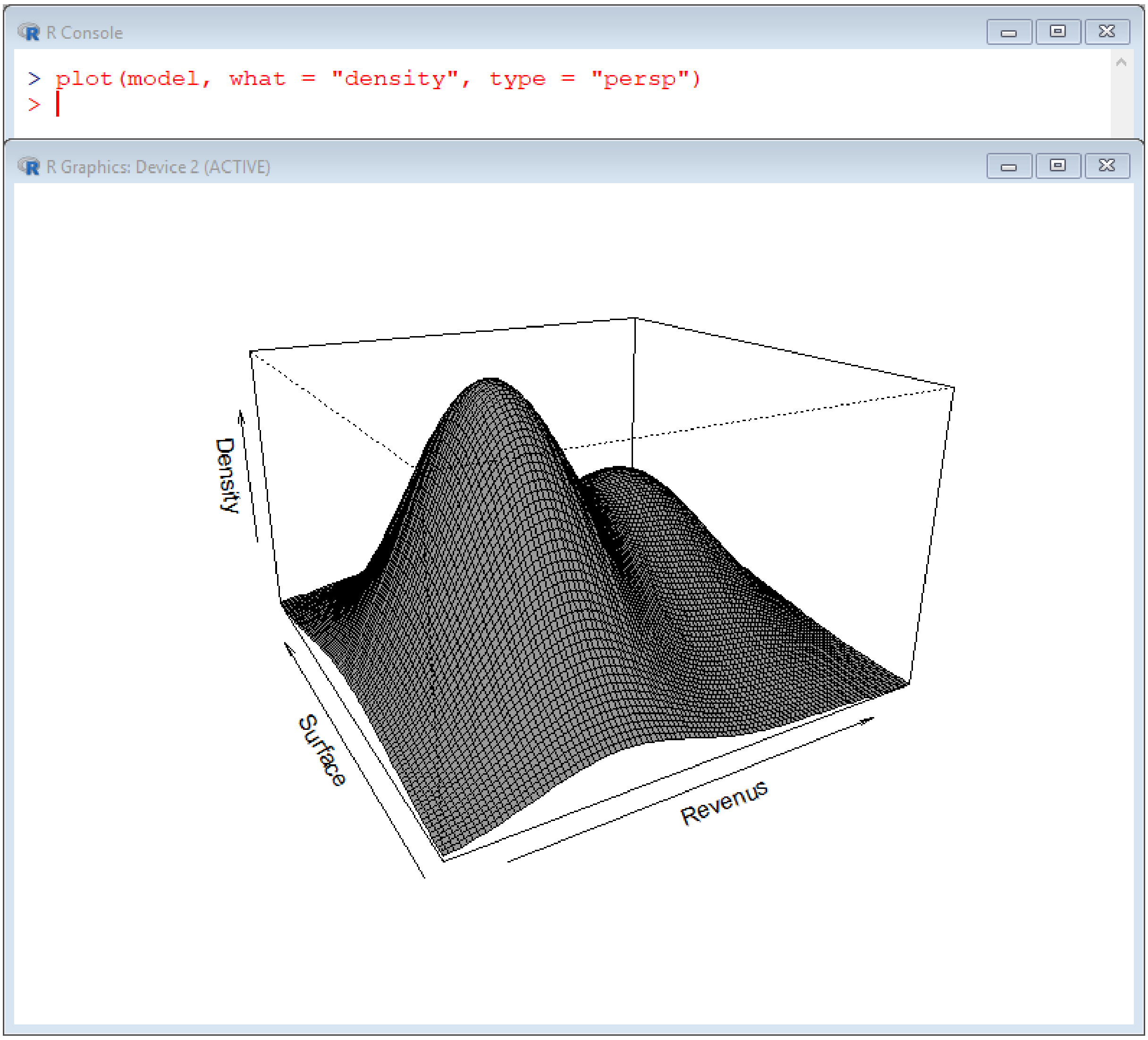
Mclust EEI (diagonal, equal volume and shape) model with 2 components:

 log.likelihood  n df       BIC       ICL
      -158.4601 24  7 -339.1666 -343.2481

Clustering table:
 1  2
16  8
> plot(model, what = "density", data = mydata)
>
```



Nous pouvons même avoir le graph en 3 dimensions:



Voyons un autre exemple condensé:

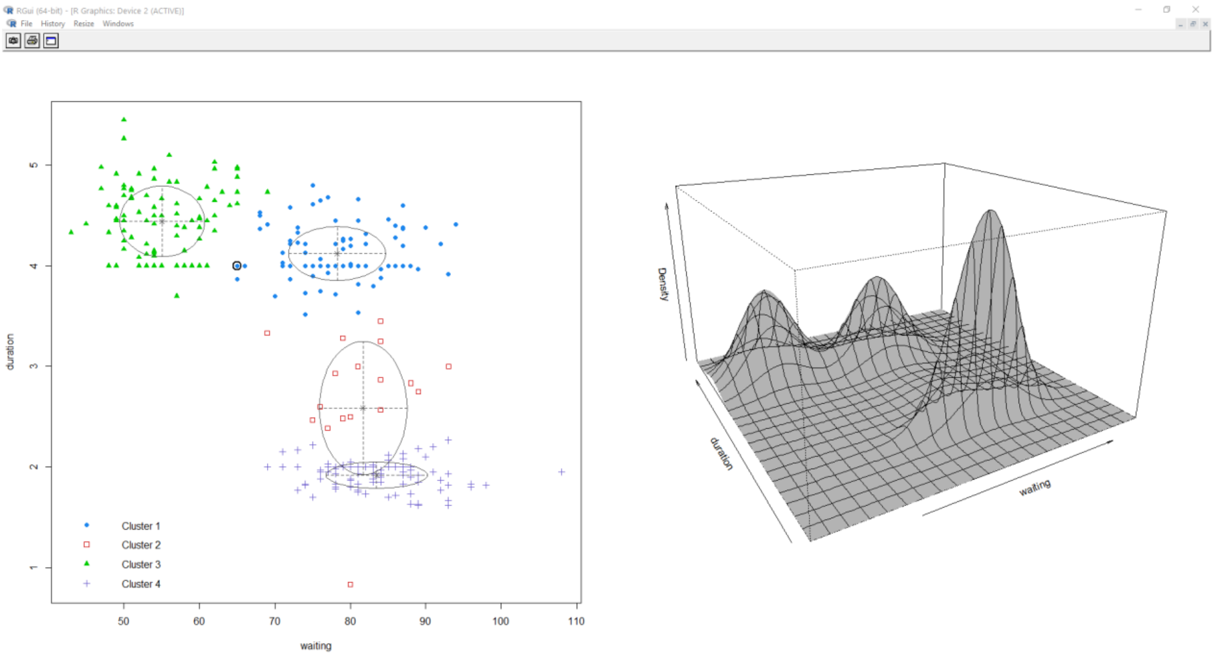
```
> library("mclust")
> data("geyser", package="MASS")
> mcl <- Mclust(geyser)
fitting ...
  |==================================================================================| 100%
> summary(mcl)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVI (diagonal, varying volume and shape) model with 4 components:

 log-likelihood   n df       BIC       ICL
       -1330.13 299 19 -2768.568 -2798.746

Clustering table:
 1  2  3  4
90 17 98 94
> par(mfrow = c(1, 2))
> plot( mcl, what="classification")
> legend("bottomleft", legend=paste("Cluster", 1:4), bty='n',
+ pch=mclust.options("classPlotSymbols")[1:4],
+ col=mclust.options("classPlotColors")[1:4])
> uncert <- which.max( mcl$uncertainty)
> points( geyser[uncert,], cex=2, lwd=2)
> round( mcl$z[which.max(mcl$uncertainty),], 3) #prob of belonging to cluster
[1] 0.535 0.001 0.464 0.000
> plot( mcl, what="density", type="persp")
> mcl$parameters$pro #mixture weights pi_k
[1] 0.29283469 0.07968364 0.33392296 0.29355871
>
```

Ce qui donne graphiquement:

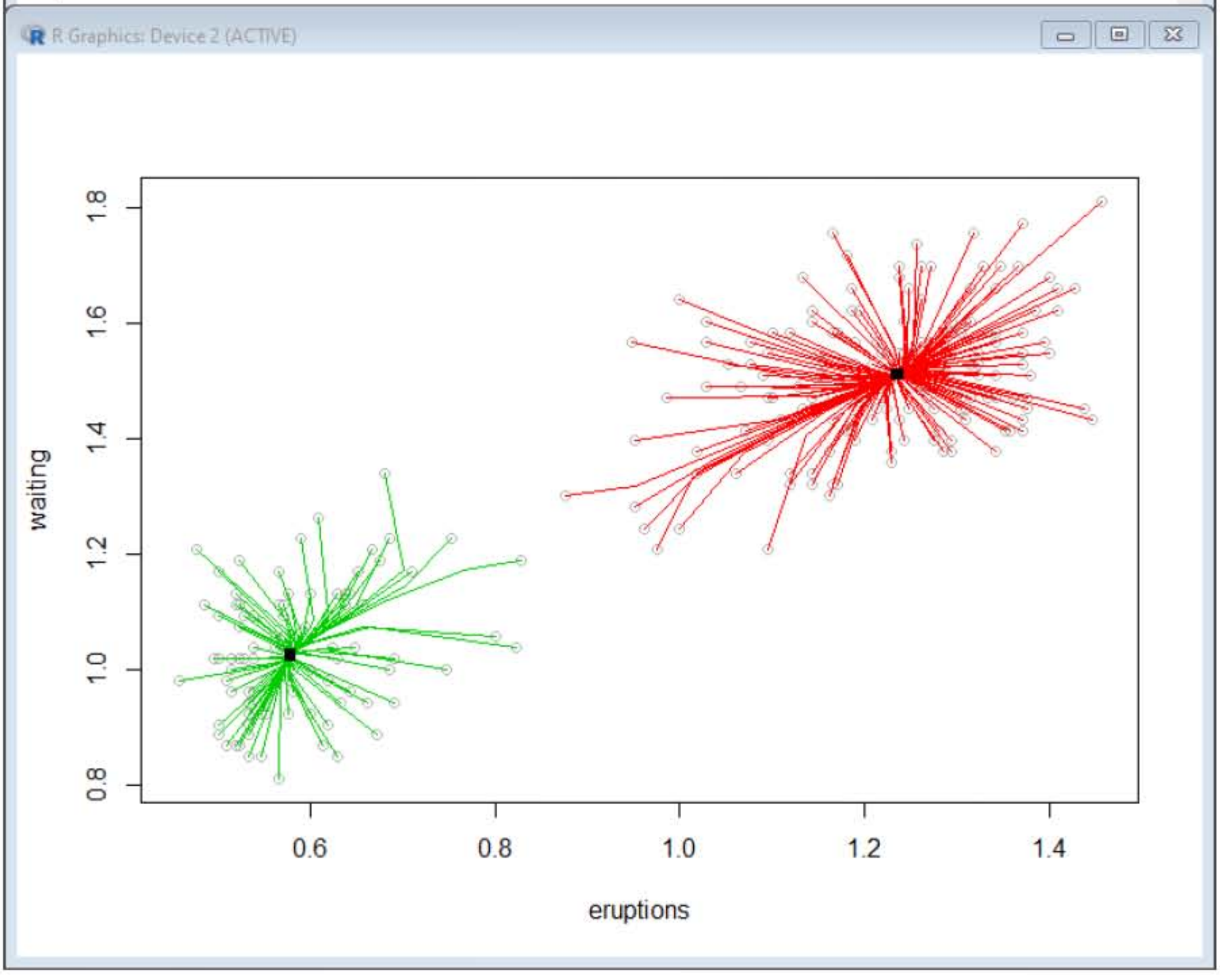


Exemple 348.: Clustering par Mean Shift

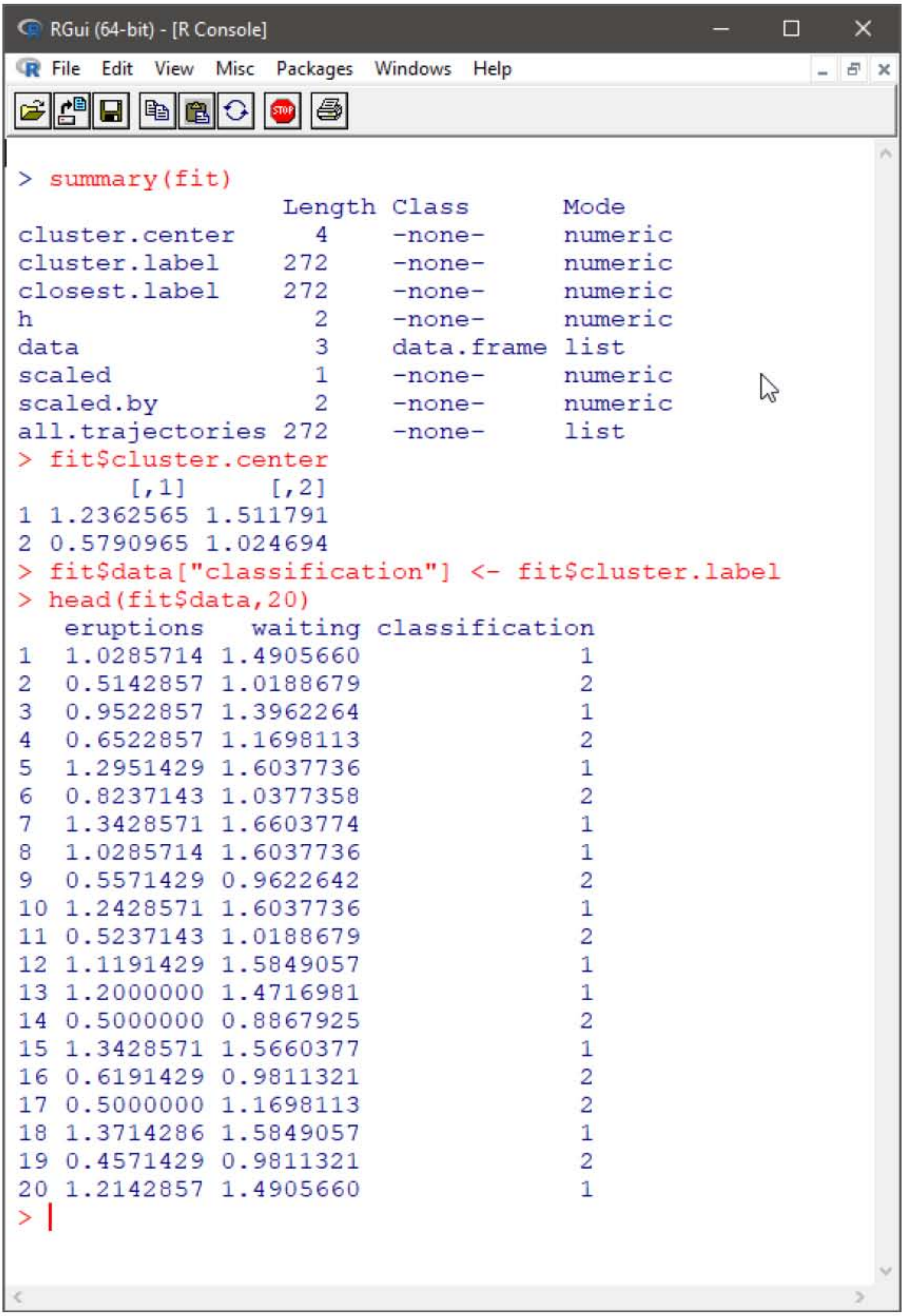
R 3.6.3

Pour traiter de ce sujet, nous allons utiliser le package **LPCM** et son jeu de données *faithful* et sa fonction **ms( )** suivant:

```
> library("LPCM")
> data(faithful)
> head(faithful)
  eruptions waiting
1     3.600      79
2     1.800      54
3     3.333      74
4     2.283      62
5     4.533      85
6     2.883      55
> fit <- ms(faithful, h=0.2)
> 
```



Et on peut faire des analyses supplémentaires sur le résultat:

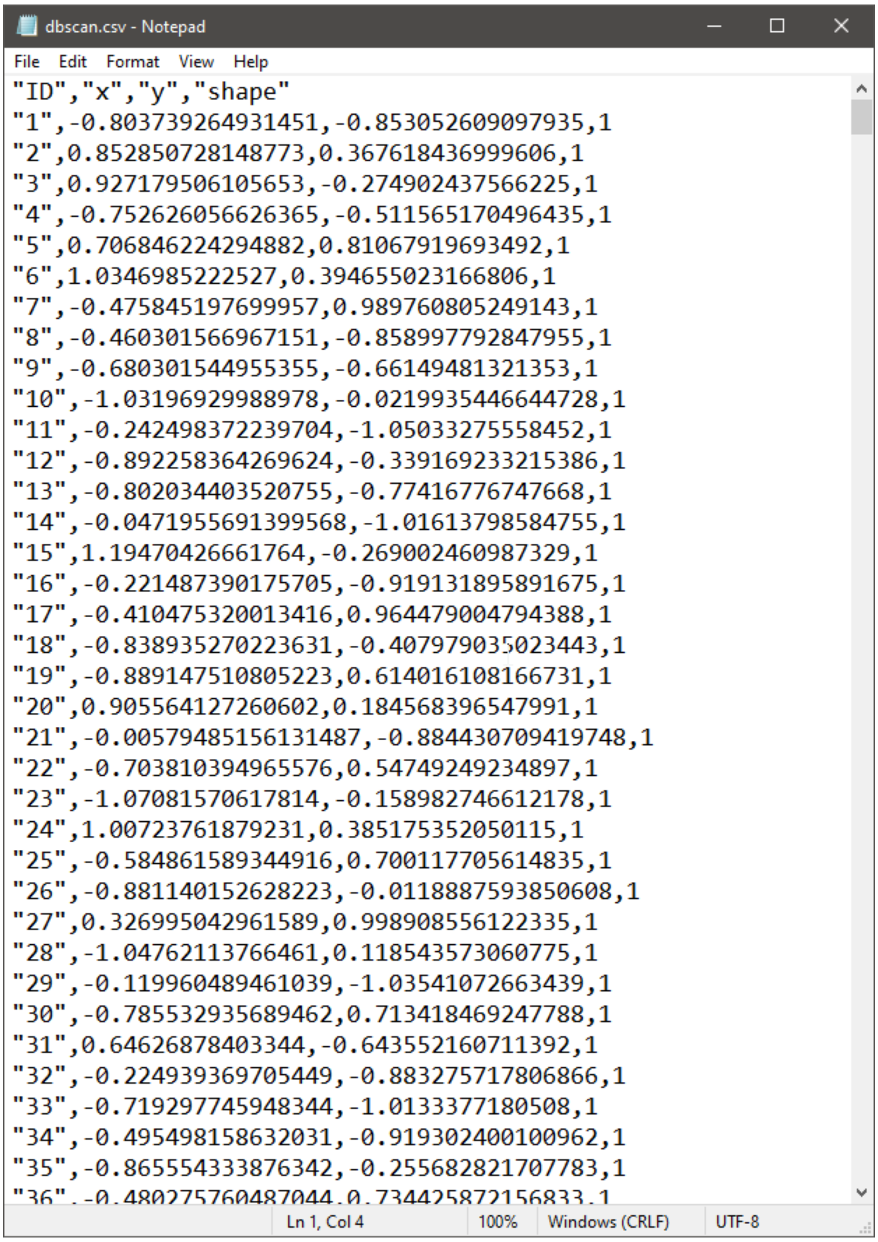

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> summary(fit)
                   Length Class   Mode
cluster.center   4   -none-   numeric
cluster.label   272   -none-   numeric
closest.label   272   -none-   numeric
h   2   -none-   numeric
data   3   data.frame list
scaled   1   -none-   numeric
scaled.by   2   -none-   numeric
all.trajectories   272   -none-   list
> fit$cluster.center
       [,1]   [,2]
1 1.2362565 1.511791
2 0.5790965 1.024694
> fit$data["classification"] <- fit$cluster.label
> head(fit$data,20)
   eruptions   waiting classification
1 1.0285714 1.4905660   1
2 0.5142857 1.0188679   2
3 0.9522857 1.3962264   1
4 0.6522857 1.1698113   2
5 1.2951429 1.6037736   1
6 0.8237143 1.0377358   2
7 1.3428571 1.6603774   1
8 1.0285714 1.6037736   1
9 0.5571429 0.9622642   2
10 1.2428571 1.6037736   1
11 0.5237143 1.0188679   2
12 1.1191429 1.5849057   1
13 1.2000000 1.4716981   1
14 0.5000000 0.8867925   2
15 1.3428571 1.5660377   1
16 0.6191429 0.9811321   2
17 0.5000000 1.1698113   2
18 1.3714286 1.5849057   1
19 0.4571429 0.9811321   2
20 1.2142857 1.4905660   1
>

Exemple 349.: Clustering DBSCAN

R 4.3.2

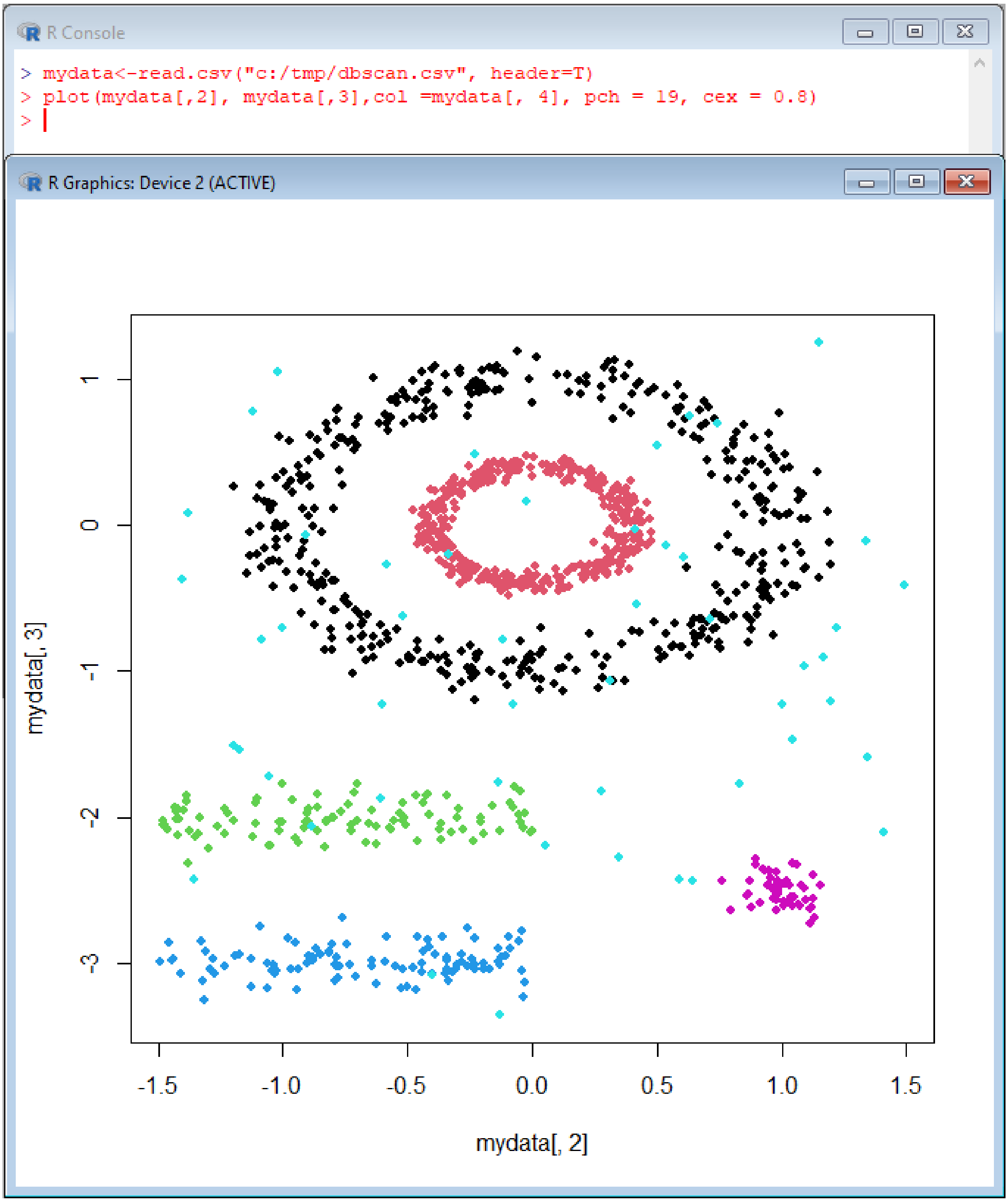
Nous avons u dans le cours théoriquement que le DBSCAN est un algorithem de clustering non-linéaire très simple. Nous allons voir ici comment le mettre en pratique.

Nous chargeons le jeu de données suivant:

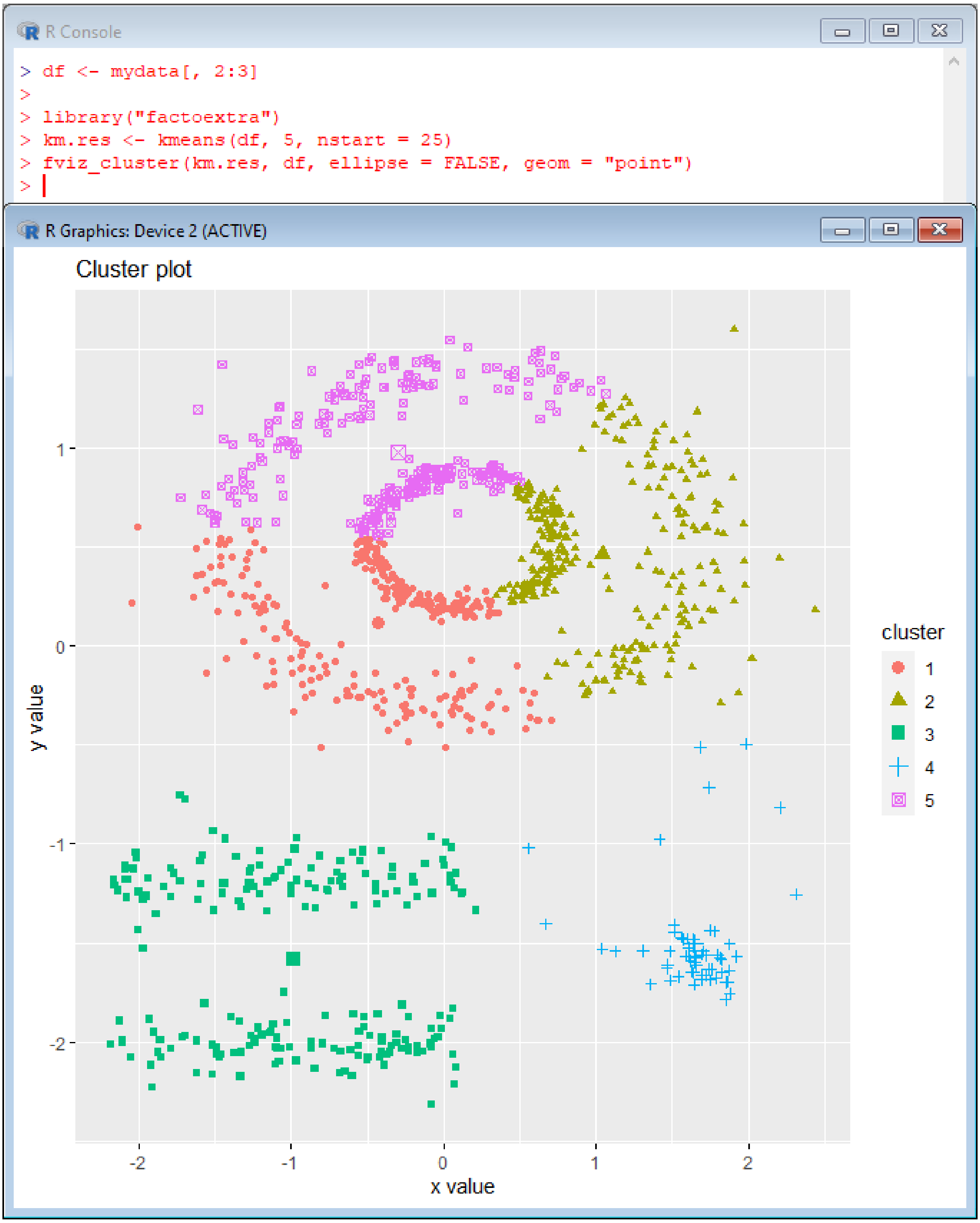


```
dbscan.csv - Notepad
File Edit Format View Help
"ID","x","y","shape"
"1",-0.803739264931451,-0.853052609097935,1
"2",0.852850728148773,0.367618436999606,1
"3",0.927179506105653,-0.27490243756625,1
"4",-0.752626056626365,-0.511565170496435,1
"5",0.706846224294882,0.81067919693492,1
"6",1.0346985222527,0.394655023166806,1
"7",-0.475845197699957,0.989760805249143,1
"8",-0.460301566967151,-0.858997792847955,1
"9",-0.680301544955355,-0.66149481321353,1
"10",-1.03196929988978,-0.0219935446644728,1
"11",-0.242498372239704,-1.05033275558452,1
"12",-0.892258364269624,-0.339169233215386,1
"13",-0.802034403520755,-0.77416776747668,1
"14",-0.0471955691399568,-1.01613798584755,1
"15",1.19470426661764,-0.269002460987329,1
"16",-0.221487390175705,-0.919131895891675,1
"17",-0.410475320013416,0.964479004794388,1
"18",-0.838935270223631,-0.407979035023443,1
"19",-0.889147510805223,0.614016108166731,1
"20",0.905564127260602,0.184568396547991,1
"21",-0.00579485156131487,-0.884430709419748,1
"22",-0.703810394965576,0.54749249234897,1
"23",-1.07081570617814,-0.158982746612178,1
"24",1.00723761879231,0.385175352050115,1
"25",-0.584861589344916,0.700117705614835,1
"26",-0.881140152628223,-0.0118887593850608,1
"27",0.326995042961589,0.998908556122335,1
"28",-1.04762113766461,0.118543573060775,1
"29",-0.119960489461039,-1.03541072663439,1
"30",-0.785532935689462,0.713418469247788,1
"31",0.64626878403344,-0.643552160711392,1
"32",-0.224939369705449,-0.883275717806866,1
"33",-0.719297745948344,-1.0133377180508,1
"34",-0.495498158632031,-0.919302400100962,1
"35",-0.865554333876342,-0.255682821707783,1
"36",-0.480275760487044,0.734425872156833,1
Ln 1, Col 4 100% Windows (CRLF) UTF-8
```

Voyons à quoi cela ressemble visuellement dans un premier temps:

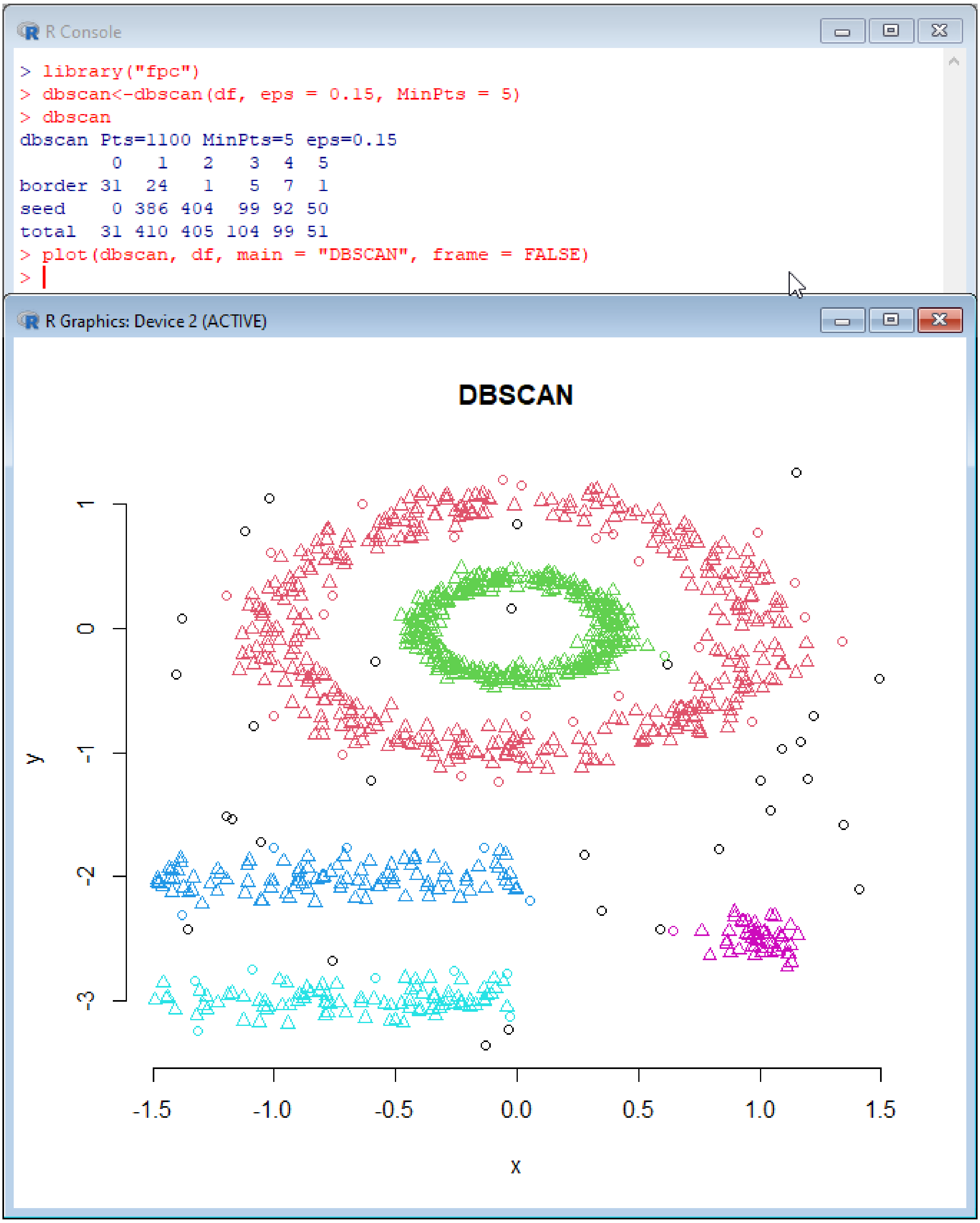


Faisons d'abord un clustering avec la méthode des k-means juste pour voir… :



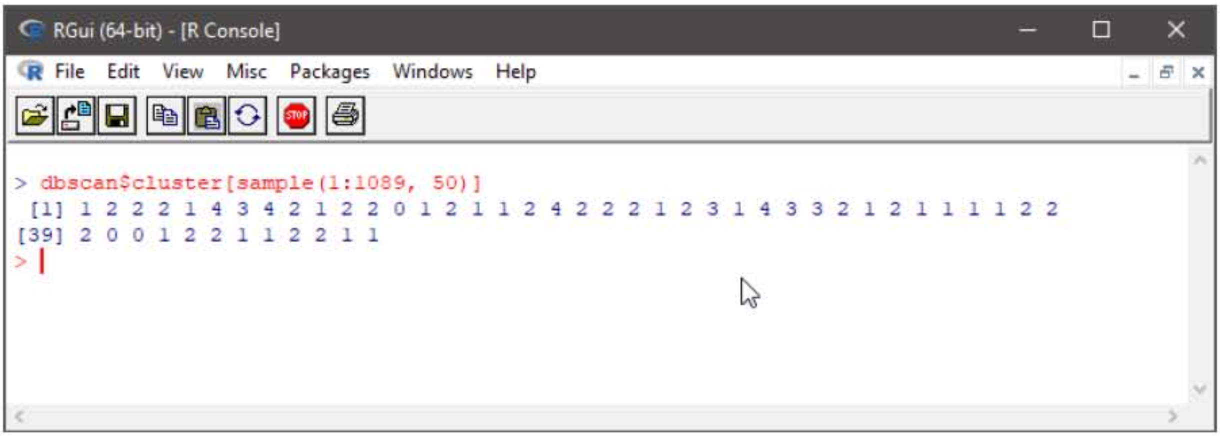
Bref vu la non-linéarité du jeu de données, nous ne sommes pas trop étonnés de la médiocrité du résultat.

Essayons maintenant avec la méthode DBSCAN:



C'est beaucoup mieux et ce sans surprises!

Nous pouvons aussi voir à quel cluster appartient quel ligne de notre jeu de données comme suit:

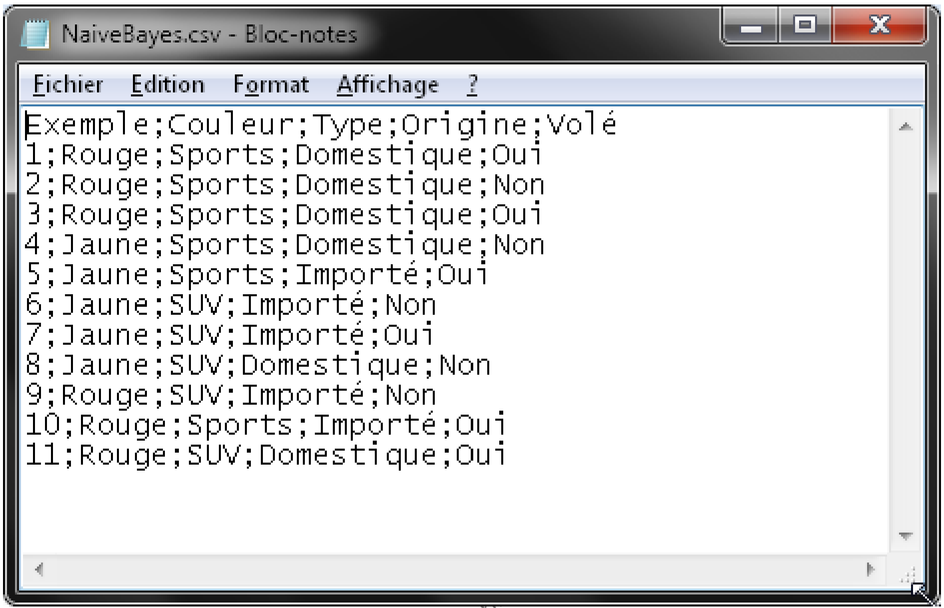


Exemple 350.: Classification bayésienne naïve

R 3.0.2

Classification bayésienne naïve binomiale

Nous allons ici encore et encore une fois utiliser les mêmes données que dans le cours théorique:



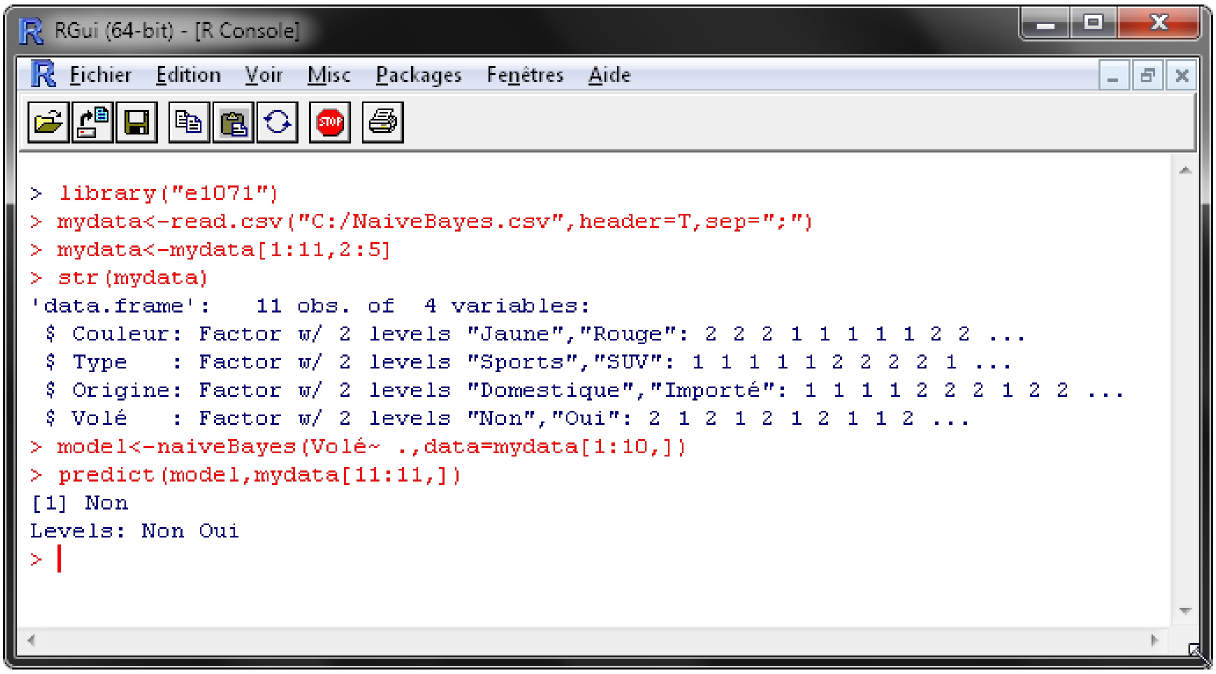
```
Exemple;Couleur;Type;Origine;Volé
1;Rouge;Sports;Domestique;Oui
2;Rouge;Sports;Domestique;Non
3;Rouge;Sports;Domestique;Oui
4;Jaune;Sports;Domestique;Non
5;Jaune;Sports;Importé;Oui
6;Jaune;SUV;Importé;Non
7;Jaune;SUV;Importé;Oui
8;Jaune;SUV;Domestique;Non
9;Rouge;SUV;Importé;Non
10;Rouge;Sports;Importé;Oui
11;Rouge;SUV;Domestique;Oui
```

et comparer le résultat avec ceux calculés à la main.

Pour cela, nous allons utiliser la commande **naiveBayes( )** du package **e1071**.

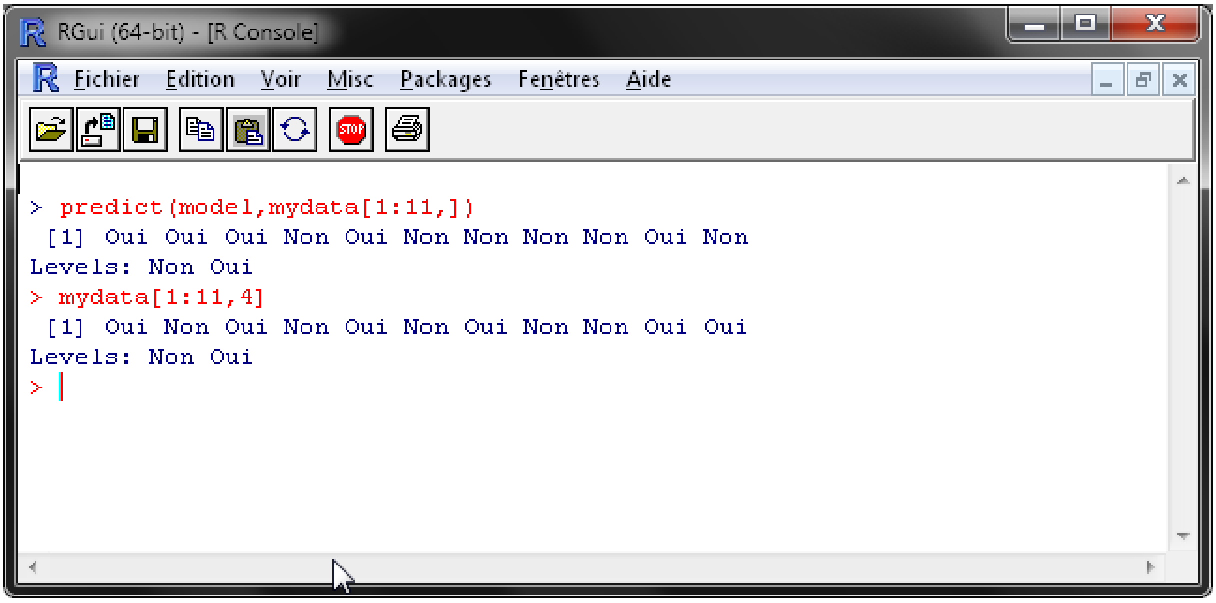
Attention!!! Le package **e1071**, ne s'installera pas dans R3.0.2 ou d'autres versions si vous ne spécifiez pas le serveur source comme étant l'*Autriche* (Austria) qui est le pays d'origine de ceux qui ont développé ce package.

Nous utilisons ensuite la commande **naiveBayes( )** pour entraîner le modèle et la commande **predict( )** pour le test sur la dernière ligne:



Nous obtenons donc le même résultat que dans le cours théorique.

Nous pouvons même faire un contrôle sur les données d'entraînement elles-mêmes:

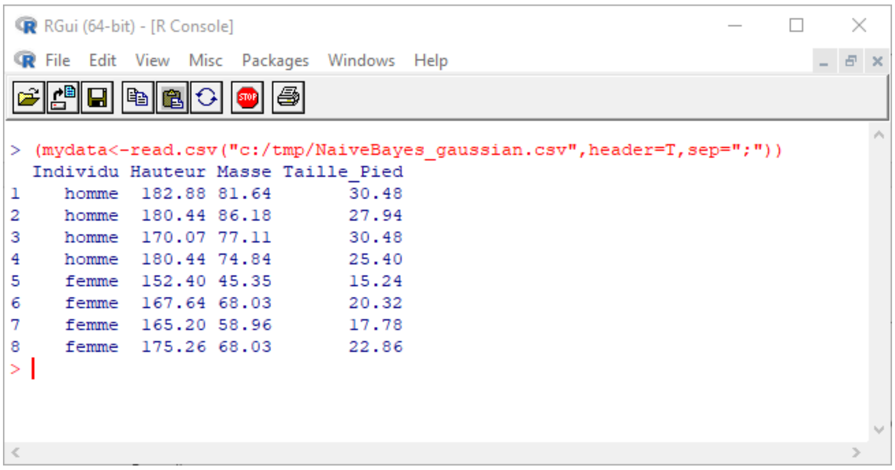


Nous réutiliserons cette fonction lors de notre étude du Texte Mining pour l'analyse de sentiments (voir page 2779).

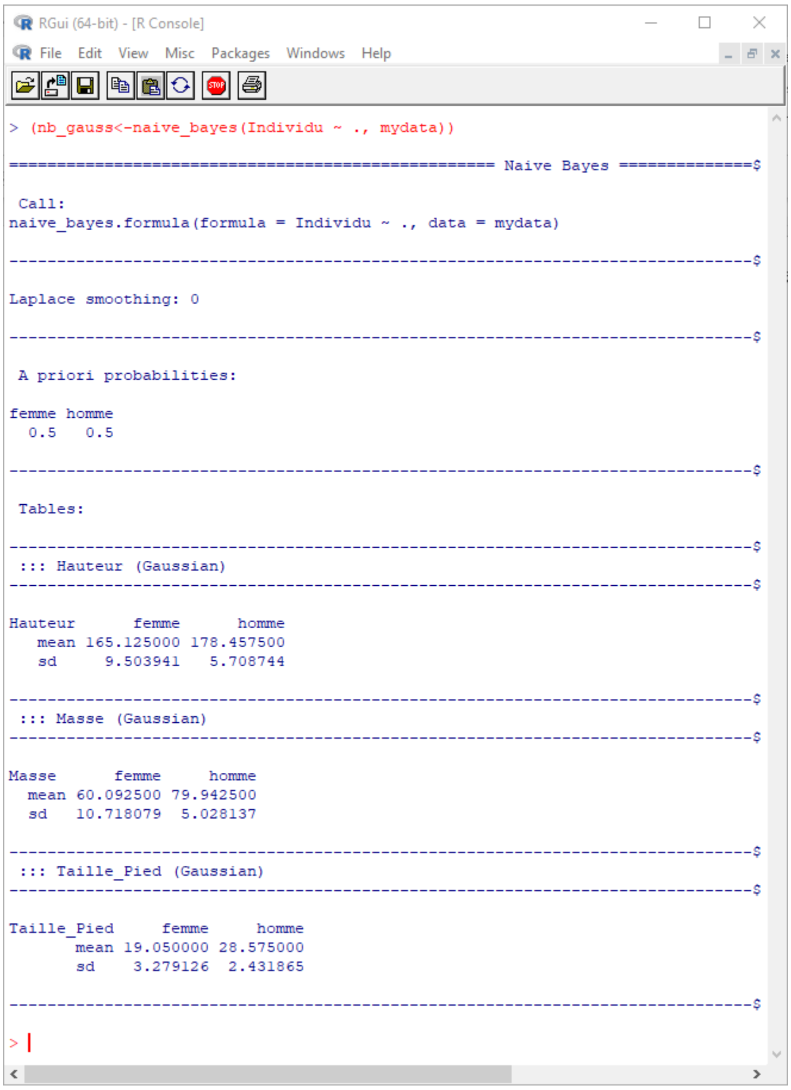
Donc comme nous pouvons le voir... un modèle sur une technique empirique reste une modèle et de plus avec aussi peu de données...

Classification bayésienne naïve gaussienne

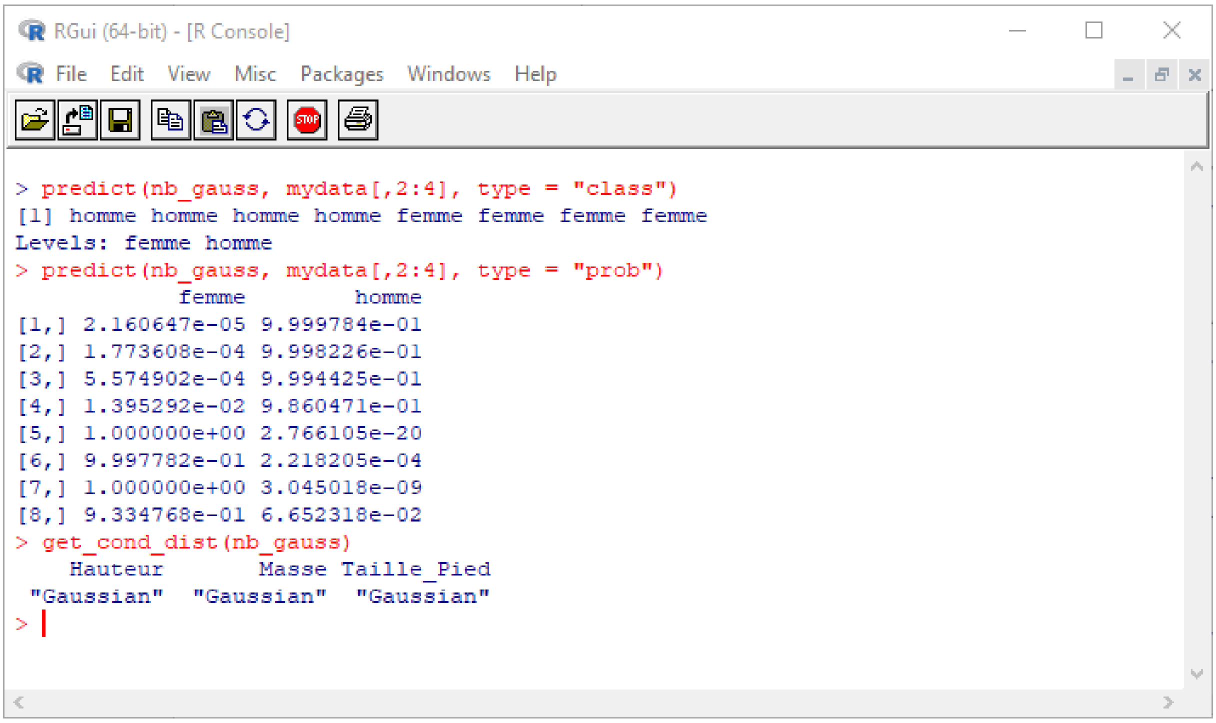
Nous allons ici encore et encore une fois utiliser les mêmes données que dans le cours théorique:



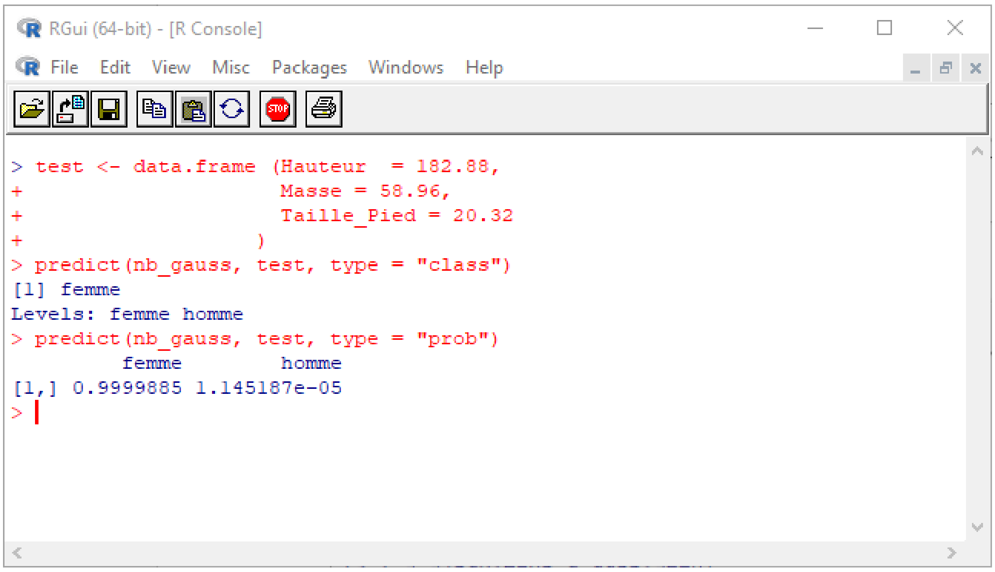
Et maintenant nous faisons usage du package **naivebayes** et de sa fonction naive_bayes( ):



On peut prédire maintenant sur le jeu de données initial pour le fun et obtenir les fonction de probabilités conditionnelles utilisées par le package:



Et mettons faisons cela sur notre individu test:

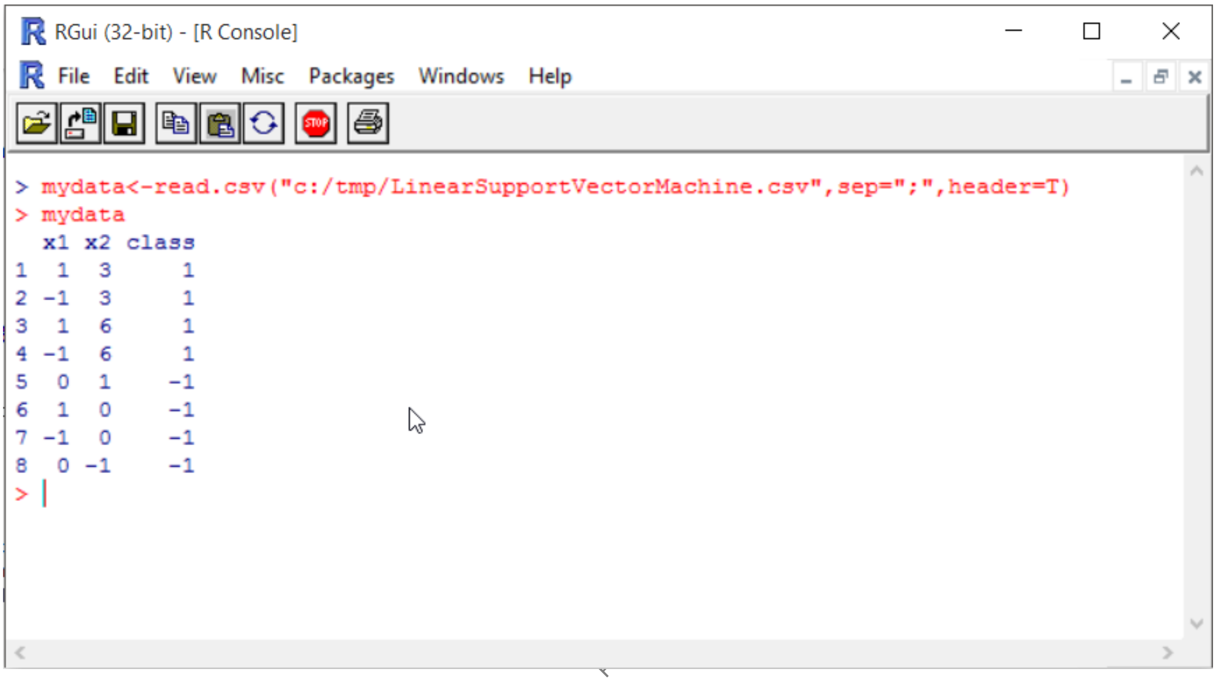


Nous retombons bien exactement sur le résultat obtenu dans le cours théorique!

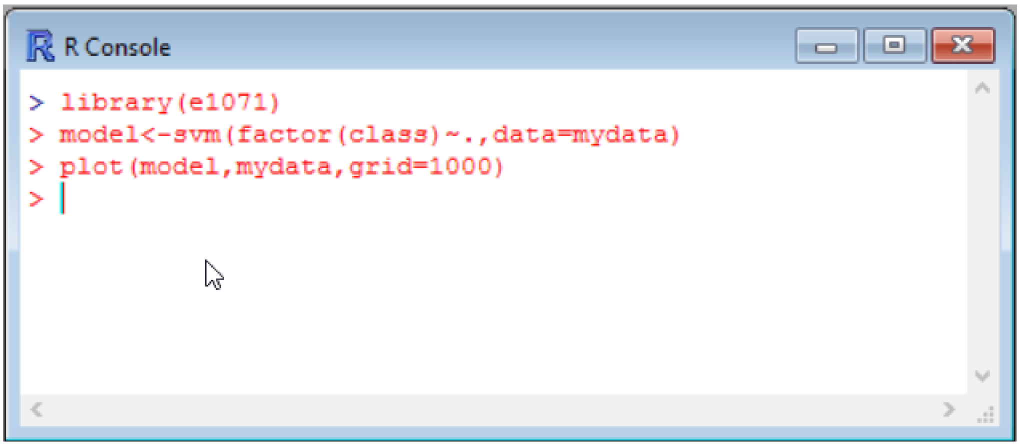
Exemple 351.: Classification SVM (Support Vector Machine)

R 3.2.1 x86

Nous allons commencer comme dans le cours théorique avec le cas linéaire. Les données seront donc les mêmes et dans ce cas particulier réduit à un exemple à deux dimensions seulement:



Ensuite nous utilisons la fonction **svm( )** du package e1701 pour créer le modèle:



Ce qui nous donne:

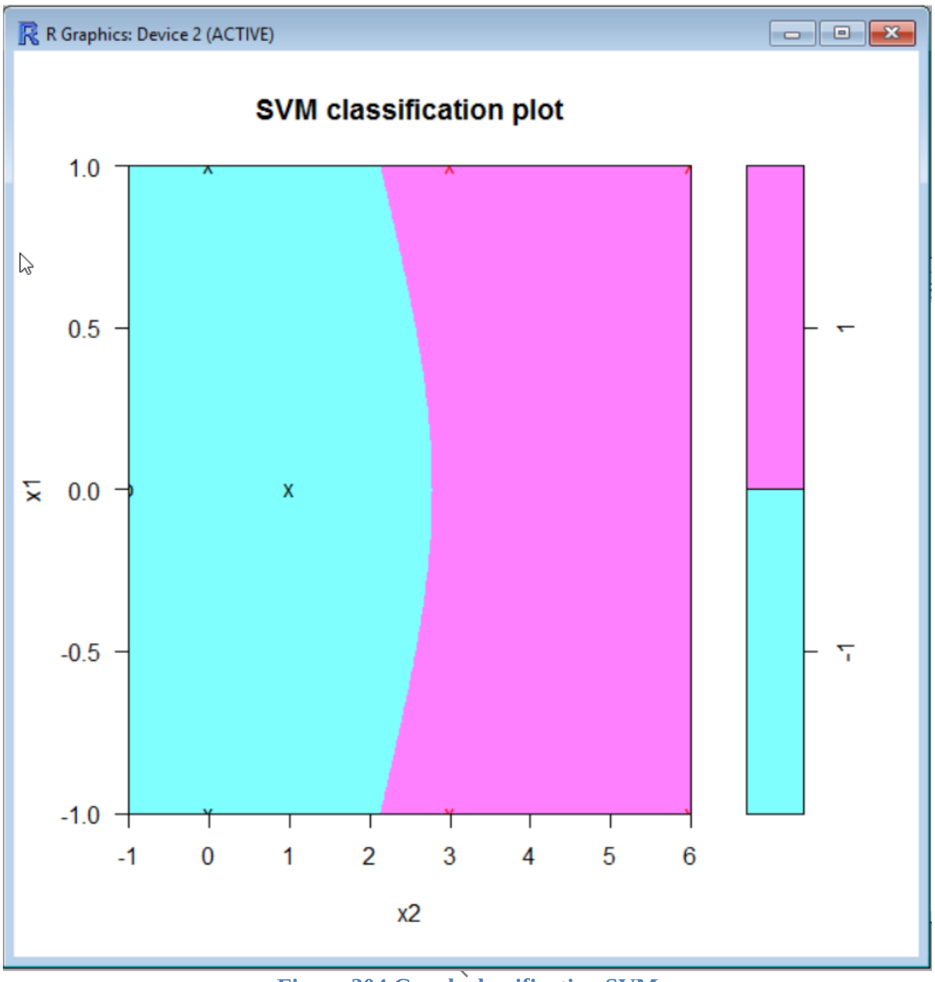
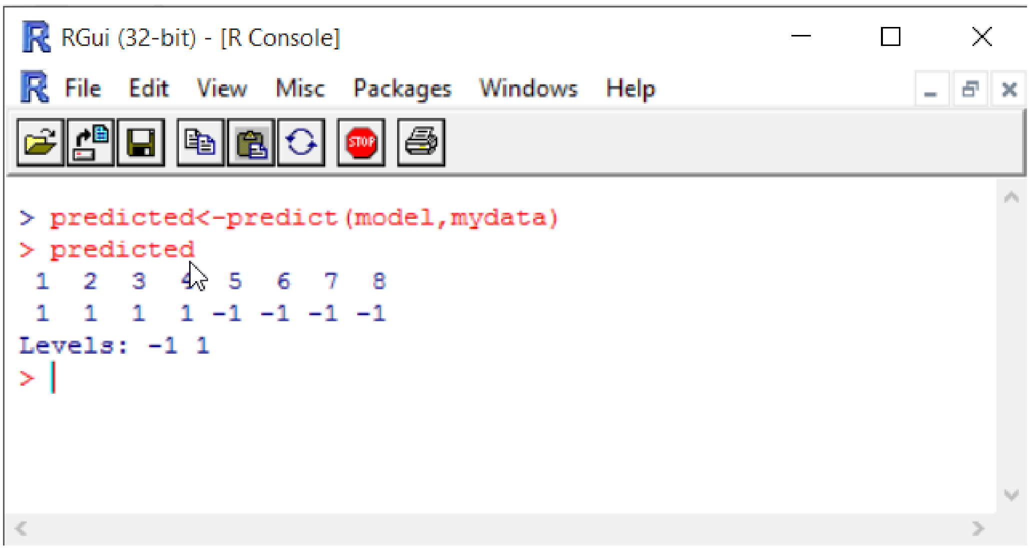


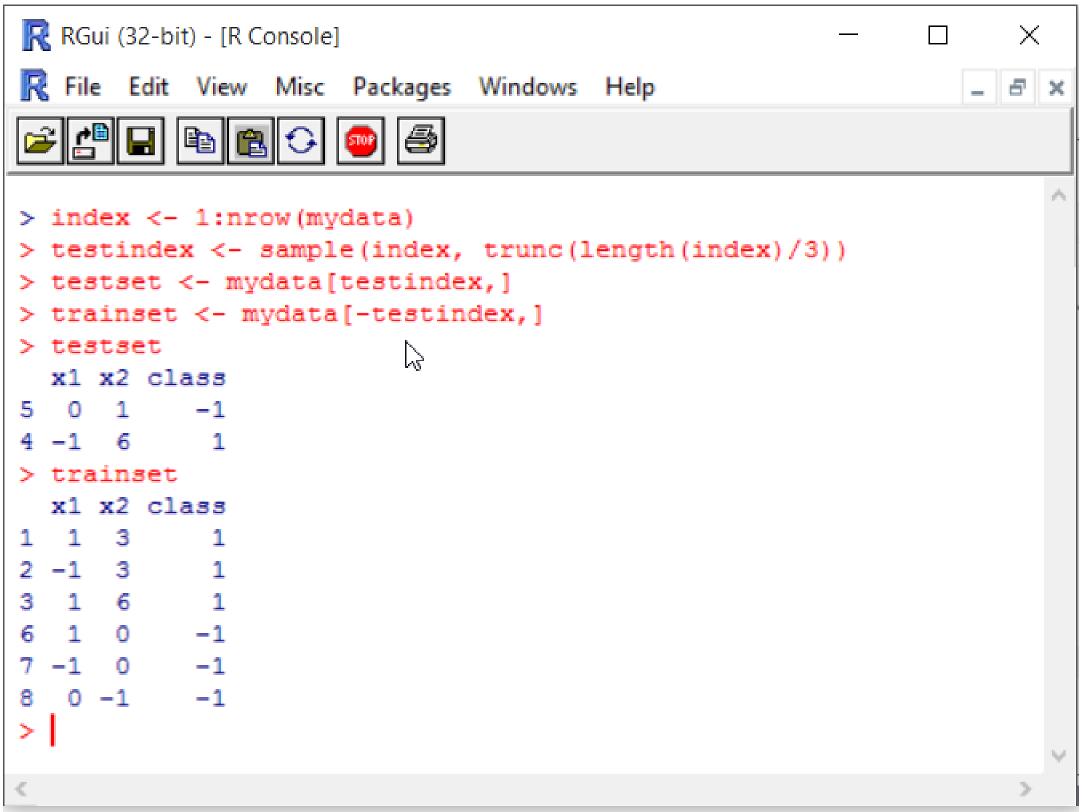
Figure 304 Graph classification SVM

Outre le fait que le résultat est esthétiquement mauvais (certains points sont sur la bordure, il n'y a pas de grille, ni d'axes…), nous remarquerons que par rapport au cours théorique la fonction intégrée prend immédiatement un modèle quadratique. Même si le résultat est toutefois OK il n'est pas possible à ce jour d'obtenir exactement le résultat du cours théorique étant donné qu'il n'existe pas d'option pour forcer un modèle linéaire simple.

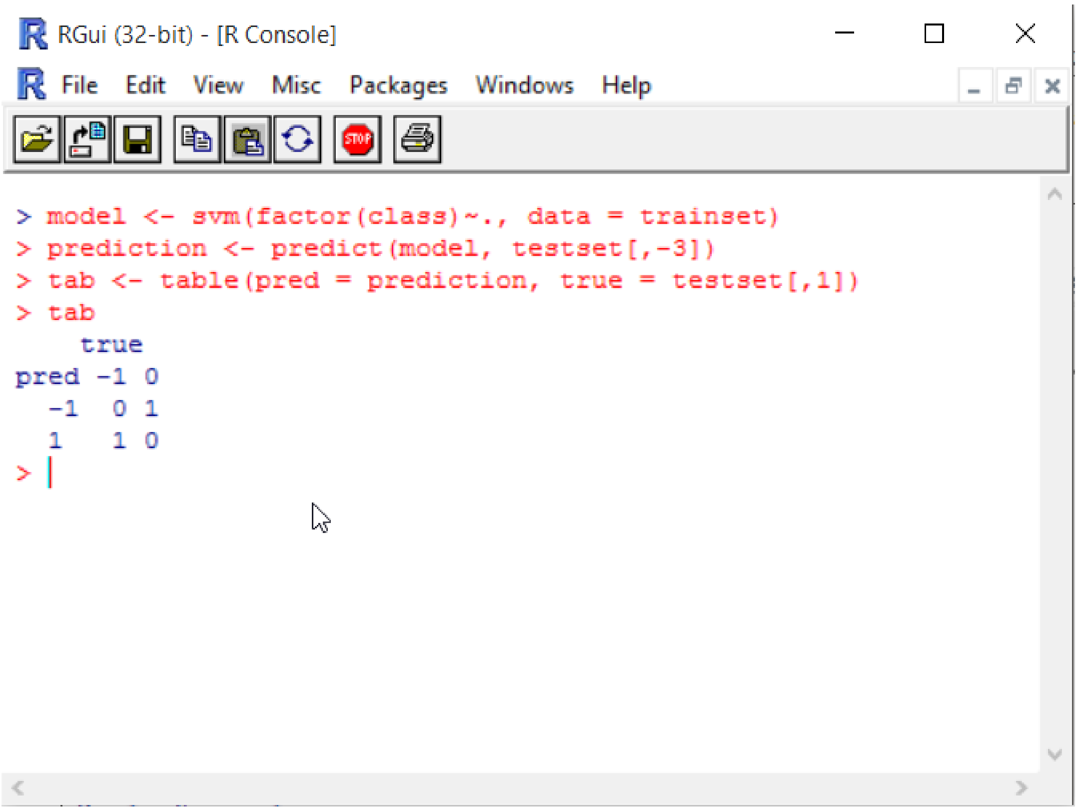
Nous pouvons avec la fonction **predicted( )** vérifier le résultat obtenu:



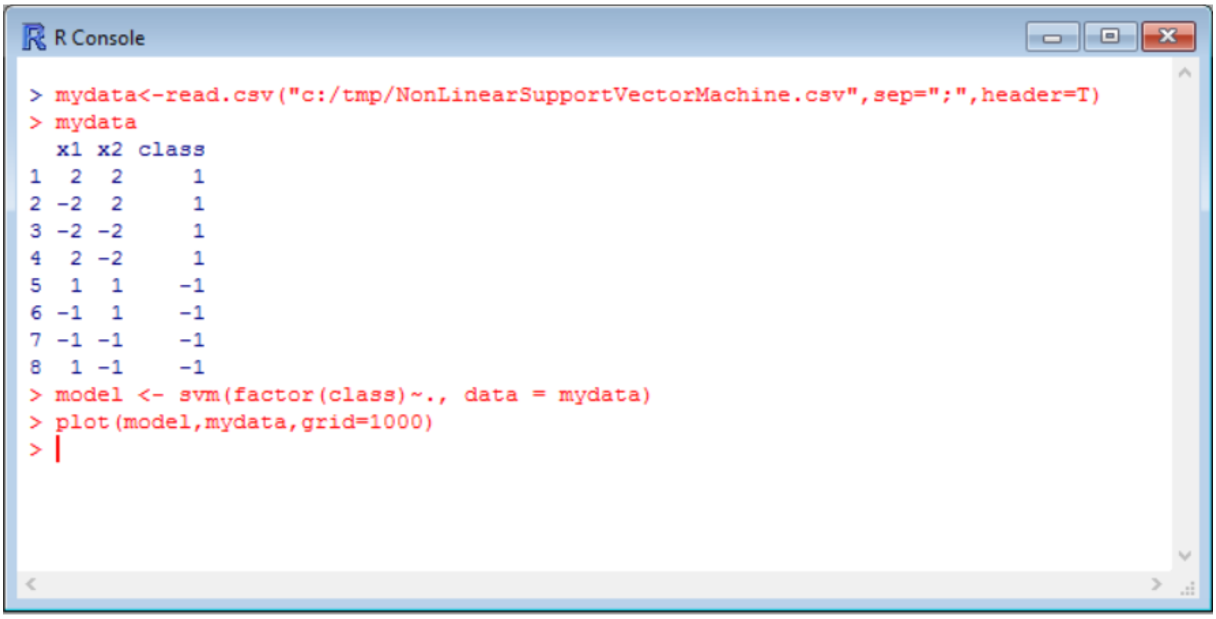
Nous pouvons maintenant avoir une approche plus "Data Mining" avec un jeu d'entraînement et un jeu de test:



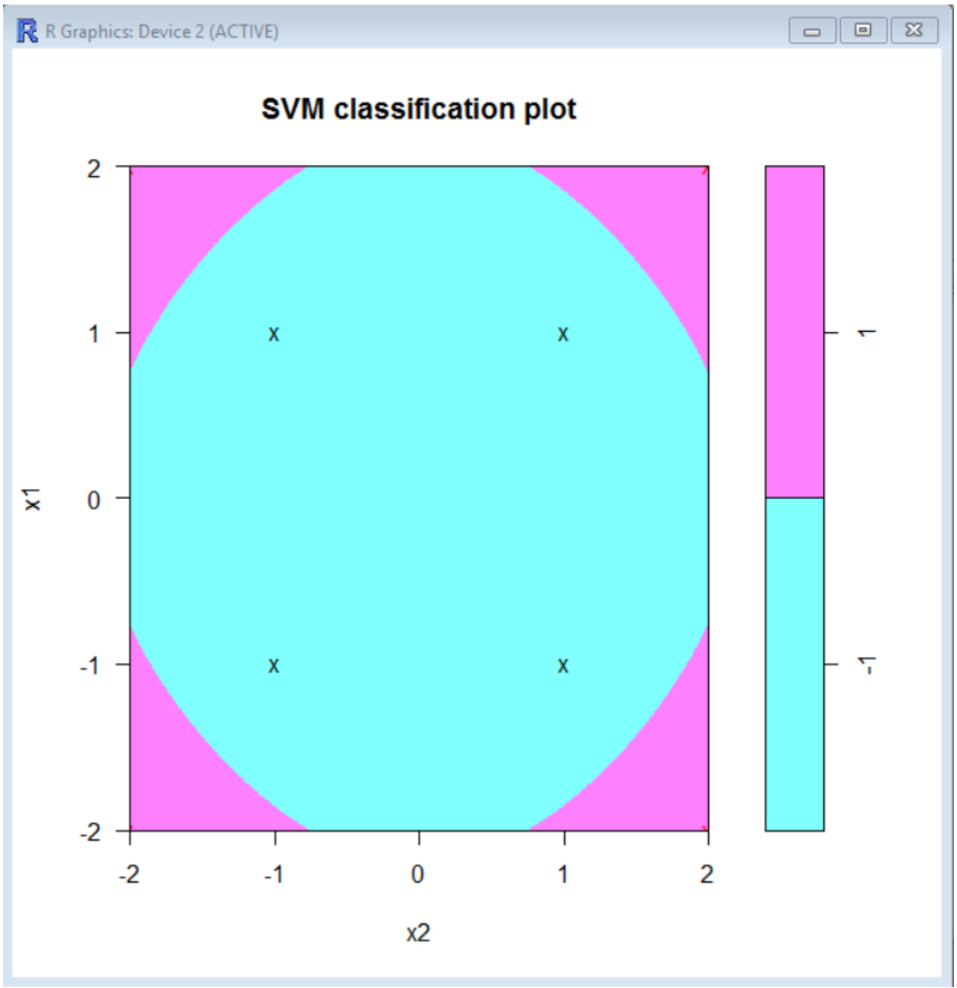
Et nous avons donc dans notre cas présent les commandes suivantes pour notre mini matrice de confusion (ben oui étant donné la taille du jeu de test… les valeurs qui sont dedans ne sont pas bien grandes):



Nous pouvons faire la même chose avec le jeu de données non linéaire du cours théorique:



Ce qui donne:



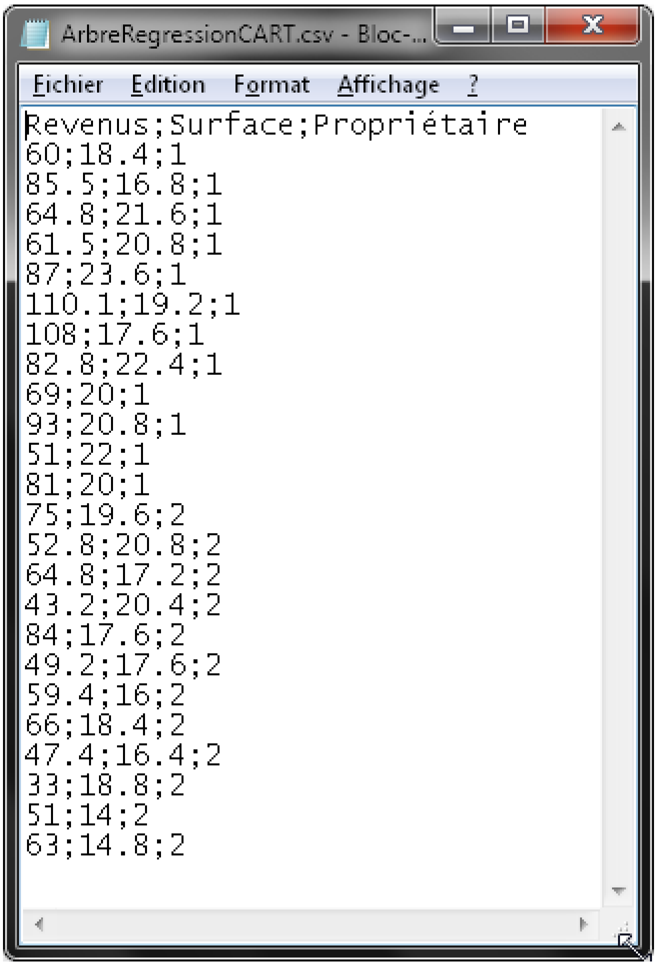
Il est donc dommage que nous n'ayons pas automatique un plot de l'espace de projection tel que vue dans le cours théorique. Sinon nous voyons aussi que la fonction choisie n'est pas la même que dans le cours théorique mais que le résultat test toutefois le même.

Exemple 352.: Classification CART (arbres de régression et de classification)

R 3.0.2

Arbre de classification

Nous allons ici encore et encore une fois utiliser les mêmes données que dans le cours théorique:



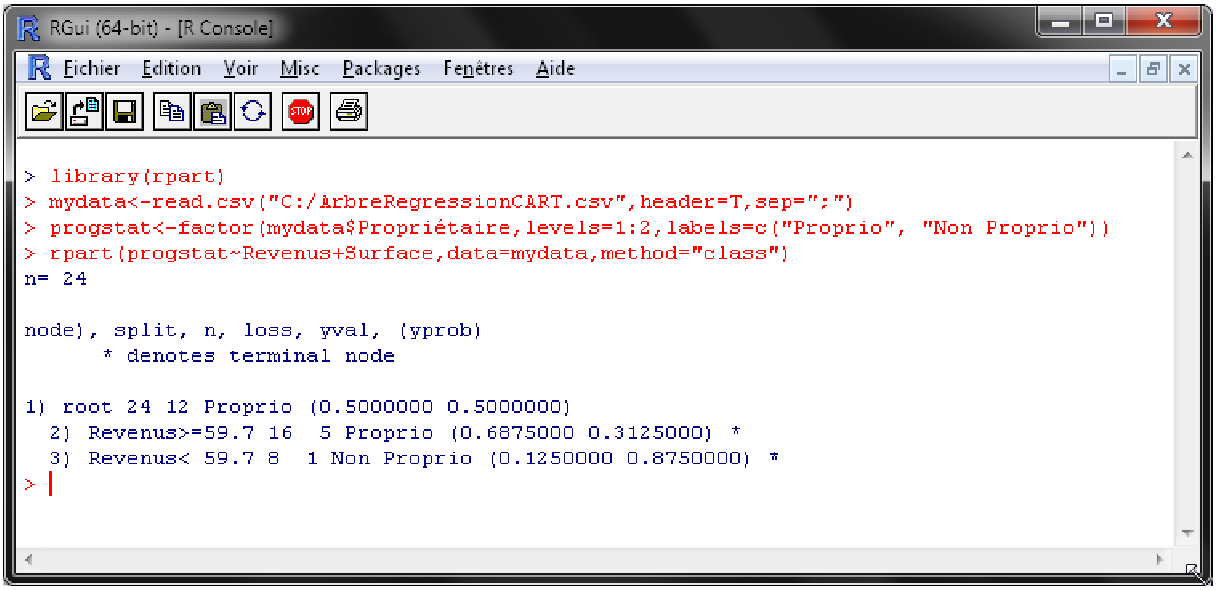
```
Revenus;Surface;Propriétaire
60;18.4;1
85.5;16.8;1
64.8;21.6;1
61.5;20.8;1
87;23.6;1
110.1;19.2;1
108;17.6;1
82.8;22.4;1
69;20;1
93;20.8;1
51;22;1
81;20;1
75;19.6;2
52.8;20.8;2
64.8;17.2;2
43.2;20.4;2
84;17.6;2
49.2;17.6;2
59.4;16;2
66;18.4;2
47.4;16.4;2
33;18.8;2
51;14;2
63;14.8;2
```

et comparer le résultat avec ceux calculés à la main et obtenus avec Tanagra.

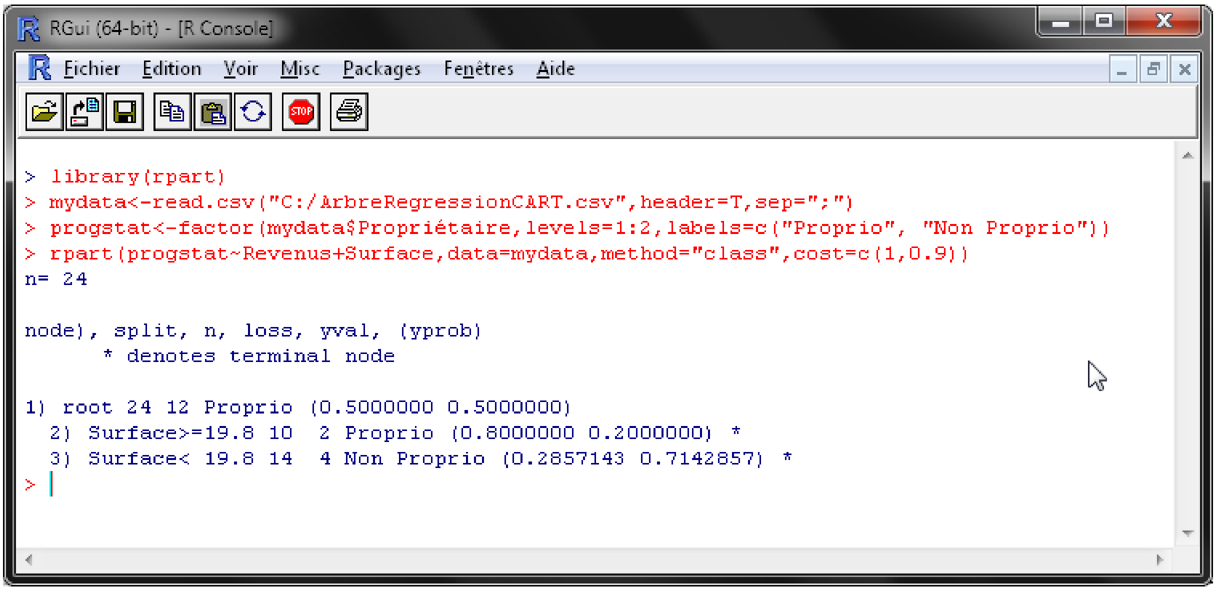
Rappel: Les techniques de classification (segmentations) par arbres sont souvent utilisées en marketing pour analyser les cibles répondant le mieux à certaines campagnes marketing. Les praticiens du marketing appellent cela des "modèles uplift" et certains statisticiens des "partionnement récursif".

Pour cela, nous allons utiliser la commande **rpart( )** du package **rpart** avec des paramètres qui comme nous allons le voir sont loin d'être triviaux à deviner.

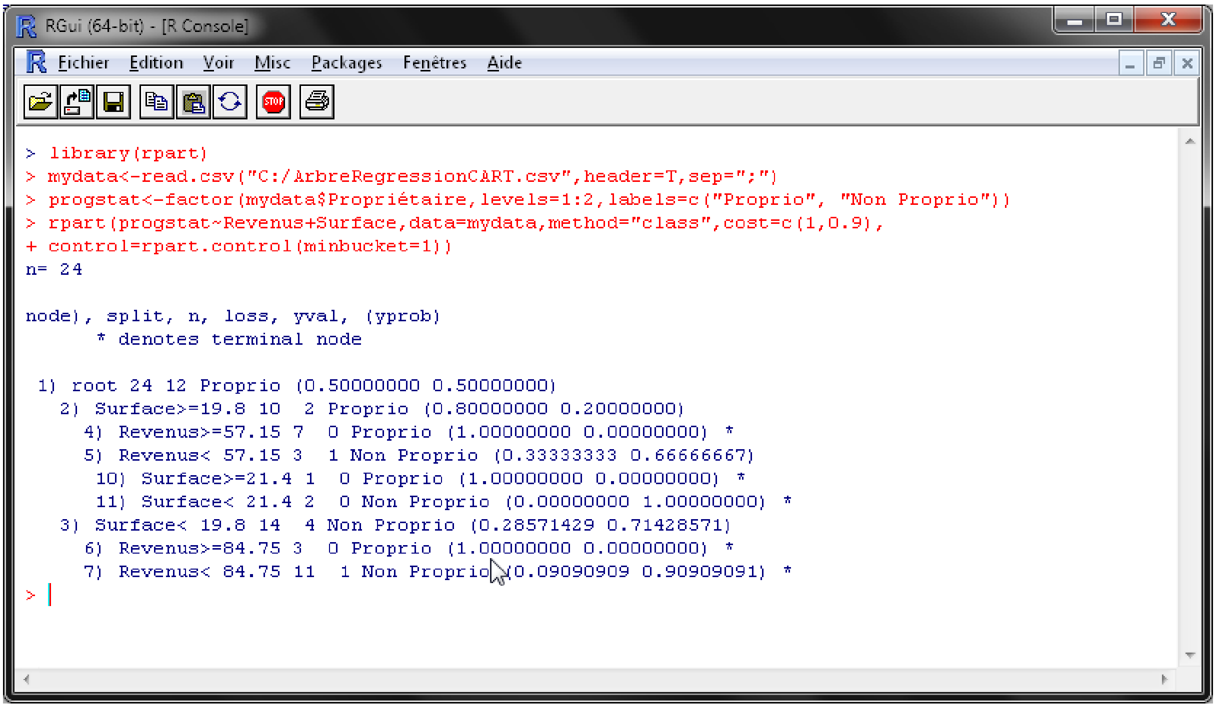
D'abord nous indiquons le minimum syndical:



Comme nous pouvons le voir, le résultat n'est vraiment pas intuitif si nous comparons avec ce que nous avons fait dans le cours théorique. Comme c'est choquant, nous allons donner un poids plus grand à la surface qu'aux revenus:



et là c'est parfait!!! Par contre l'arbre s'arrête bien trop vite. Nous allons donc chercher à l'étendre:



et là encore c'est parfait!!!

Pour information, *minbucket*=1 signifie que l'algorithme s'arrête s'il y a moins d'une feuille dans la branche. Si vous utilisez le paramètre *minsplit*=10 (pas présent dans l'exemple ci-dessus) cela signifie que l'algorithme ne coupera pas une branche s'il y a moins de 10 éléments dedans.

Mais nous avons vu dans le cours théorique que nous pouvons creuser l'arbre encore plus. Alors qu'en est-il avec ce package? Essayons:

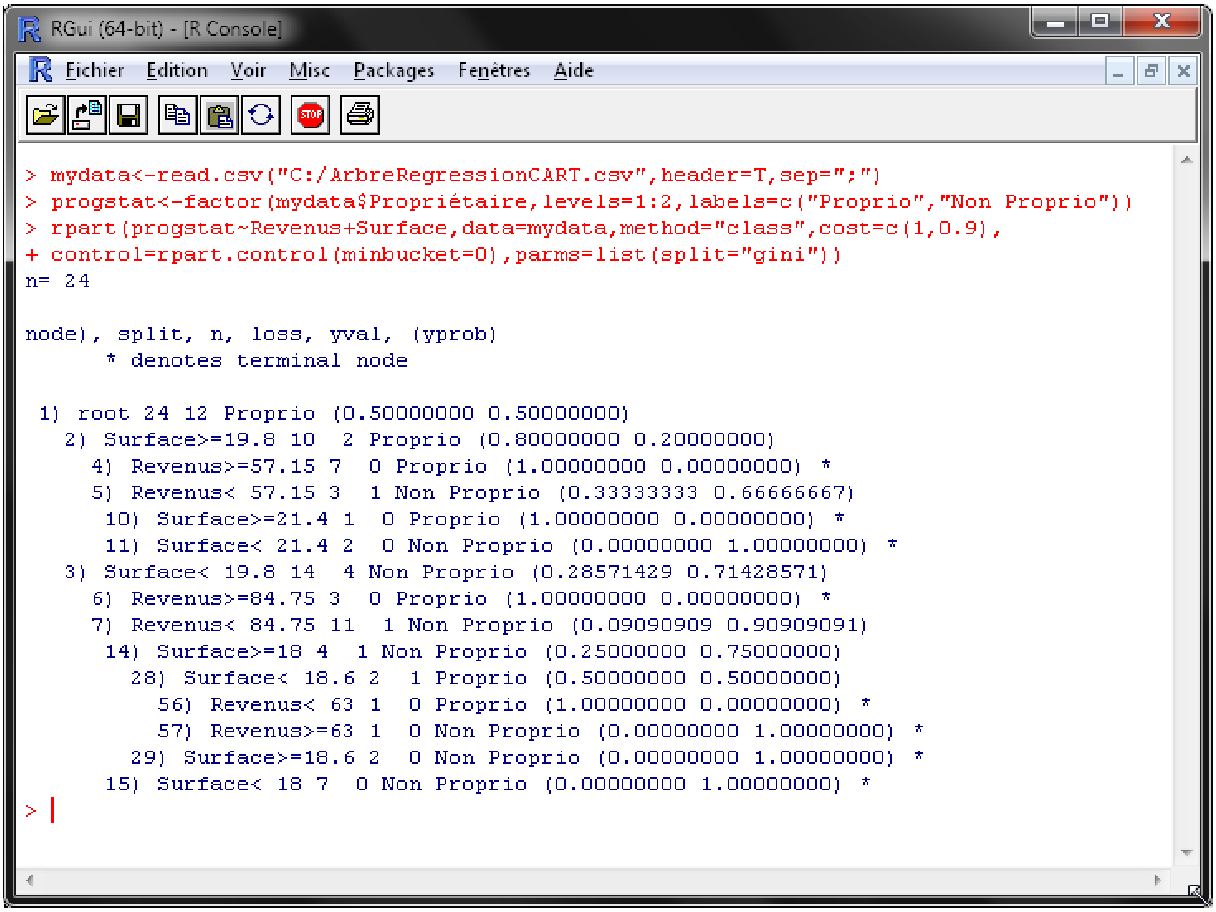
```
> mydata<-read.csv("C:/ArbreRegressionCART.csv",header=T,sep=";")
> progstat<-factor(mydata$Propriétaire,levels=1:2,labels=c("Proprio", "Non Proprio"))
> rpart(progstat~Revenus+Surface,data=mydata,method="class",cost=c(1,0.9),
+ control=rpart.control(minbucket=0))
n= 24

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 24 12 Proprio (0.50000000 0.50000000)
   2) Surface>=19.8 10  2 Proprio (0.80000000 0.20000000)
     4) Revenus>=57.15 7  0 Proprio (1.00000000 0.00000000) *
     5) Revenus< 57.15 3  1 Non Proprio (0.33333333 0.66666667)
      10) Surface>=21.4 1  0 Proprio (1.00000000 0.00000000) *
      11) Surface< 21.4 2  0 Non Proprio (0.00000000 1.00000000) *
   3) Surface< 19.8 14  4 Non Proprio (0.28571429 0.71428571)
     6) Revenus>=84.75 3  0 Proprio (1.00000000 0.00000000) *
     7) Revenus< 84.75 11  1 Non Proprio (0.09090909 0.90909091)
      14) Surface>=18 4  1 Non Proprio (0.25000000 0.75000000)
        28) Surface< 18.6 2  1 Proprio (0.50000000 0.50000000)
          56) Revenus< 63 1  0 Proprio (1.00000000 0.00000000) *
          57) Revenus>=63 1  0 Non Proprio (0.00000000 1.00000000) *
        29) Surface>=18.6 2  0 Non Proprio (0.00000000 1.00000000) *
      15) Surface< 18 7  0 Non Proprio (0.00000000 1.00000000) *
>
```

Que céééé beau! Nous obtenons donc la même chose que dans le cours théorique et avec un niveau de détail supérieur à celui de Tanagra.

Remarquons que la commande **rpart( )** utilise bien par défaut l'indicateur de Gini pour vu dans le cours théorique séparer les données. Effectivement, si nous les spécifions:



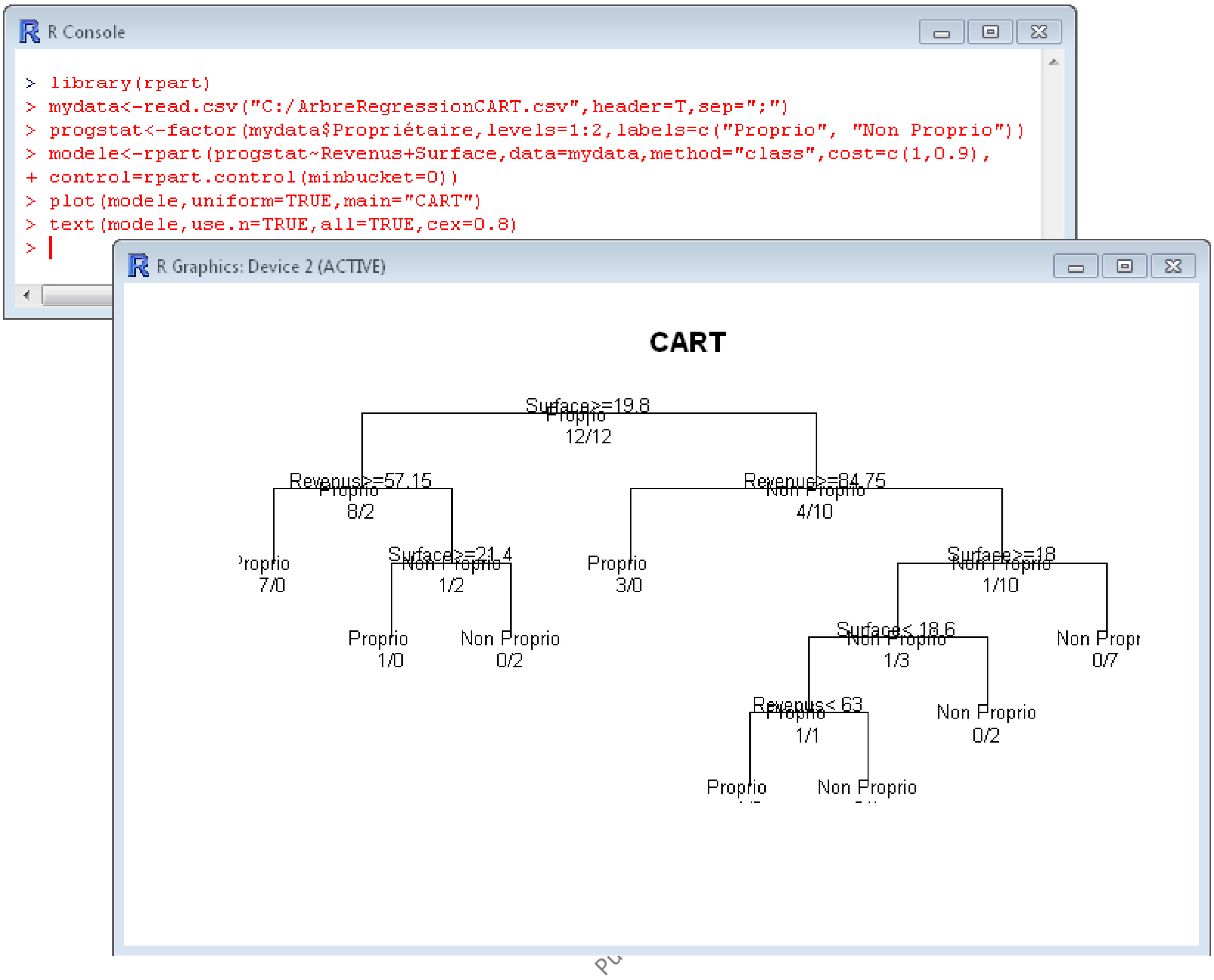
```
> mydata<-read.csv("C:/ArbreRegressionCART.csv",header=T,sep=";")
> progstat<-factor(mydata$Propriétaire,levels=1:2,labels=c("Proprio","Non Proprio"))
> rpart(progstat~Revenus+Surface,data=mydata,method="class",cost=c(1,0.9),
+ control=rpart.control(minbucket=0),parms=list(split="gini"))
n= 24

node), split, n, loss, yval, (yprob)
      * denotes terminal node

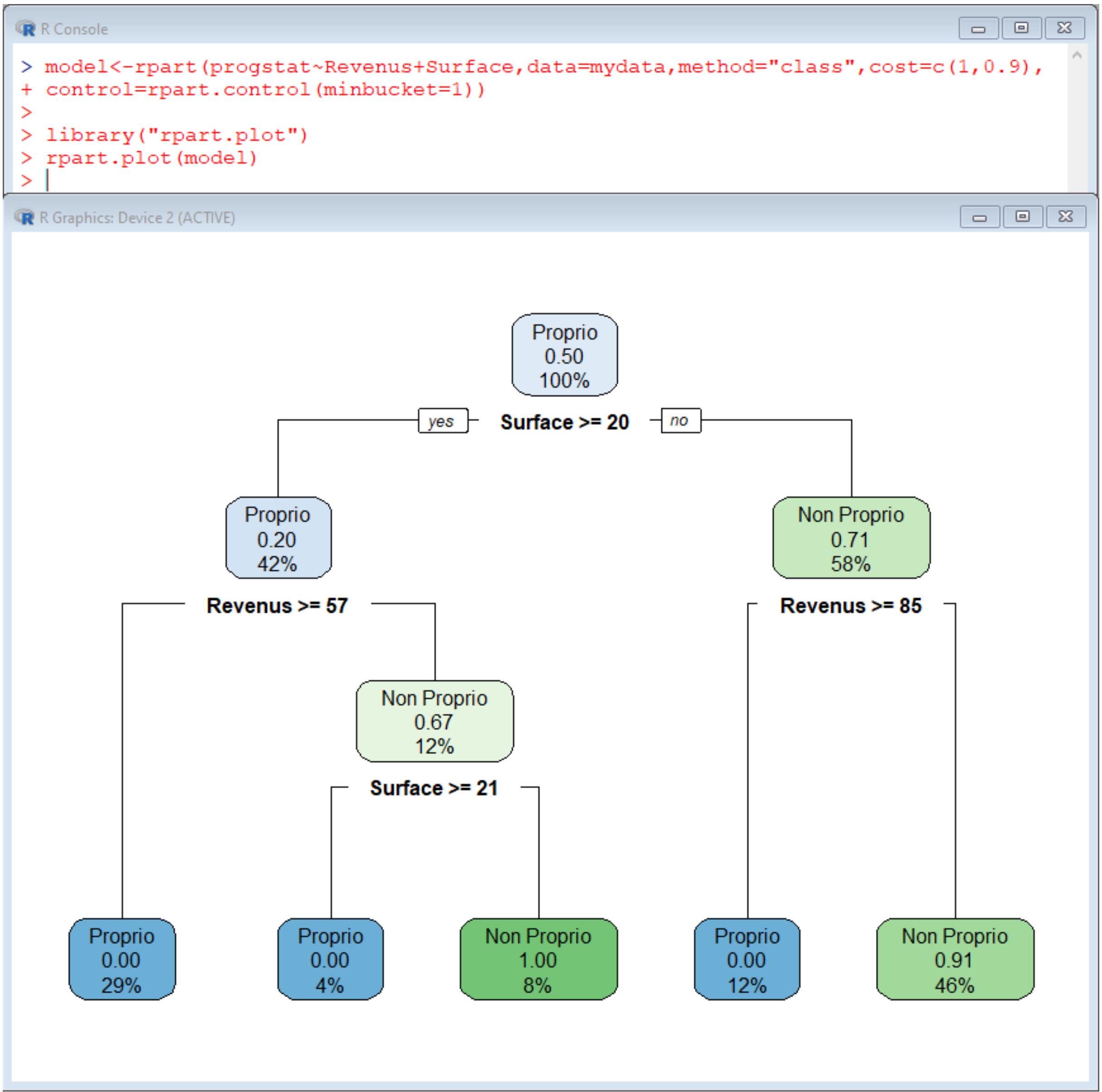
 1) root 24 12 Proprio (0.50000000 0.50000000)
   2) Surface>=19.8 10  2 Proprio (0.80000000 0.20000000)
     4) Revenus>=57.15 7  0 Proprio (1.00000000 0.00000000) *
     5) Revenus< 57.15 3  1 Non Proprio (0.33333333 0.66666667)
      10) Surface>=21.4 1  0 Proprio (1.00000000 0.00000000) *
      11) Surface< 21.4 2  0 Non Proprio (0.00000000 1.00000000) *
   3) Surface< 19.8 14  4 Non Proprio (0.28571429 0.71428571)
     6) Revenus>=84.75 3  0 Proprio (1.00000000 0.00000000) *
     7) Revenus< 84.75 11  1 Non Proprio (0.09090909 0.90909091)
      14) Surface>=18 4  1 Non Proprio (0.25000000 0.75000000)
        28) Surface< 18.6 2  1 Proprio (0.50000000 0.50000000)
          56) Revenus< 63 1  0 Proprio (1.00000000 0.00000000) *
          57) Revenus>=63 1  0 Non Proprio (0.00000000 1.00000000) *
        29) Surface>=18.6 2  0 Non Proprio (0.00000000 1.00000000) *
      15) Surface< 18 7  0 Non Proprio (0.00000000 1.00000000) *
>
```

Nous obtenons bien les mêmes données.

Nous pouvons faire un graph du résultat contrairement à Tanagra:



Sinon avec le package **rpart.plot** et sa fonction **rpart.plot( )** nous pouvons obtenir quelque chose de très élégant:

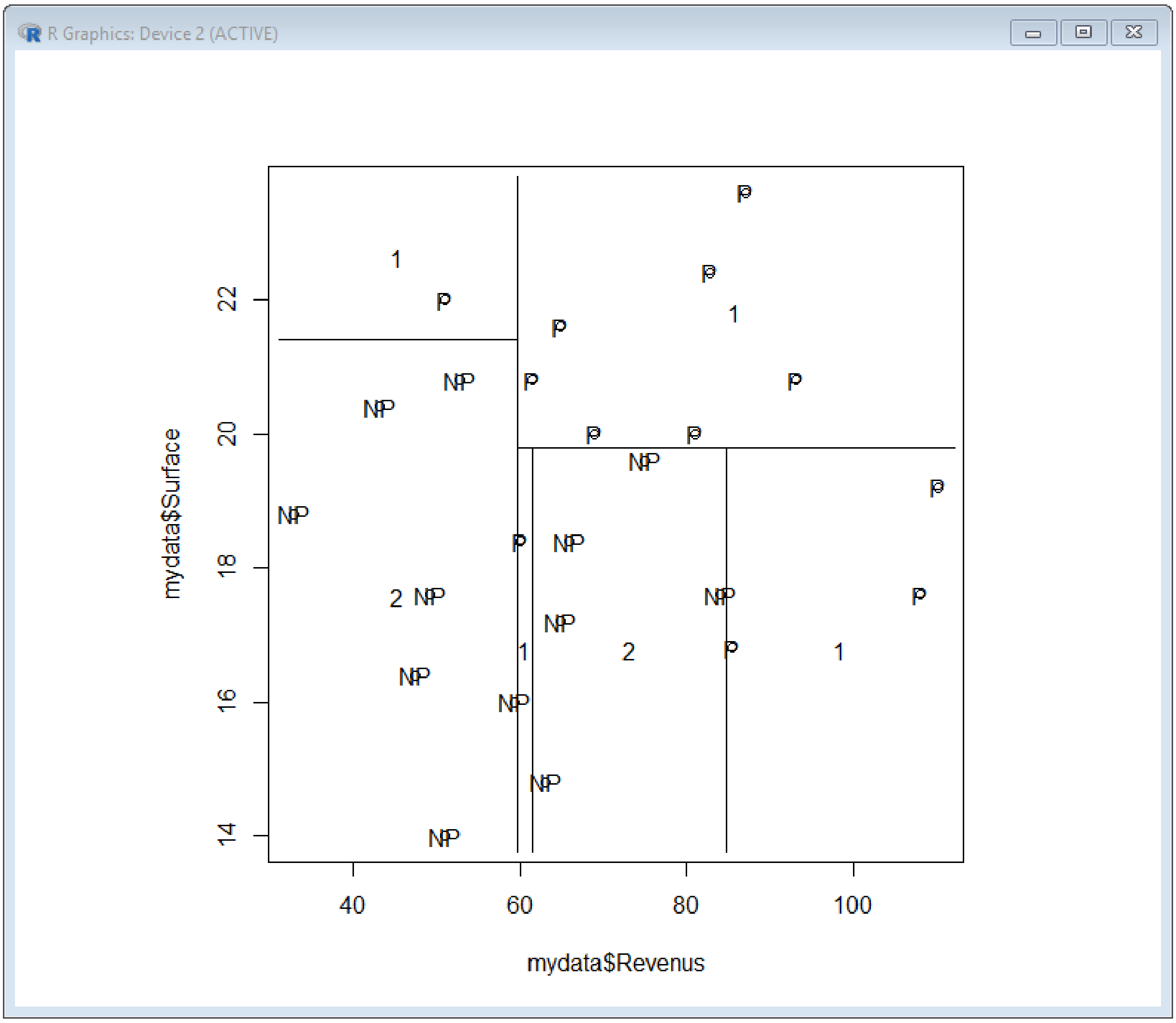


Cependant ce serait sympathique de réobtenir le même résultat visuel que dans le cours théorique avec le graph 2D dont l'aire est partitionnée, nous pouvons obtenir cela avec le package **tree** et ses fonctions **tree( )** et **partition.tree( )** mais qui cependant n'offrent pas (a priori!) le même niveau de flexibilité que la fonction **rpart( )** utilisée précédemment, ce qui fait que nous n'arriverons pas à obtenir le même résultat que dans le cours théorique (toutefois le résultat correspond bien à celui retourné par le premier exemple où nous avons utilisé **rpart( )**):

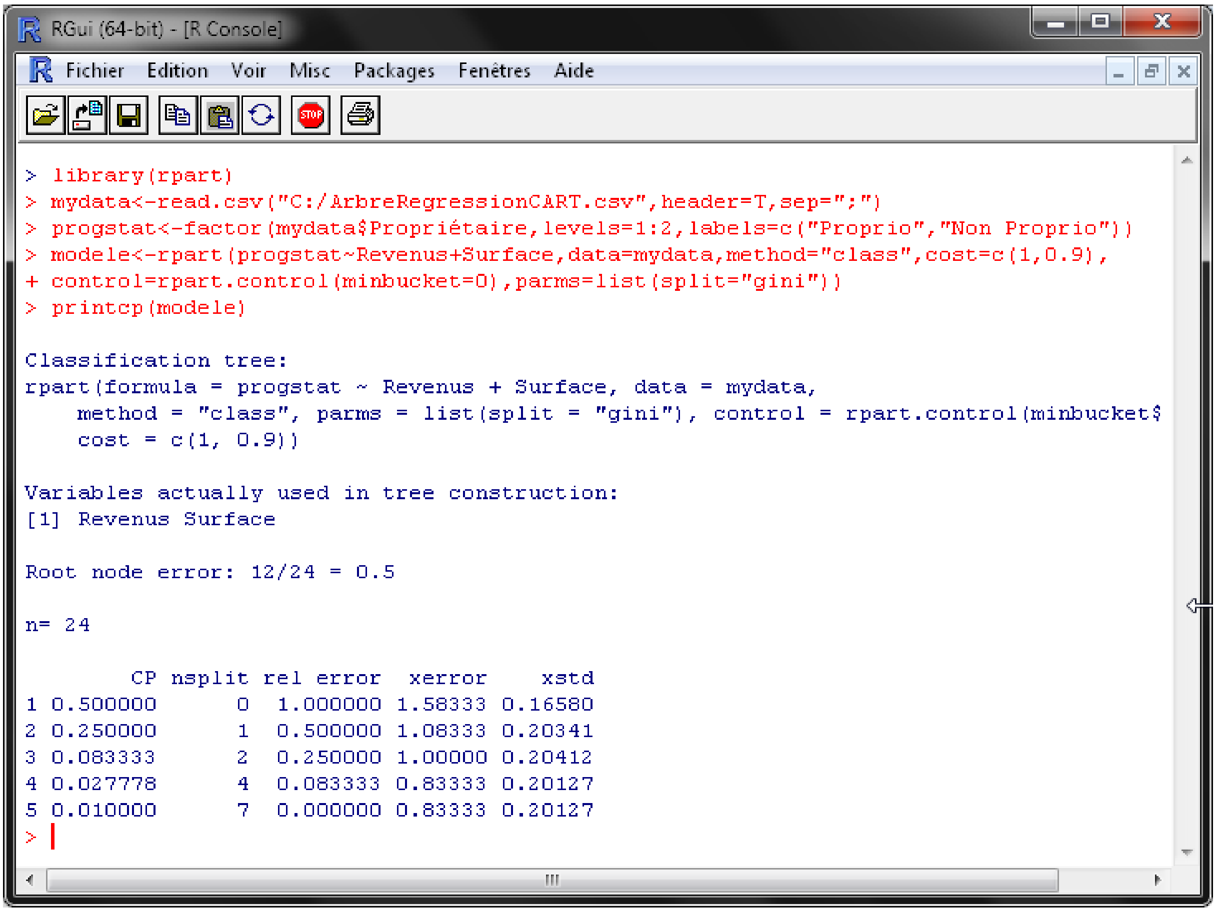
```
> library("tree")
> Tree<-tree(Propriétaire~Revenus+Surface,data=mydata,
+ control=tree.control(24, mincut = 1, minsize = 2, mindev = 0.01))
> Tree
node), split, n, deviance, yval
      * denotes terminal node

 1) root 24 6.0000 1.500
   2) Revenus < 59.7 8 0.8750 1.875
     4) Surface < 21.4 7 0.0000 2.000 *
     5) Surface > 21.4 1 0.0000 1.000 *
   3) Revenus > 59.7 16 3.4380 1.312
     6) Surface < 19.8 9 2.2220 1.556
      12) Revenus < 84.75 6 0.8333 1.833
        24) Revenus < 61.5 1 0.0000 1.000 *
        25) Revenus > 61.5 5 0.0000 2.000 *
      13) Revenus > 84.75 3 0.0000 1.000 *
     7) Surface > 19.8 7 0.0000 1.000 *
> plot(mydata$Revenus,mydata$Surface)
> text(mydata$Revenus,mydata$Surface, c("P", "NP")[mydata$Propriétaire])
> partition.tree(Tree,add=TRUE)
> 
```

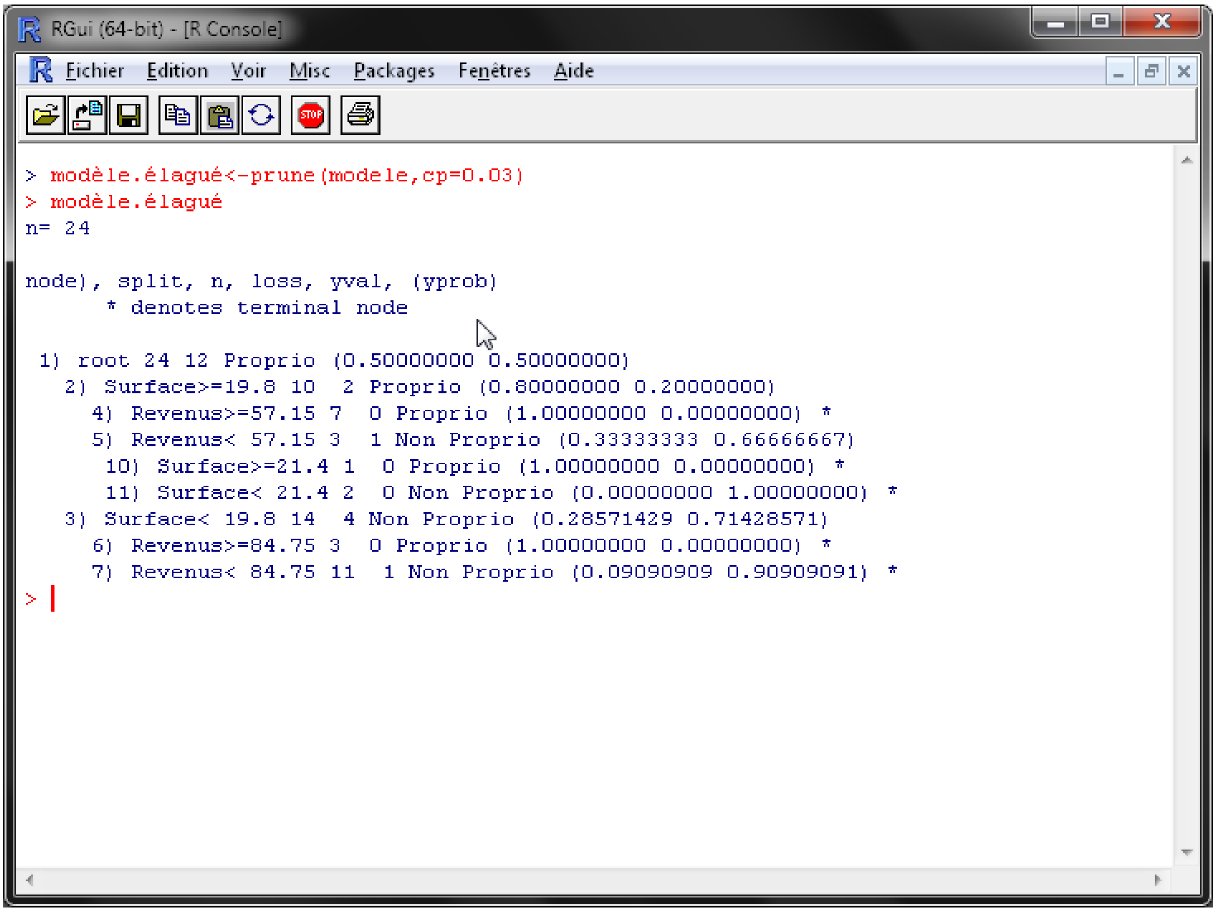
Ce qui donne graphiquement le résultat attendu (outre les couleurs et le type de points qui peuvent être bien évidemment rajoutés):



Maintenant faisons du "tree post-pruning". C'est-à-dire, prendre l'arbre qui minimise l'erreur de catégorisation. Pour cela, nous allons d'abord lister les arbres possibles à l'aide de la commande **printcp( )** :

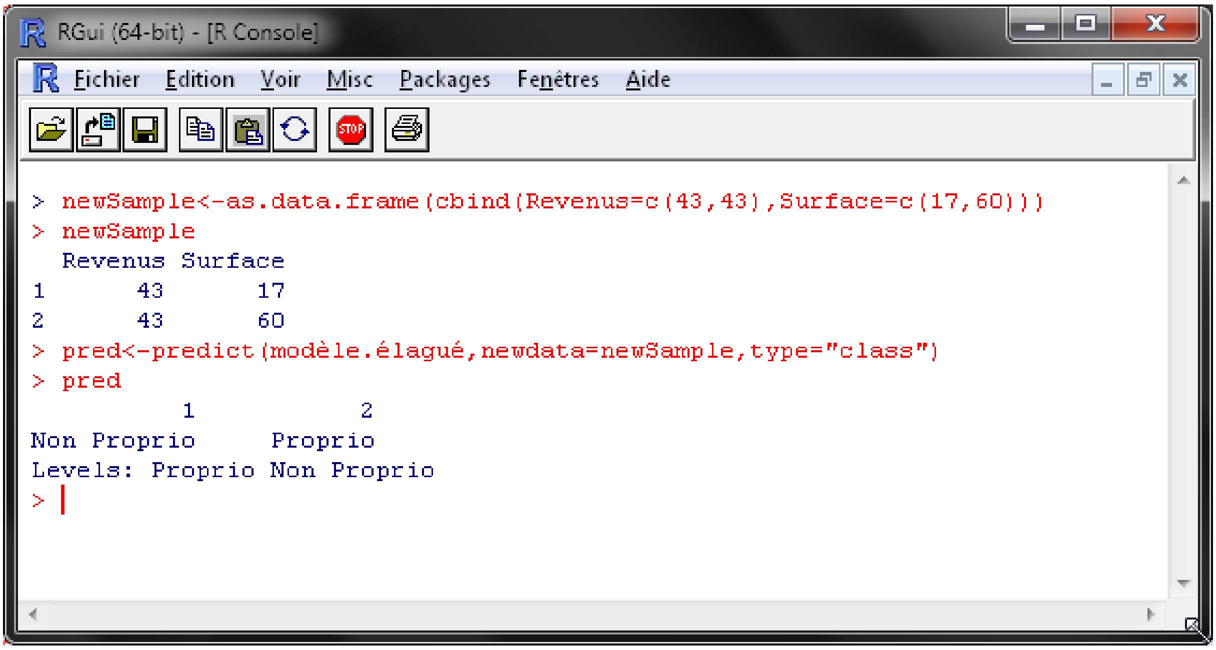


Donc il y aurait deux arbres qui minimisent la *xerror* qui sont respectivement celui à 4 et 7 sectionnements. Si nous voulions voir celui à 4 (plutôt que de chercher par tâtonnement) il suffit alors de spécifier un *CP* à l'algorithme compris entre 0.010000 et 0.08333. Voyons cela en utilisant la commande **prune( )** :

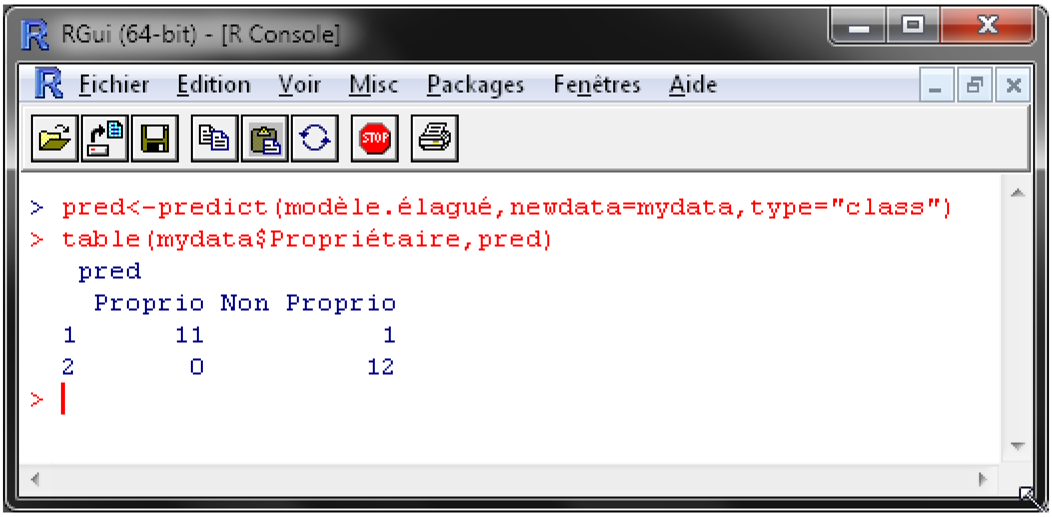


et ainsi de suite...

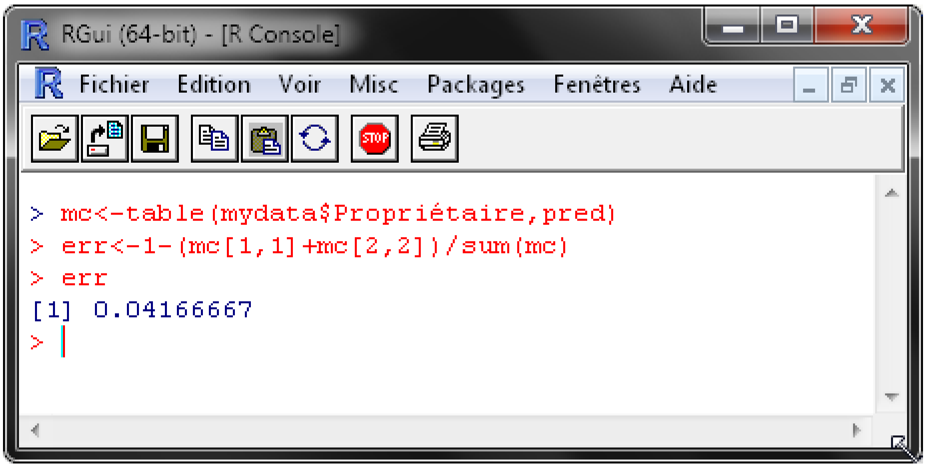
On peut ensuite faire des prédictions de clustering avec des données test en utilisant la commande **predict( )** :



Nous pouvons même faire un test sur notre propre jeu de données d'entraînement et construire la matrice de confusion (rappelez-vous que nous avons "1" qui signifie "Proprio" et "2" qui signifie "Non Proprio"):



Donc le modèle est très bon. Nous pouvons calculer le taux d'erreur assez rapidement:



Arbre de régression

Pour les arbres de régression, la démarche est exactement la même qu'avant à la différent qu'on va mettre *method="anova"* au lieu de *methode="class"*.

Voyons un exemple avec un jeu de données qui nous est déjà connu (…):

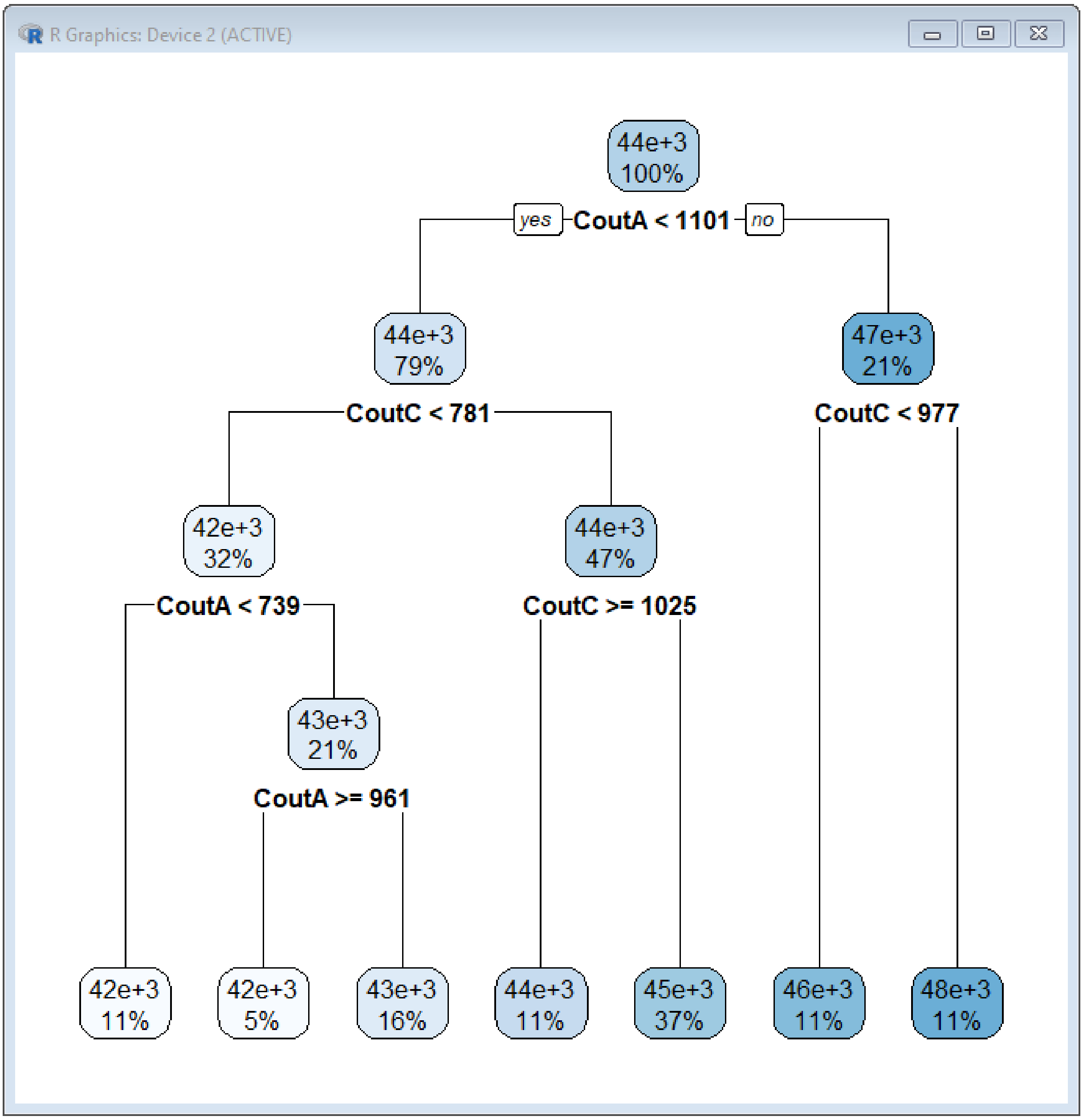
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("rpart")
> library("rpart.plot")
> mydata<-read.csv("c:/tmp/RegressionLineaireMultiple.csv",header=T,sep=";")
> mydata
   Mois Couts CoutA CoutB CoutC
1     1 44439   515   541   928
2     2 43936   929   692   711
3     3 44464   800   710   824
4     4 41533   979   675   758
5     5 46343  1165  1147   635
6     6 44922   651   939   901
7     7 43203   847   755   580
8     8 43000   942   908   589
9     9 40967   630   738   682
10   10 48582  1113  1175  1050
11   11 45003  1086  1075   984
12   12 44303   843   640   828
13   13 42070   500   752   708
14   14 44353   813   989   804
15   15 45968  1190   823   904
16   16 47781  1200  1108  1120
17   17 43202   731   590  1065
18   18 44074  1089   607  1132
19   19 44610   786   513   839
> (model<-rpart(Couts ~ CoutA + CoutB + CoutC, method="anova", data=mydata,
+ control=rpart.control(minbucket=1)))
n= 19

node), split, n, deviance, yval
      * denotes terminal node

 1) root 19 66397490.0 44355.42
   2) CoutA< 1101 15 21807070.0 43605.27
     4) CoutC< 781 6  6262270.0 42451.50
       8) CoutA< 738.5 2   608304.5 41518.50 *
       9) CoutA>=738.5 4  3042498.0 42918.00
        18) CoutA>=960.5 1        0.0 41533.00 *
        19) CoutA< 960.5 3   484864.7 43379.67 *
     5) CoutC>=781 9  2233030.0 44374.44
      10) CoutC>=1024.5 2   380192.0 43638.00 *
      11) CoutC< 1024.5 7   458222.9 44584.86 *
   3) CoutA>=1101 4  4495789.0 47168.50
     6) CoutC< 977 2    70312.5 46155.50 *
     7) CoutC>=977 2   320800.5 48181.50 *
> rpart.plot(model)
> |
```

Et avec le graphique d'arbre correspondant:



Le reste (prédiction, validation, etc.) est exactement identique à ce que nous avons vu précédemment!

Exemple 353.: Random Forests (CART Boostrap)

R 3.0.2

Donc comme nous l'avons mentionné dans le cours théorique, les Random Forests (forêts aléatoires, appelées aussi "bootstrap Forests") ne consistent qu'en un bootstrapping de la population initiale et ne garde à la fin que le modèle ayant le meilleur pouvoir prédictif!

Le package **randomForest** bien que fonctionnant très bien ne représente pas les données sous forme structurée du meilleur arbre. Pour cela, nous allons utiliser le script suivant:

```
#*************************
#Build tree structure for Random forest
#Author: Andrea DAL POZZOLO
#*************************
getConds<-function(tree){
  #store all conditions into a list
  conds<-list()
  #start by the terminal nodes and find previous conditions
  id.leafs<-which(tree$status==-1)
        j<-0
        for(i in id.leafs){
            j<-j+1
            prevConds<-prevCond(tree,i)
            conds[[j]]<-prevConds$cond
            while(prevConds$id>1){
              prevConds<-prevCond(tree,prevConds$id)
              conds[[j]]<-paste(conds[[j]]," & ",prevConds$cond)
              if(prevConds$id==1){
                  conds[[j]]<-paste(conds[[j]]," => ",tree$prediction[i])
        break()
      }
    }
  }
  return(conds)
}

prevCond<-function(tree,i){
  if(i %in% tree$right_daughter){
            id<-which(tree$right_daughter==i)
            cond<-paste(tree$split_var[id],">",tree$split_point[id])
        }
        if(i %in% tree$left_daughter){
    id<-which(tree$left_daughter==i)
            cond<-paste(tree$split_var[id],"<",tree$split_point[id])
  }
  return(list(cond=cond,id=id))
}

#remove spaces in a word
collapse<-function(x){
  x<-sub(" ","_",x)
  return(x)
}
```

Pour étudier cela dans un contexte ultra-simplifié, nous allons utiliser le jeu de données suivant:

```
Revenus;Surface;Propriétaire
60;18.4;1
85.5;16.8;1
64.8;21.6;1
61.5;20.8;1
87;23.6;1
110.1;19.2;1
108;17.6;1
82.8;22.4;1
69;20;1
93;20.8;1
51;22;1
81;20;1
75;19.6;2
52.8;20.8;2
64.8;17.2;2
43.2;20.4;2
84;17.6;2
49.2;17.6;2
59.4;16;2
66;18.4;2
47.4;16.4;2
33;18.8;2
51;14;2
63;14.8;2
```

Nous utilisons alors la commande **randomForest( )** du package **randomForest**:

```
> source("C:/RandomForestStructure.R")
> library(randomForest)
> mydata<-read.csv("C:/ArbreRegressionCART.csv",header=T,sep=";")
> progstat<-factor(mydata$Propriétaire,levels=1:2,labels=c("Proprio", "Non Proprio"))
> mod.rf <- randomForest(progstat~Revenus+Surface, data=mydata)
> tree<-getTree(mod.rf, k=1, labelVar=TRUE)
> #rename the name of the column
> colnames(tree)<-sapply(colnames(tree),collapse)
> rules<-getConds(tree)
> print(rules)
[[1]]
[1] "Surface < 19"

[[2]]
[1] "Revenus < 47.1  &  Surface > 19  =>  Non Proprio"

[[3]]
[1] "Revenus > 78.9  &  Revenus > 47.1  &  Surface > 19  =>  Proprio"

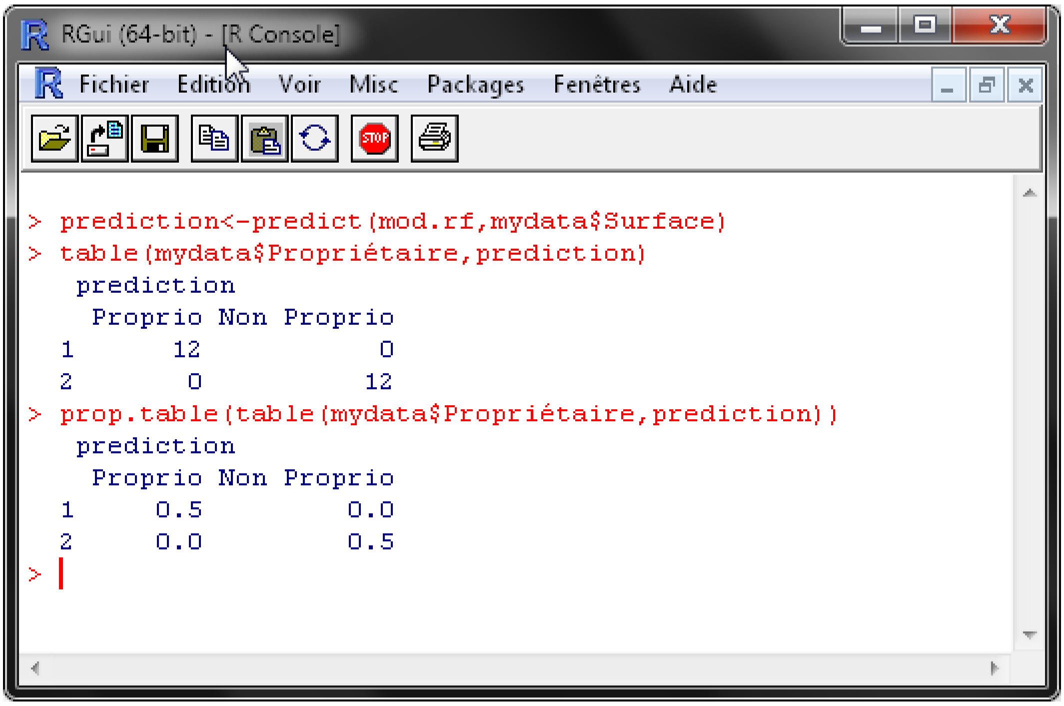
[[4]]
[1] "Revenus < 51.9  &  Revenus < 78.9  &  Revenus > 47.1  &  Surface > 19  =>  Proprio"

[[5]]
[1] "Surface < 21.2  &  Revenus > 51.9  &  Revenus < 78.9  &  Revenus > 47.1  &  Surface > 19  =>  Non Proprio"

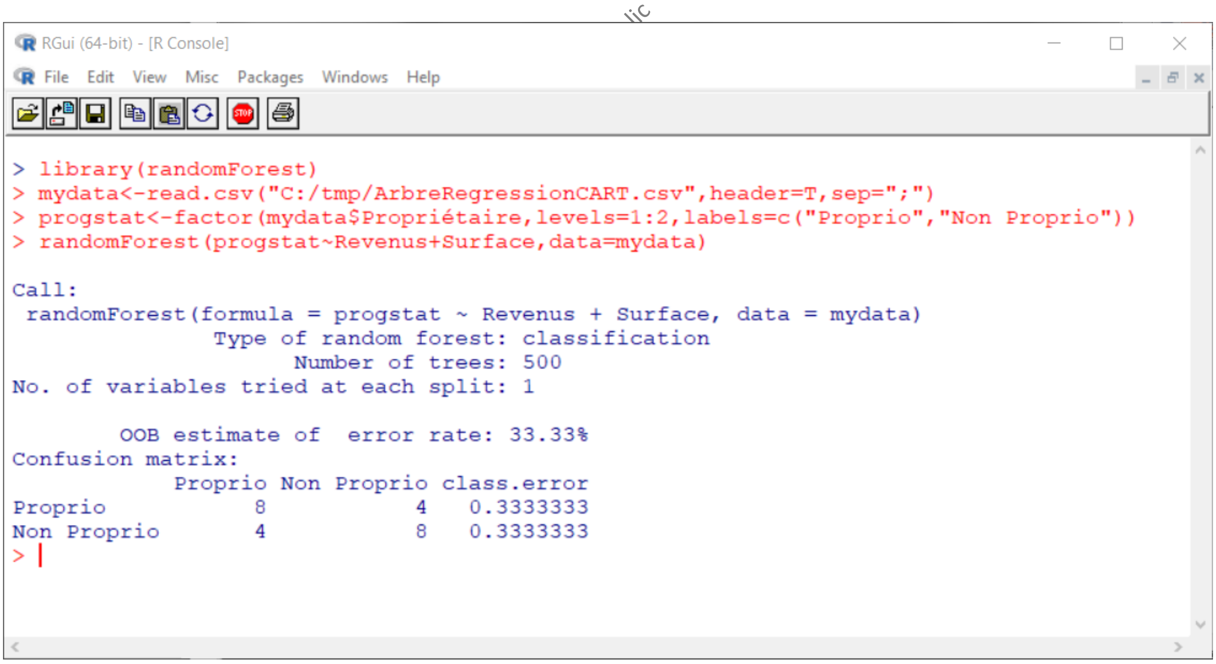
[[6]]
[1] "Surface > 21.2  &  Revenus > 51.9  &  Revenus < 78.9  &  Revenus > 47.1  &  Surface > 19  =>  Proprio"

>
```

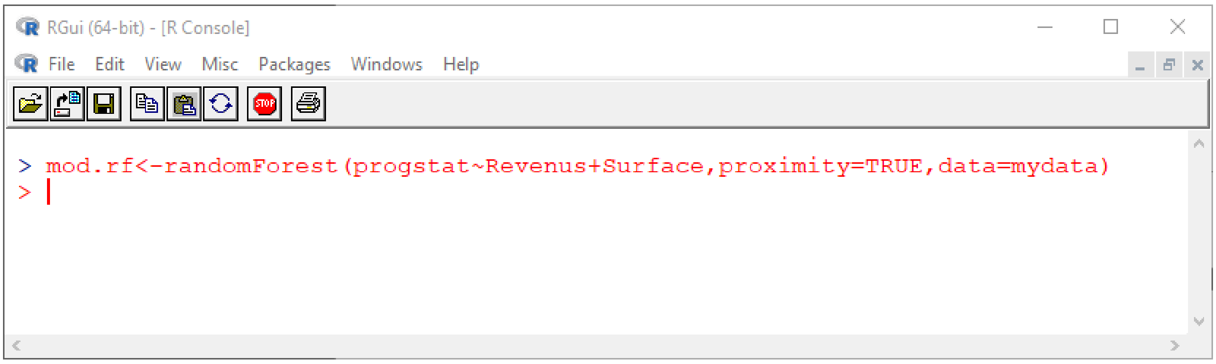
Nous pouvons aussi vérifier que les random Forest performent très bien à la classification pour les surfaces seules par exemple:



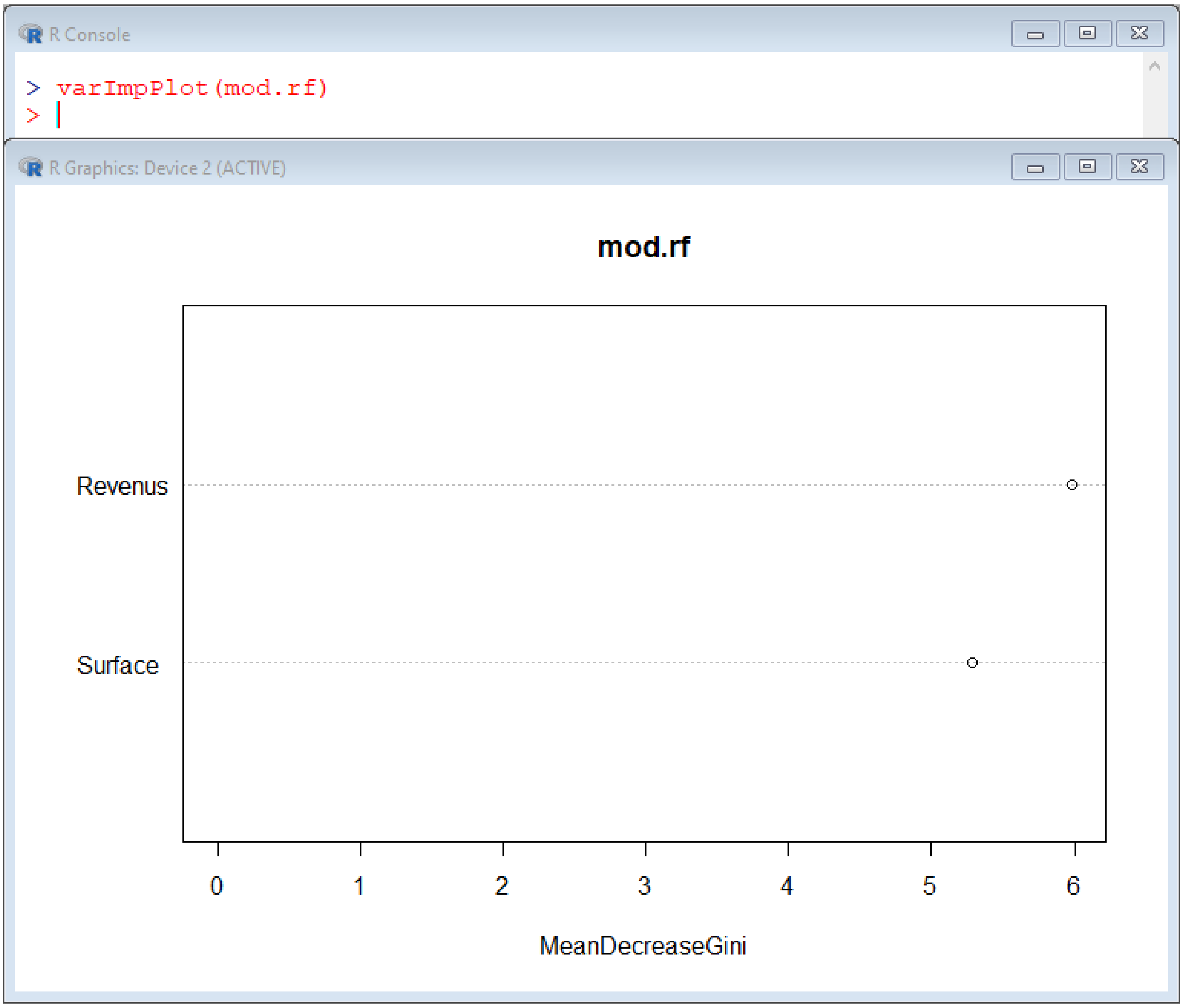
Nous pouvons également bien évidemment observer le résultat d'une des étapes intermédiaires:



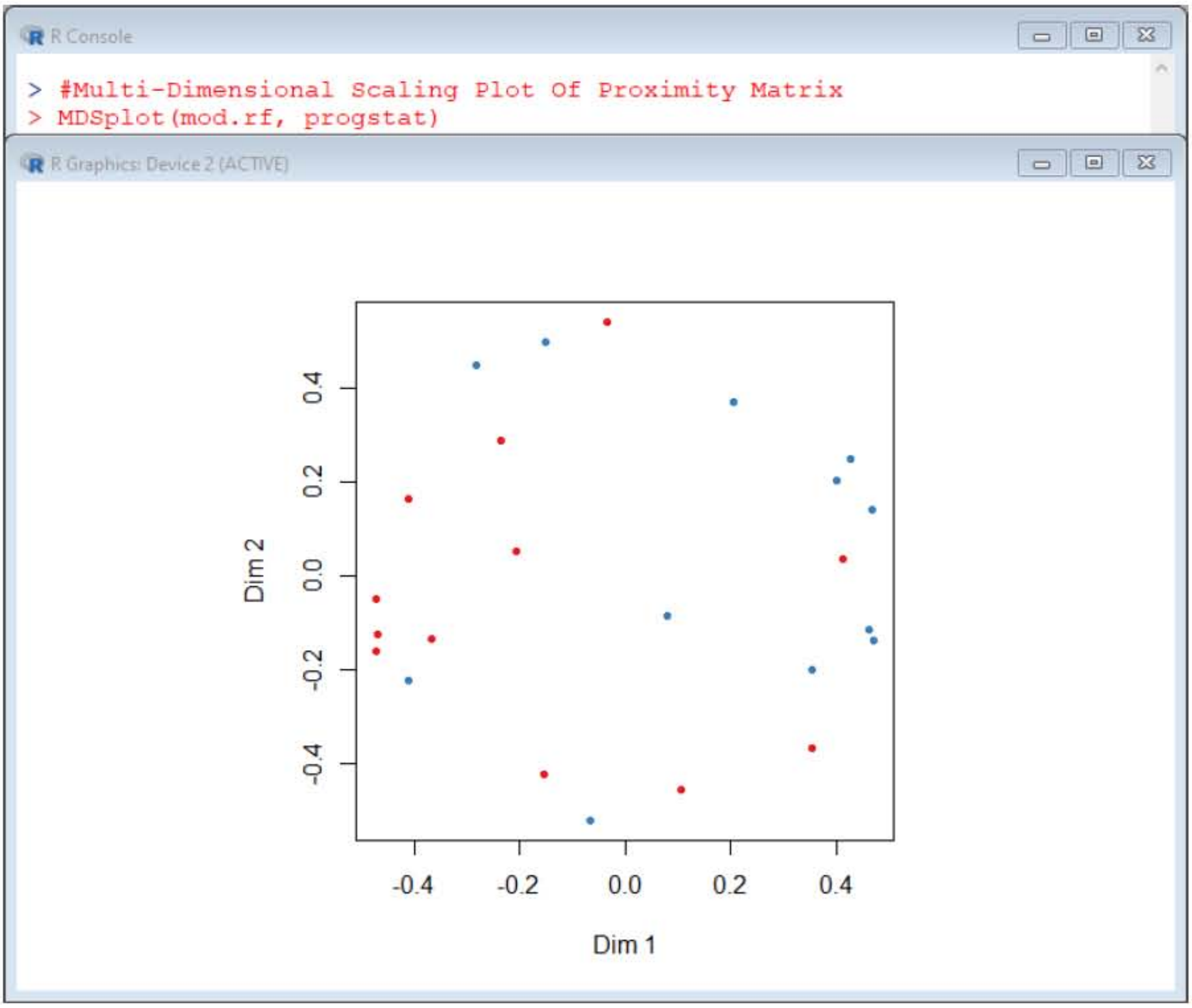
On peut aussi rajouter quelques graphiques (je sais qu'une majorité de gens aiment bien malheureusement lire certains graphiques…). Mais pour cela on relancera le modèle avec un paramètre supplémentaire:



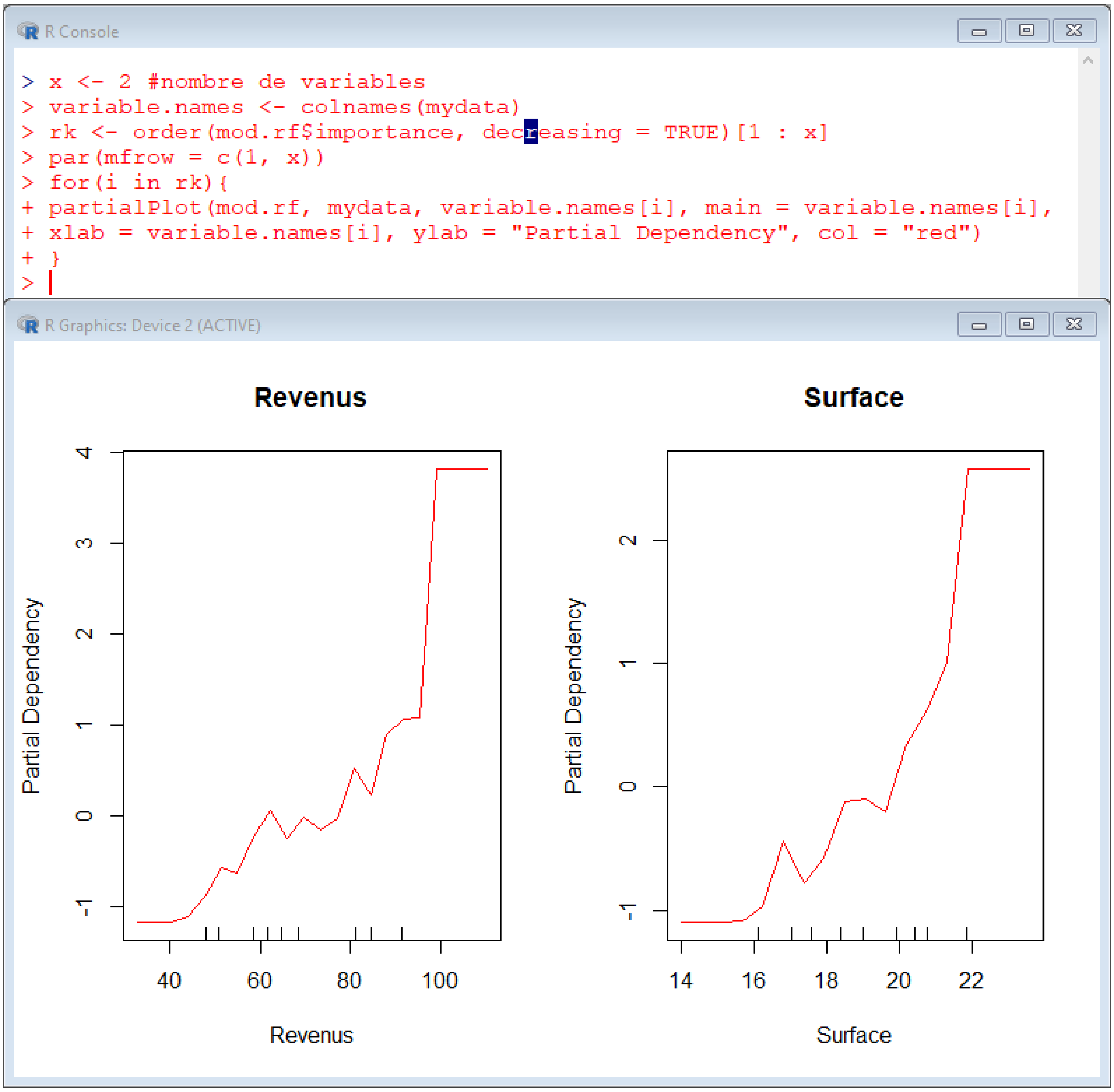
Et allons-y pour le premier graphique:



Ensuite pour le deuxième…:



Et le dernier graphique:



Et je vais m'arrêter là à ce jour pour les graphiques car cela me saoule…

Exemple 354.: Comparaison d'apprentissage de modèles

R 3.4.2

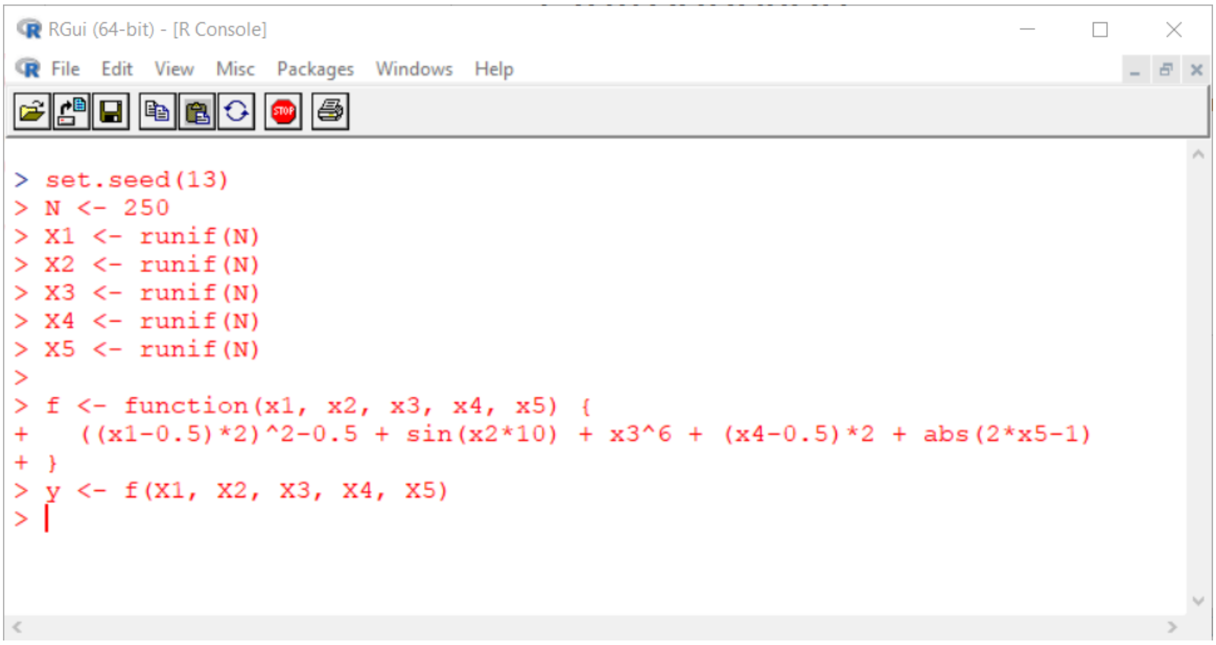
Il peut être très utile pour des objectifs d'enseignements, ou de présentations à un comité, ou pour faire une analyse qualitative, de comparer différents modèles de Machine Learning qui sont entraînés sur un même jeu de modèle et voir comme ces derniers prédisent sur le même jeu de modèles en question.

Pour cela il existe un magnifique package nommé **DALEX** et sa fonction **explain( )**. Voyons comment cela fonctionne en reprenant à l'identique l'exemple donnée dans la vignette du package par l'auteur lui-même (quand l'exemple est parfait… faut pas réinventer la roue!).

Considérons pour l'exemple la fonction suivante à 5 arguments:

$$f(\vec{x}) = 2(x_1 - 1)^2 + \sin(10x_2) + x_3^6 + (2x_4 - 1) + |2x_5 - 1|$$

où chacune des 5 variables suit une loi uniforme continue sur l'intervalle [0,1]. Simulons 250 échantillons de chaque et générons la fonction:



Maintenant comparons 5 modèles:

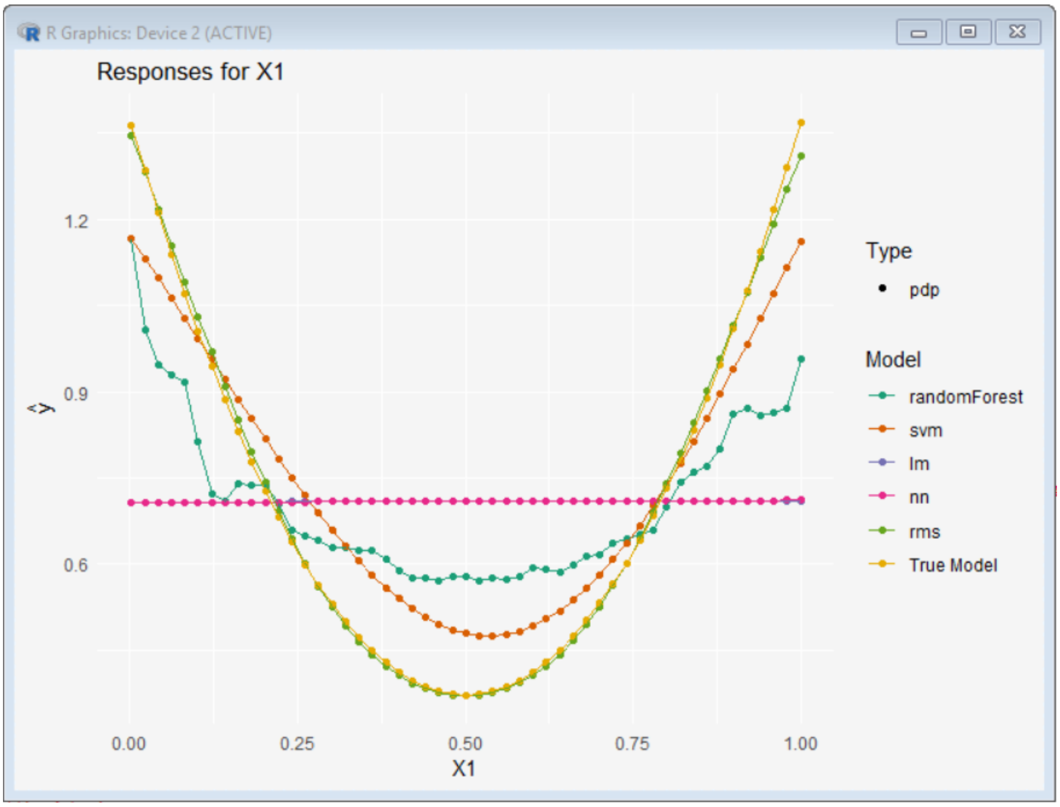
- Random forest
- Modèle linéaire
- Support Vector Machine
- Restricted Cubic Spline regression[36]
- Réseau de neurones feed-forward
- Le modèle "vrai" lui-même…

[36] qui pour rappel ne marche pas avec lm( ) mais seulement avec ols( )

Voyons cela:

```
> library(randomForest)
> library(DALEX)
> library(e1071)
> library(rms)
> library(neuralnet)
>
>
> df <- data.frame(y, X1, X2, X3, X4, X5)
>
> model_rf<-randomForest(y~., df)
> model_svm<-svm(y~., df)
> model_lm<-lm(y~., df)
> model_nn<-neuralnet(y~X1+X2+X3+X4+X5,df,hidden=1)
>
>
> dd <- datadist(df)
> options(datadist="dd")
> model_rms <- ols(y ~ rcs(X1) + rcs(X2) + rcs(X3) + rcs(X4) + rcs(X5), df)
>
> ex_rf<-explain(model_rf)
> ex_svm<-explain(model_svm)
> ex_lm<-explain(model_lm)
> ex_rms<-explain(model_rms, label = "rms", data = df[, -1], y = df$y)
> ex_nn<-explain(model_nn,data = df[, -1],y = df[, 1],predict_function = function(x, y) compute(x, y)$net.result)
> ex_tr<-explain(model_lm, data = df[,-1],
+                  predict_function = function(m, x) f(x[,1], x[,2], x[,3], x[,4], x[,5]),
+                  label = "True Model")
>
> library(ggplot2)
> plot(single_variable(ex_rf, "X1"),
+      single_variable(ex_svm, "X1"),
+      single_variable(ex_lm, "X1"),
+      single_variable(ex_nn, "X1"),
+      single_variable(ex_rms, "X1"),
+      single_variable(ex_tr, "X1")) +
+   ggtitle("Responses for X1")
>
```

Ce qui donne:



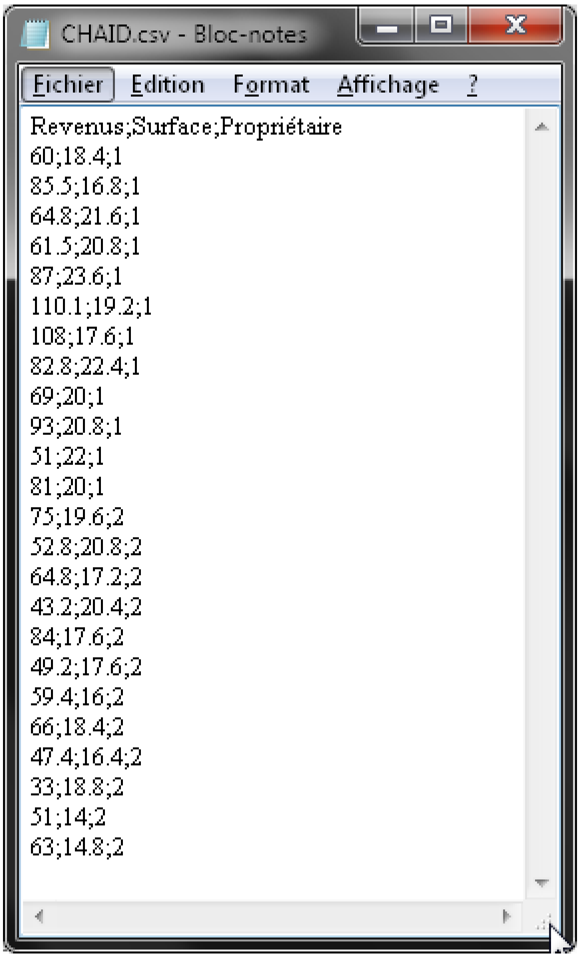
Voilà! Un jolie petit outil pour diagnostiquer qualitativement un modèle!

Exemple 355.: Chi-squared Automated Interaction Detection (CHAID)

R 3.1.1.

Nous avons parlé dans le cours théorique de cette méthode empirique (à la base conçue pour ne fonctionner que pour des variables catégorielles) de partitionnement/classification. Il n'y avait rien à démontrer mathématiquement car elle n'utilise que des éléments statistiques triviaux.

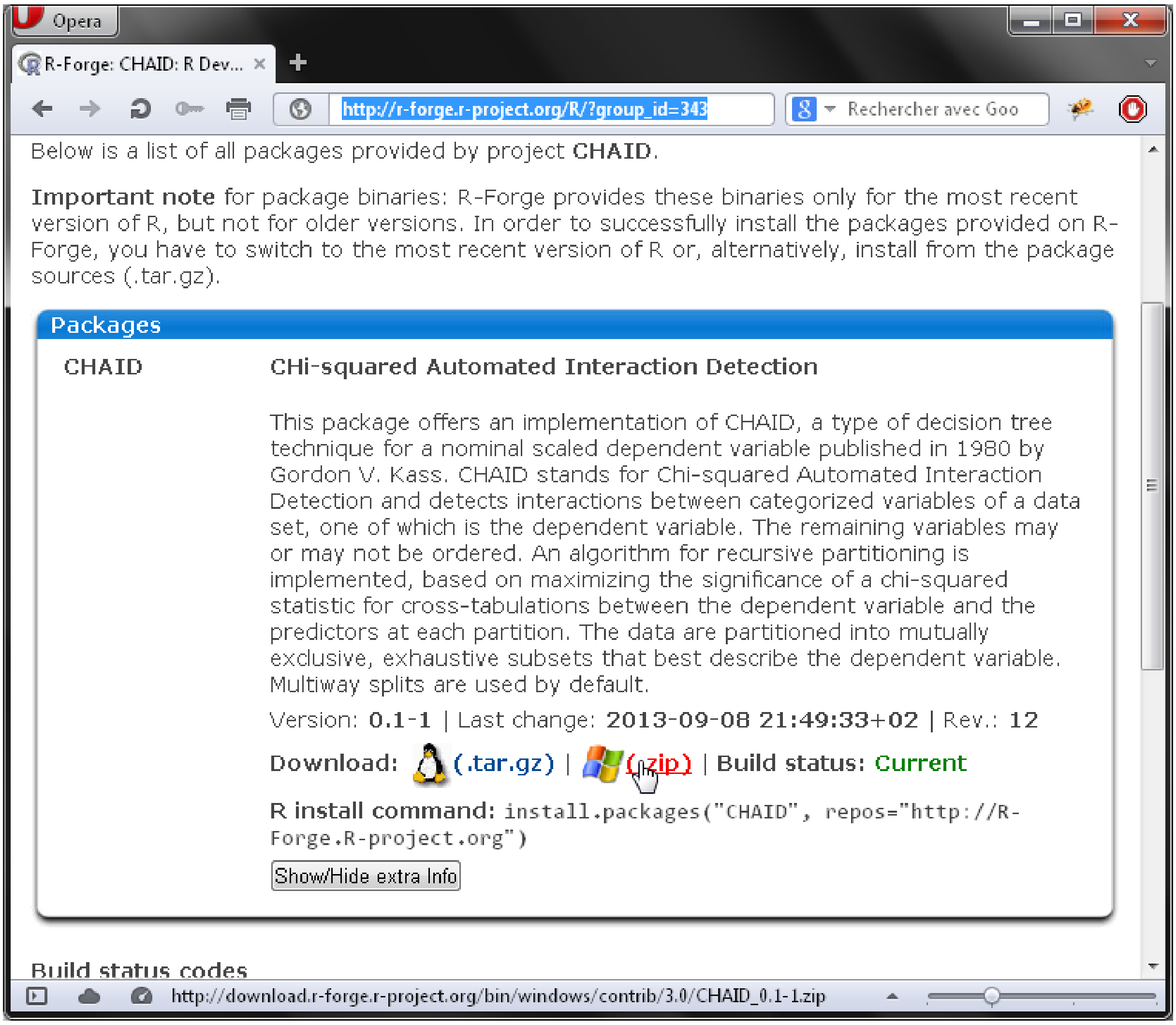
Voyons-en un exemple pratique avec le jeu de données suivant où puisqu'il y a des variables non catégorielles nous allons devoir fractionner les valeurs continues et catégories en nous basant sur notre retour d'expérience (pour rappel il s'agit là de la faiblesse majeure de cette méthode):


CHAID.csv - Bloc-notes

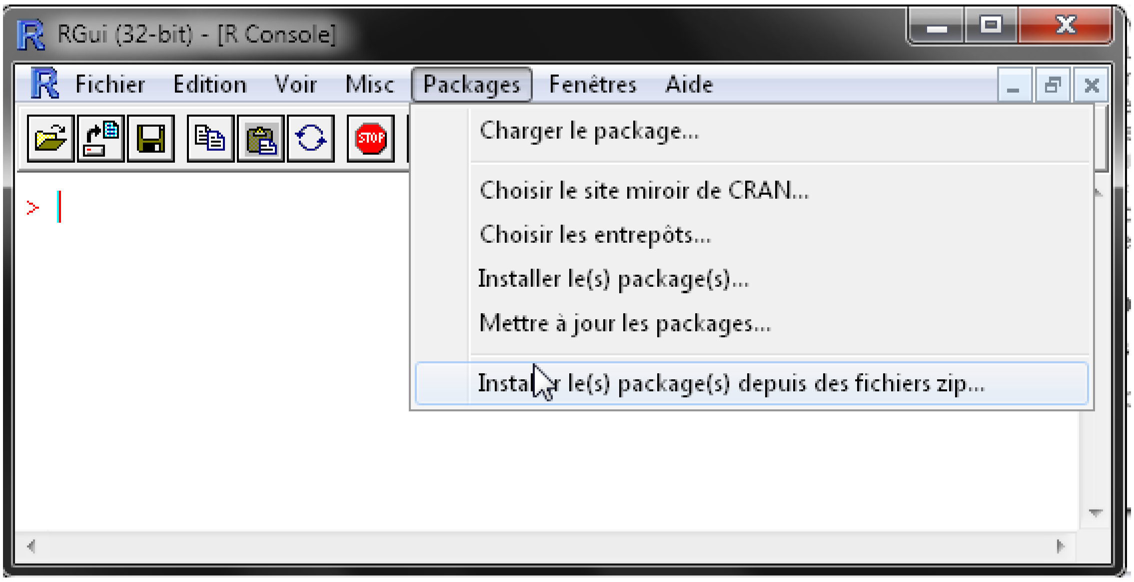
Fichier Edition Format Affichage ?

Revenus;Surface;Propriétaire
60;18.4;1
85.5;16.8;1
64.8;21.6;1
61.5;20.8;1
87;23.6;1
110.1;19.2;1
108;17.6;1
82.8;22.4;1
69;20;1
93;20.8;1
51;22;1
81;20;1
75;19.6;2
52.8;20.8;2
64.8;17.2;2
43.2;20.4;2
84;17.6;2
49.2;17.6;2
59.4;16;2
66;18.4;2
47.4;16.4;2
33;18.8;2
51;14;2
63;14.8;2

Nous téléchargeons d'abord le fichier *.zip de ce package depuis le site web suivant car à ce jour il ne peut pas être installé depuis l'interface directement:

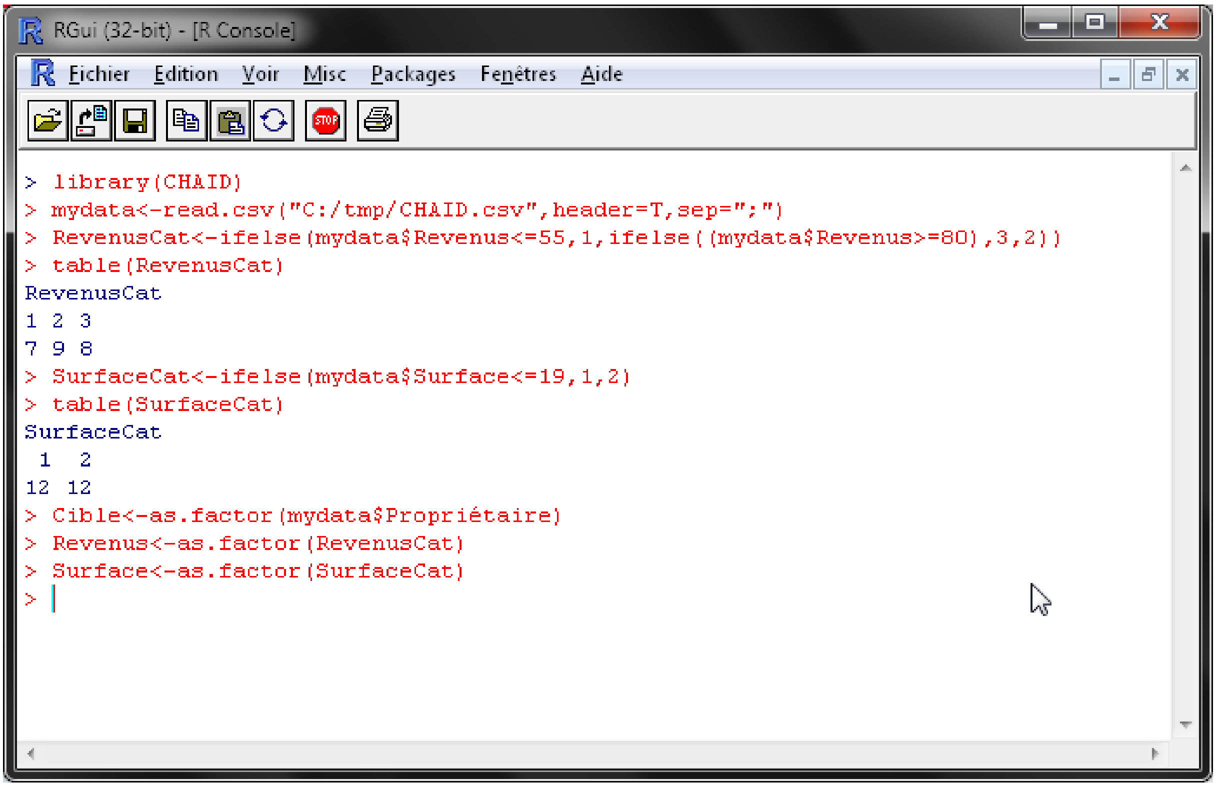


Une fois ceci fait, nous installons le package directement:

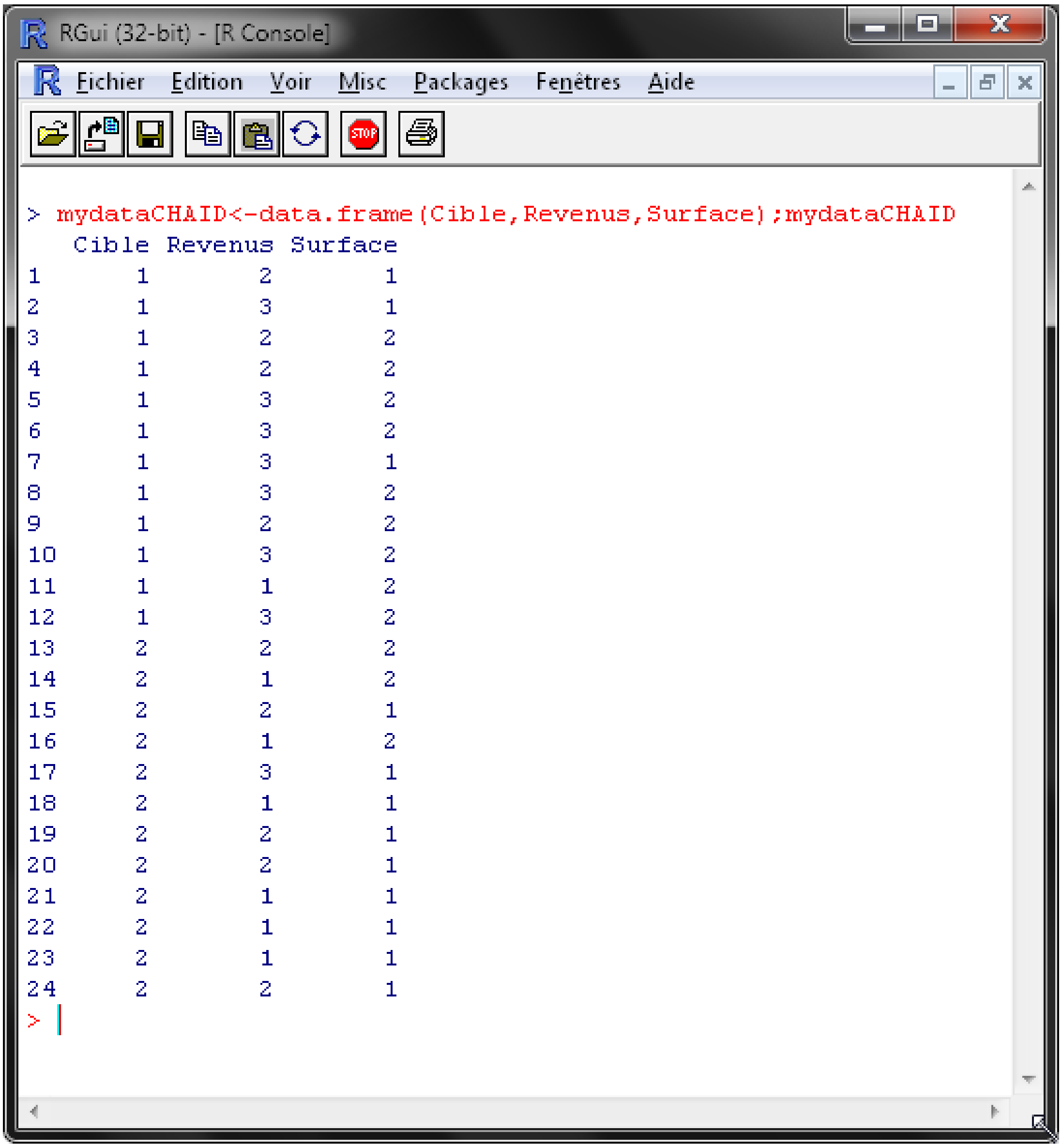


et nous pouvons poursuivre comme à l'habitude.

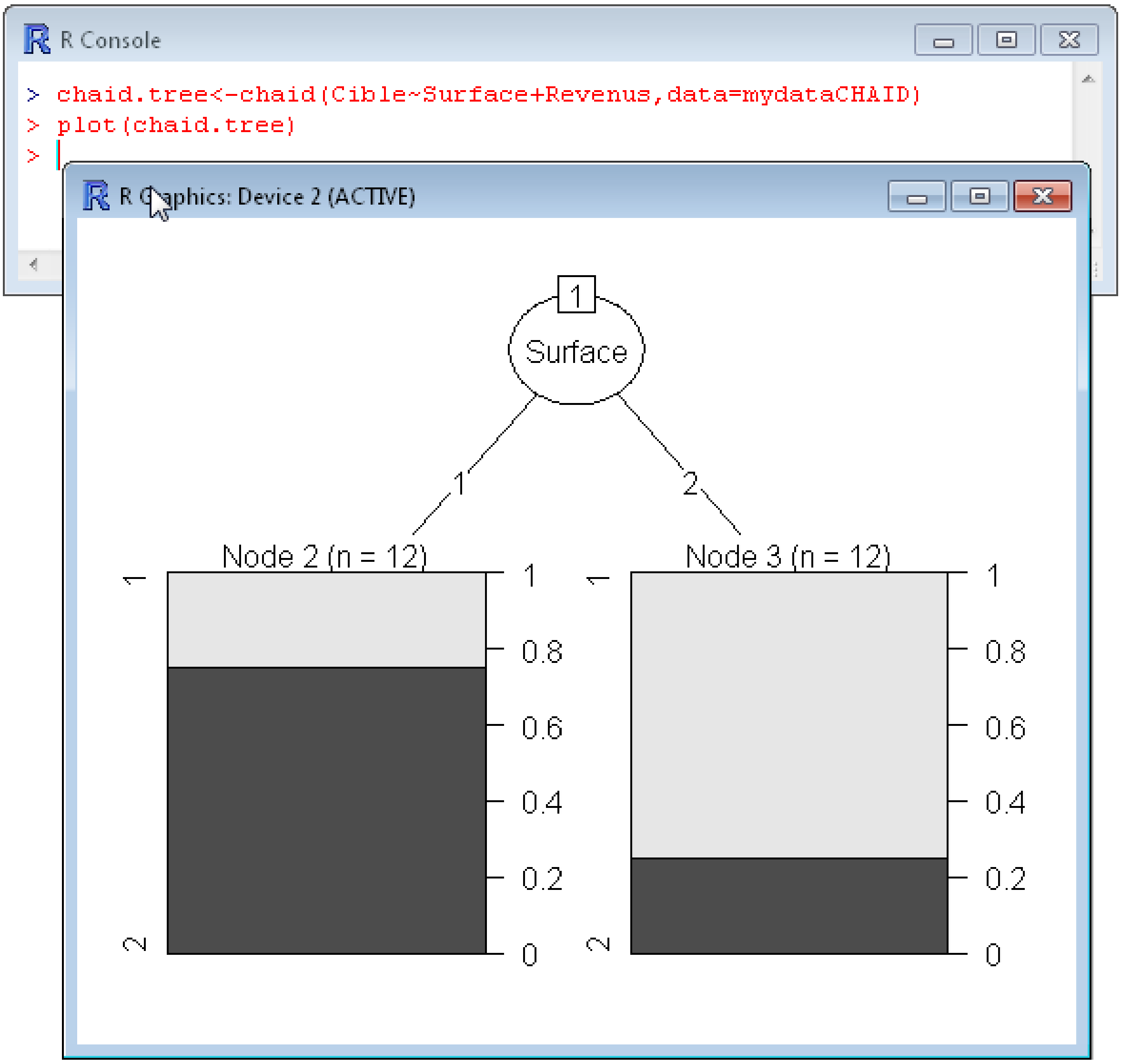
D'abord nous préparons les variables:



Nous reconstruisons un data frame en correspondance:



Et enfin nous construisons l'arbre:



Le résultat n'est guère convaincant avec ce package. Au fait nous pouvons deviner qu'il utilise le package **rpart** et donc il s'arrête au premier niveau pour les mêmes raisons que celles vues à la page 2168 (SPSS fait de même):

```
> print(chaid.tree)

Model formula:
Cible ~ Revenus + Surface

Fitted party:
[1] root
|   [2] Surface in 1: 2 (n = 12, err = 25.0%)
|   [3] Surface in 2: 1 (n = 12, err = 25.0%)

Number of inner nodes:    1
Number of terminal nodes: 2
>
```

Mais ici comme nous ne pouvons a priori définir des poids nous sommes coincés pour cet exemple scolaire.

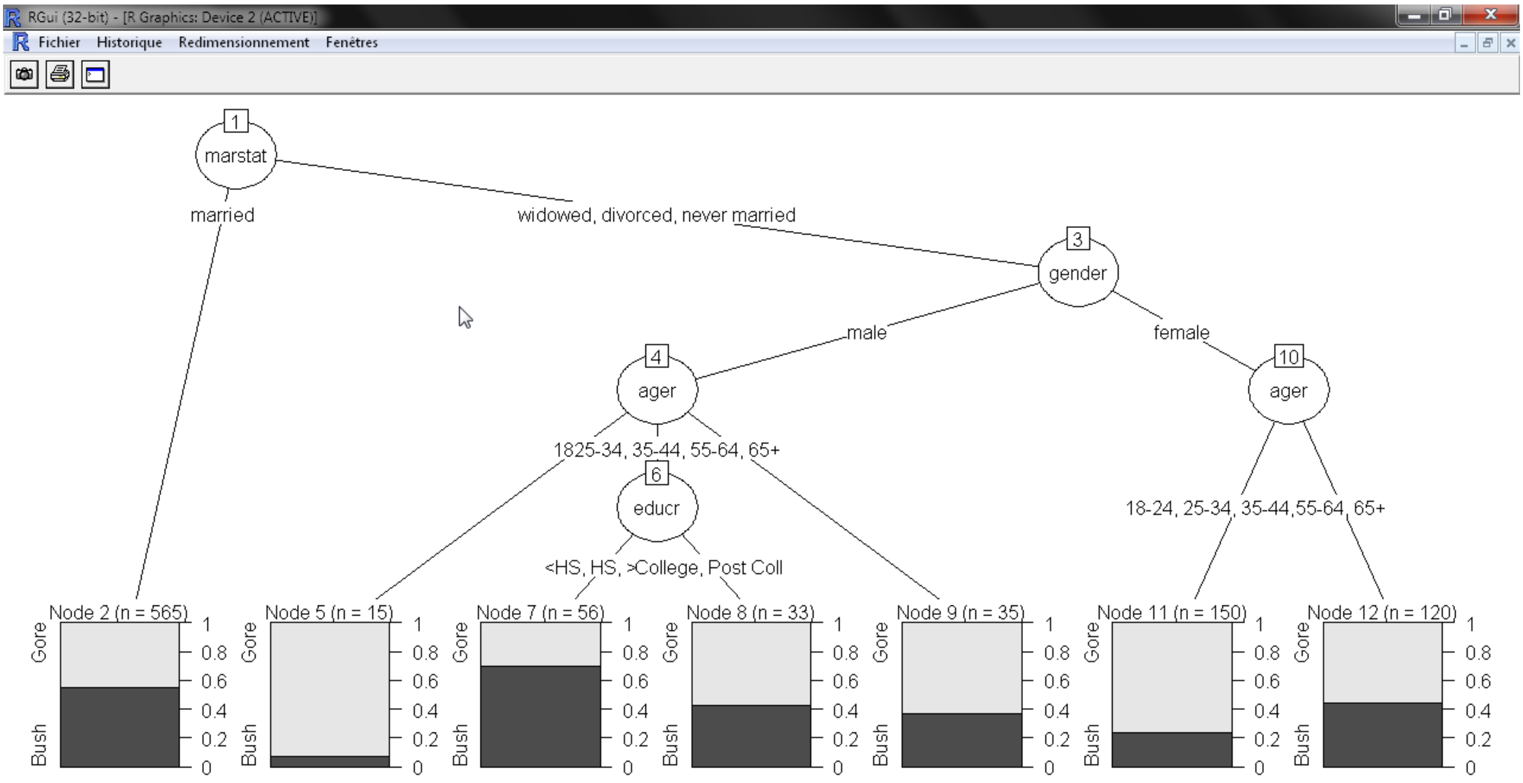
Dès lors, prenons un jeu de données par défaut du package CHAID et jouons avec:

```
> summary(USvote)
  vote3          gender          ager         empstat             educr                  marstat
 Gore:5558   male  :4777   18-24: 559   yes     :6773   <HS         : 614   married      :6133
 Bush:5087   female:5868   25-34:1521   no      :1740   HS          :2570   widowed      :1006
                           35-44:2471   retired:2132    >HS         :3293   divorced     :1499
                           45-54:2172                   College     :2744   never married:1717
                           55-64:1796                   Post Coll:1404      NA's         : 290
                           65+  :2096                   NA's        :  20
                           NA's :  30
>
```

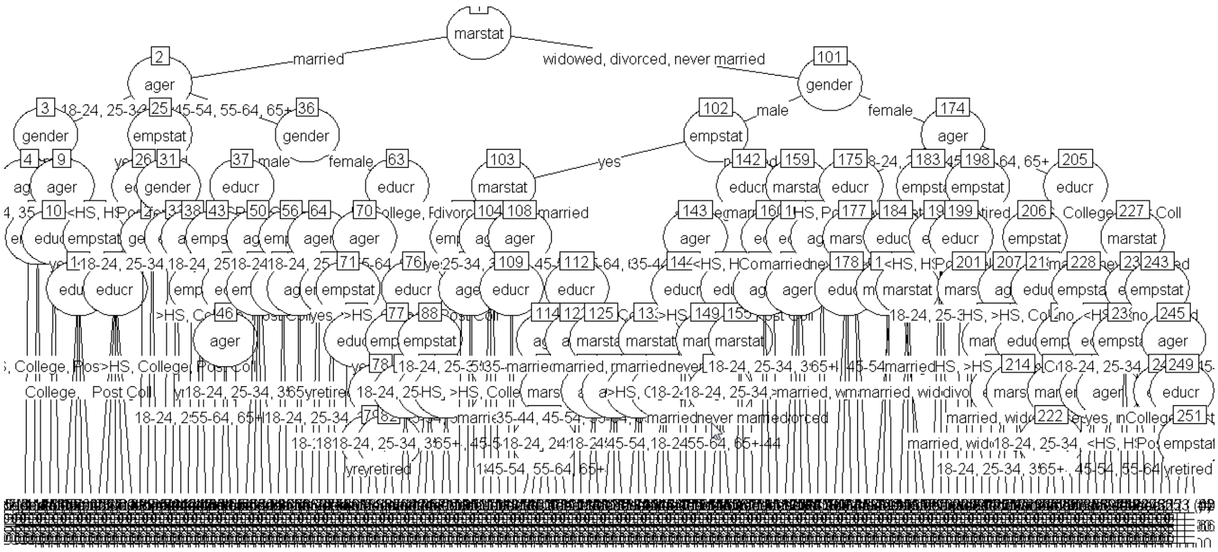
Et les commandes vues plus haut:

```
> library("CHAID")
> USvoteS <- USvote[sample(1:nrow(USvote), 1000),]
> chaidUS <- chaid(vote3 ~ ., data = USvoteS)
> plot(chaidUS)
>
```

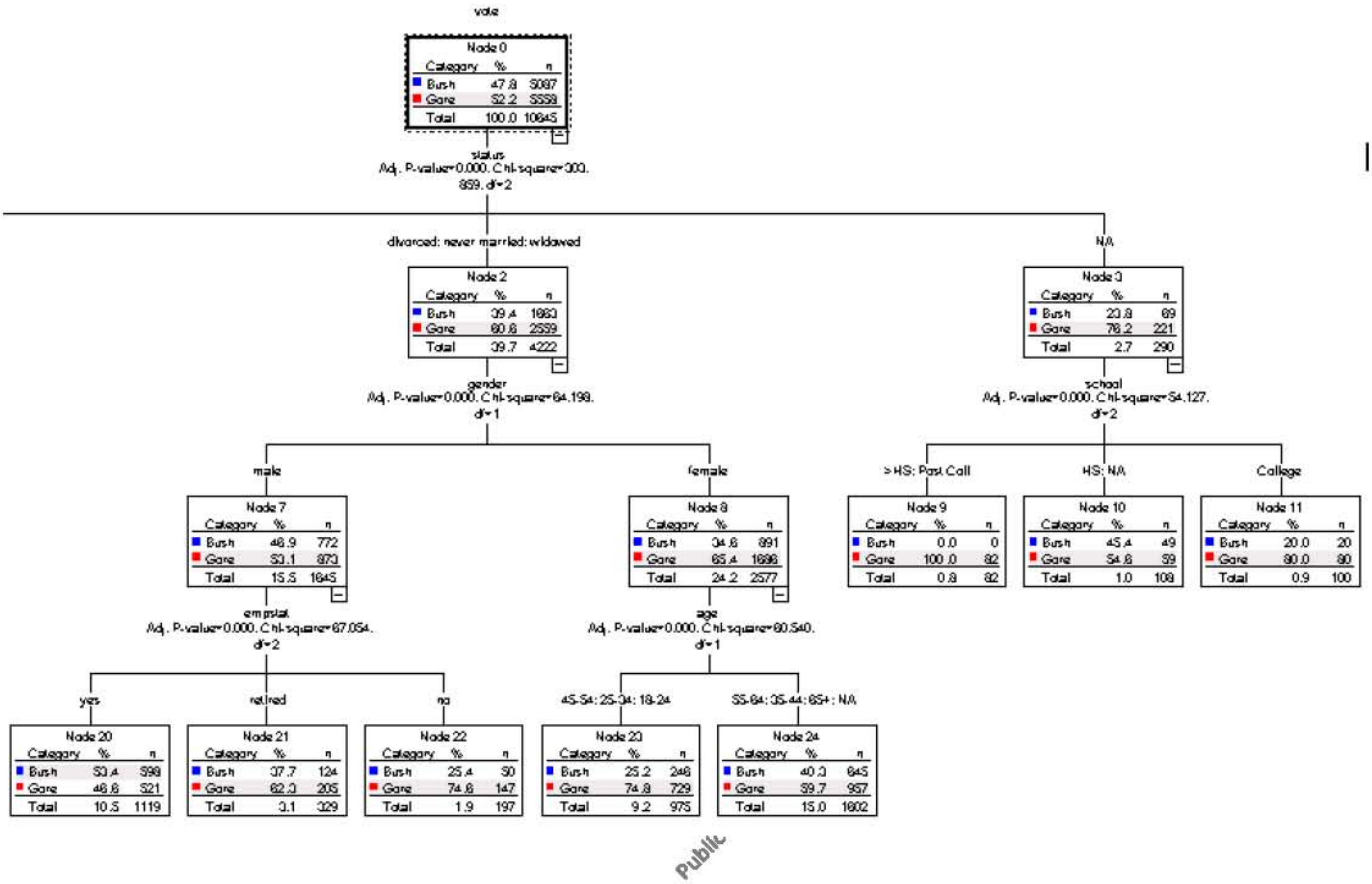
Ce qui donne:



Nous n'avons malheureusement pas le temps dans le cours théorique de vérifier que le résultat soit juste… La comparaison avec SPSS montre que le résultat test très différent entre les deux logiciels… mais cela est du au fait que nous avons pris un échantillon de taille 1'000. Si nous prenons tout le jeu de données cela nous donne:



Ce qui est illisible (contrairement à SPSS) mais qui correspond par contre au niveau des nœuds à ce que donne SPSS dont voici un morceau à la page suivante:

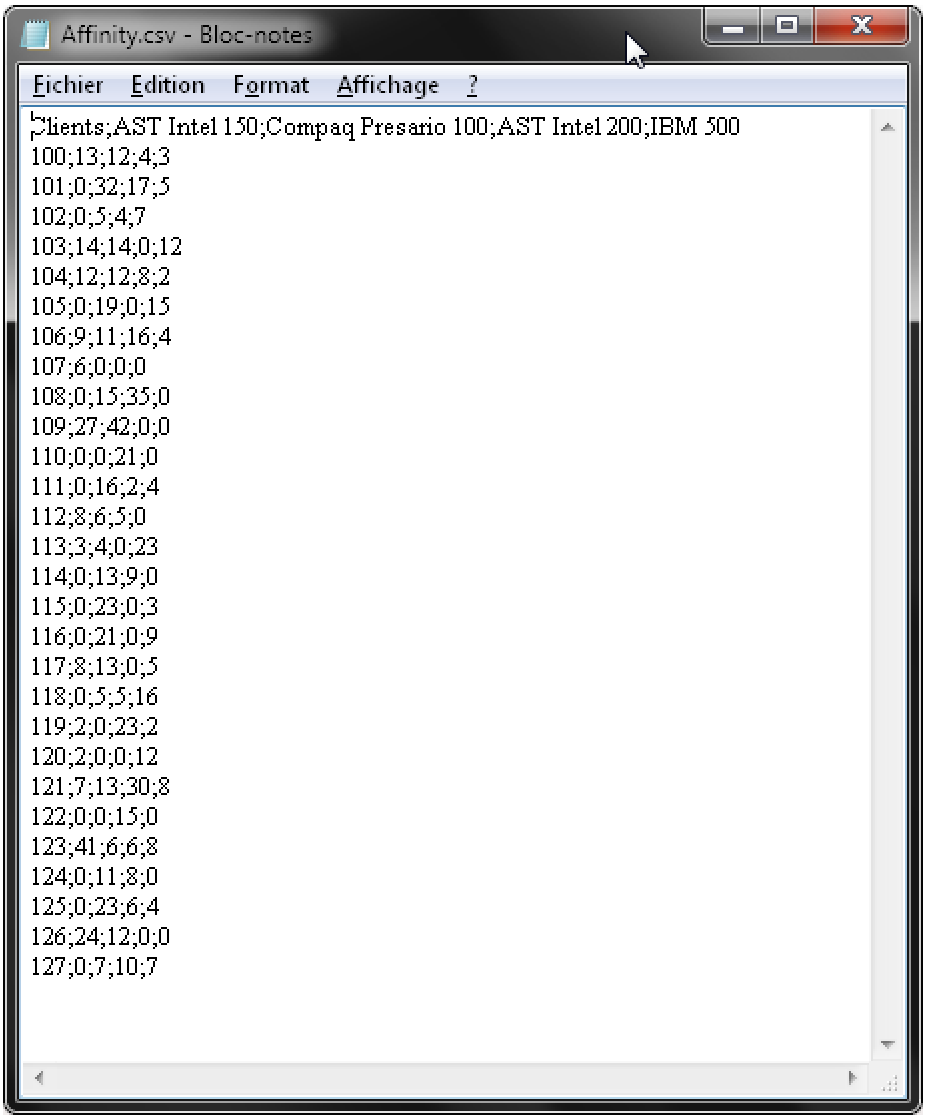

vote
Node 0
status
Adj. P-value=0.000, Chi-square=300.859, df=2
divorced; never married; widowed
N/A
Node 2
Node 3
gender
Adj. P-value=0.000, Chi-square=64.198, df=1
school
Adj. P-value=0.000, Chi-square=54.127, df=2
male
female
>HS; Post Coll
HS; N/A
College
Node 7
Node 8
Node 9
Node 10
Node 11
empstat
Adj. P-value=0.000, Chi-square=67.054, df=2
age
Adj. P-value=0.000, Chi-square=60.540, df=1
yes
retired
no
45-54; 25-34; 18-24
55-64; 35-44; 65+; N/A
Node 20
Node 21
Node 22
Node 23
Node 24
Category % n
Bush
Gore
Total
Public

Exemple 356.: Analyse d'affinité (affinity analysis)

R 3.1.1

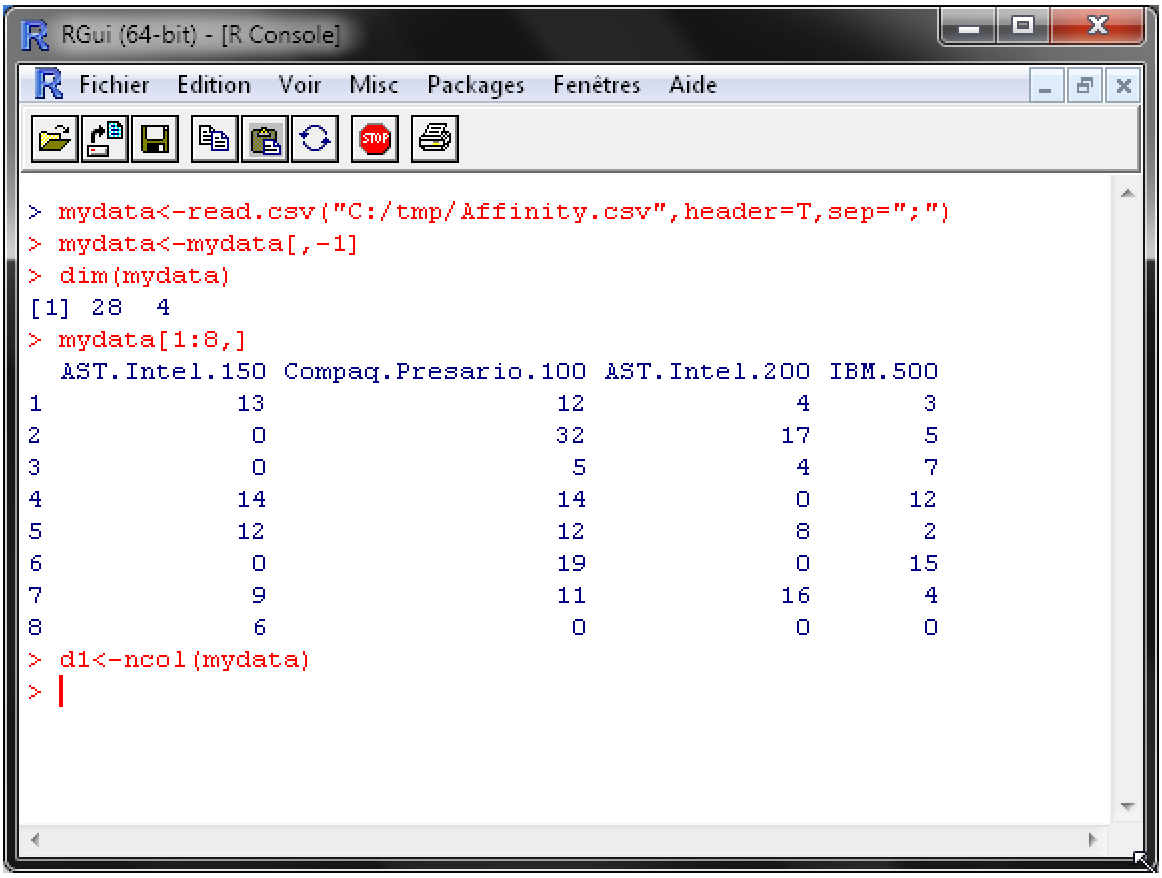
L'analyse d'affinité est très utilisée sur les sites Internet implémentant des modèles naïfs de proposition d'articles par rapport aux achats précédents de consommateurs.

Voyons en quoi consiste l'idée en partant du fichier suivant où chaque ligne représente un panier commandé par un client donné (certains clients sont revenus plusieurs fois pour des paniers différents) et où les colonnes représentées les articles achetés et les composantes les quantités correspondantes:

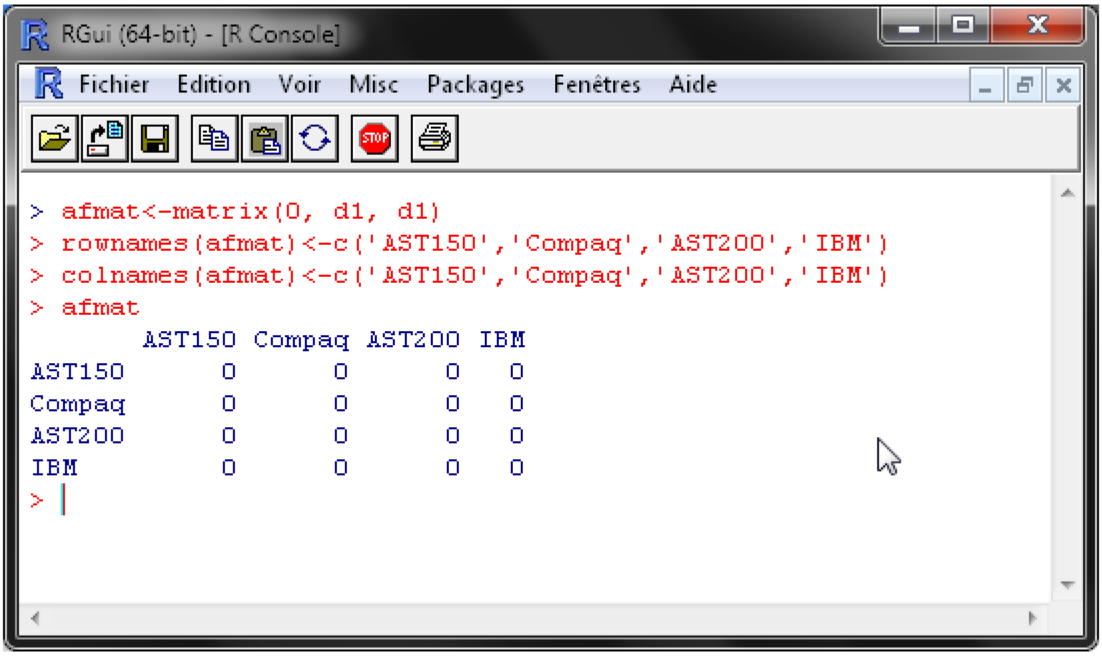



```
Clients;AST Intel 150;Compaq Presario 100;AST Intel 200;IBM 500
100;13;12;4;3
101;0;32;17;5
102;0;5;4;7
103;14;14;0;12
104;12;12;8;2
105;0;19;0;15
106;9;11;16;4
107;6;0;0;0
108;0;15;35;0
109;27;42;0;0
110;0;0;21;0
111;0;16;2;4
112;8;6;5;0
113;3;4;0;23
114;0;13;9;0
115;0;23;0;3
116;0;21;0;9
117;8;13;0;5
118;0;5;5;16
119;2;0;23;2
120;2;0;0;12
121;7;13;30;8
122;0;0;15;0
123;41;6;6;8
124;0;11;8;0
125;0;23;6;4
126;24;12;0;0
127;0;7;10;7
```

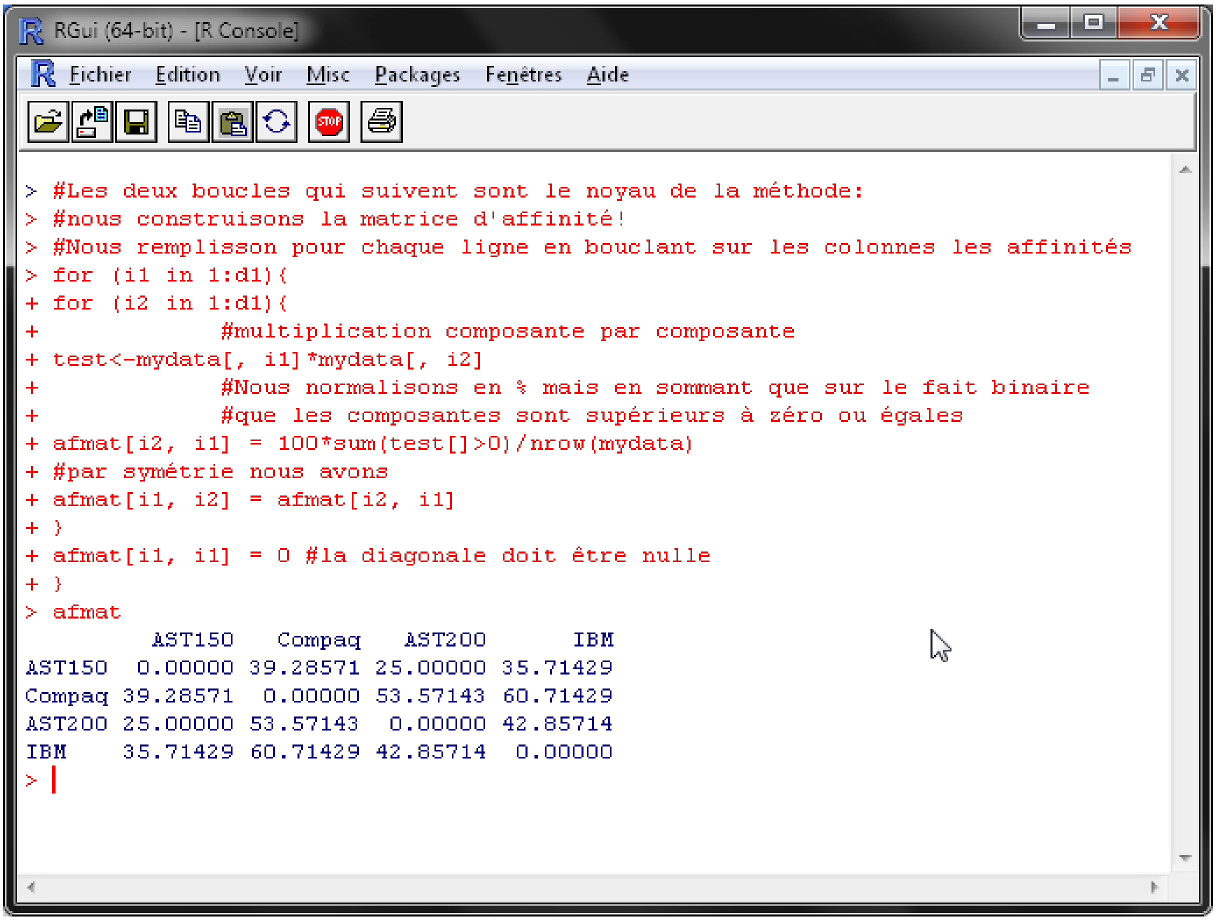
D'abord nous importons les données et vérifions que celles-ci soient bien importées et extrayons le nombre de colonnes:



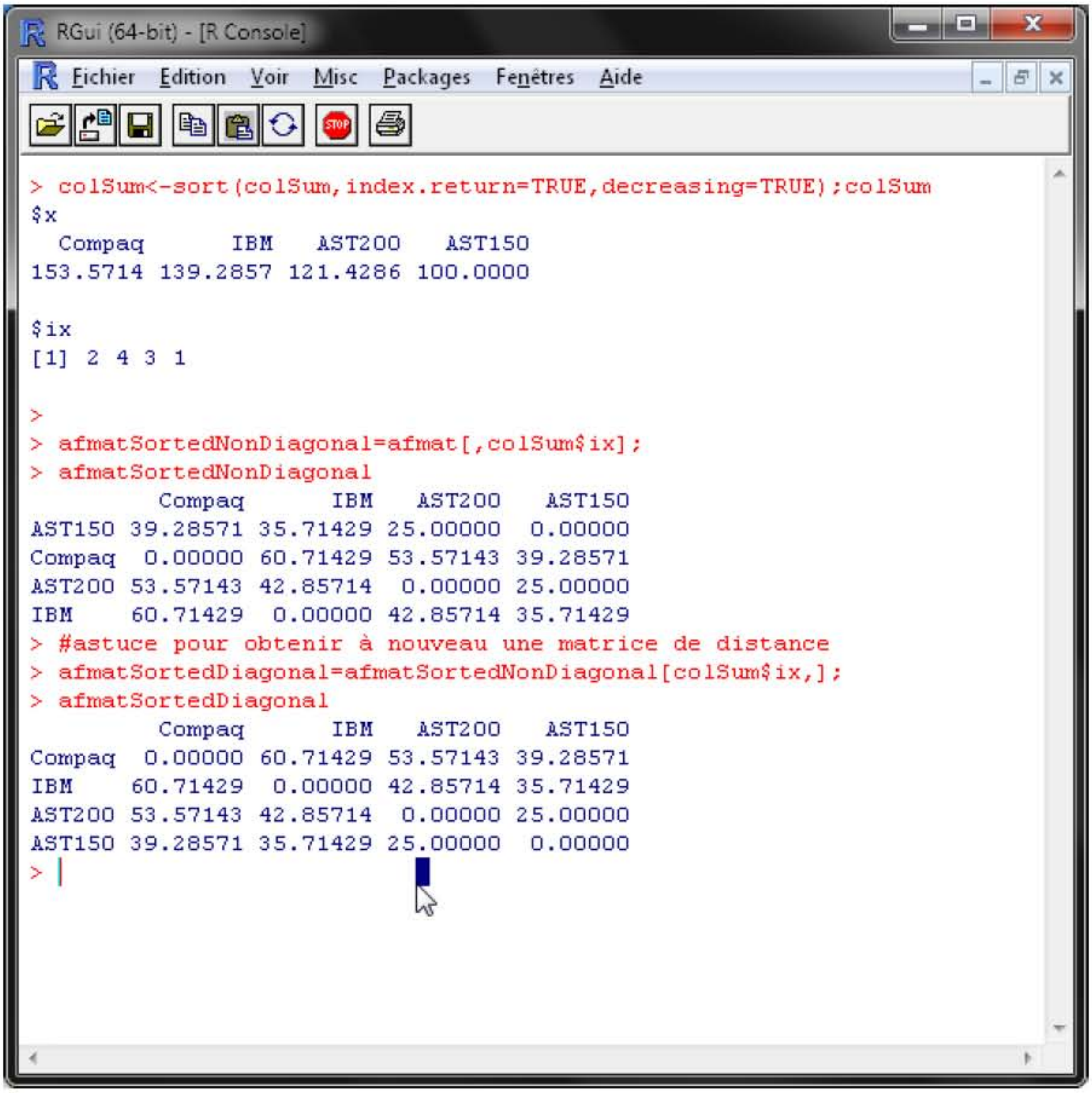
Une fois ceci fait, nous préparons la matrice d'affinité et nous renommons les colonnes au passage pour que leurs intitulés soient plus petits:



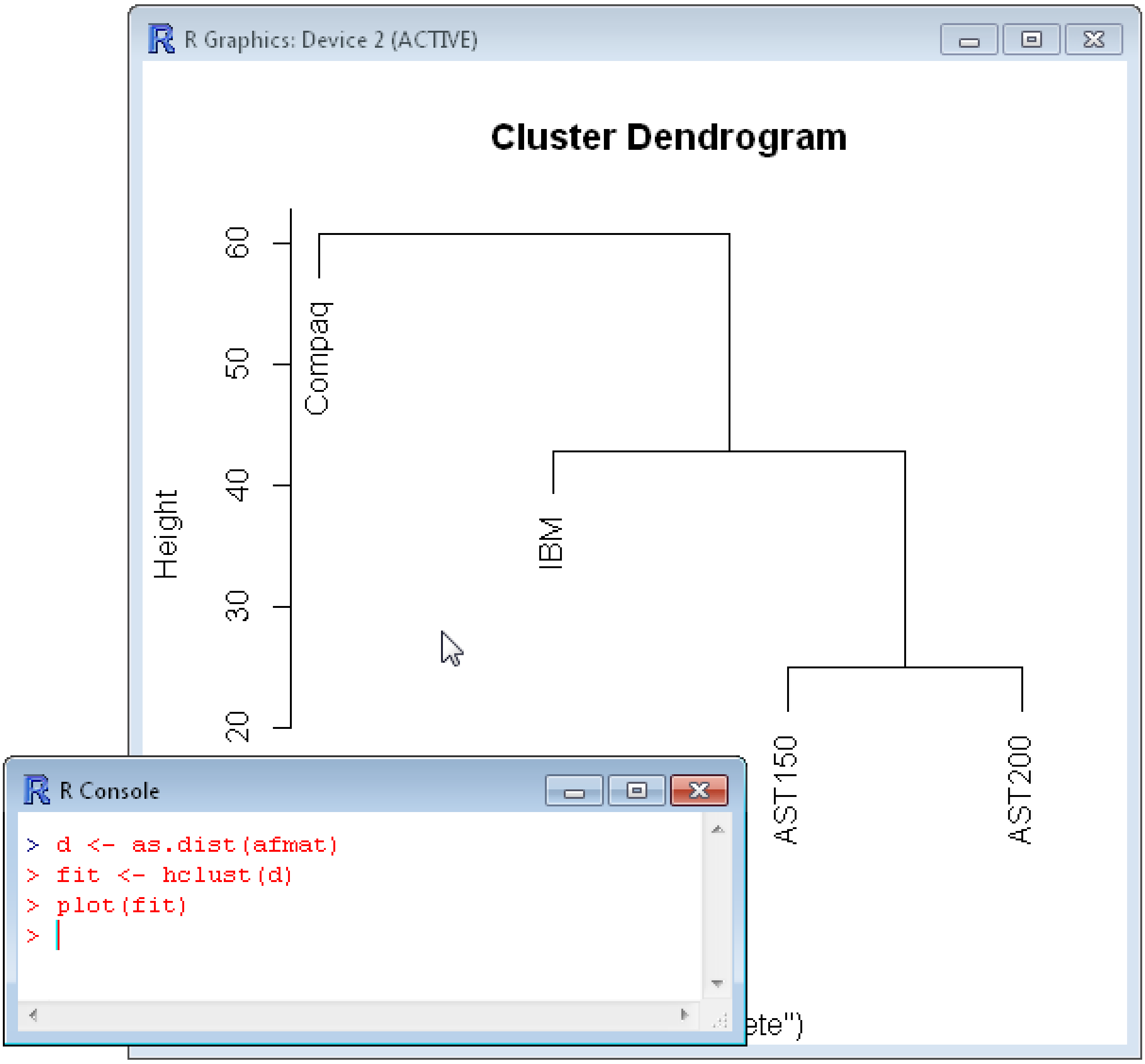
Maintenant vient le point névralgique de la méthode (c'est la partie "difficile"):



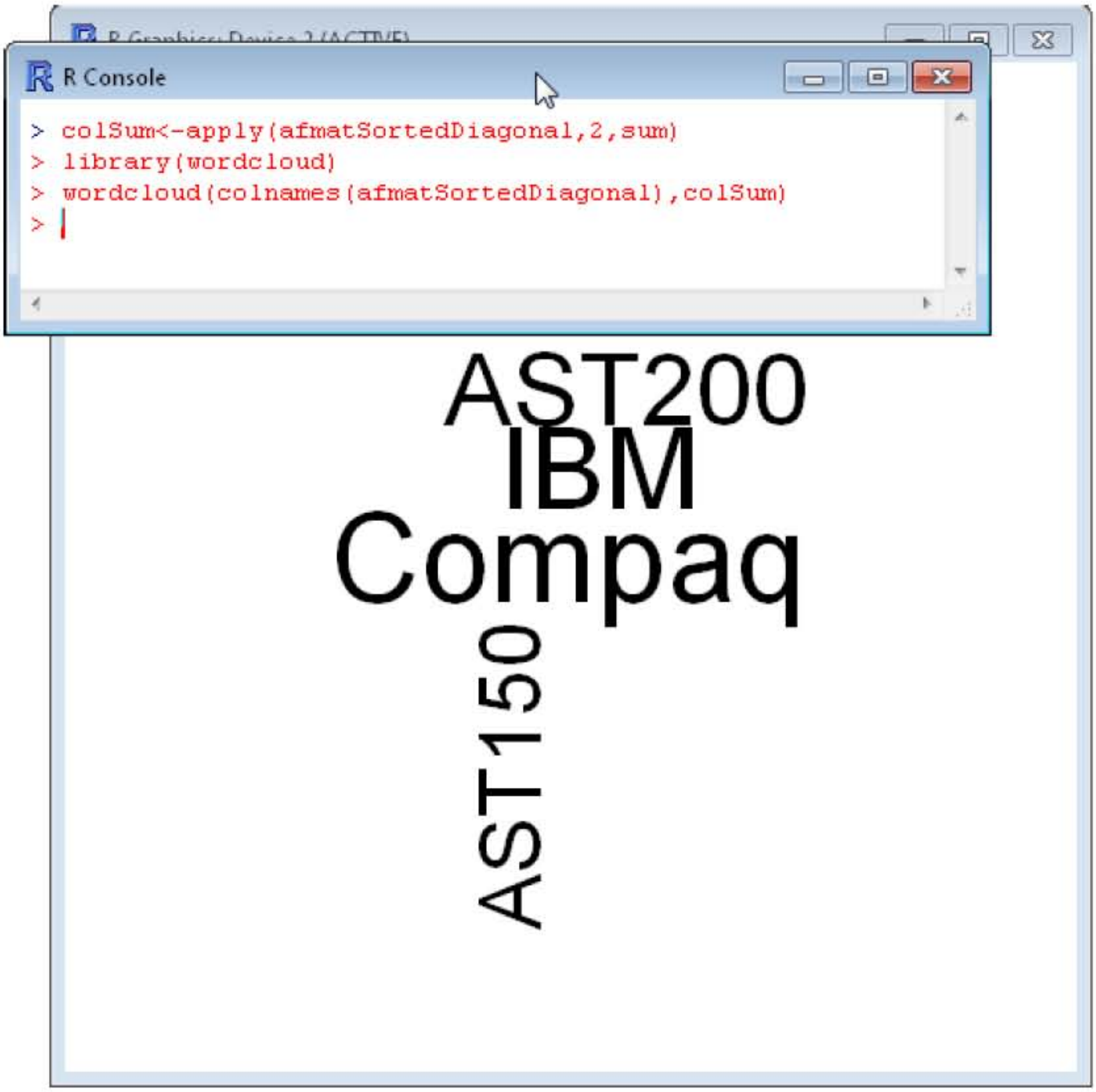
Pour le confort de lecture de cette matrice de distance nous allons trier les colonnes de plus influentes au moins influentes et en refaire une matrice de distance (mais cette petite transformation est totalement facultative et n'est pas une nécessité pour les graphiques que nous verrons plus loin):


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> colSum<-sort(colSum,index.return=TRUE,decreasing=TRUE);colSum
$x
Compaq IBM AST200 AST150
153.5714 139.2857 121.4286 100.0000
$ix
[1] 2 4 3 1
>
> afmatSortedNonDiagonal=afmat[,colSum$ix];
> afmatSortedNonDiagonal
Compaq IBM AST200 AST150
AST150 39.28571 35.71429 25.00000 0.00000
Compaq 0.00000 60.71429 53.57143 39.28571
AST200 53.57143 42.85714 0.00000 25.00000
IBM 60.71429 0.00000 42.85714 35.71429
> #astuce pour obtenir à nouveau une matrice de distance
> afmatSortedDiagonal=afmatSortedNonDiagonal[colSum$ix,];
> afmatSortedDiagonal
Compaq IBM AST200 AST150
Compaq 0.00000 60.71429 53.57143 39.28571
IBM 60.71429 0.00000 42.85714 35.71429
AST200 53.57143 42.85714 0.00000 25.00000
AST150 39.28571 35.71429 25.00000 0.00000
>

Et nous pouvons aussi faire quelques graphiques:



et pourquoi pas un petit nuage de mots:


R Console
> colSum<-apply(afmatSortedDiagonal,2,sum)
> library(wordcloud)
> wordcloud(colnames(afmatSortedDiagonal),colSum)
AST200
IBM
Compaq
AST150

Exemple 357.: Positionnement multidimensionnel

R 3.1.1

Nous avons étudié dans le cours théorique le positionnement multidimensionnel simple ("Multidimensional scaling" ou "M.D.S." en anglais) utilisant le coefficient de stress 1 de Kruskal.

Nous avons calculé "à la main" dans Microsoft Excel la matrice des distances et les vecteurs de position plongé dans un cas particulier de $\mathbb{R}^2$ à partir d'une matrice de dissimilarité et le but ici va être de vérifier si quelques fonctions disponibles dans R à cet effet donnent les mêmes valeurs ou permettent au moins d'arriver aux même conclusions (sachant qu'il de nombreuses méthodes empiriques d'implémentation).

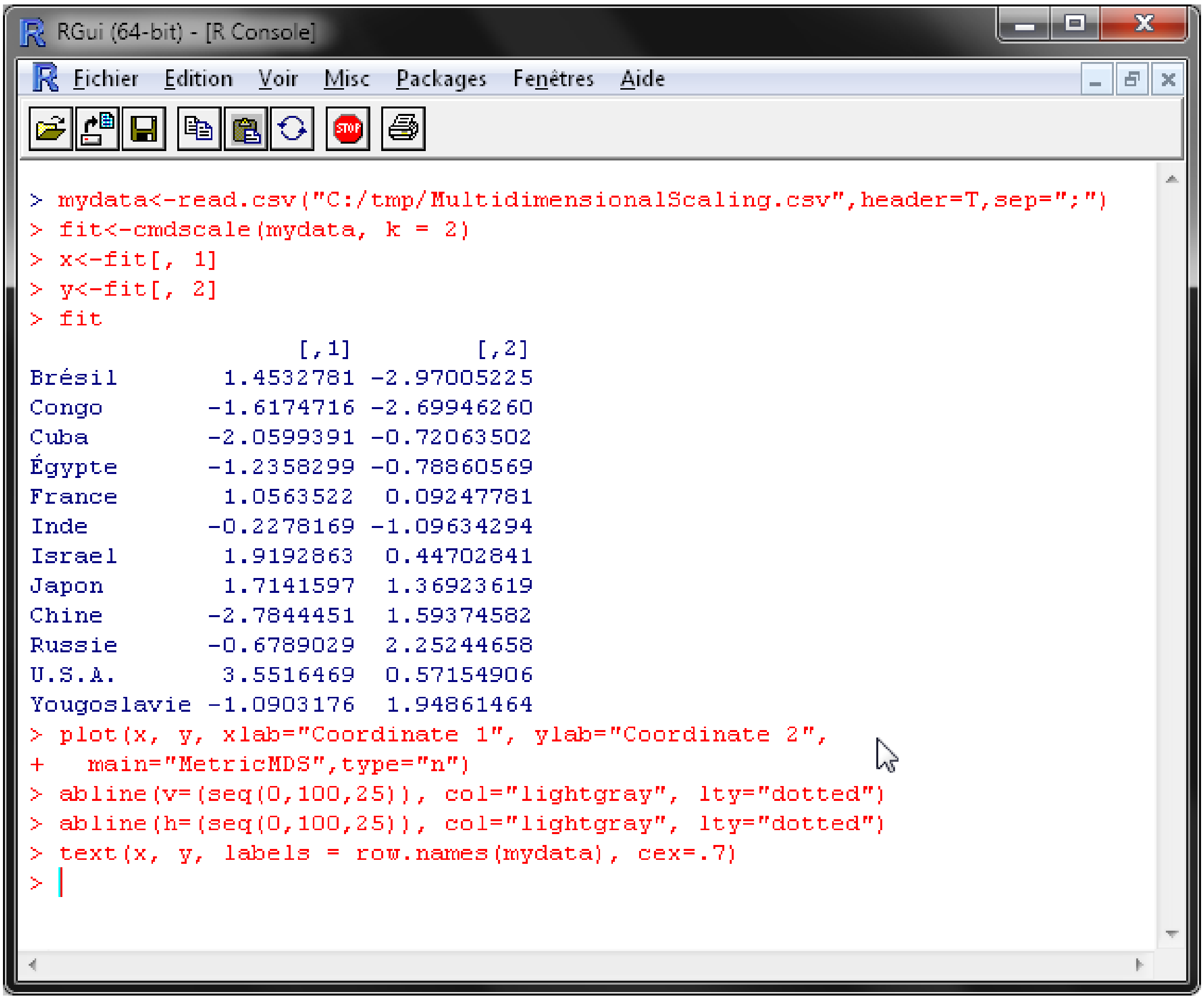
Le fichier brut utilisé sera le même que dans le cours théorique:

MultidimensionalScaling.csv - Bloc-notes

Fichier Edition Format Affichage ?

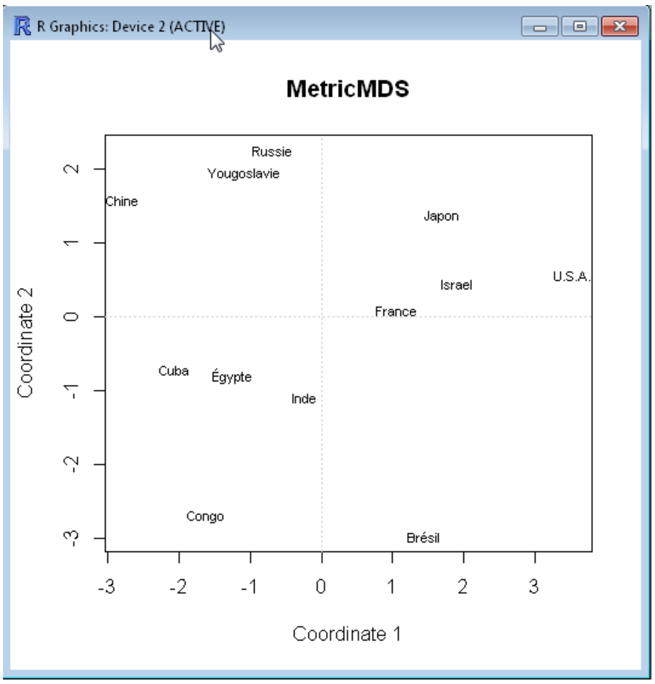
```
Brésil;Congo;Cuba;Égypte;France;Inde;Israel;Japon;Chine;Russie;U.S.A.;Yougoslavie
Brésil;0;4.17;3.72;5.56;4.28;4.5;5.17;5.5;6.61;5.94;3.61;5.83
Congo;4.17;0;4.44;4;5;4.17;5.67;5.61;5;5.61;6.61;5.5
Cuba;3.72;4.44;0;3.83;4.89;5;5.39;6.06;3.5;3.56;5.83;3.89
Égypte;5.56;4;3.83;0;4.22;3.17;4.33;5.17;4.61;4.61;5.67;4.72
France;4.28;5;4.89;4.22;0;5.56;5;4.78;5.33;3.94;3.06;4.28
Inde;4.5;4.17;5;3.17;5.56;0;4.89;4.5;4.89;4.5;4.72;5
Israel;5.17;5.67;5.39;4.33;5;4.89;0;4.17;6;4.83;3.06;4.56
Japon;5.5;5.61;6.06;5.17;4.78;4.5;4.17;0;4.83;4.39;2.94;4.72
Chine;6.61;5;3.5;4.61;5.33;4.89;6;4.83;0;3.28;6.44;3.94
Russie;5.94;5.61;3.56;4.61;3.94;4.5;4.83;4.39;3.28;0;4;2.33
U.S.A.;3.61;6.61;5.83;5.67;3.06;4.72;3.06;2.94;6.44;4;0;5.44
Yougoslavie;5.83;5.5;3.89;4.72;4.28;5;4.56;4.72;3.94;2.33;5.44;0
```

Voyons la première fonction **cmdscale( )** qui est une fonction native à R:

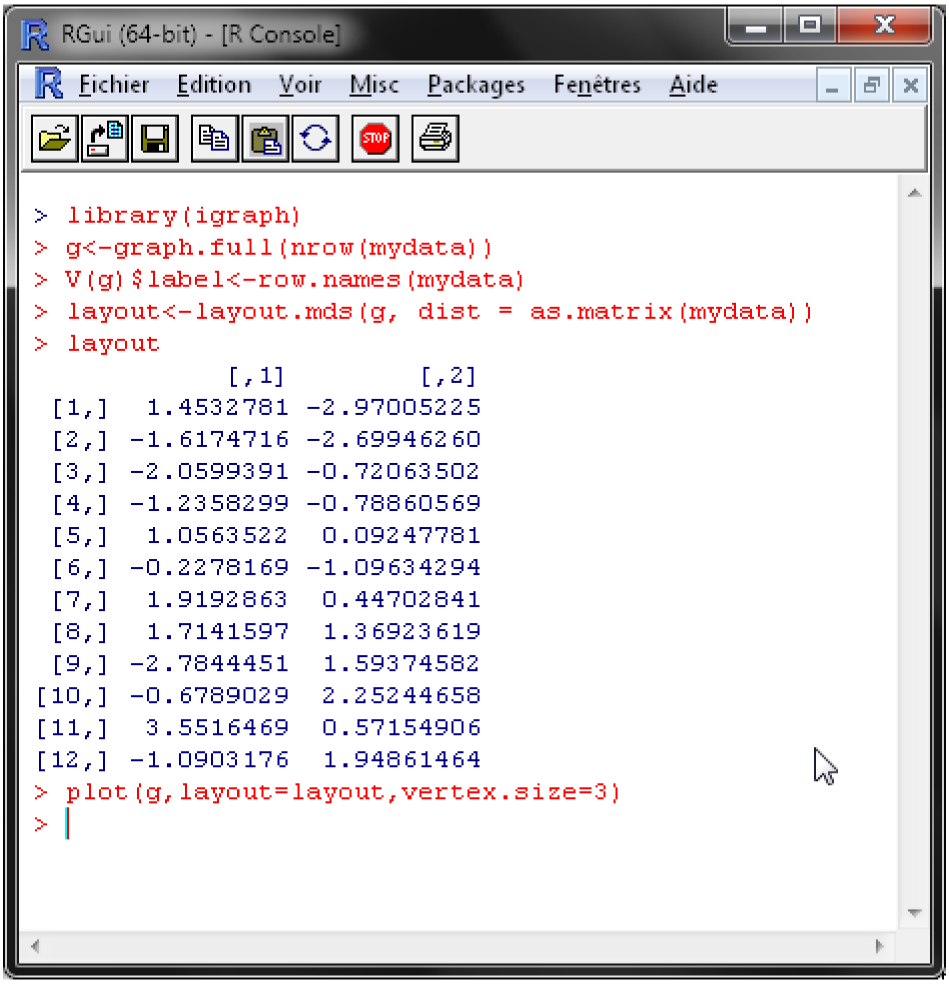


```
> mydata<-read.csv("C:/tmp/MultidimensionalScaling.csv",header=T,sep=";")
> fit<-cmdscale(mydata, k = 2)
> x<-fit[, 1]
> y<-fit[, 2]
> fit
                  [,1]        [,2]
Brésil       1.4532781 -2.97005225
Congo       -1.6174716 -2.69946260
Cuba        -2.0599391 -0.72063502
Égypte      -1.2358299 -0.78860569
France       1.0563522  0.09247781
Inde        -0.2278169 -1.09634294
Israel       1.9192863  0.44702841
Japon        1.7141597  1.36923619
Chine       -2.7844451  1.59374582
Russie      -0.6789029  2.25244658
U.S.A.       3.5516469  0.57154906
Yougoslavie -1.0903176  1.94861464
> plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
+   main="MetricMDS",type="n")
> abline(v=(seq(0,100,25)), col="lightgray", lty="dotted")
> abline(h=(seq(0,100,25)), col="lightgray", lty="dotted")
> text(x, y, labels = row.names(mydata), cex=.7)
> 
```

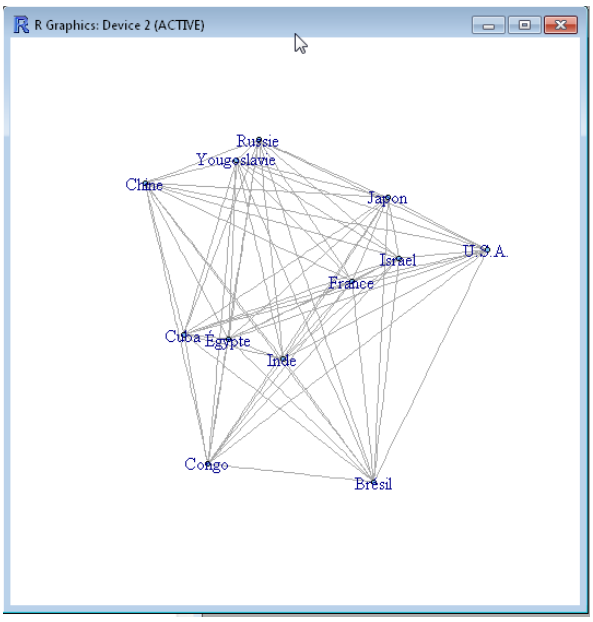
et graphiquement:



Donc a posteriori aussi au niveau des coordonnées obtenues nous sommes très loin de ce que nous avons obtenu dans le cours théorique mais au niveau qualitatif du graphique nous sommes très proches. Voyons cela visuellement avec la fonction **layout.mds( )** du package **igraph**:

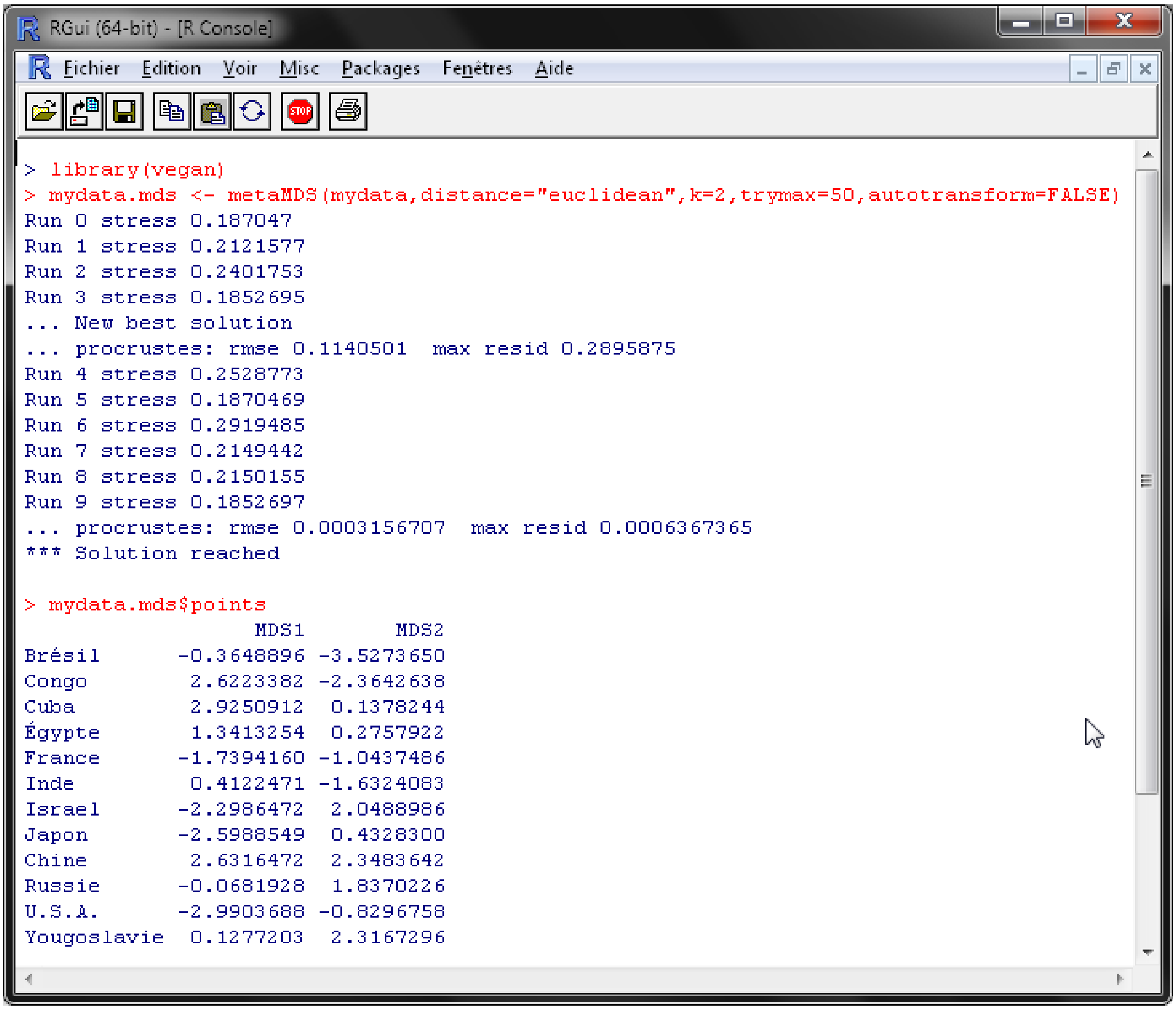


Et visuellement:



Donc bien que les coordonnées ne soient pas conformes au cours théorique (car le modèle à nouveau était le plus simpliste existant), les associations sont conformes.

Et maintenant voyons que les coordonnées peuvent vraiment varier suivant la méthode mais que la conclusion qualitative reste bien la même. L'avantage des fonctions du package **vegan** que nous allons voir maintenant est que nous retrouvons la fameuse variable de Stress explicitement avec l'optimisation relativement à sa minimisation (nous nous passerons de la construction du graphique qui est la même qu'avec la première méthode):



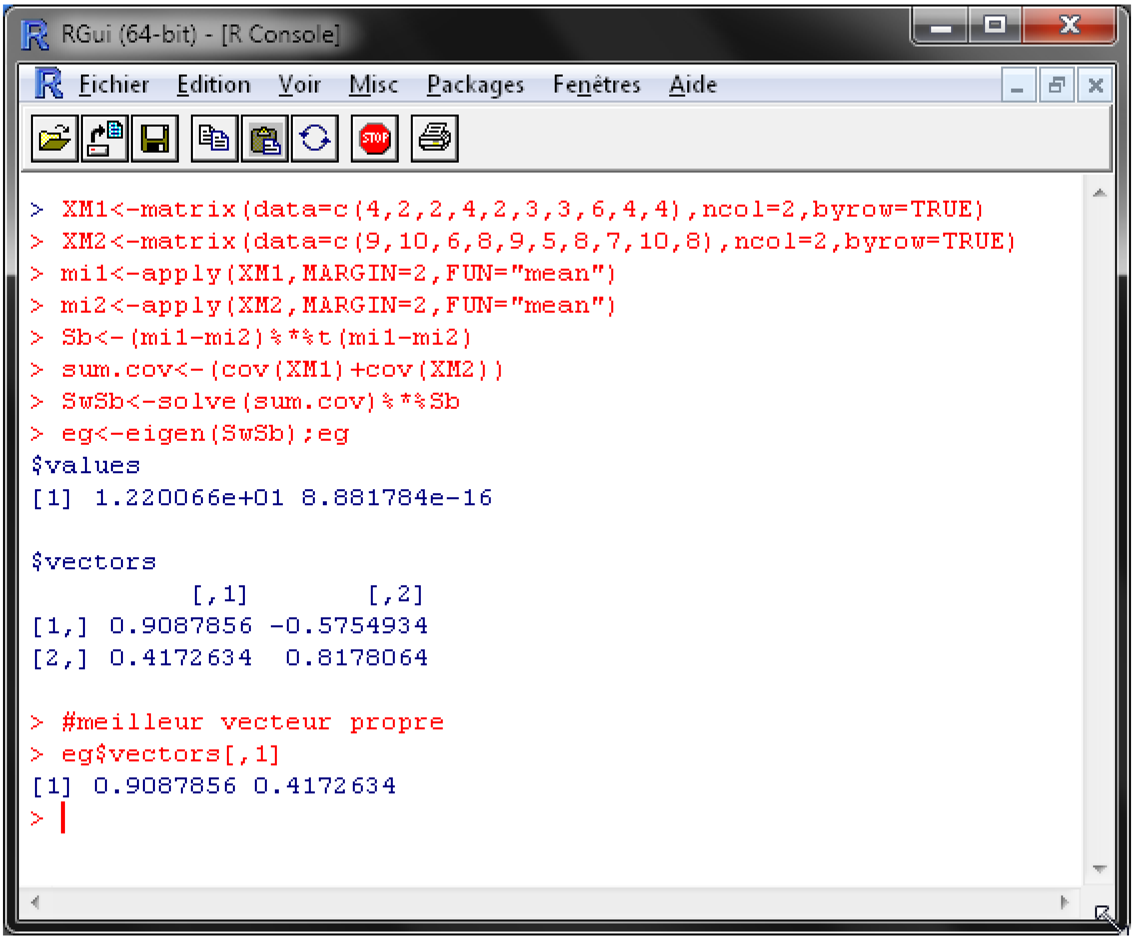
Ce dernier est par ailleurs le plus proche de MATLAB.

Exemple 358.: Analyse discriminante linéaire (ADL)

R 3.0.2

Nous allons voir ici si nous retombons avec R sur les calculs effectués dans le cours théorique où nous avons utilisé la LDA avec le discriminant de Fisher.

Nous utiliserons donc les mêmes données et pour des raisons pédagogiques nous allons d'abord le faire sans package:

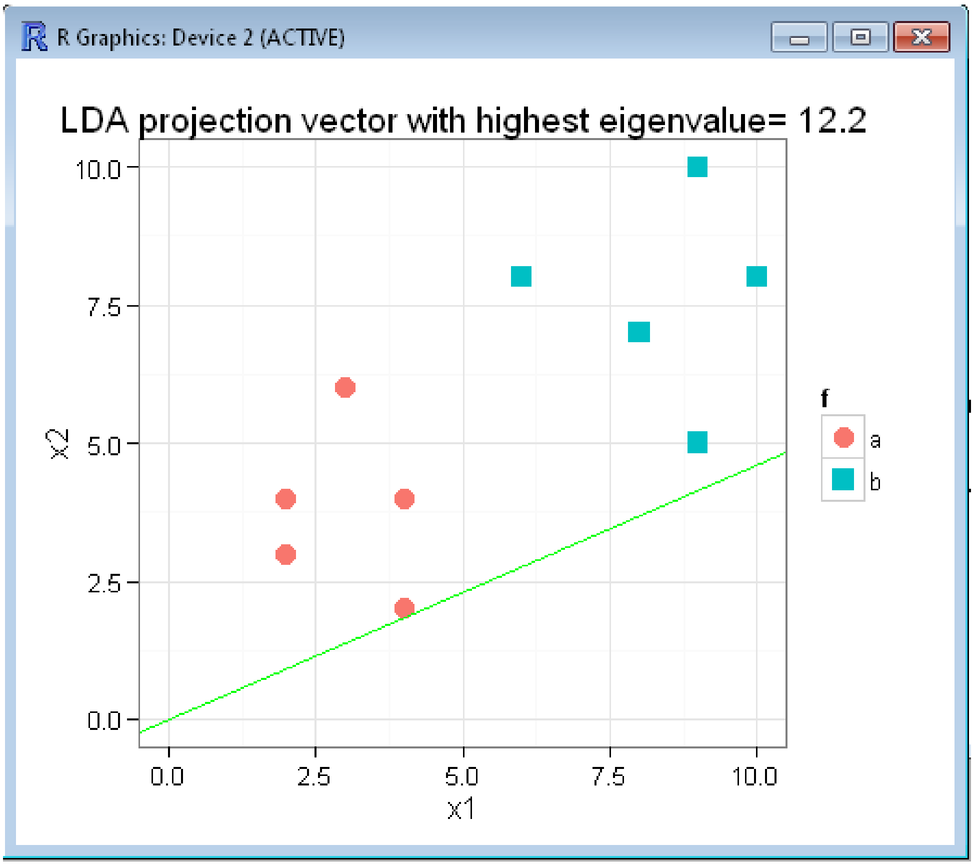


Nous retrouvons donc exactement la même valeur et le même vecteur propre que dans les calculs effectués à la main dans le cours théorique.

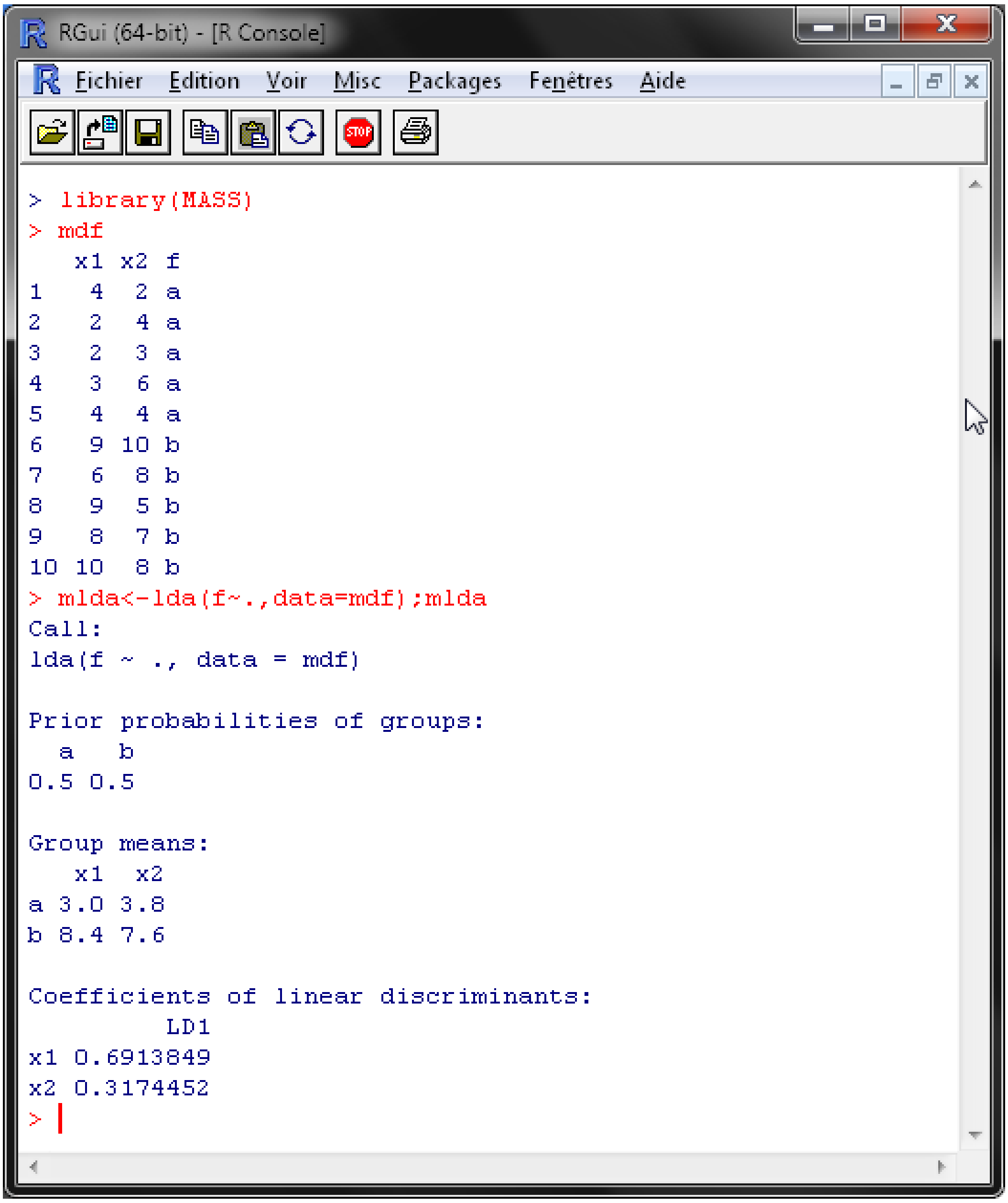
Faisons maintenant une représentation graphique de cela avec le package **ggplot2**:

```
> mdf<-as.data.frame(rbind(XM1,XM2))
> names(mdf)<-c("x1","x2")
> mdf$f<-c(rep("a",nrow(XM1)),rep("b",nrow(XM2)))
> mdf
   x1 x2 f
1   4  2 a
2   2  4 a
3   2  3 a
4   3  6 a
5   4  4 a
6   9 10 b
7   6  8 b
8   9  5 b
9   8  7 b
10 10  8 b
> library(ggplot2)
> ggplot(mdf,aes(x=x1,y=x2,col=f))+
+ geom_point(size=4,aes(shape=f))+
+ geom_abline(slope=eg$vectors[2,1]/eg$vectors[1,1],colour="green")+
+ scale_shape_manual(values=c(16,15))+
+ expand_limits(y=0,x=0)+
+ labs(title=paste("LDA projection vector with highest eigenvalue=",round(eg$values[1],2)))+
+ theme_bw()
> 
```

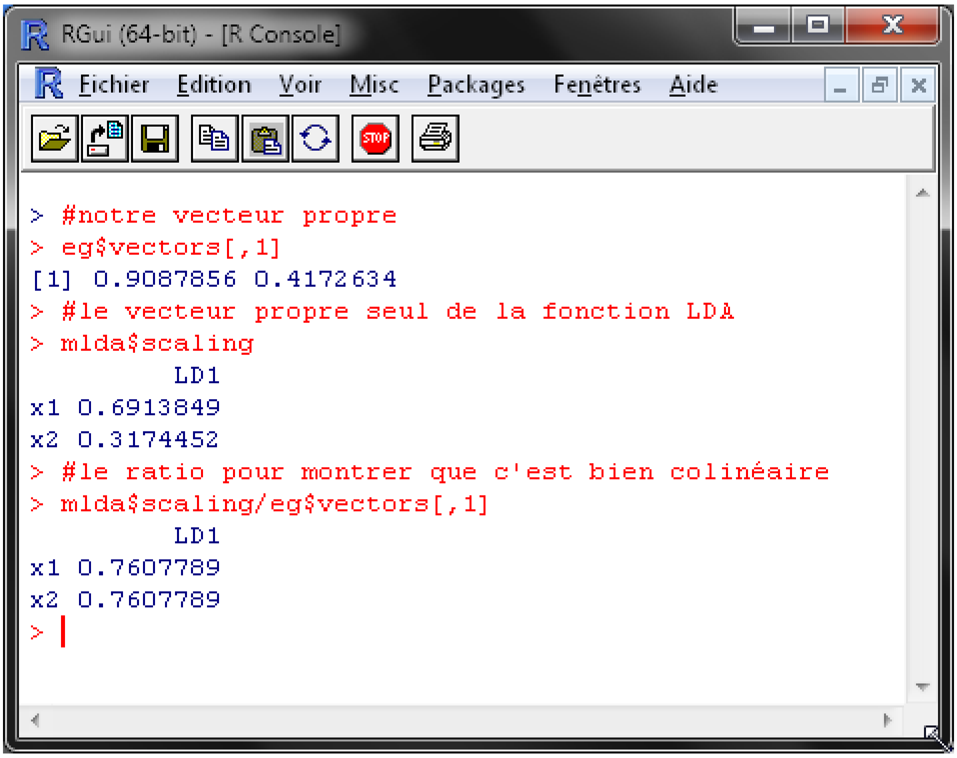
Ce qui donne:



Maintenant voyons avec la fonction **lda( )** intégrée à R:

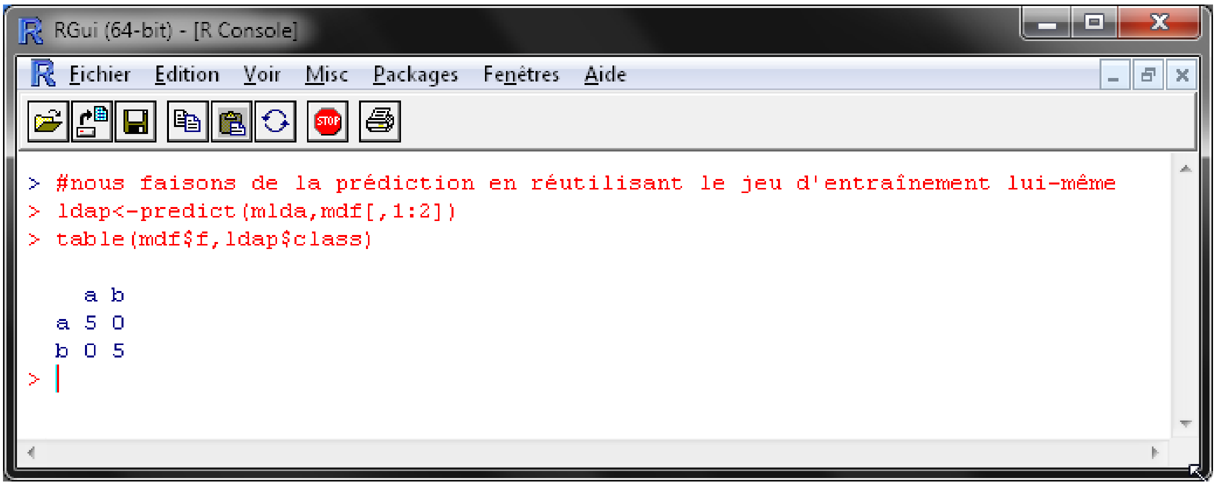


A priori nous ne retrouvons pas le vecteur propre mais en réalité il n'en est rien. C'est juste qu'avec la méthode vue dans le cours théorique le vecteur propre était normalisé alors qu'ici ce n'est pas le cas. Nous pouvons vérifier que le vecteur LD1 est bien colinéaire au vecteur propre obtenu avec la méthode vue dans le cours théorique:

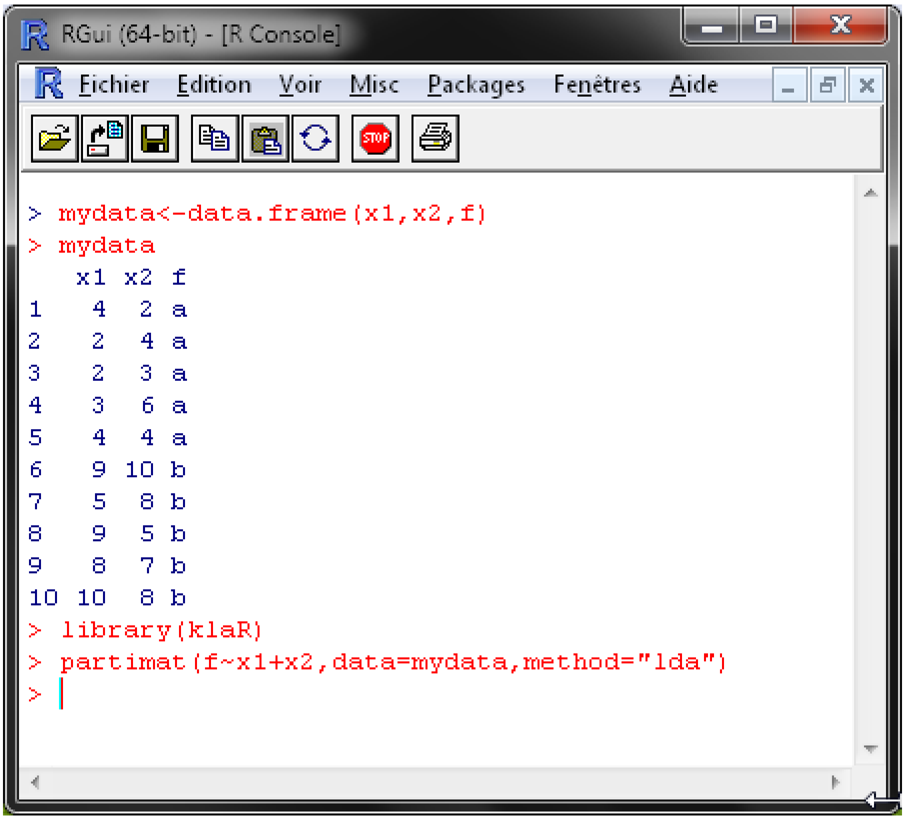


C'est donc bien colinéaire!

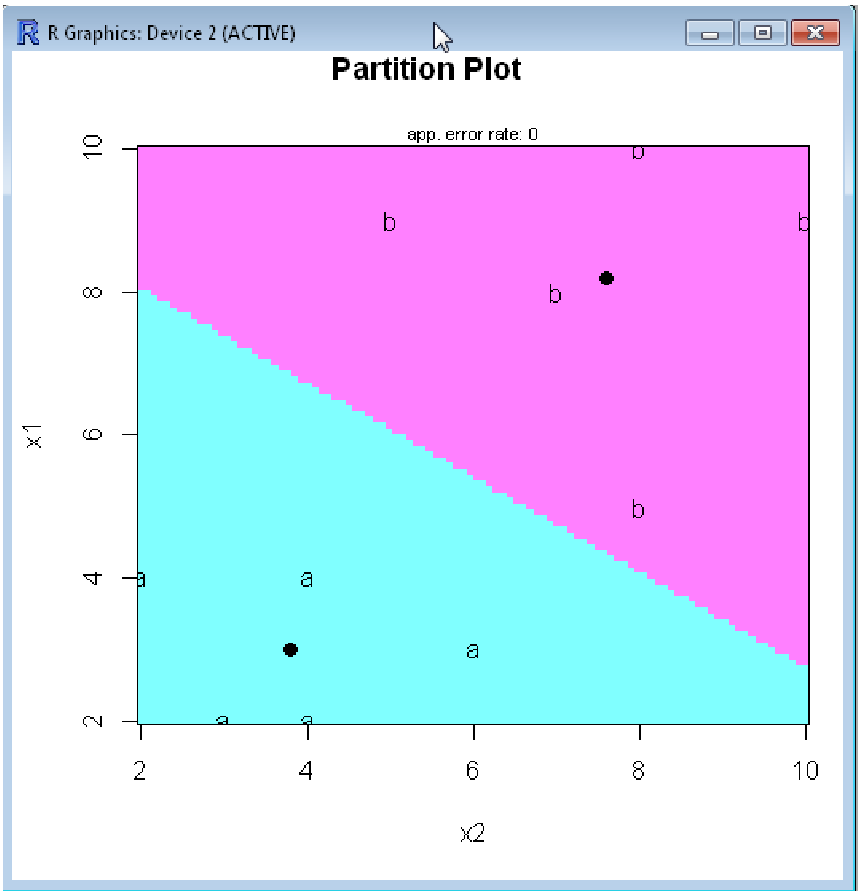
Maintenant voyons qu'il est possible d'utiliser la LDA pour faire de la classification (via la distance à la moyenne des groupes) et obtenons par la même occasion une matrice de confusion:



Avec la commande **partimat( )** du package **klaR** nous pouvons afficher les polygones de séparation des groupements:



Ce qui donne la frontière (droite) qui maximise les distances entre les points des groupes:

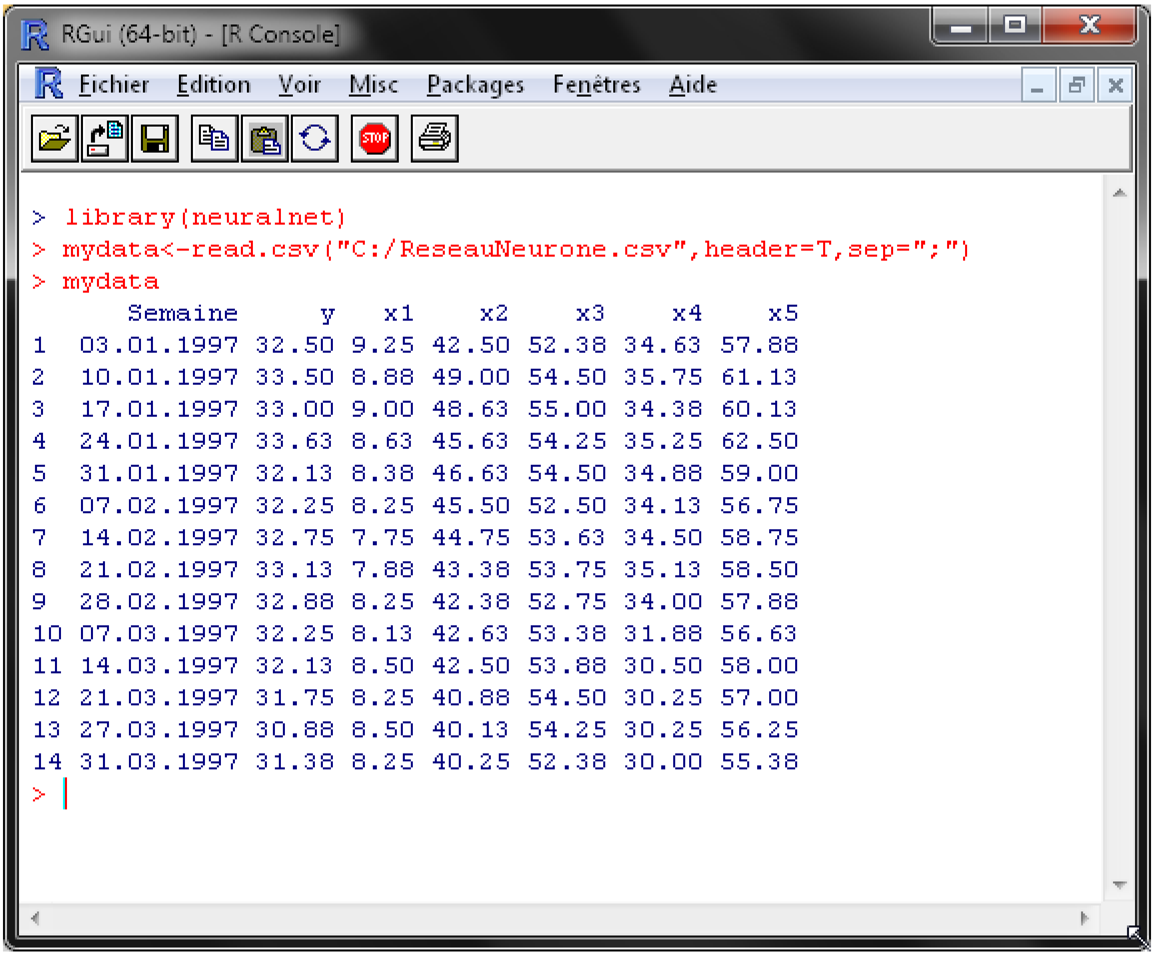


Exemple 359.: Réseaux de neurones feed-forward (FFNN)

R 3.0.2

Réseaux de neurones FFNN avec fonction d'activation logistique (binomiale)

Le but ici va être de comparer le modèle de neurones construit dans le cours théorique avec le tableur Microsoft Excel (avec pour avantage que l'on peut construire un réseau de neurones multicouches avec n'importe quelle fonction de cut-off) par rapport à celui que nous allons obtenir ci-dessous avec la commande **neuralnet( )** du package **neuralnet** (il y a d'autres package comme **nnet**, **RSNNS** ou encore **AMORE**) en utilisant le même jeu de données:



```
> library(neuralnet)
> mydata<-read.csv("C:/ReseauNeurone.csv",header=T,sep=";")
> mydata
      Semaine     y   x1    x2    x3    x4    x5
1  03.01.1997 32.50 9.25 42.50 52.38 34.63 57.88
2  10.01.1997 33.50 8.88 49.00 54.50 35.75 61.13
3  17.01.1997 33.00 9.00 48.63 55.00 34.38 60.13
4  24.01.1997 33.63 8.63 45.63 54.25 35.25 62.50
5  31.01.1997 32.13 8.38 46.63 54.50 34.88 59.00
6  07.02.1997 32.25 8.25 45.50 52.50 34.13 56.75
7  14.02.1997 32.75 7.75 44.75 53.63 34.50 58.75
8  21.02.1997 33.13 7.88 43.38 53.75 35.13 58.50
9  28.02.1997 32.88 8.25 42.38 52.75 34.00 57.88
10 07.03.1997 32.25 8.13 42.63 53.38 31.88 56.63
11 14.03.1997 32.13 8.50 42.50 53.88 30.50 58.00
12 21.03.1997 31.75 8.25 40.88 54.50 30.25 57.00
13 27.03.1997 30.88 8.50 40.13 54.25 30.25 56.25
14 31.03.1997 31.38 8.25 40.25 52.38 30.00 55.38
> 
```

Notez qu'au jour où nous écrivons ces lignes, ces packages ne prennent pas en compte les variables catégorielles (si non recodées en valeurs entières: "hot encoding").

Nous ne nous attarderons pas trop longtemps sur cette fonctionnalité dans R car les neurones à choix (quel que soit le package) sont très basiques et le manque de flexibilité de construction fait que même un tableur performe souvent mieux (c'est d'ailleurs ce que nous allons pouvoir constater ici).

Un réseau de neurones à une couche nous donnera dans le cas présent:

```
> nn_h1<-neuralnet(y~x1+x2+x3+x4+x5,data=mydata,hidden=1)
> nn_h1
Call: neuralnet(formula = y ~ x1 + x2 + x3 + x4 + x5, data = mydata,     hidden = 1)

1 repetition was calculated.

        Error Reached Threshold Steps
1 3.960200004    0.0003203286332   182

> plot(nn_h1)
>
```

Remarque: La fonction **neuralnet( )** ne prend pas en compte au jour où nous écrivons ces lignes la syntaxe y~. Donc si vous avez un long data frame, le mieux est d'utiliser l'astuce suivante (ce qui au passage est un superbe exemple d'application de **as.formula( )** !):

```
> n<-names(mydata)
> f<-as.formula(paste("y~",paste(n[!n %in% "pred_con"],collapse="+")))
> f
y ~ y + X1 + X2 + X3 + X4 + X5
>
```

Et finalement avec le graphique associé (**n'oubliez pas qu'à chaque fois que vous relancerez le modèle sans utiliser un seet.seed( ), les poids ne seront pas les mêmes!**):

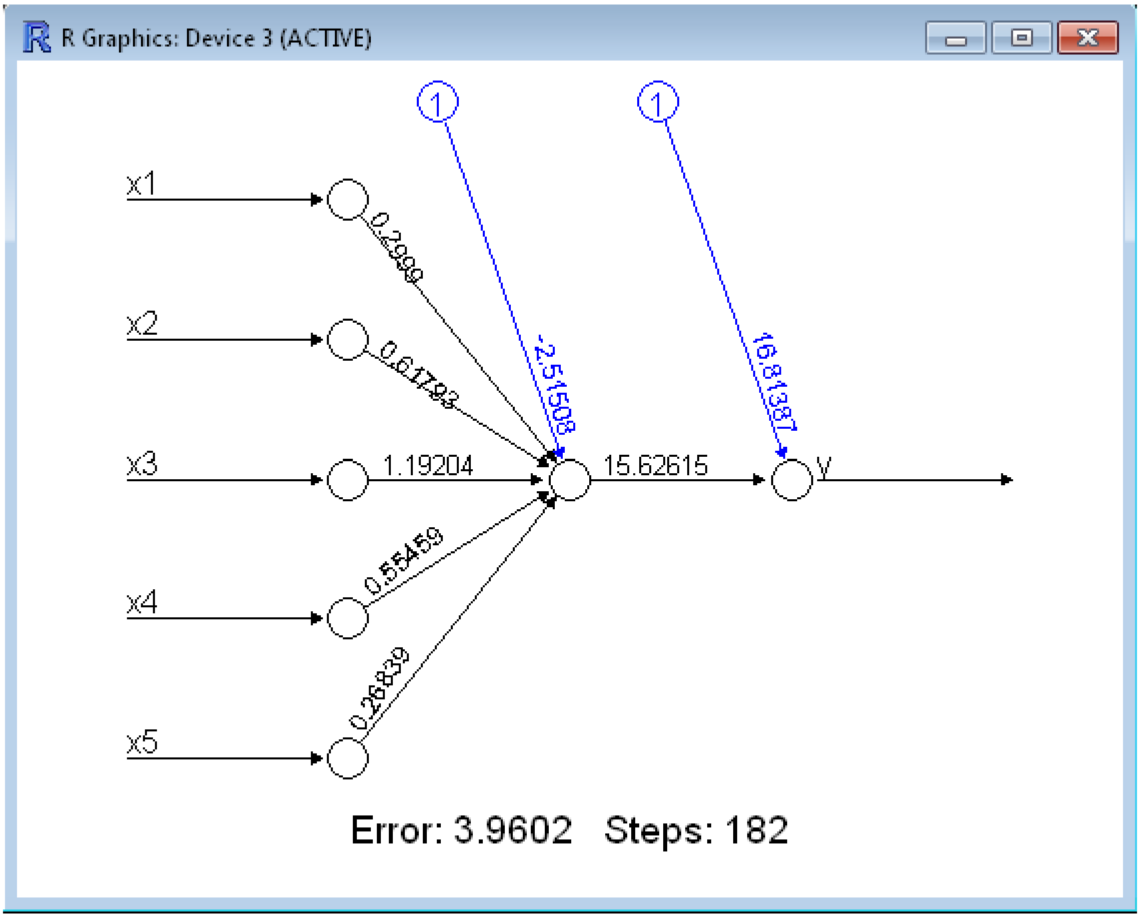


Figure 305 Graph réseau de neurones

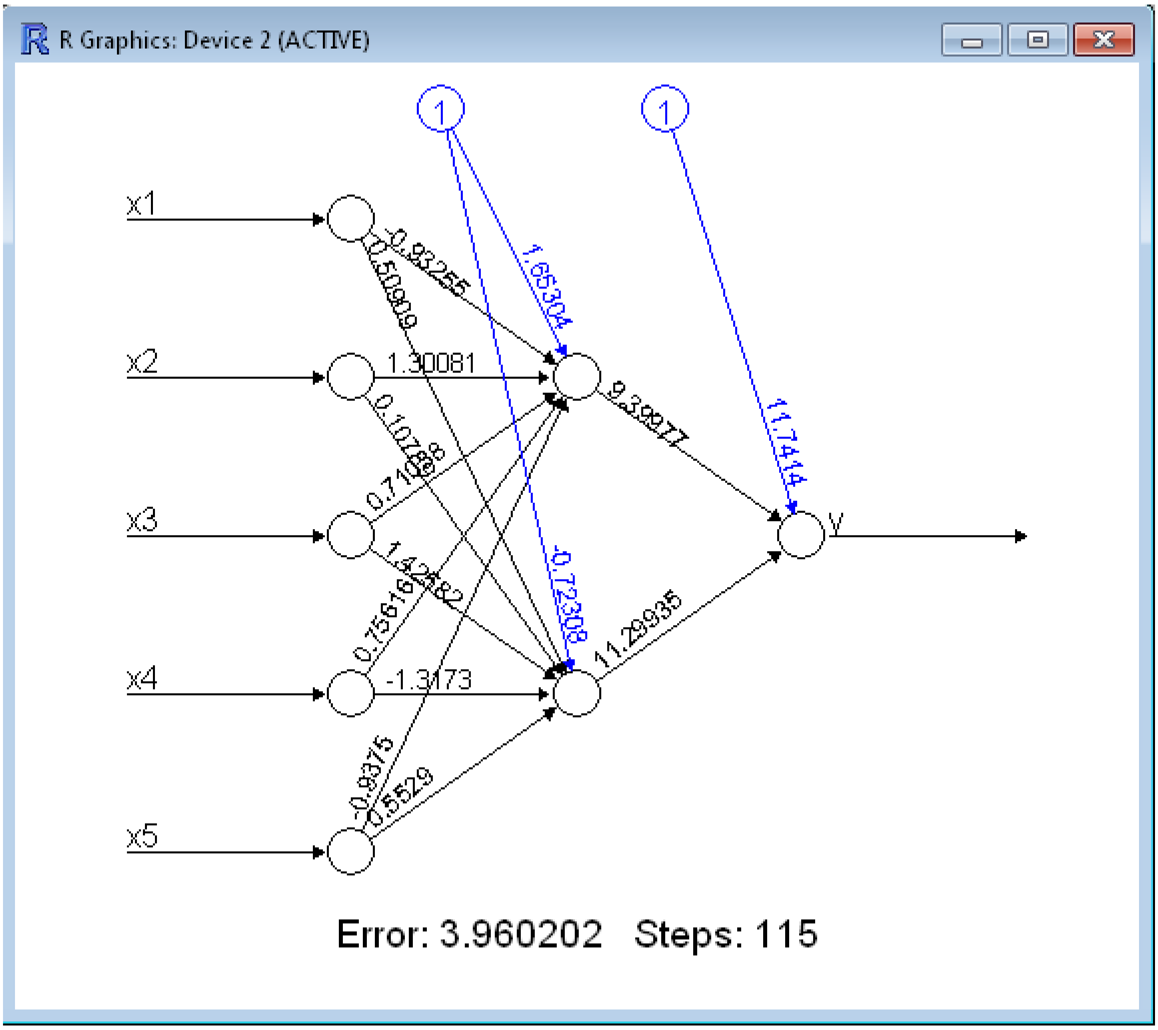
La somme des carrés des erreurs est de 3.9602 donc plus élevé que ce que nous avons obtenu avec le tableur. Faisons donc un réseau à deux couches:

```
> nn_h2<-neuralnet(y~x1+x2+x3+x4+x5,data=mydata,hidden=2)
> nn_h2
Call: neuralnet(formula = y ~ x1 + x2 + x3 + x4 + x5, data = mydata,     hidden = 2)

1 repetition was calculated.

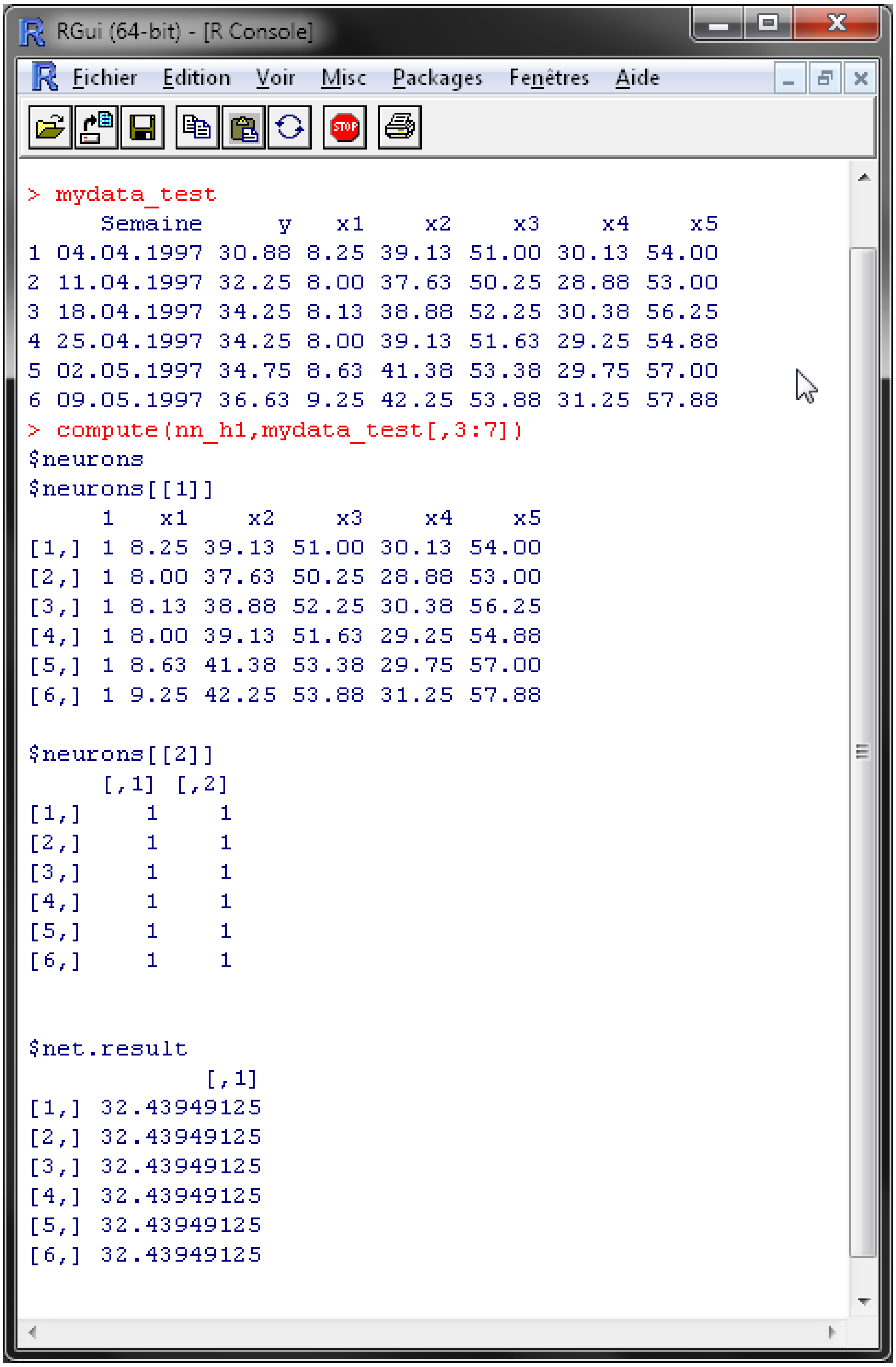
        Error Reached Threshold Steps
1 3.960201924    0.007339769443   115

> plot(nn_h2)
> |
```



Aucune amélioration significative... et nous pourrions continuer ainsi longtemps sans obtenir aucune amélioration.

Regardons cependant si ce modèle performe mieux sur un jeu de test n'étant pas inclus dans le jeu d'entraînement. Nous allons utiliser le même jeu que dans le cours théorique:

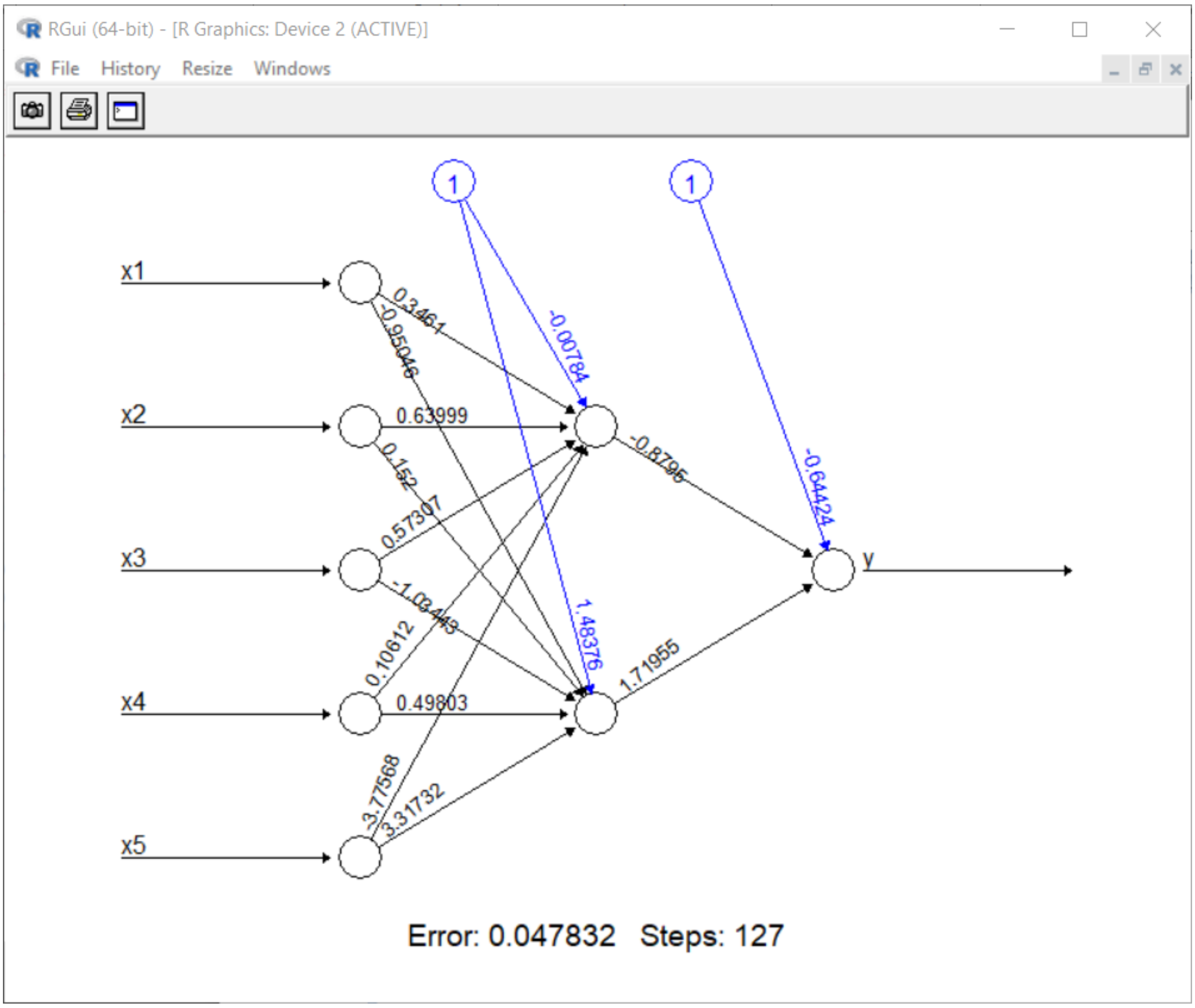


Comme nous pouvons le voir, ce réseau de neurone a un problème puisqu'il considère chaque entrée comme une valeur binaire. Ainsi, s'il existe un nombre en entrée il met celle-ci à la valeur 1 et ensuite il multiplie par le poids et fait la somme des biais, raison pour laquelle on se retrouve toujours avec la même valeur en sortie dans le cas présent. En réalité ce package est plutôt fait pour de la classification et moins pour prédire des valeurs des continues...

Cela n'empêche toutefois pas que nous pouvons normaliser les données en entrée et les dénormaliser ensuite en sortie:

```
> library("neuralnet")
>
> mydata<-read.csv("C:/tmp/ReseauNeurone.csv",header=T,sep=";")
>
> norm.fun = function(x){(x - min(x))/(max(x) - min(x))}
> train_data= apply(mydata[,2:7], 2, norm.fun)
>
> nn_ffd<-neuralnet(y~x1+x2+x3+x4+x5,data=train_data,hidden=2)
>
> mydata<-read.csv("C:/tmp/ReseauNeuroneTest.csv",header=T,sep=";")
>
> test_data= apply(mydata[,3:7], 2, norm.fun)
>
> output_data<-compute(nn_ffd,test_data)$net.result
>
> denormalized_output_data<-data.frame('y'=(min(mydata[, 'y'])
+ + output_data* (max(mydata[, 'y'])-min(mydata[, 'y']))))
>
> denormalized_output_data
            y
1 33.84199638
2 32.71658907
3 36.09684932
4 34.96983876
5 35.98005059
6 36.22073714
>
```

avec le diagramme correspondant où on peut voir que cette fois l'erreur est significativement inférieure:



Nous allons voir si nous pouvons faire mieux avec un autre package, nommé **nnet** (**n'oubliez pas qu'à chaque fois que vous relancerez le modèle, les poids ne seront pas les mêmes!**) et sa fonction **nnet( )**:

```
R Console
> ?nnet
> library(nnet)
> mydata<-read.csv("C:/ReseauNeurone.csv",header=T,sep=";")
> mnnet<-nnet(y~x1+x2+x3+x4+x5,data=mydata,size=5,skip=TRUE,linout=TRUE,decay=0.025)
# weights:  41
initial  value 3733.721913
iter  10 value 5.112277
iter  20 value 2.200506
iter  30 value 2.076809
iter  40 value 2.063702
iter  50 value 1.923754
iter  60 value 1.856845
iter  70 value 1.844442
iter  80 value 1.827847
iter  90 value 1.813017
iter 100 value 1.811880
final  value 1.811880
stopped after 100 iterations
>
```

Ensuite, comparons déjà la valeur prédite aux valeurs originales pour le jeu d'entraînement:

```
>
> result<-as.data.frame(cbind(mydata$y,predict(mnnet,mydata[,3:7])))
> names(result)<-c("Valeurs originales","Valeurs prédites")
> result
   Valeurs originales Valeurs prédites
1               32.50      32.54855144
2               33.50      33.37719399
3               33.00      33.08323113
4               33.63      33.74447270
5               32.13      32.79131699
6               32.25      31.97742705
7               32.75      32.77867100
8               33.13      33.18870976
9               32.88      32.65281478
10              32.25      31.68170711
11              32.13      32.12829152
12              31.75      31.76838707
13              30.88      31.39874607
14              31.38      30.99853430
> ErreurQuadratique<-sum((result[,1]-result[,2])^2)
> ErreurQuadratique
[1] 1.342899524
>
```

Ce qui pour cette exécution est moins bon que ce que nous avons obtenu avec le tableur mais si vous exécutez plusieurs fois vous verrez que la valeur peut diminuer en-dessous de 0.8 que nous avions obtenu avec Microsoft Excel. Il est aussi possible d'augmenter le nombre d'unités dans la couche neuronale.

Nous pouvons comparer aussi avec le même jeu test:

```
R Console
> mydata_test<-read.csv("C:/ReseauNeuroneTest.csv",header=T,sep=";")
> mydata_test<-read.csv("C:/ReseauNeuroneTest.csv",header=T,sep=";")
> result<-as.data.frame(cbind(mydata_test$y,predict(mnnet,mydata_test[,3:7])))
> names(result)<-c("Valeurs originales","Valeurs prédites")
> result
  Valeurs originales Valeurs prédites
1              30.88      30.95488298
2              32.25      30.30051776
3              34.25      31.18967699
4              34.25      30.98574011
5              34.75      31.65250404
6              36.63      31.98806444
> ErreurQuadratique<-sum((result[,1]-result[,2])^2)
> ErreurQuadratique
[1] 54.96910505
> 
```

Donc nous faisons juste un peu moins bien... mais c'est le hasard car si vous relancez vous verrez que vous ferez mieux que le réseau de neurone créé dans le cours théorique.

Réseaux de neurones FFNN avec fonction d'activation Softmax (multinomiale)

Nous allons reprendre l'exemple pris lors de notre étude la régression logistique multinomiale:

```
> mydata<-read.csv("c:/tmp/Iris.csv",header=T,sep=",")
> head(mydata)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> summary(mydata)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> 
```

Nous entraînons le réseau de neurones avec la fonction **multinom( )** du package **nnet**:

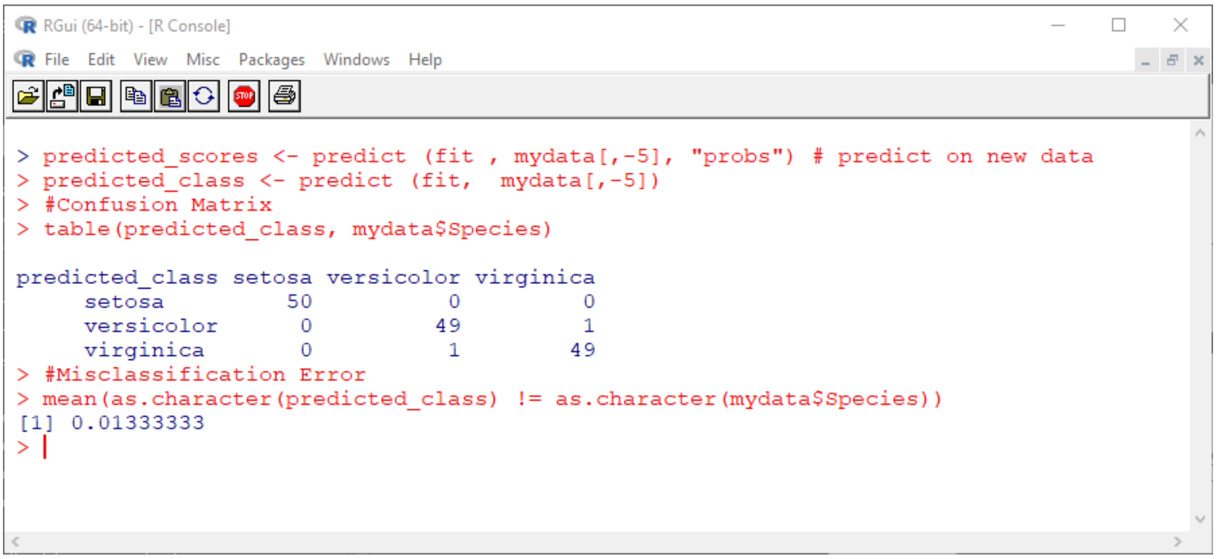
```
> library("nnet")
> fit <- multinom(Species~.,data=mydata)
# weights:  18 (10 variable)
initial  value 164.791843
iter  10 value 16.177348
iter  20 value 7.111438
iter  30 value 6.182999
iter  40 value 5.984028
iter  50 value 5.961278
iter  60 value 5.954900
iter  70 value 5.951851
iter  80 value 5.950343
iter  90 value 5.949904
iter 100 value 5.949867
final  value 5.949867
stopped after 100 iterations
> summary(fit)
Call:
multinom(formula = Species ~ ., data = mydata)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    18.69037    -5.458424   -8.707401     14.24477   -3.097684
virginica    -23.83628    -7.923634  -15.370769     23.65978   15.135301

Std. Errors:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    34.97116     89.89215    157.0415     60.19170    45.48852
virginica     35.76649     89.91153    157.1196     60.46753    45.93406

Residual Deviance: 11.89973
AIC: 31.89973
> 
```

Nous pouvons voir comment le réseau de neurones entraîne prédit de nouvelles données ou les données d'entraînement elles-mêmes toujours à l'aide de la fameux fonction **predict( )**:



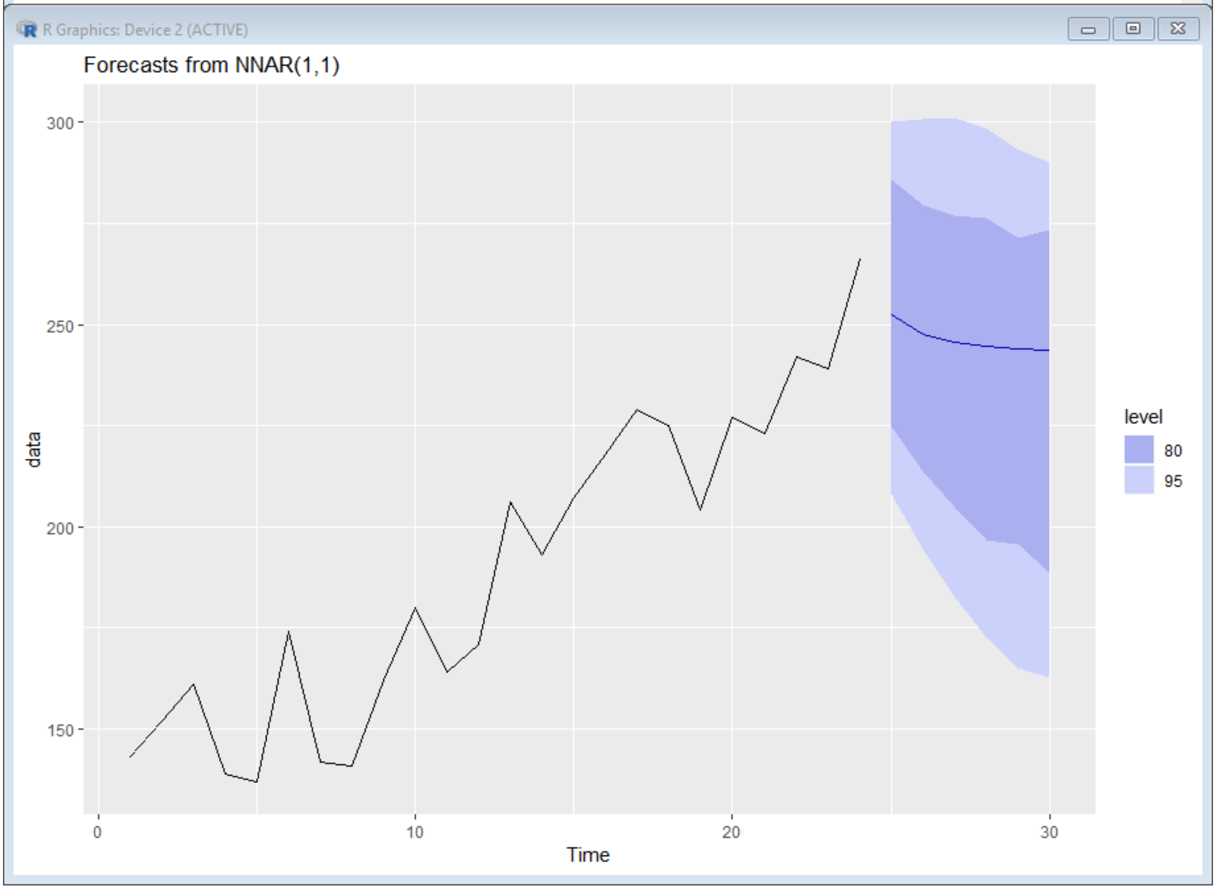
Le lecteur remarquera que nous avons ajouté la matrice de confusion et l'erreur moyenne de mauvaise classification.

Réseaux de neurones FFNN auto-régressif

Il s'agit ici d'un réseau de neurones assez classique on dans la série temporelle on choisit que les *p* dernières valeurs et *k* le nombre de nœuds du réseau de neurone FFNN à une couche masquée (les nœuds sont des fonctions sigmoïdes) qui au final passent toutes par une fonction linéaire (il y plein de schémas sur internet qu'on peut trouver).

Voici un exemple avec un jeu de données que nous réutiliseront beaucoup dans le chapitre sur les séries temporelles:

```
> library("forecast")
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ 204,227,223,242,239,266)
> #réseau de neurone NNAR(p,k)
> #p le nombre des dernières valeurs à prendre en compte
> #k le nombre de noeuds (nodes) dans la couche cachée
> fit <- nnetar(data, lambda=0)
> #prédiction avec intervalles de prédictions (PI) simulés
> #(attention la simulation peut être longue)
> autoplot(forecast(fit, PI=TRUE, h=6))
>
```



Exemple 360.: Réseaux de neurones récurrents Vanilla RNN, GRU et LSTM

R 3.6.2

En un seul script, voyons comment entraîner les trois réseaux de neurones dont nous avons parcouru quelques éléments mathématiques dans le cours théoriques:

- Vanilla RNN (Recurrent Neural Network)
- GRU (Gated Recurrent Unit)
- LSTM (Long-Short Term Memory)

L'exemple est pris et adapté du lien suivant (Ivo Dinov):

http://www.socr.umich.edu/people/dinov/courses/DSPA_notes/18_BigLongitudinalDataAnalysis_RNN.Rmd

Voyons d'abord comment importer des données les données climatiques 2009-2017 de l'Institut Max Planck de Jena, en Allemagne:

```
> rm(list=ls())
> #le fichier est aussi disponible en local sous le nom: ClimateData_2009_2017_Jena_Germany_MaxPlanck.zip
> clim_data_url <- "https://umich.instructure.com/files/8014703/download?download_frd=1"
> clim_data_zip_file <- tempfile(); download.file(clim_data_url, clim_data_zip_file, mode="wb")
trying URL 'https://umich.instructure.com/files/8014703/download?download_frd=1'
Content type 'application/x-zip-compressed' length 15173218 bytes (14.5 MB)
downloaded 14.5 MB

> climate_data <- read.csv(unzip(clim_data_zip_file))
> str(climate_data)
'data.frame':   473111 obs. of  15 variables:
 $ Date_Time     : Factor w/ 472784 levels "1/1/2009 0:10",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ p_mbar        : num  997 997 997 997 997 ...
 $ T_degC        : num  -8.02 -8.41 -8.51 -8.31 -8.27 -8.05 -7.62 -7.62 -7.91 -8.43 ...
 $ Tpot_K        : num  265 265 265 265 265 ...
 $ Tdew_degC     : num  -8.9 -9.28 -9.31 -9.07 -9.04 -8.78 -8.3 -8.36 -8.73 -9.34 ...
 $ rh_percent    : num  93.3 93.4 93.9 94.2 94.1 94.4 94.8 94.4 93.8 93.1 ...
 $ VPmax_mbar    : num  3.33 3.23 3.21 3.26 3.27 3.33 3.44 3.44 3.36 3.23 ...
 $ VPact_mbar    : num  3.11 3.02 3.01 3.07 3.08 3.14 3.26 3.25 3.15 3 ...
 $ VPdef_mbar    : num  0.22 0.21 0.2 0.19 0.19 0.19 0.18 0.19 0.21 0.22 ...
 $ sh_g_kg       : num  1.94 1.89 1.88 1.92 1.92 1.96 2.04 2.03 1.97 1.88 ...
 $ H2OC_mmol_mol: num  3.12 3.03 3.02 3.08 3.09 3.15 3.27 3.26 3.16 3.02 ...
 $ rho_g_m3      : num  1308 1310 1310 1309 1309 ...
 $ wv_m_s        : num  1.03 0.72 0.19 0.34 0.32 0.21 0.18 0.19 0.28 0.59 ...
 $ max_wv_m_s    : num  1.75 1.5 0.63 0.5 0.63 0.63 0.63 0.5 0.75 0.88 ...
 $ wd_deg        : num  152 136 172 198 214 ...
> head(climate_data)
      Date_Time p_mbar T_degC Tpot_K Tdew_degC rh_percent VPmax_mbar VPact_mbar VPdef_mbar
1 1/1/2009 0:10 996.52  -8.02 265.40     -8.90       93.3       3.33       3.11       0.22
2 1/1/2009 0:20 996.57  -8.41 265.01     -9.28       93.4       3.23       3.02       0.21
3 1/1/2009 0:30 996.53  -8.51 264.91     -9.31       93.9       3.21       3.01       0.20
4 1/1/2009 0:40 996.51  -8.31 265.12     -9.07       94.2       3.26       3.07       0.19
5 1/1/2009 0:50 996.51  -8.27 265.15     -9.04       94.1       3.27       3.08       0.19
6 1/1/2009 1:00 996.50  -8.05 265.38     -8.78       94.4       3.33       3.14       0.19
  sh_g_kg H2OC_mmol_mol rho_g_m3 wv_m_s max_wv_m_s wd_deg
1    1.94          3.12  1307.75   1.03       1.75  152.3
2    1.89          3.03  1309.80   0.72       1.50  136.1
3    1.88          3.02  1310.24   0.19       0.63  171.6
4    1.92          3.08  1309.19   0.34       0.50  198.0
5    1.92          3.09  1309.00   0.32       0.63  214.3
6    1.96          3.15  1307.86   0.21       0.63  192.7
> unlink(clim_data_zip_file)
>
```

Voici le contenu du fichier *.csv pour les curieux:

```
ClimateData_2009_2017_Jena_Germany_MaxPlanck.csv - Notepad (Not Responding)
File Edit Format View Help
Date_Time,p_mbar,T_degC,Tpot_K,Tdew_degC,rh_percent,VPmax_mbar,VPact_mbar,VPdef_mbar,sh_g_kg,H2OC_mmol_mol,rho_g_m3,wv_m_s,max_wv_m_s,w
d_deg
1/1/2009 0:10,996.52,-8.02,265.4,-8.9,93.3,3.33,3.11,0.22,1.94,3.12,1307.75,1.03,1.75,152.3
1/1/2009 0:20,996.57,-8.41,265.01,-9.28,93.4,3.23,3.02,0.21,1.89,3.03,1309.8,0.72,1.5,136.1
1/1/2009 0:30,996.53,-8.51,264.91,-9.31,93.9,3.21,3.01,0.2,1.88,3.02,1310.24,0.19,0.63,171.6
1/1/2009 0:40,996.51,-8.31,265.12,-9.07,94.2,3.26,3.07,0.19,1.92,3.08,1309.19,0.34,0.5,198
1/1/2009 0:50,996.51,-8.27,265.15,-9.04,94.1,3.27,3.08,0.19,1.92,3.09,1309,0.32,0.63,214.3
1/1/2009 1:00,996.5,-8.05,265.38,-8.78,94.4,3.33,3.14,0.19,1.96,3.15,1307.86,0.21,0.63,192.7
1/1/2009 1:10,996.5,-7.62,265.81,-8.3,94.8,3.44,3.26,0.18,2.04,3.27,1305.68,0.18,0.63,166.5
1/1/2009 1:20,996.5,-7.62,265.81,-8.36,94.4,3.44,3.25,0.19,2.03,3.26,1305.69,0.19,0.5,118.6
1/1/2009 1:30,996.5,-7.91,265.52,-8.73,93.8,3.36,3.15,0.21,1.97,3.16,1307.17,0.28,0.75,188.5
1/1/2009 1:40,996.53,-8.43,264.99,-9.34,93.1,3.23,3,0.22,1.88,3.02,1309.85,0.59,0.88,185
1/1/2009 1:50,996.62,-8.76,264.66,-9.66,93.1,3.14,2.93,0.22,1.83,2.94,1311.64,0.45,0.88,183.2
1/1/2009 2:00,996.62,-8.88,264.54,-9.77,93.2,3.12,2.9,0.21,1.81,2.91,1312.25,0.25,0.63,190.3
1/1/2009 2:10,996.63,-8.85,264.57,-9.7,93.5,3.12,2.92,0.2,1.82,2.93,1312.11,0.16,0.5,158.3
1/1/2009 2:20,996.74,-8.83,264.58,-9.68,93.5,3.13,2.92,0.2,1.83,2.93,1312.15,0.36,0.63,184.8
1/1/2009 2:30,996.81,-8.66,264.74,-9.46,93.9,3.17,2.98,0.19,1.86,2.99,1311.37,0.33,0.75,155.9
1/1/2009 2:40,996.81,-8.66,264.74,-9.5,93.6,3.17,2.97,0.2,1.85,2.98,1311.38,0.07,0.5,272.4
```

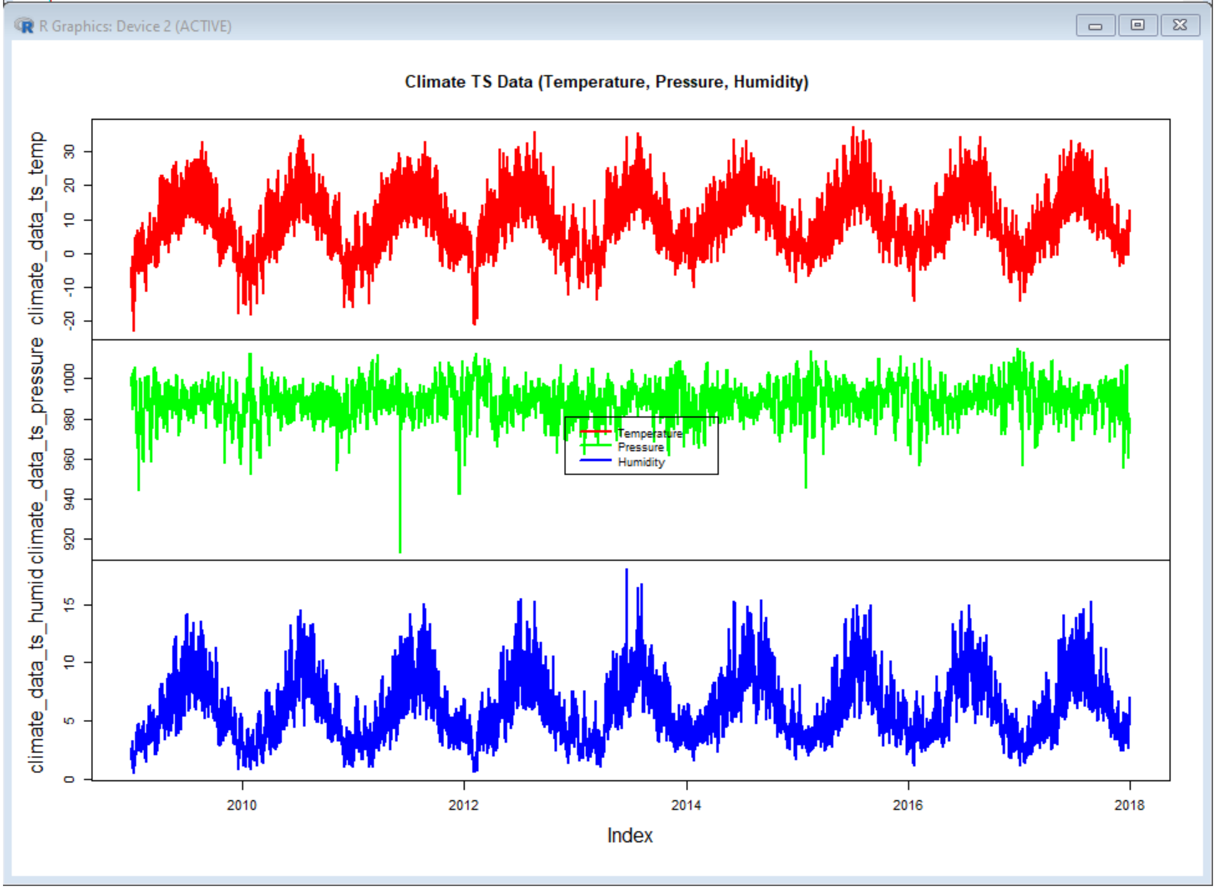
En bref, ces données de séquence représentent des séries chronologiques météorologiques enregistrées à la station météorologique de l'Institut Max Planck de biogéochimie à Jena, en Allemagne. Les données comprennent la date-heure et 14 mesures / caractéristiques climatiques (par exemple, la pression atmosphérique, la température et l'humidité), qui sont enregistrées toutes les 10 minutes sur 9 ans.

Maintenant on fait un peu de nettoyage et d'extraction des données qui nous intéressent:

```
> #on convertit la colonne de dates (caractères) en POSIXct
> climate_time <- as.POSIXct(climate_data$Date_Time, format = "%d/%m/%Y %H:%M")
> head(climate_time,10)
 [1] "2009-01-01 00:10:00 CET" "2009-01-01 00:20:00 CET" "2009-01-01 00:30:00 CET"
 [4] "2009-01-01 00:40:00 CET" "2009-01-01 00:50:00 CET" "2009-01-01 01:00:00 CET"
 [7] "2009-01-01 01:10:00 CET" "2009-01-01 01:20:00 CET" "2009-01-01 01:30:00 CET"
[10] "2009-01-01 01:40:00 CET"
> length(climate_time)
[1] 473111
>
> library("zoo")
>
> #on extrait les éléments uniques qui nous intéressent
> #Température
> climate_data_ts_temp <- zoo(climate_data$T_degC[!duplicated(climate_time)],
+                             climate_time[!duplicated(climate_time)])
> head(climate_data_ts_temp)
2009-01-01 00:10:00 2009-01-01 00:20:00 2009-01-01 00:30:00 2009-01-01 00:40:00
              -8.02               -8.41               -8.51               -8.31
2009-01-01 00:50:00 2009-01-01 01:00:00
              -8.27               -8.05
> length(climate_data_ts_temp)
[1] 472731
> #Pression
> climate_data_ts_pressure <- zoo(climate_data$p_mbar[!duplicated(climate_time)],
+                                 climate_time[!duplicated(climate_time)])
> head(climate_data_ts_pressure)
2009-01-01 00:10:00 2009-01-01 00:20:00 2009-01-01 00:30:00 2009-01-01 00:40:00
             996.52              996.57              996.53              996.51
2009-01-01 00:50:00 2009-01-01 01:00:00
             996.51              996.50
> length(climate_data_ts_pressure)
[1] 472731
> #Humidité
> climate_data_ts_humid <- zoo(climate_data$sh_g_kg[!duplicated(climate_time)],
+                              climate_time[!duplicated(climate_time)])
> head(climate_data_ts_humid)
2009-01-01 00:10:00 2009-01-01 00:20:00 2009-01-01 00:30:00 2009-01-01 00:40:00
               1.94                1.89                1.88                1.92
2009-01-01 00:50:00 2009-01-01 01:00:00
               1.92                1.96
> length(climate_data_ts_humid)
[1] 472731
> |
```

Faisons maintenant un plot de toutes ces données:

```
> climate_data_ts_aggregate = cbind(climate_data_ts_temp,
+                                   climate_data_ts_pressure, climate_data_ts_humid)
>
> plot(climate_data_ts_aggregate, main="Climate TS Data (Temperature, Pressure, Humidity)",
+      col=c("red", "green", "blue"), lty=1, lwd=2)
> legend("center", legend=c("Temperature", "Pressure", "Humidity"),
+        col=c("red", "green", "blue"), lty=1, lwd=2, cex=0.6, x.intersp=0.5)
>
```



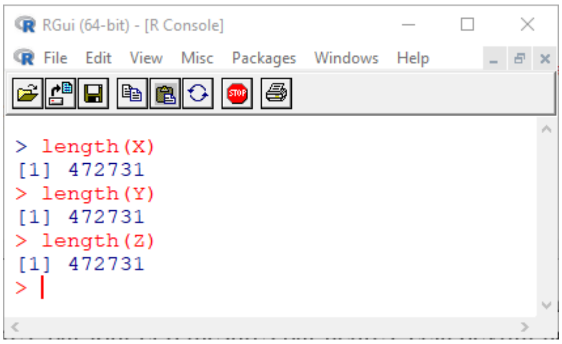
Normalisons les données ci-dessus et préparons aussi des fonctions de dé-normalisation:

```
> #Now we can proceed with the RNN time-series modeling and prediction.
> #Recall: Bivariate Input=(X,Y); Univariate Outcome (Z)
> X <- climate_data_ts_pressure
> Y <- climate_data_ts_humid
> Z <- climate_data_ts_temp
>
> #RNN requores standardization in the interval 0 - 1
> min_x <- min(X); min_y <- min(Y); min_z <- min(Z)
> max_x <- max(X); max_y <- max(Y); max_z <- max(Z)
>
> #Generic transformation:
> #forward (scale), for raw data, and reverse (unscale), for predicted data
> my.scale <- function (x, forward=TRUE, input="X") {
+   if (input=="X" && forward) {        # X=Input==predictors & Forward scaling
+     x <- (x - min_x) / (max_x - min_x)
+   } else if (input=="X" && !forward) {    # X=Input==predictors & Reverse scaling
+     x <- x * (max_x - min_x) + min_x
+   } else if (input=="Y" && forward) {     # Y=Input==predictors & Forward scaling
+     x <- (x - min_y) / (max_y - min_y)
+   } else if (input=="Y" && !forward) {    # Y=Input==predictors & Reverse scaling
+     x <- x * (max_y - min_y) + min_y
+   } else if (input=="Z" && forward) {     # Z=Output==predictors & Forward scaling
+     x <- (x - min_z) / (max_z - min_z)
+   } else if (input=="Z" && !forward) {    # Z=Output==predictors & Reverse scaling
+     x <- x * (max_z - min_z) + min_z
+   }
+   return (x)
+ }
>
> #Save these transform parameters; center/scale, so we can invert the transforms
> #after we get the predicted values.
> X <- my.scale(X, forward=TRUE, input="X")
> Y <- my.scale(Y, forward=TRUE, input="Y")
> Z <- my.scale(Z, forward=TRUE, input="Z")
>
```

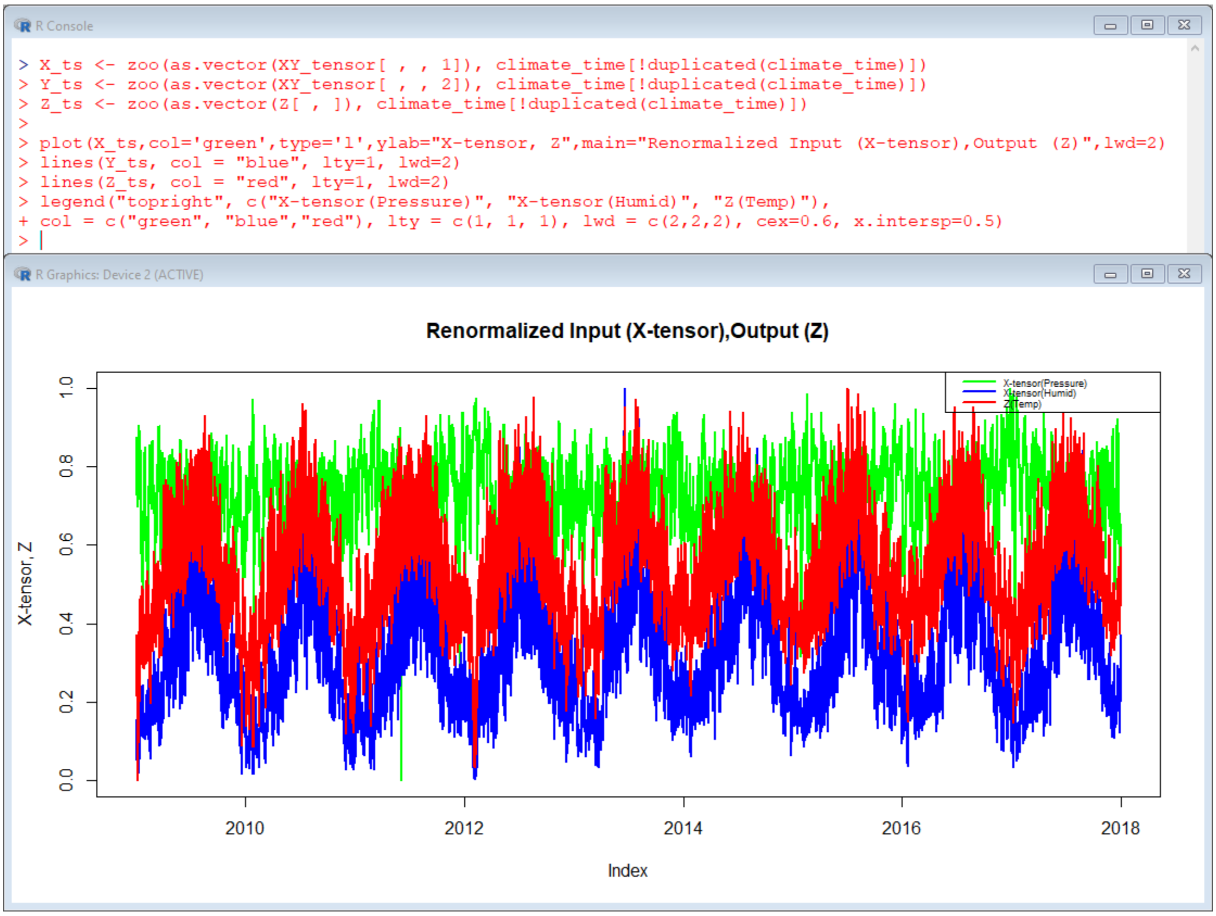
Une fois ceci fait, rappelons que nous avons 9 années de données, en supposant 365.25 jours par année, 24 heures, par jour et 6 mesures par heures, cela devrait nous faire grossièrement un total de points de mesure de:

```
> 365.25*24*6*9
[1] 473364
>
```

Mais nous avons moins de points donc il y a des lacunes (certaines journées ou heures non mesurées) comme l'atteste cette capture d'écran:

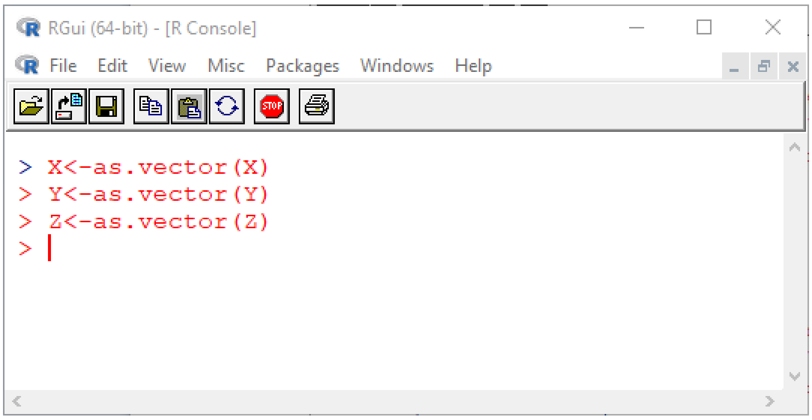


Voyons comment faire un graphique temporel à partir des ces tenseurs avant de passer à la partie sur l'entraînement du réseau de neurones:

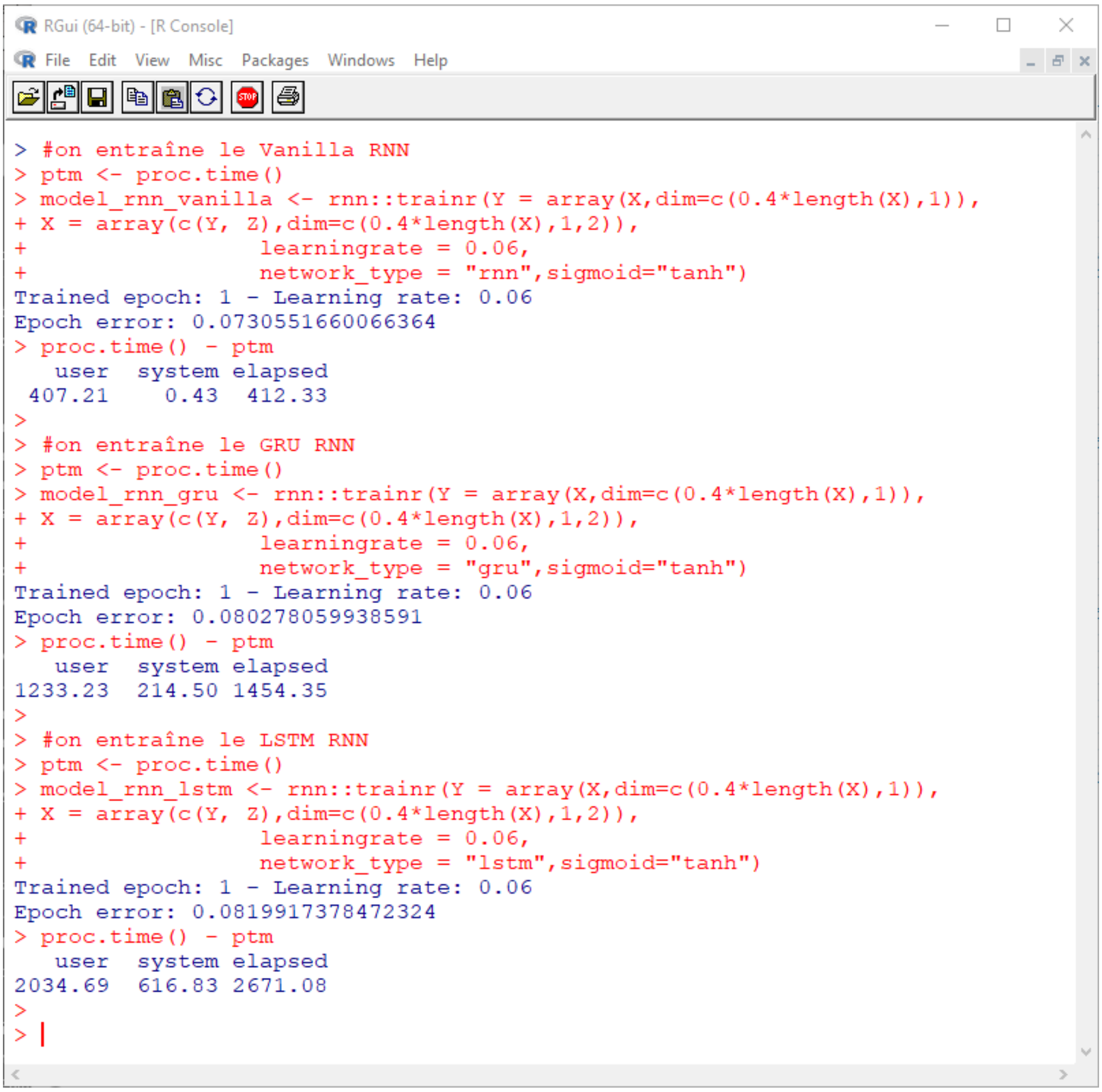


Maintenant on entraîne les trois différentes familles de réseaux de neurones avec la même fonction d'activation que celle vue dans le cours théorique (tangente hyperbolique) et en précisant aussi le *learningrate* qui est un paramètre obligatoire dans la fonction **trainr( )** du package **rnn** (le lecteur pourra ajouter un set.seed s'il souhaite pouvoir reproduire ses résultat sur son propre ordinateur).

Coemme la fonction **trainr( )**veut des arrays en entrée basés sur des vecteurs il va d'abord nous falloir faire une petite transformation de nos trois facteurs:

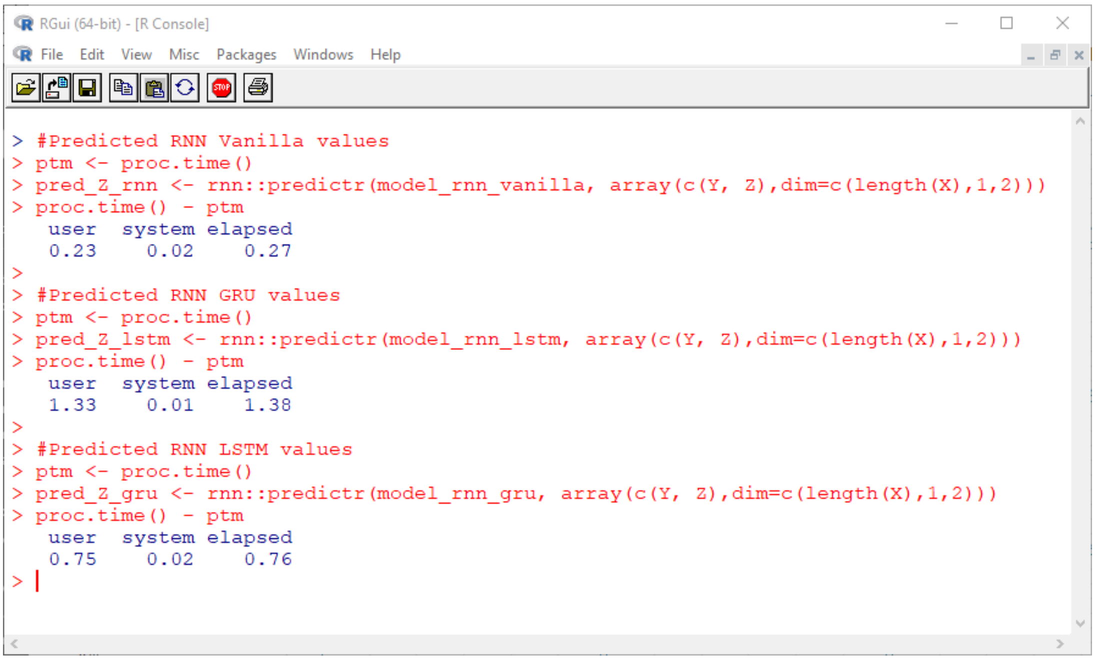


Maintenant entraînons les trois différents types de réseaux de neurones vu dans le cours théorique avec la fonction d'activation Tangente Hyperbolique et avec 40% du jeu de données complet pour l'entraînement (observez bien la syntaxe de transformation des vecteurs en array – surtout la deuxième transformation - c'est crucial pour être autonome par la suite[37]!!!):

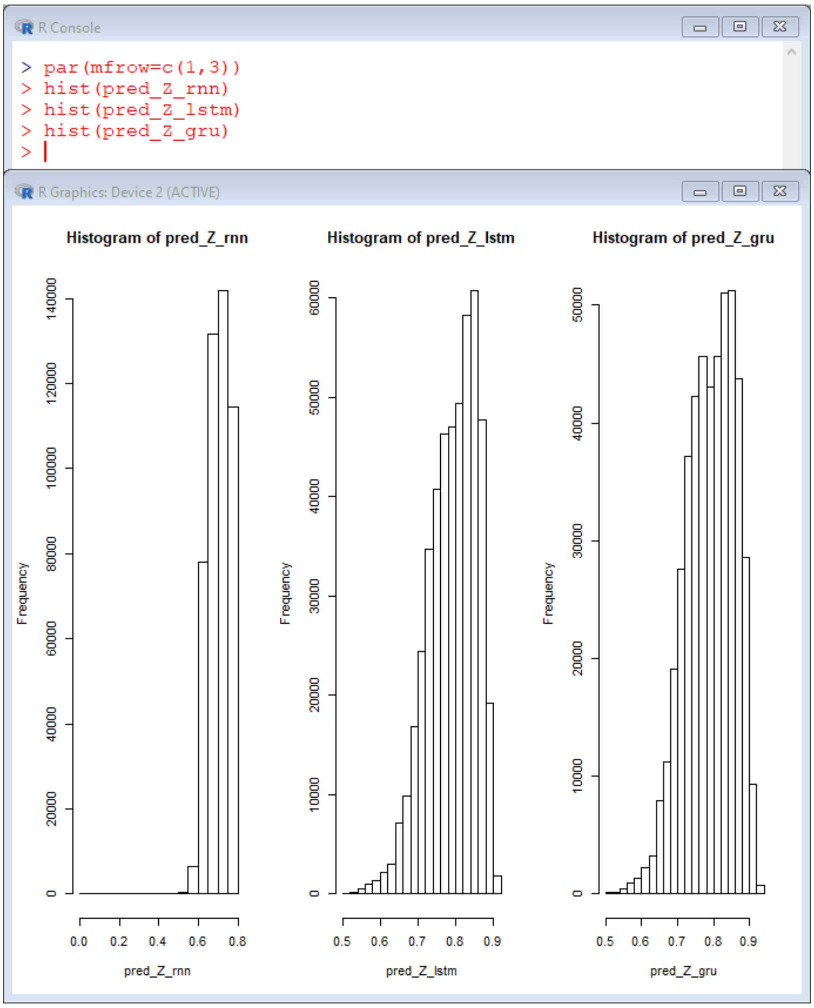


[37] En particulier la partie c(0.4*length(X),1,2) où le "1" signifie que l'on UN paramètre de sortie et le "2" signifie que l'on a DEUX paramètres d'entrée.

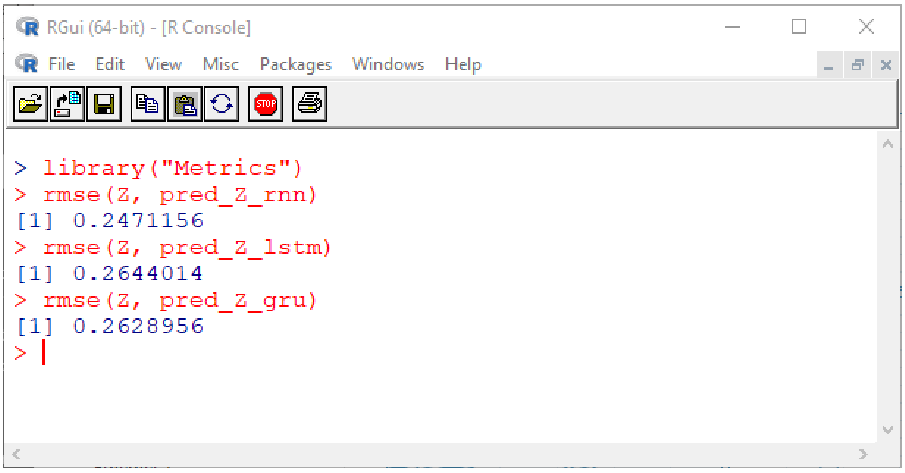
Ensuite, nous pouvons faire une prédiction sur les bases des réseaux de neurones construits. Prédiction se faisant sur le jeu de données d'entraînement + sur la partie non utilisée:



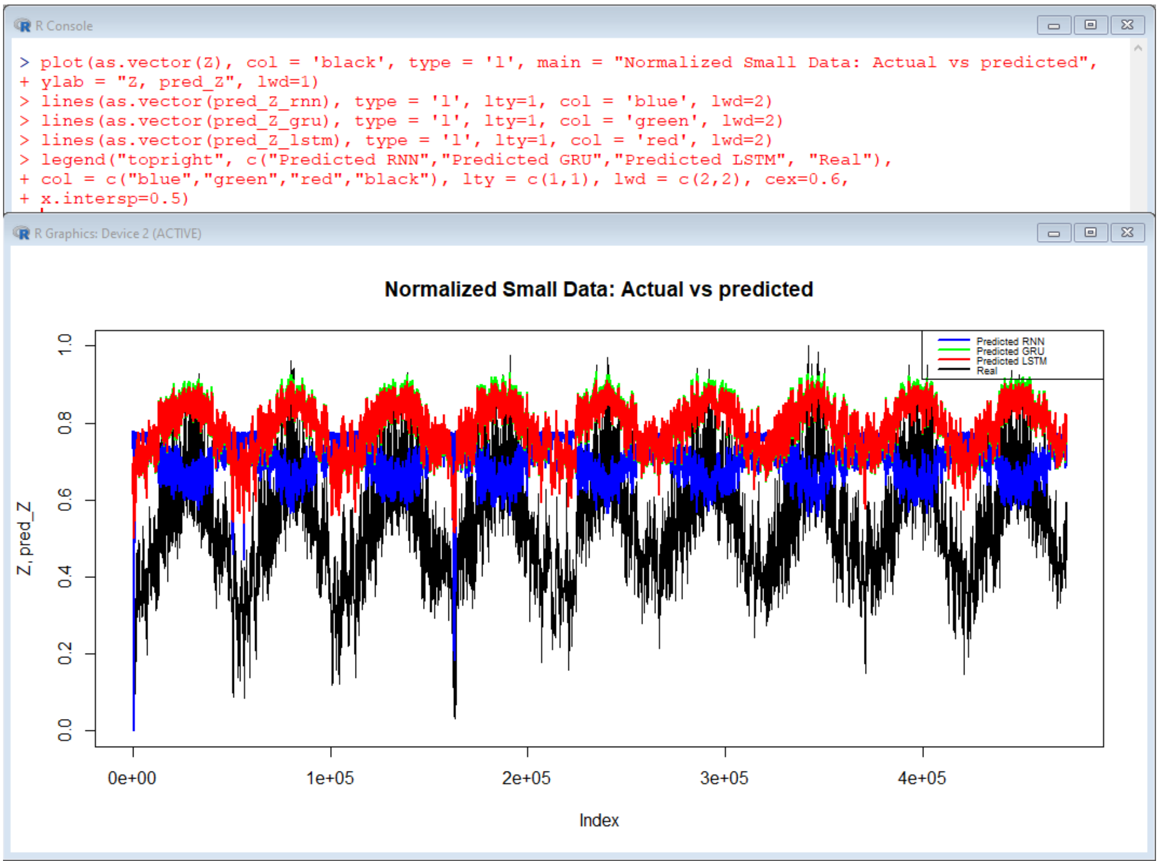
On peut comparer l'histogramme des prédiction pour le fun:



ou on peut aussi comparer la performance de chacun des réseaux en calculant sa RMSE (root mean square error):



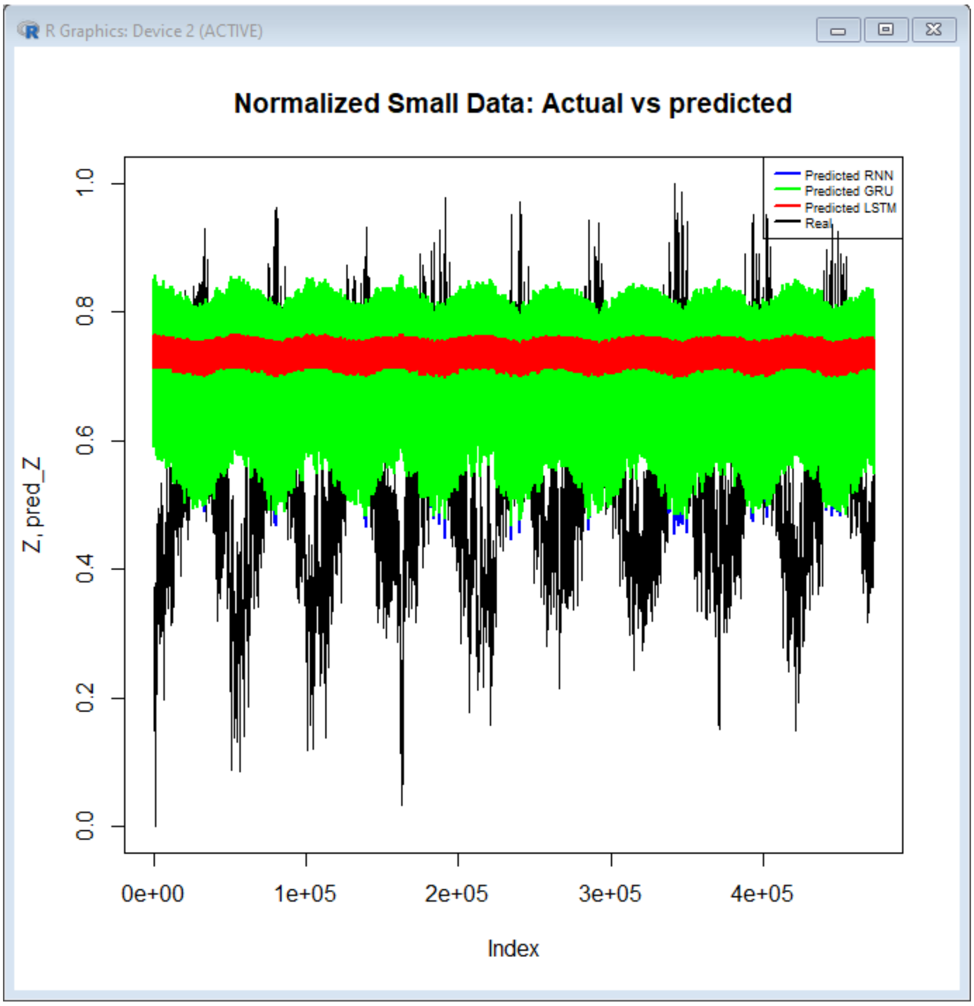
Maintenant voyons graphiquement la comparaison entre les trois réseaux de neurones entraînés avec les données réelles normalisées:



Les résultats ne sont guère mieux avec les biais:

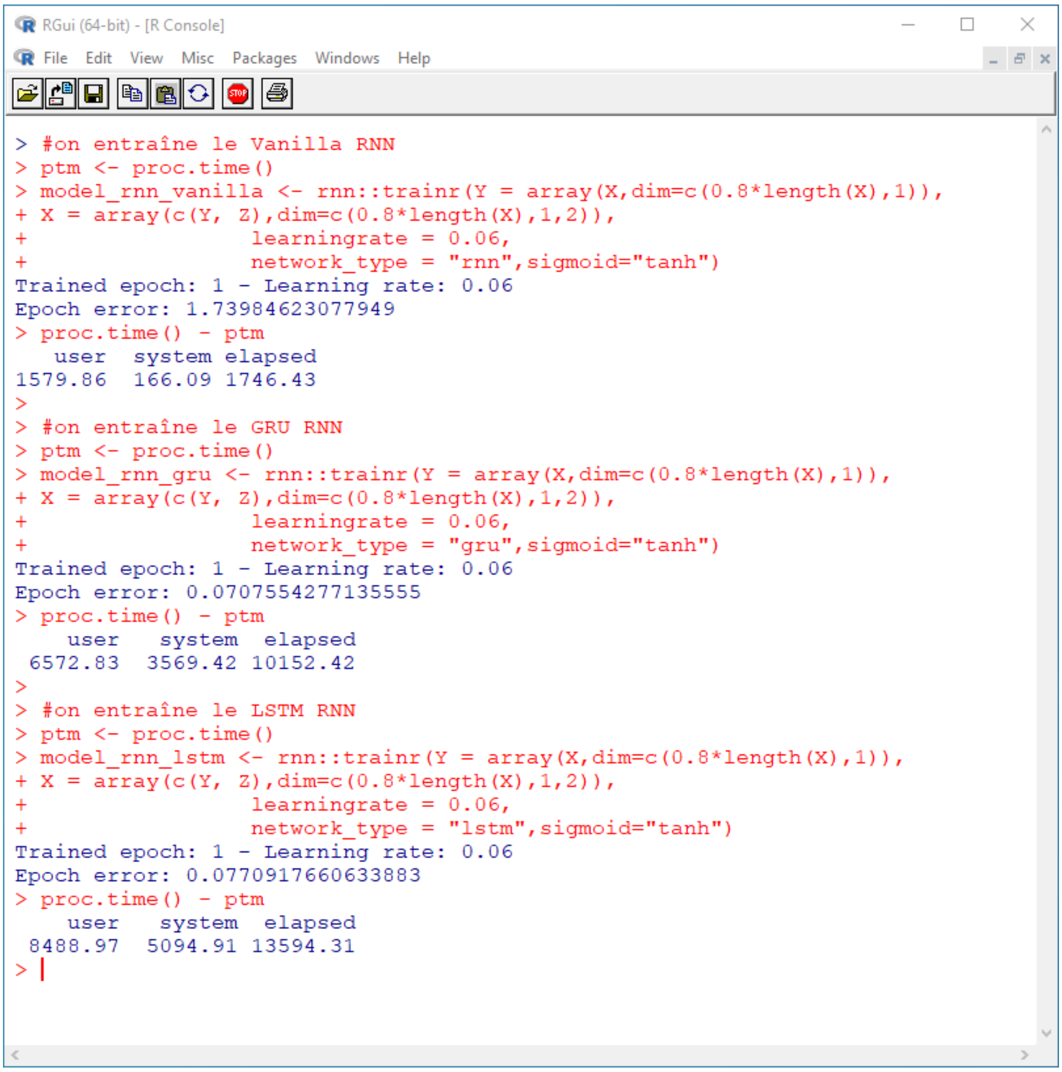
```
> #on entraîne le Vanilla RNN
> ptm <- proc.time()
> model_rnn_vanilla <- rnn::trainr(Y = array(X,dim=c(0.4*length(X),1)),
+ X = array(c(Y, Z),dim=c(0.4*length(X),1,2)),
+                learningrate = 0.06, use_bias = T,
+                network_type = "rnn",sigmoid="tanh")
Trained epoch: 1 - Learning rate: 0.06
Epoch error: 0.0698041840375521
> proc.time() - ptm
   user  system elapsed
 394.76    6.33  401.27
>
> #on entraîne le GRU RNN
> ptm <- proc.time()
> model_rnn_gru <- rnn::trainr(Y = array(X,dim=c(0.4*length(X),1)),
+ X = array(c(Y, Z),dim=c(0.4*length(X),1,2)),
+                learningrate = 0.06, use_bias = T,
+                network_type = "gru",sigmoid="tanh")
Trained epoch: 1 - Learning rate: 0.06
Epoch error: 0.0656295816455269
> proc.time() - ptm
   user  system elapsed
1230.48  418.36 1649.55
>
> #on entraîne le LSTM RNN
> ptm <- proc.time()
> model_rnn_lstm <- rnn::trainr(Y = array(X,dim=c(0.4*length(X),1)),
+ X = array(c(Y, Z),dim=c(0.4*length(X),1,2)),
+                learningrate = 0.06, use_bias = T,
+                network_type = "lstm",sigmoid="tanh")
Trained epoch: 1 - Learning rate: 0.06
Epoch error: 0.0674019914193665
> proc.time() - ptm
   user  system elapsed
2132.45  868.61 3002.00
>
```

Ce qui donne graphiquement:

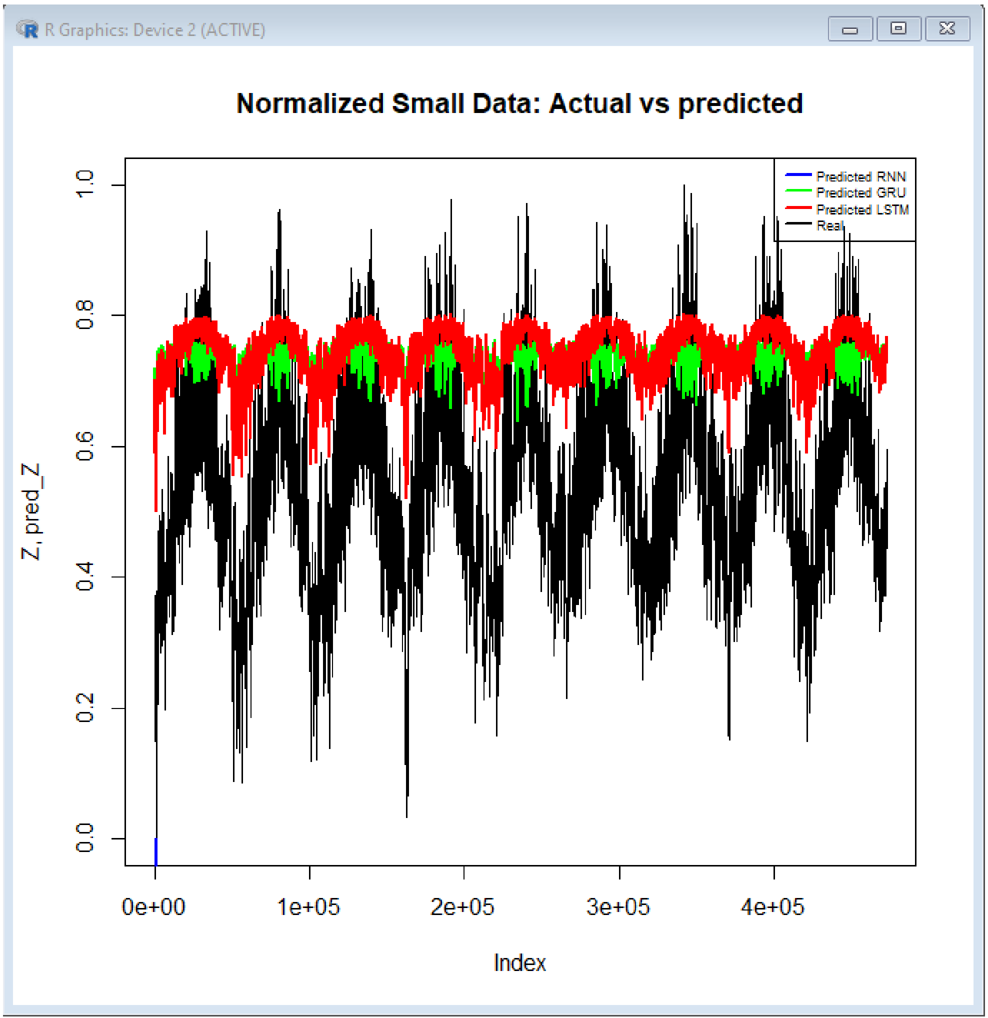


Et ce n'est pas mieux non plus avec les autres fonctions d'activation!

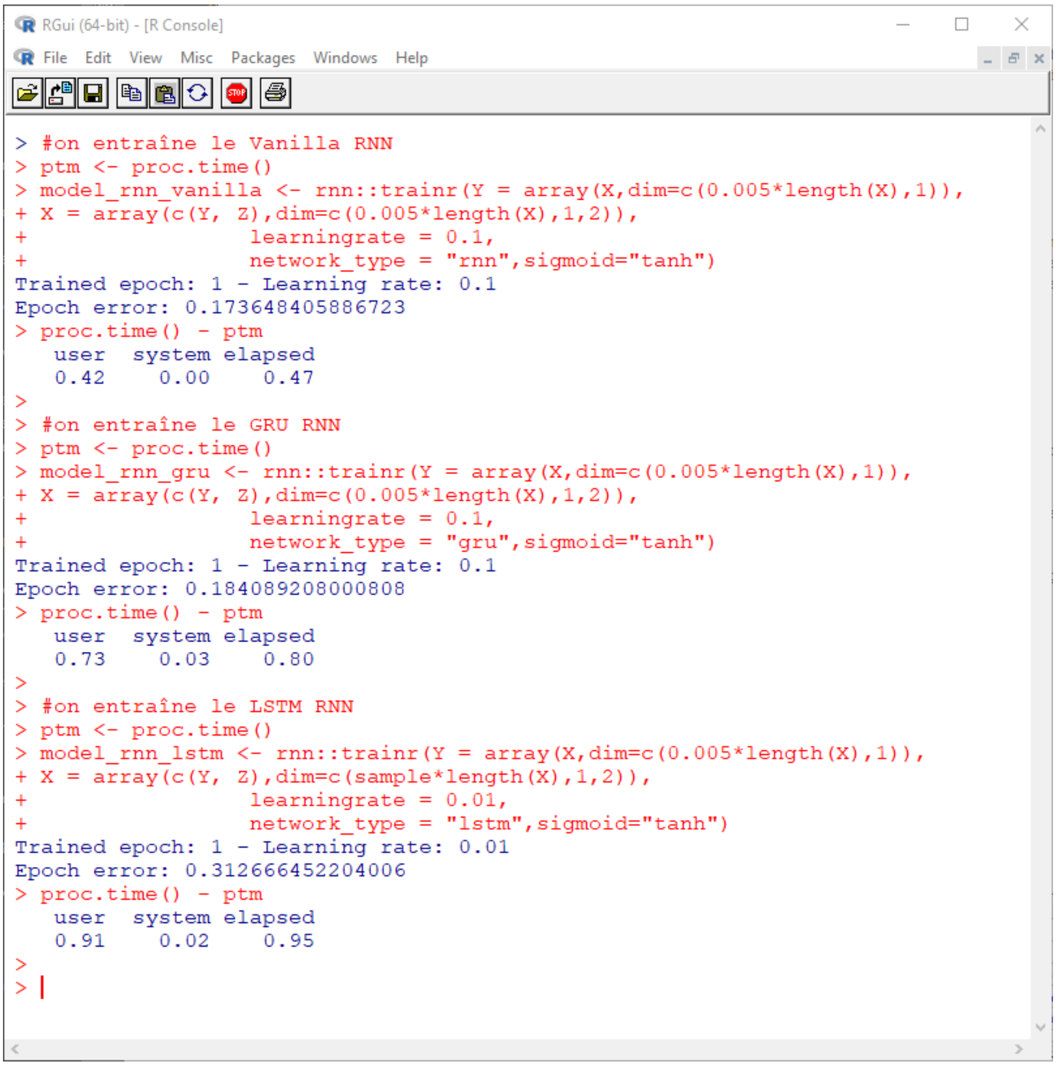
Et pour ceux qui ont la patience de faire l'entraînement sur 80% (ici sur un Alienwaer Area 51m avec toutes les options!):



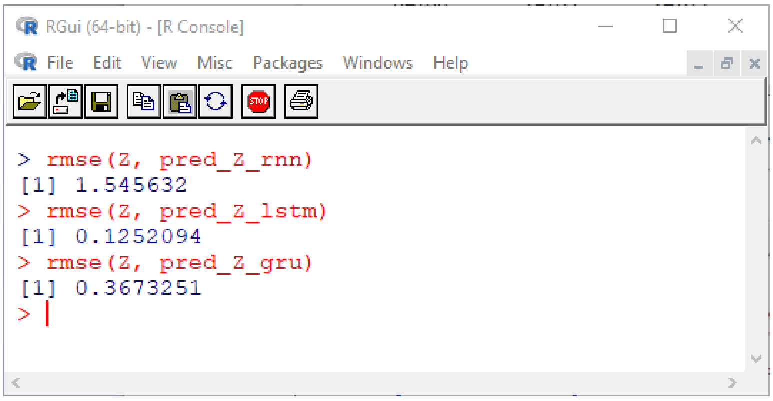
C'est la catastrophe (surtout pour le réseau RNN…!):



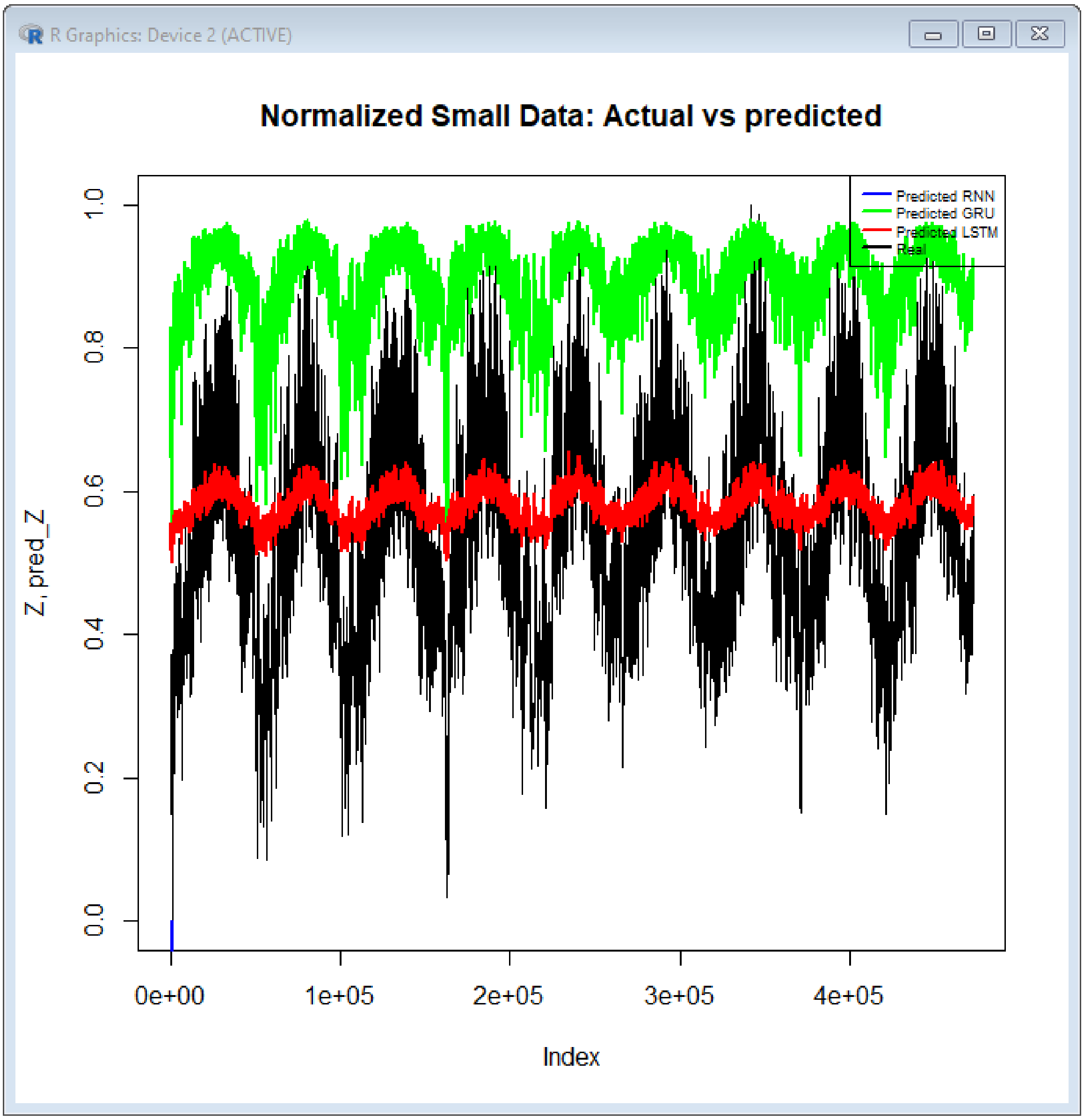
On est typiquement dans un cas de sur-apprentissage. Relaxons alors un peu tout cela significativement en faisant l'entraînement que sur un très faible je de données:



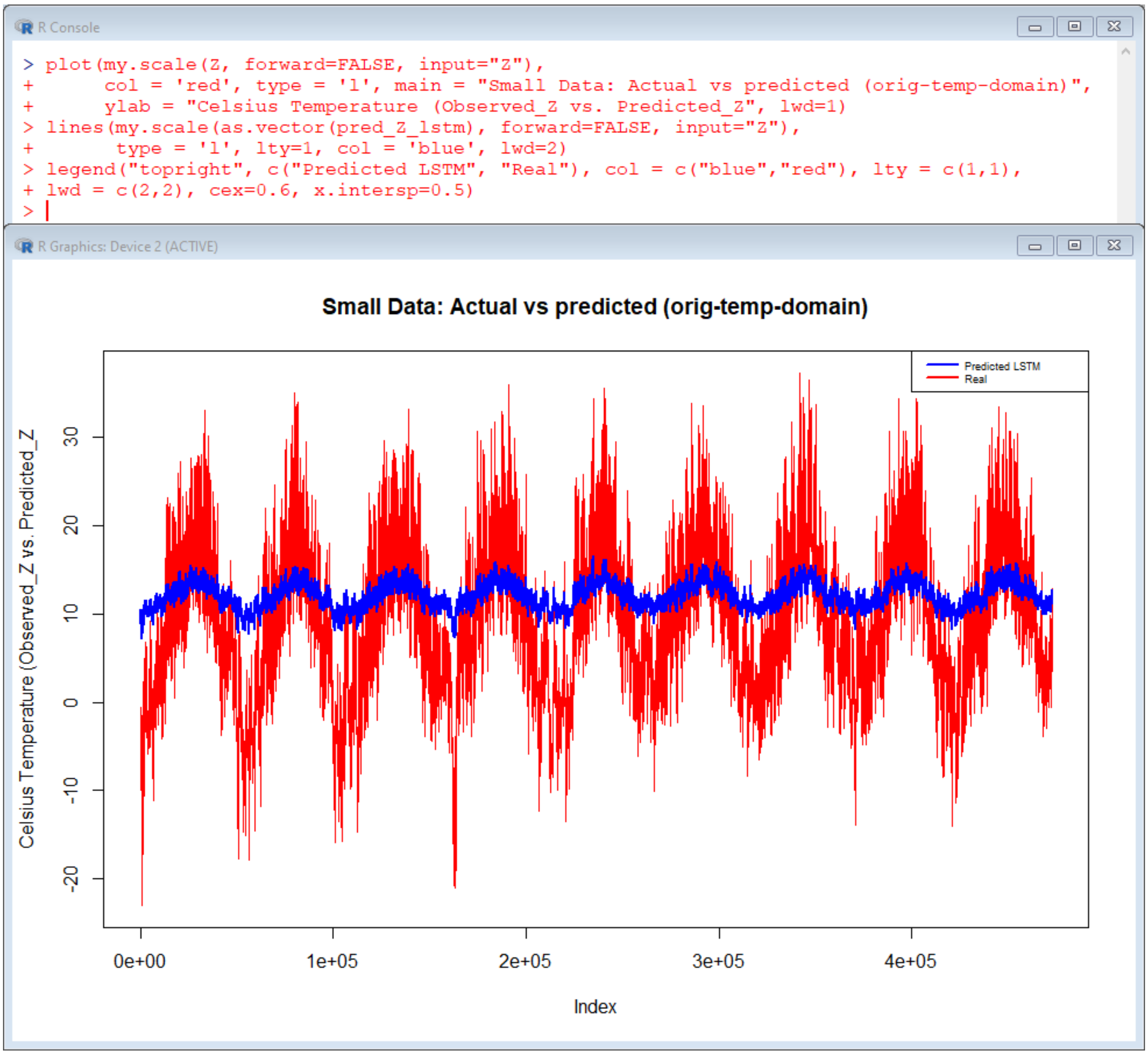
Le réseau LSTM performe relativement bien ici comme on peut le constater:



Il semble difficile de faire mieux:

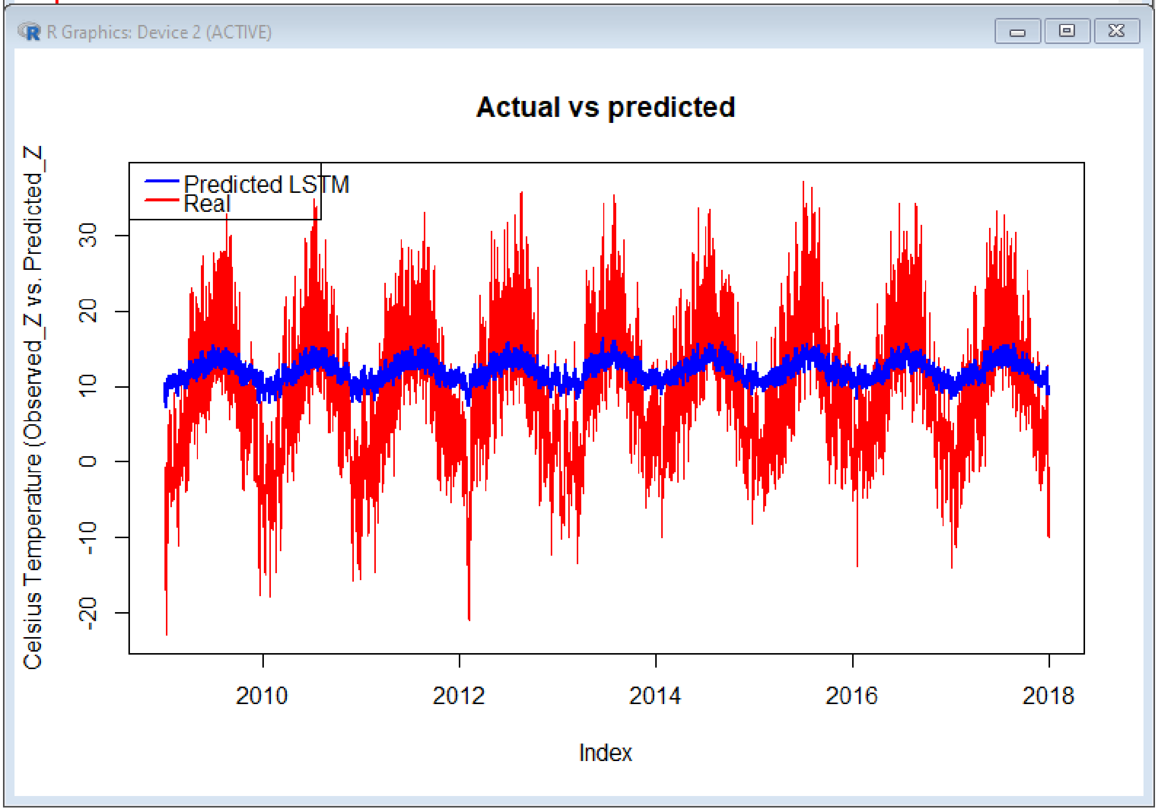


Remettons à l'échelle d'origine:



et avec le bon axe:

```
> obs_Temp_Z_ts <- zoo(my.scale(Z, forward=FALSE, input="Z"),
+                               climate_time)
Warning message:
In zoo(my.scale(Z, forward = FALSE, input = "Z"), climate_time) :
  some methods for "zoo" objects do not work if the index entries in 'ord$
> pred_Temp_Z_ts <- zoo(my.scale(as.vector(pred_Z_lstm),
+                        forward=FALSE, input="Z"), climate_time)
Warning message:
In zoo(my.scale(as.vector(pred_Z_lstm), forward = FALSE, input = "Z"),  :
  some methods for "zoo" objects do not work if the index entries in 'ord$
> plot(obs_Temp_Z_ts, col = 'red', type = 'l',
+      main = "Actual vs predicted",
+      ylab = "Celsius Temperature (Observed_Z vs. Predicted_Z", lwd=1)
> lines(pred_Temp_Z_ts, type = 'l', lty=1, col = 'blue', lwd=2)
> legend("topleft", c("Predicted LSTM", "Real"), col = c("blue","red"),
+ lty = c(1,1), lwd = c(2,2), cex=1.0, x.intersp=0.4)
>
```



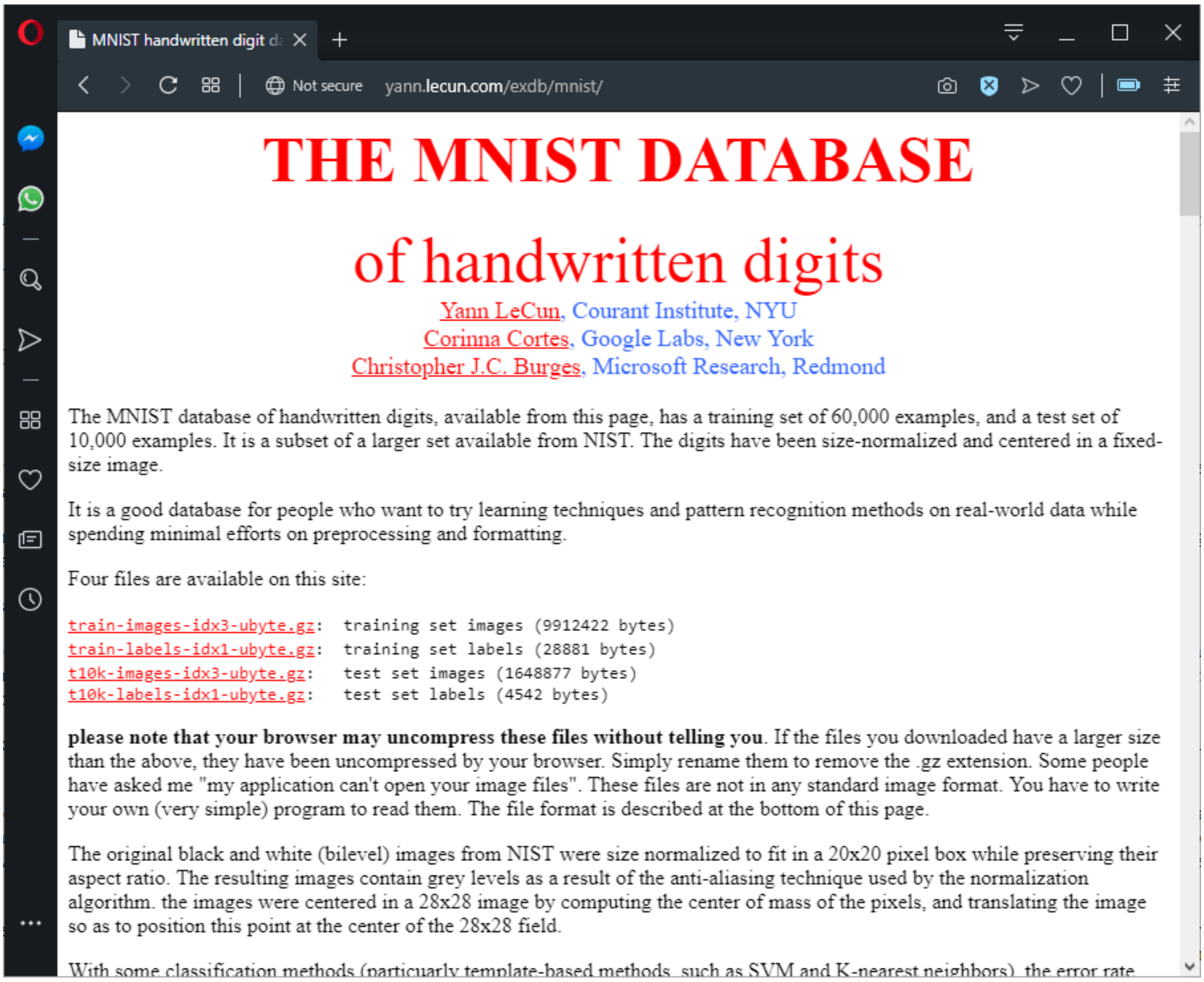
Cet exemple nous apprends a priori quelque chose que l'on savait déjà: Un modèle complexe et moderne ou à la mode, n'est pas nécessairement mieux qu'un modèle simple!

Exemple 361.: Réseaux de neurones convolutionnels GAN (Generative Adversial Networks)

R 3.6.3

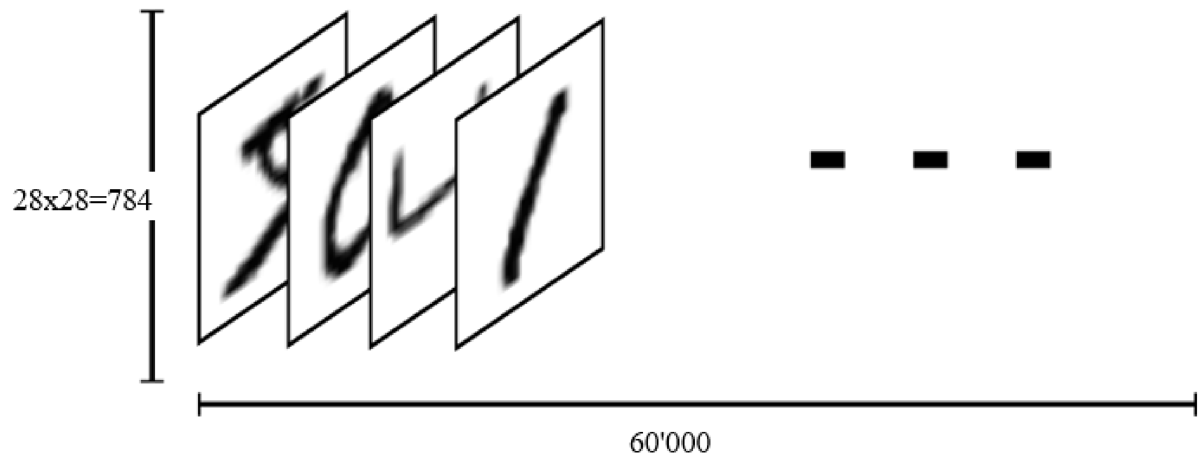
Nous allons faire un exemple de GAN ici sur de l'imagerie et plus particulièrement sur un exemple et jeu de données académique fameux qu'est le MNIST:

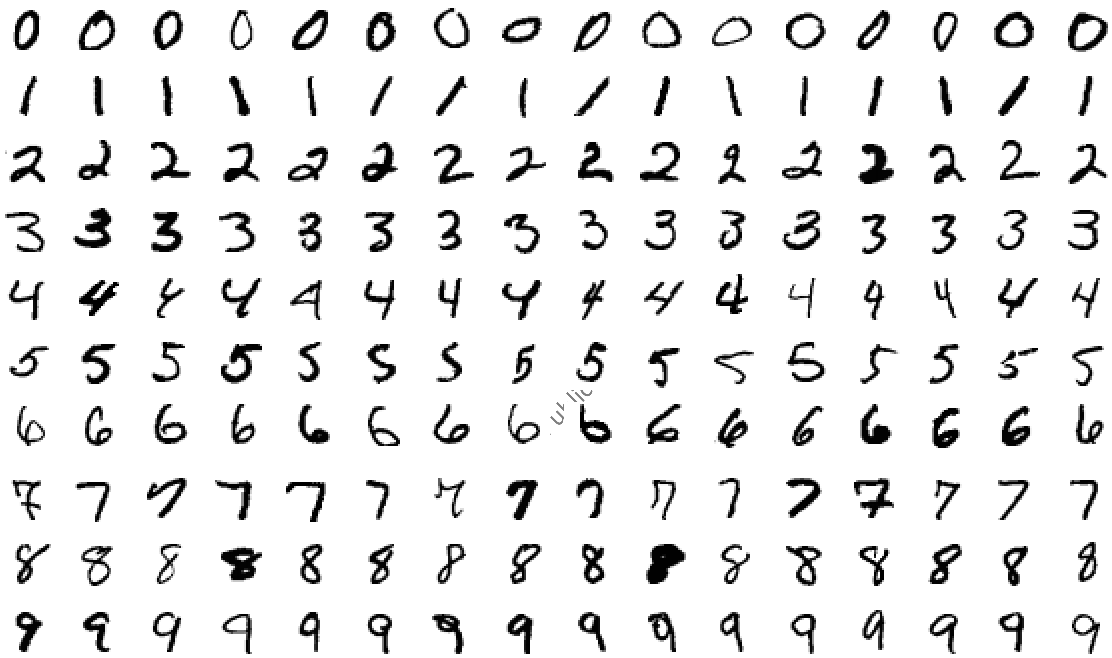
http://yann.lecun.com/exdb/mnist/



La base de données MNIST pour Mixed National Institute of Standards and Technology, est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenu un test standard. Elle regroupe 60'000 images d'apprentissage et 10'000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté (soit 28 x 28 = 784 points):





L'exemple qui va suivre vient d'un article de blog de Sigrid Keydana:

https://tensorflow.rstudio.com/guide/keras/examples/eager_dcgan/

dont on peut encore trouver normalement la source ici:

https://tensorflow.rstudio.com/guide/keras/examples/eager_dcgan/

ou ici:

https://github.com/rstudio/keras/blob/master/vignettes/examples/eager_dcgan.R

Pour ceux qui souhaitent apprendre à créer des jeux d'images de ce type à partir de leur propre bibliothèque, nous les renvoyons à le lecteur sur la gestion de listes R (voir page 296) et du Traitement d'images dans R (voir page 3005).

Voici l'ensemble du code en text brut qui fonctionne sur R 3.6.3:

```
library("keras") #pour charger les jeux de données
library("tensorflow") #pour le CNN
#tensorflow::install_tensorflow()
#Notons que pour utiliser la version GPU de tensorflow (ou de keras...):
#install_tensorflow(version = "gpu")
library("tfdatasets") #pour adapter les données R au format TensorFlow

#on charge les données en ligne détaillées précédemment
mnist <- keras::dataset_mnist()
```

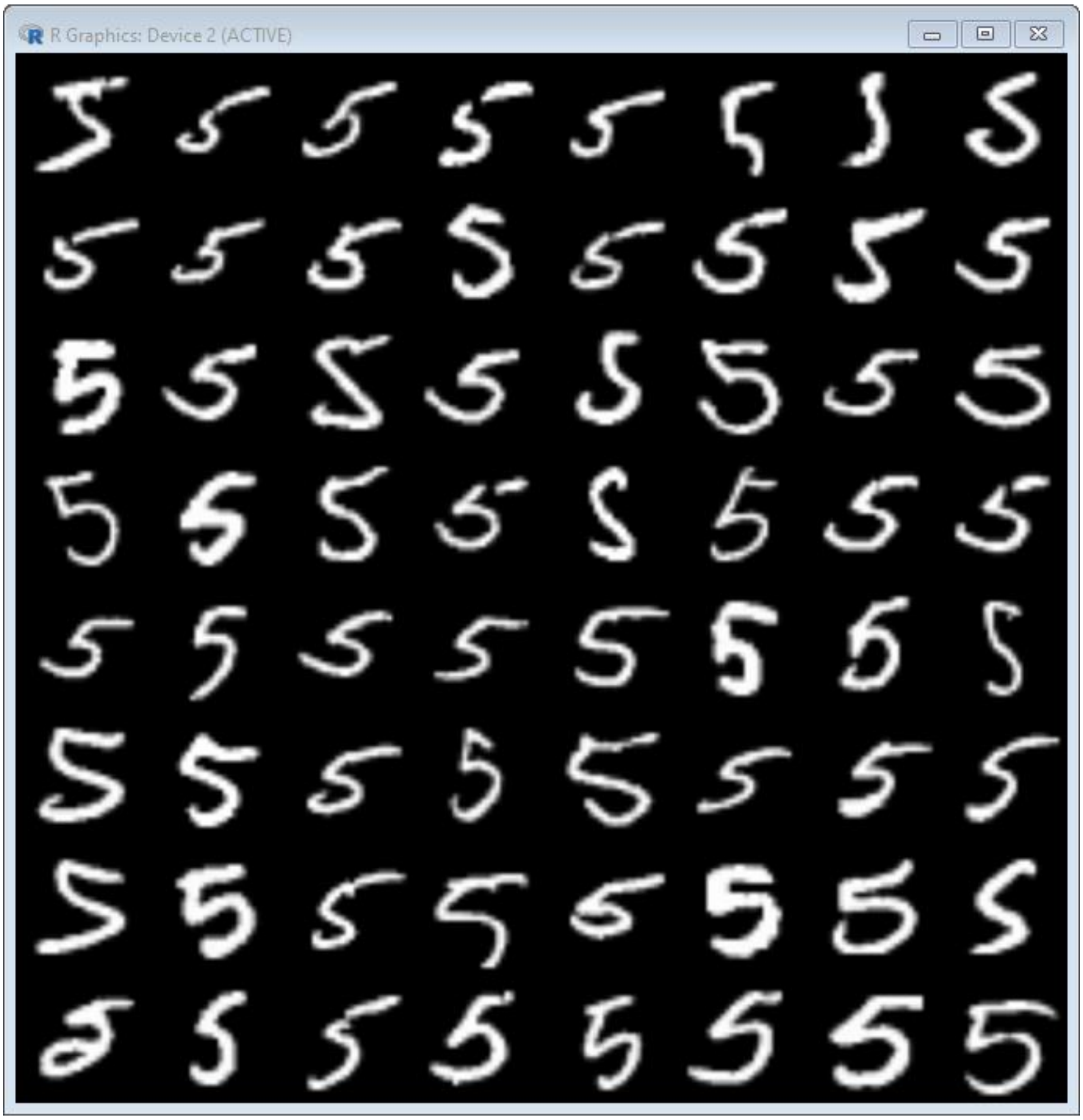
Voici ce que contient ce jeu de données dont nous avons au besoin (si la source en ligne venait à disparaître) une copie locale dans un fichier RData:

```
mnist                          Large list (2 elements, 209.6 Mb)
   train:List of 2
   ..$ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ y: int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4 ...
   test :List of 2
   ..$ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
   ..$ y: int [1:10000(1d)] 7 2 1 0 4 1 4 9 5 9 ...
```

```
MNIST_handwriting_dataset.RData    2020-05-07 14:12    R Workspace    20'215 KB
```

```
#on crée un jeu de données contenant les données pour entraîner le CNN
c(train_images, train_labels) %<-% mnist$train

#avant l'entraînement, faisons juste un plot des 64 premières images (8x8)
du jeu de test pour les images ayant comme label "5"
c(c(trainx, trainy), c(testx, testy)) %<-% mnist
trainx <- trainx[trainy == 5,,]
par(mfrow = c(8, 8), mar = rep(0, 4))
for (i in 1:64) plot(as.raster(trainx[i,,], max = 255))
```



```
#préparation du jeu de données au format Tensorflow
train_images <- train_images %>%
  k_expand_dims() %>%
  k_cast(dtype = "float32")

#on normalize les niveaux de gris (255/2=127.5)
train_images <- (train_images - 127.5) / 127.5

buffer_size <- 60000 #les 60000 images
batch_size <- 256 #le nombre d'images que l'on souhaite traiter par EPOCH
batches_per_epoch <- (buffer_size / batch_size) %>% round()

#on utilise le package tfdatasets encore une fois pour préparer les données
train_dataset <- tensor_slices_dataset(train_images) %>%
  dataset_shuffle(buffer_size) %>% #on randomize le dataset
  dataset_batch(batch_size) #on la randomisation dans un batch

generator <-
  function(name = NULL) {
    keras_model_custom(name = name, function(self) {
      self$fc1 <- layer_dense(units = 7 * 7 * 64, use_bias = FALSE)
      self$batchnorm1 <- layer_batch_normalization()
      self$leaky_relu1 <- layer_activation_leaky_relu()
```

```
      self$conv1 <-
        layer_conv_2d_transpose(
          filters = 64,
          kernel_size = c(5, 5),
          strides = c(1, 1),
          padding = "same",
          use_bias = FALSE
        )
      self$batchnorm2 <- layer_batch_normalization()
      self$leaky_relu2 <- layer_activation_leaky_relu()

      self$conv2 <-
        layer_conv_2d_transpose(
          filters = 32,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same",
          use_bias = FALSE
        )
      self$batchnorm3 <- layer_batch_normalization()
      self$leaky_relu3 <- layer_activation_leaky_relu()

      self$conv3 <-
        layer_conv_2d_transpose(
          filters = 1,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same",
          use_bias = FALSE,
          activation = "tanh"
        )

      function(inputs,
               mask = NULL,
               training = TRUE) {
        self$fc1(inputs) %>%
          self$batchnorm1(training = training) %>%
          self$leaky_relu1() %>%
          k_reshape(shape = c(-1, 7, 7, 64)) %>%

          self$conv1() %>%
          self$batchnorm2(training = training) %>%
          self$leaky_relu2() %>%

          self$conv2() %>%
          self$batchnorm3(training = training) %>%
          self$leaky_relu3() %>%

          self$conv3()
      }
    })
  }

discriminator <-
  function(name = NULL) {
    keras_model_custom(name = name, function(self) {
      self$conv1 <- layer_conv_2d(
        filters = 64,
        kernel_size = c(5, 5),
        strides = c(2, 2),
```

```
        padding = "same"
      )
      self$leaky_relu1 <- layer_activation_leaky_relu()
      self$dropout <- layer_dropout(rate = 0.3)

      self$conv2 <-
        layer_conv_2d(
          filters = 128,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same"
        )
      self$leaky_relu2 <- layer_activation_leaky_relu()
      self$flatten <- layer_flatten()
      self$fc1 <- layer_dense(units = 1)

      function(inputs,
               mask = NULL,
               training = TRUE) {
        inputs %>% self$conv1() %>%
          self$leaky_relu1() %>%
          self$dropout(training = training) %>%
          self$conv2() %>%
          self$leaky_relu2() %>%
          self$flatten() %>%
          self$fc1()

      }
    })
  }

generator <- generator()
discriminator <- discriminator()

generator$call <- tf$`function`(generator$call)
discriminator$call <- tf$`function`(discriminator$call)

cross_entropy = tf$keras$losses$BinaryCrossentropy(from_logits = TRUE)

discriminator_loss <- function(real_output, generated_output) {
  real_loss <- cross_entropy(k_ones_like(real_output), real_output)
  generated_loss <- cross_entropy(k_zeros_like(generated_output),
generated_output)
  real_loss + generated_loss
}

generator_loss <- function(generated_output) {
  cross_entropy(tf$ones_like(generated_output), generated_output)
}

discriminator_optimizer <- tf$optimizers$Adam(1e-4)
generator_optimizer <- tf$optimizers$Adam(1e-4)

num_epochs <- 150
noise_dim <- 100
num_examples_to_generate <- 25L

random_vector_for_generation <-
  k_random_normal(c(num_examples_to_generate,
                    noise_dim))
```

```
generate_and_save_images <- function(model, epoch, test_input) {
  predictions <- model(test_input, training = FALSE)
  png(paste0("c:/tmp/images_epoch_", epoch, ".png"))
  par(mfcol = c(5, 5))
  par(mar = c(0.5, 0.5, 0.5, 0.5),
      xaxs = 'i',
      yaxs = 'i')
  for (i in 1:25) {
    img <- predictions[i, , , 1]
    img <- t(apply(img, 2, rev))
    image(
      1:28,
      1:28,
      img * 127.5 + 127.5,
      col = gray((0:255) / 255),
      xaxt = 'n',
      yaxt = 'n'
    )
  }
  dev.off()
}

train <- function(dataset, epochs, noise_dim) {
  for (epoch in seq_len(num_epochs)) {
    start <- Sys.time()
    total_loss_gen <- 0
    total_loss_disc <- 0
    iter <- make_iterator_one_shot(train_dataset)

    until_out_of_range({
      batch <- iterator_get_next(iter)
      noise <- k_random_normal(c(batch_size, noise_dim))
      with(tf$GradientTape() %as% gen_tape, {
        with(tf$GradientTape() %as% disc_tape, {
          generated_images <- generator(noise)
          disc_real_output <- discriminator(batch, training = TRUE)
          disc_generated_output <-
            discriminator(generated_images, training = TRUE)
          gen_loss <- generator_loss(disc_generated_output)
          disc_loss <-
            discriminator_loss(disc_real_output, disc_generated_output)
        })
      })

      gradients_of_generator <-
        gen_tape$gradient(gen_loss, generator$trainable_variables)
      gradients_of_discriminator <-
        disc_tape$gradient(disc_loss, discriminator$trainable_variables)

      generator_optimizer$apply_gradients(purrr::transpose(list(
        gradients_of_generator, generator$trainable_variables
      )))
      discriminator_optimizer$apply_gradients(purrr::transpose(
        list(gradients_of_discriminator, discriminator$trainable_variables)
      ))

      total_loss_gen <- total_loss_gen + gen_loss
      total_loss_disc <- total_loss_disc + disc_loss

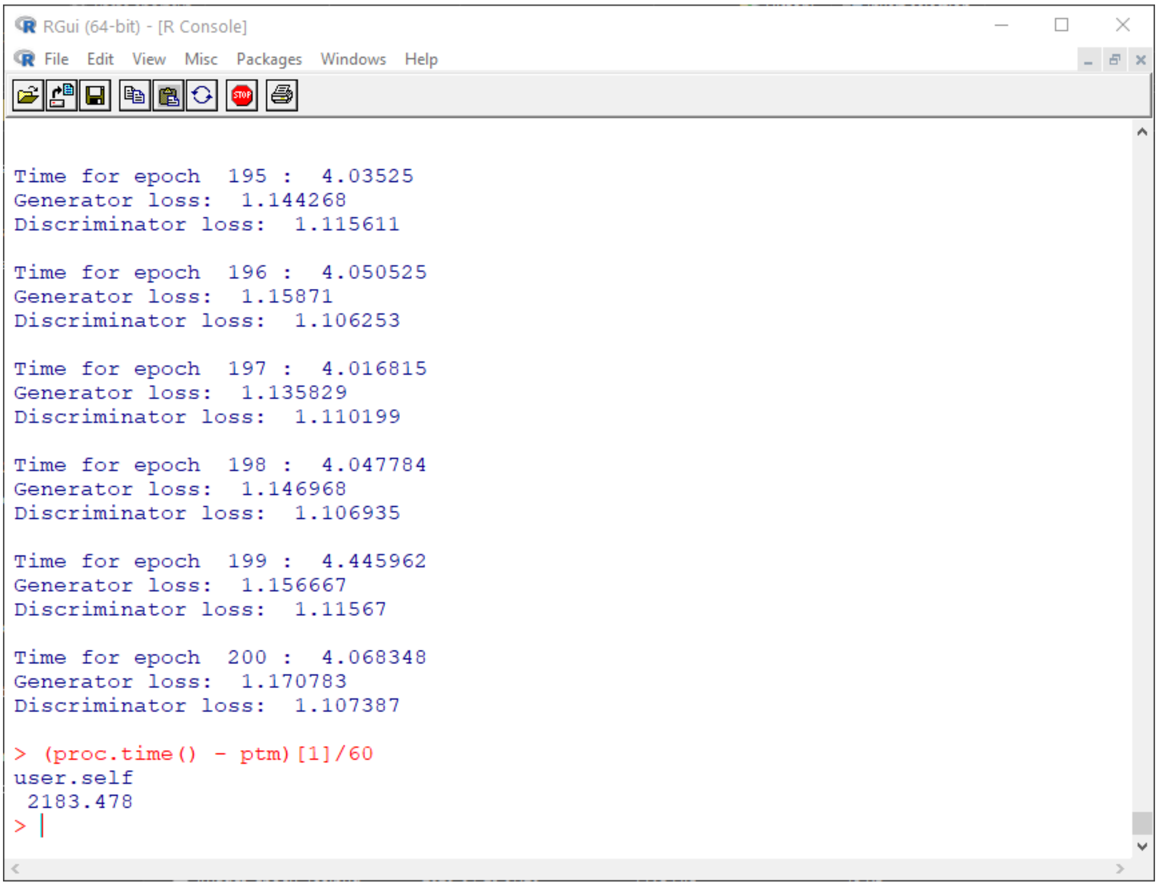
    })
```

```
    cat("Time for epoch ", epoch, ": ", Sys.time() - start, "\n")
    cat("Generator loss: ",
        total_loss_gen$numpy() / batches_per_epoch,
        "\n")
    cat("Discriminator loss: ",
        total_loss_disc$numpy() / batches_per_epoch,
        "\n\n")
    #if (epoch %% 10 == 0)
      generate_and_save_images(generator,
                               epoch,
                               random_vector_for_generation)

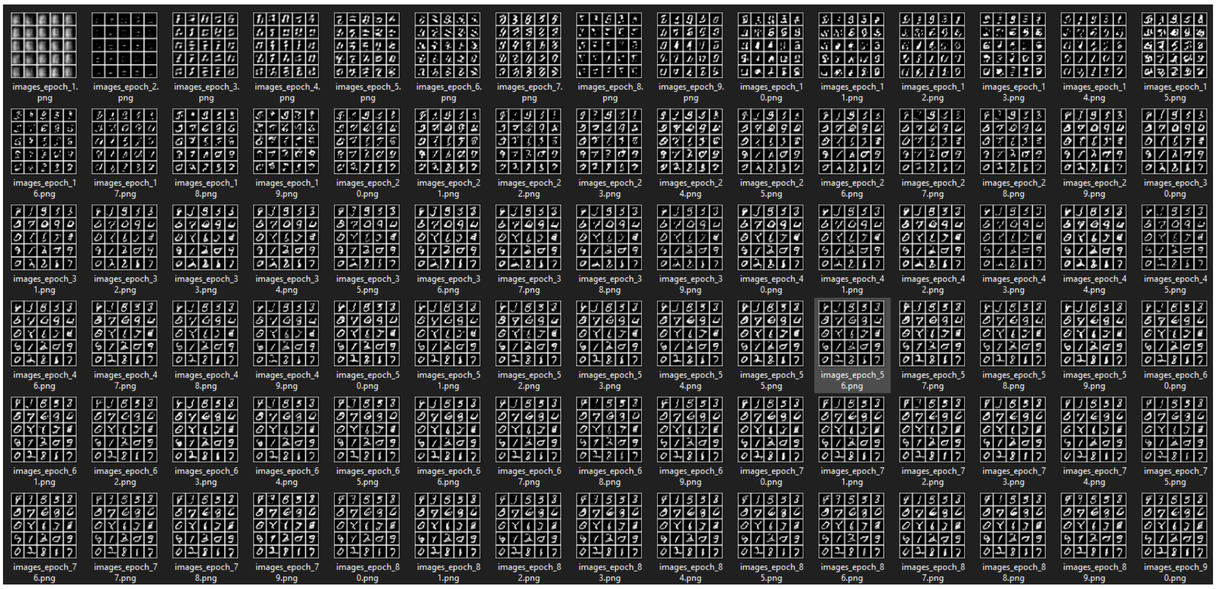
  }
}

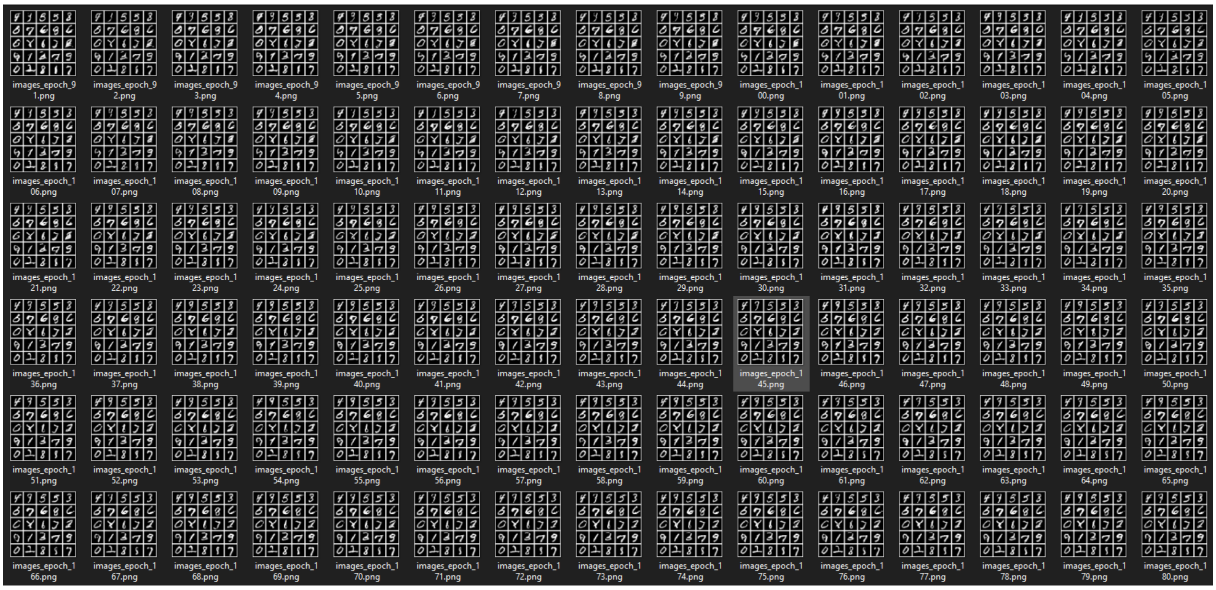
num_epochs <- 200
ptm <- proc.time()
train(train_dataset, num_epochs, noise_dim)
(proc.time() - ptm)[1]/60
```

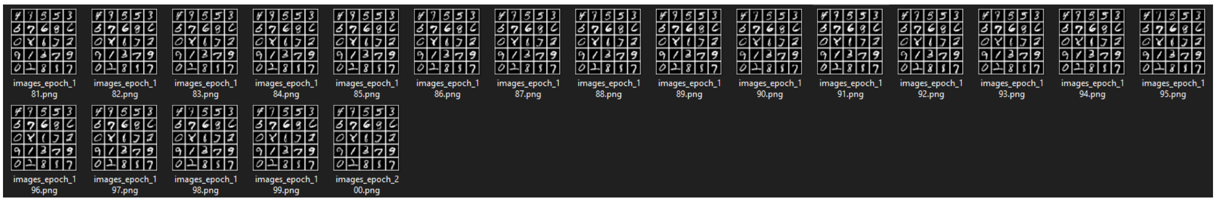
Comme on peut le voir il faut plusieurs heures pour faire les 200 EPOCH:



On peut voir l'évolution de l'apprentissage du réseau de neurone convolutionnel ci-dessous:







Vous pouvez demander l'ensemble du code dans un fichier unique nommé GAN.R au formateur ou copier/coller l'ensemble du code à partir de la page suivante:

```
#' This is the companion code to the post
#' "Generating digits with Keras and TensorFlow eager execution"
#' on the TensorFlow for R blog.
#'
#' https://blogs.rstudio.com/tensorflow/posts/2018-08-26-eager-dcgan/


library("keras")
library("tensorflow")
#tensorflow::install_tensorflow()
#Notons que pour utiliser la version GPU de tensorflow (ou de keras...):
#install_tensorflow(version = "gpu")
library("tfdatasets")

mnist <- keras::dataset_mnist()
c(train_images, train_labels) %<-% mnist$train

#just for the plot
c(c(trainx, trainy), c(testx, testy)) %<-% mnist
trainx <- trainx[trainy == 5,,]
par(mfrow = c(8, 8), mar = rep(0, 4))
for (i in 1:64) plot(as.raster(trainx[i,,], max = 255))
#finish about the plot


train_images <- train_images %>%
  k_expand_dims() %>%
  k_cast(dtype = "float32")

train_images <- (train_images - 127.5) / 127.5

buffer_size <- 60000
batch_size <- 256
batches_per_epoch <- (buffer_size / batch_size) %>% round()

train_dataset <- tensor_slices_dataset(train_images) %>%
  dataset_shuffle(buffer_size) %>%
  dataset_batch(batch_size)

generator <-
  function(name = NULL) {
    keras_model_custom(name = name, function(self) {
      self$fc1 <- layer_dense(units = 7 * 7 * 64, use_bias = FALSE)
      self$batchnorm1 <- layer_batch_normalization()
      self$leaky_relu1 <- layer_activation_leaky_relu()

      self$conv1 <-
        layer_conv_2d_transpose(
          filters = 64,
          kernel_size = c(5, 5),
          strides = c(1, 1),
          padding = "same",
          use_bias = FALSE
        )
      self$batchnorm2 <- layer_batch_normalization()
      self$leaky_relu2 <- layer_activation_leaky_relu()

      self$conv2 <-
        layer_conv_2d_transpose(
          filters = 32,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same",
          use_bias = FALSE
        )
      self$batchnorm3 <- layer_batch_normalization()
      self$leaky_relu3 <- layer_activation_leaky_relu()

      self$conv3 <-
        layer_conv_2d_transpose(
          filters = 1,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same",
          use_bias = FALSE,
          activation = "tanh"
        )

      function(inputs,
               mask = NULL,
               training = TRUE) {
        self$fc1(inputs) %>%
          self$batchnorm1(training = training) %>%
          self$leaky_relu1() %>%
          k_reshape(shape = c(-1, 7, 7, 64)) %>%

          self$conv1() %>%
          self$batchnorm2(training = training) %>%
          self$leaky_relu2() %>%

          self$conv2() %>%
          self$batchnorm3(training = training) %>%
          self$leaky_relu3() %>%

          self$conv3()
      }
    })
  }

discriminator <-
  function(name = NULL) {
    keras_model_custom(name = name, function(self) {
      self$conv1 <- layer_conv_2d(
        filters = 64,
        kernel_size = c(5, 5),
        strides = c(2, 2),
        padding = "same"
      )
      self$leaky_relu1 <- layer_activation_leaky_relu()
      self$dropout <- layer_dropout(rate = 0.3)

      self$conv2 <-
        layer_conv_2d(
          filters = 128,
          kernel_size = c(5, 5),
          strides = c(2, 2),
          padding = "same"
        )
      self$leaky_relu2 <- layer_activation_leaky_relu()
      self$flatten <- layer_flatten()
      self$fc1 <- layer_dense(units = 1)

      function(inputs,
               mask = NULL,
               training = TRUE) {
        inputs %>% self$conv1() %>%
          self$leaky_relu1() %>%
          self$dropout(training = training) %>%
          self$conv2() %>%
          self$leaky_relu2() %>%
          self$flatten() %>%
          self$fc1()

      }
    })
  }

generator <- generator()
discriminator <- discriminator()

generator$call <- tf$`function`(generator$call)
discriminator$call <- tf$`function`(discriminator$call)

cross_entropy = tf$keras$losses$BinaryCrossentropy(from_logits = TRUE)

discriminator_loss <- function(real_output, generated_output) {
  real_loss <- cross_entropy(k_ones_like(real_output), real_output)
  generated_loss <- cross_entropy(k_zeros_like(generated_output), generated_output)
  real_loss + generated_loss
}

generator_loss <- function(generated_output) {
  cross_entropy(tf$ones_like(generated_output), generated_output)
}

discriminator_optimizer <- tf$optimizers$Adam(1e-4)
generator_optimizer <- tf$optimizers$Adam(1e-4)

num_epochs <- 150
noise_dim <- 100
num_examples_to_generate <- 25L

random_vector_for_generation <-
  k_random_normal(c(num_examples_to_generate,
                    noise_dim))

generate_and_save_images <- function(model, epoch, test_input) {
  predictions <- model(test_input, training = FALSE)
  png(paste0("c:/tmp/images_epoch_", epoch, ".png"))
  par(mfcol = c(5, 5))
  par(mar = c(0.5, 0.5, 0.5, 0.5),
      xaxs = 'i',
      yaxs = 'i')
  for (i in 1:25) {
    img <- predictions[i, , , 1]
    img <- t(apply(img, 2, rev))
    image(
      1:28,
      1:28,
      img * 127.5 + 127.5,
      col = gray((0:255) / 255),
      xaxt = 'n',
      yaxt = 'n'
    )
  }
  dev.off()
}

train <- function(dataset, epochs, noise_dim) {
  for (epoch in seq_len(num_epochs)) {
    start <- Sys.time()
    total_loss_gen <- 0
    total_loss_disc <- 0
    iter <- make_iterator_one_shot(train_dataset)

    until_out_of_range({
      batch <- iterator_get_next(iter)
      noise <- k_random_normal(c(batch_size, noise_dim))
      with(tf$GradientTape() %as% gen_tape, {
        with(tf$GradientTape() %as% disc_tape, {
          generated_images <- generator(noise)
          disc_real_output <- discriminator(batch, training = TRUE)
          disc_generated_output <-
            discriminator(generated_images, training = TRUE)
          gen_loss <- generator_loss(disc_generated_output)
          disc_loss <-
            discriminator_loss(disc_real_output, disc_generated_output)
        })
      })

      gradients_of_generator <-
        gen_tape$gradient(gen_loss, generator$trainable_variables)
      gradients_of_discriminator <-
        disc_tape$gradient(disc_loss, discriminator$trainable_variables)

      generator_optimizer$apply_gradients(purrr::transpose(list(
        gradients_of_generator, generator$trainable_variables
      )))
      discriminator_optimizer$apply_gradients(purrr::transpose(
        list(gradients_of_discriminator, discriminator$trainable_variables)
      ))

      total_loss_gen <- total_loss_gen + gen_loss
      total_loss_disc <- total_loss_disc + disc_loss

    })

    cat("Time for epoch ", epoch, ": ", Sys.time() - start, "\n")
    cat("Generator loss: ",
        total_loss_gen$numpy() / batches_per_epoch,
        "\n")
    cat("Discriminator loss: ",
        total_loss_disc$numpy() / batches_per_epoch,
        "\n\n")
    #if (epoch %% 10 == 0)
      generate_and_save_images(generator,
                               epoch,
                               random_vector_for_generation)

  }
}

num_epochs <- 200
ptm <- proc.time()
train(train_dataset, num_epochs, noise_dim)
(proc.time() - ptm)[1]/60
```

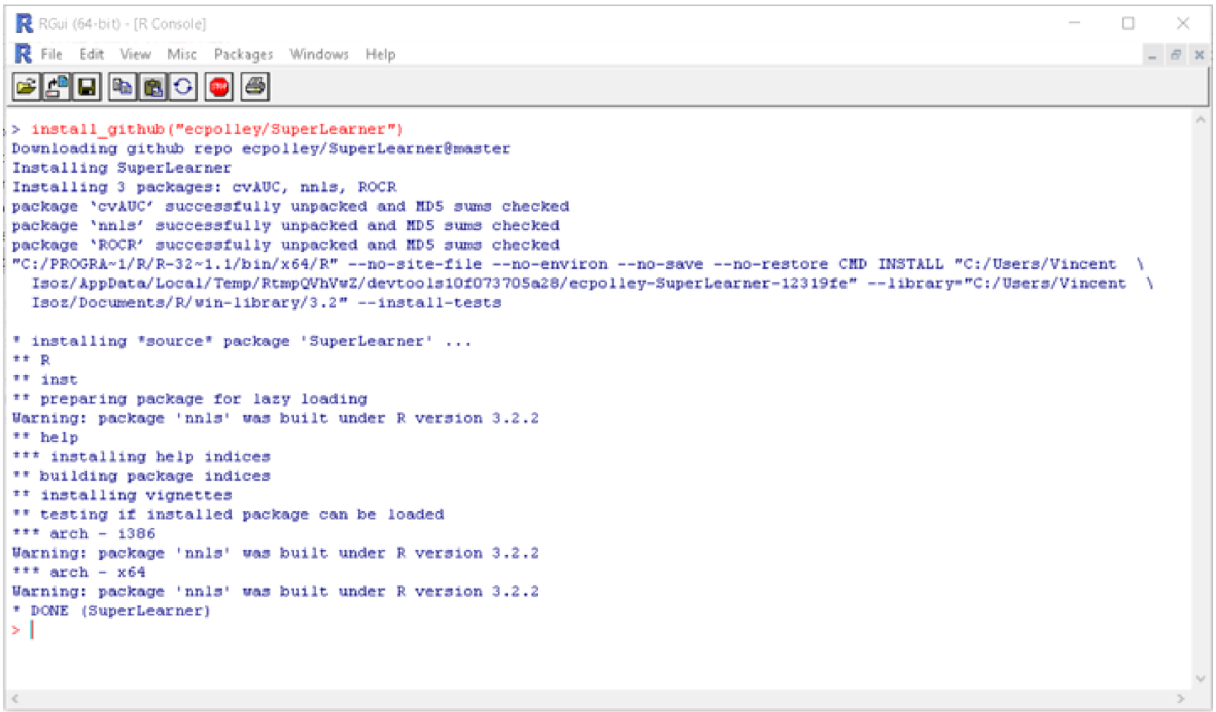
Exemple 362.: Ensemble Learning

R 3.2.1

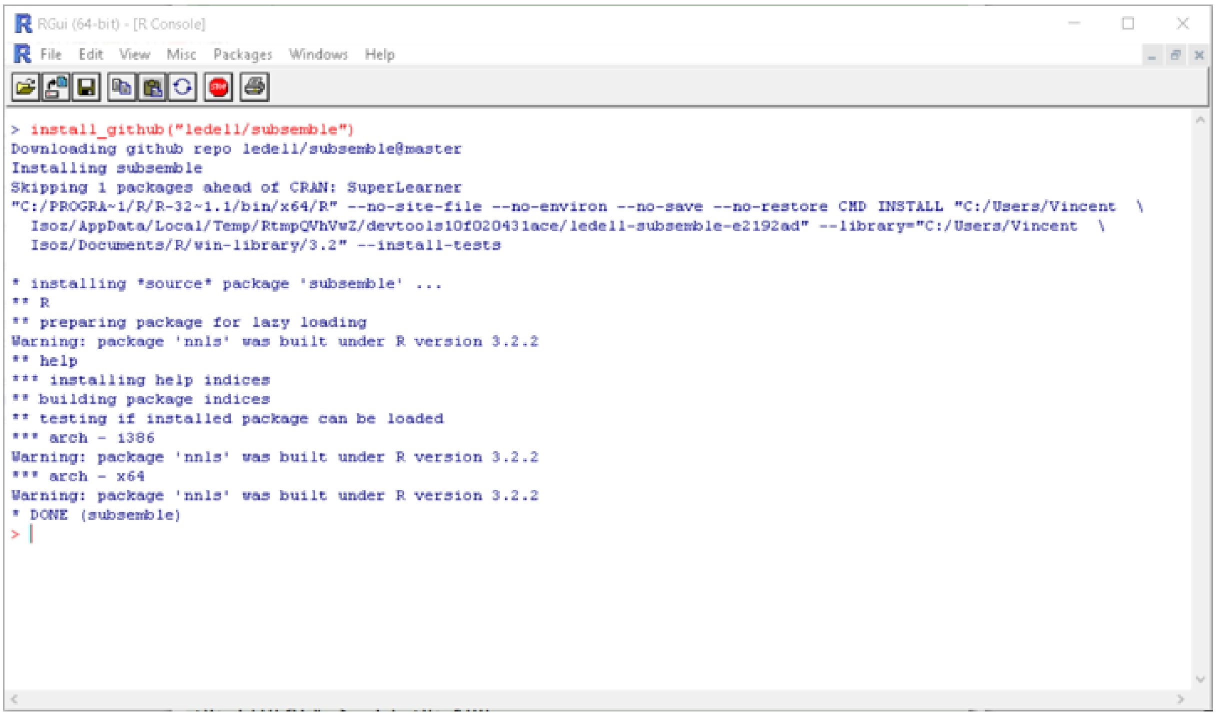
Le principe du "Ensemble Learning" est simple dans le principe. Il s'agit d'exécuter plusieurs techniques de fouilles de données sur un même jeu de données et ensuite faire de la prédiction en se basant sur une pondération optimale de chacun des modèles choisis afin de minimiser le taux d'erreur de prédiction.

Donc mathématiquement il n'y a rien de difficile car il suffit de connaître les techniques de base déjà étudiées précédemment, de les exécuter sur un même jeu de données, et ensuite par un solveur pondérer l'influence de chaque modèle.

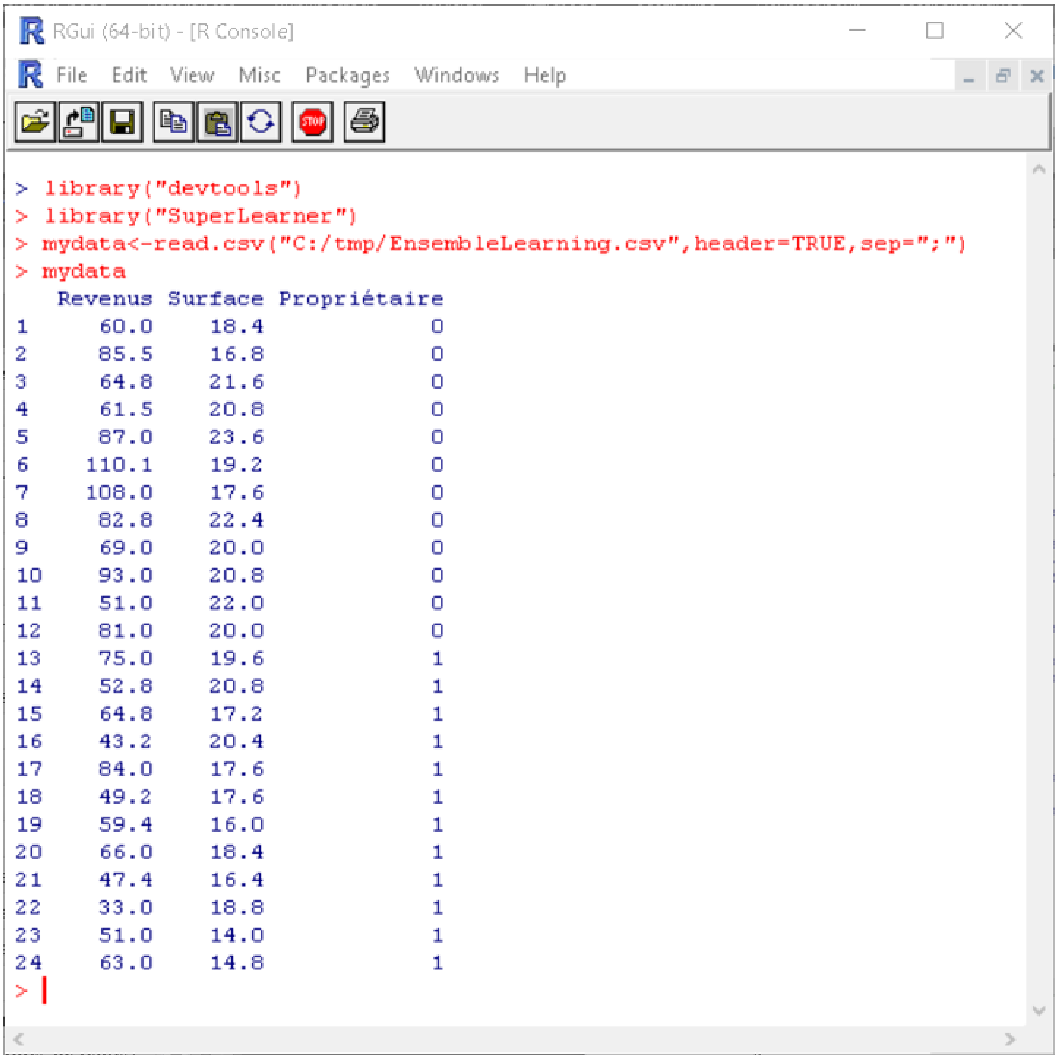
D'abord nous installons le package **SuperLearner** en passant par Github (du moins à ce jour) mais n'hésitez pas à l'installer via la méthode d'un package normal:



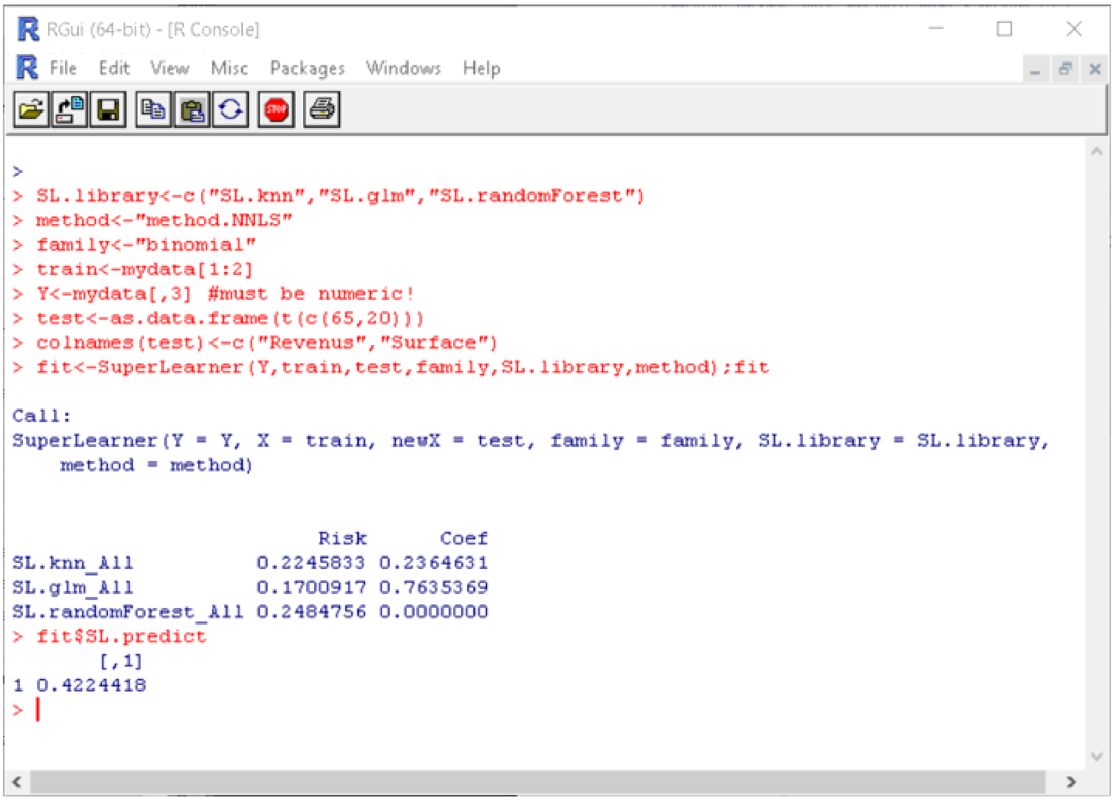
Et aussi:



On charge les données:



Maintenant on lance l'ensemble learning avec trois méthodes:



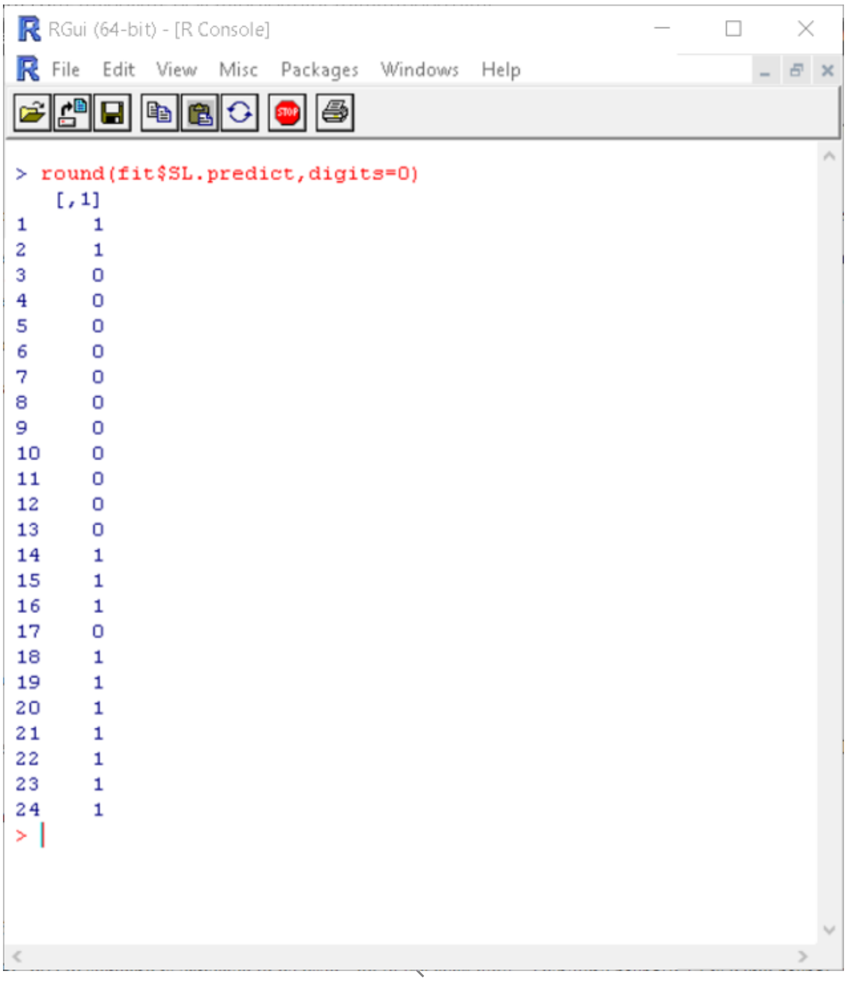
```
>
> SL.library<-c("SL.knn","SL.glm","SL.randomForest")
> method<-"method.NNLS"
> family<-"binomial"
> train<-mydata[1:2]
> Y<-mydata[,3] #must be numeric!
> test<-as.data.frame(t(c(65,20)))
> colnames(test)<-c("Revenus","Surface")
> fit<-SuperLearner(Y,train,test,family,SL.library,method);fit

Call:
SuperLearner(Y = Y, X = train, newX = test, family = family, SL.library = SL.library,
    method = method)


                         Risk      Coef
SL.knn_All          0.2245833 0.2364631
SL.glm_All          0.1700917 0.7635369
SL.randomForest_All 0.2484756 0.0000000
> fit$SL.predict
       [,1]
1 0.4224418
>
```

Nous y voyons donc les pondérations (*Coef*) de chaque modèle dans le prognostique final.

Pour notre jeu de données de départ cela donne:

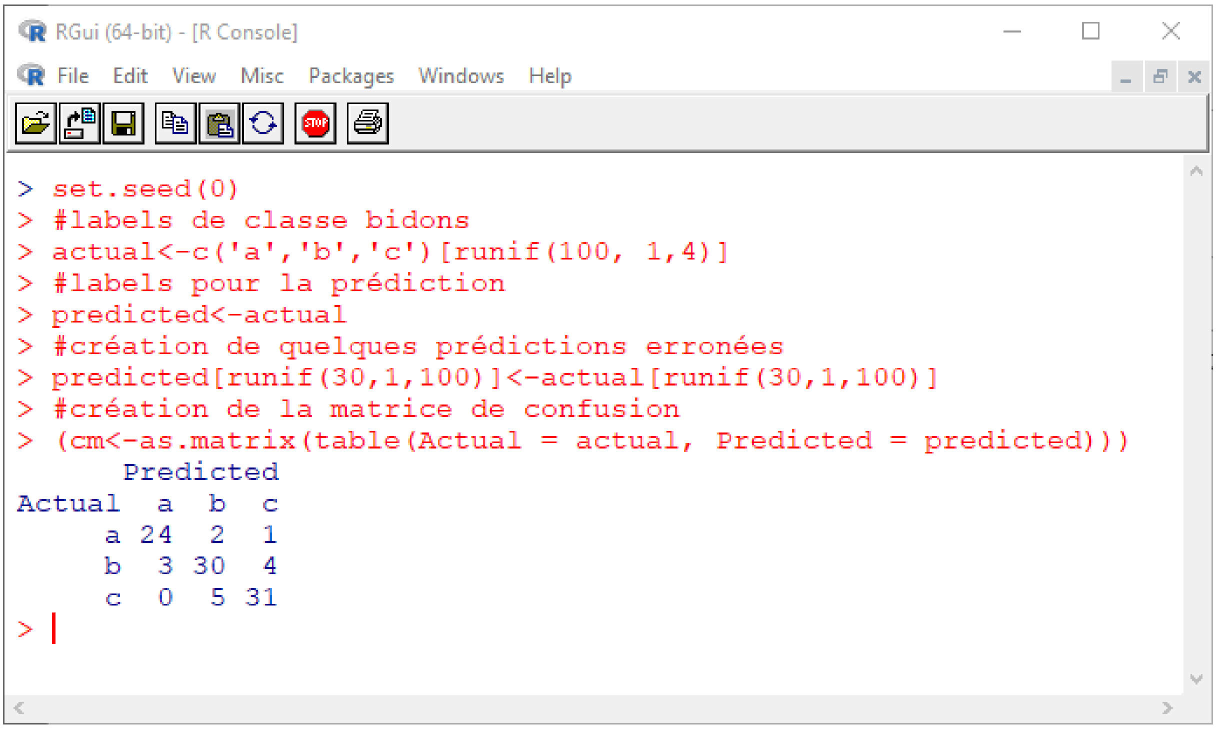


Donc la prévision n'est pas trop mauvaise (il faudrait faire une matrice de confusion mais nous avons déjà étudié cela).

Exemple 363.: Métriques de classifications

R 3.6.2

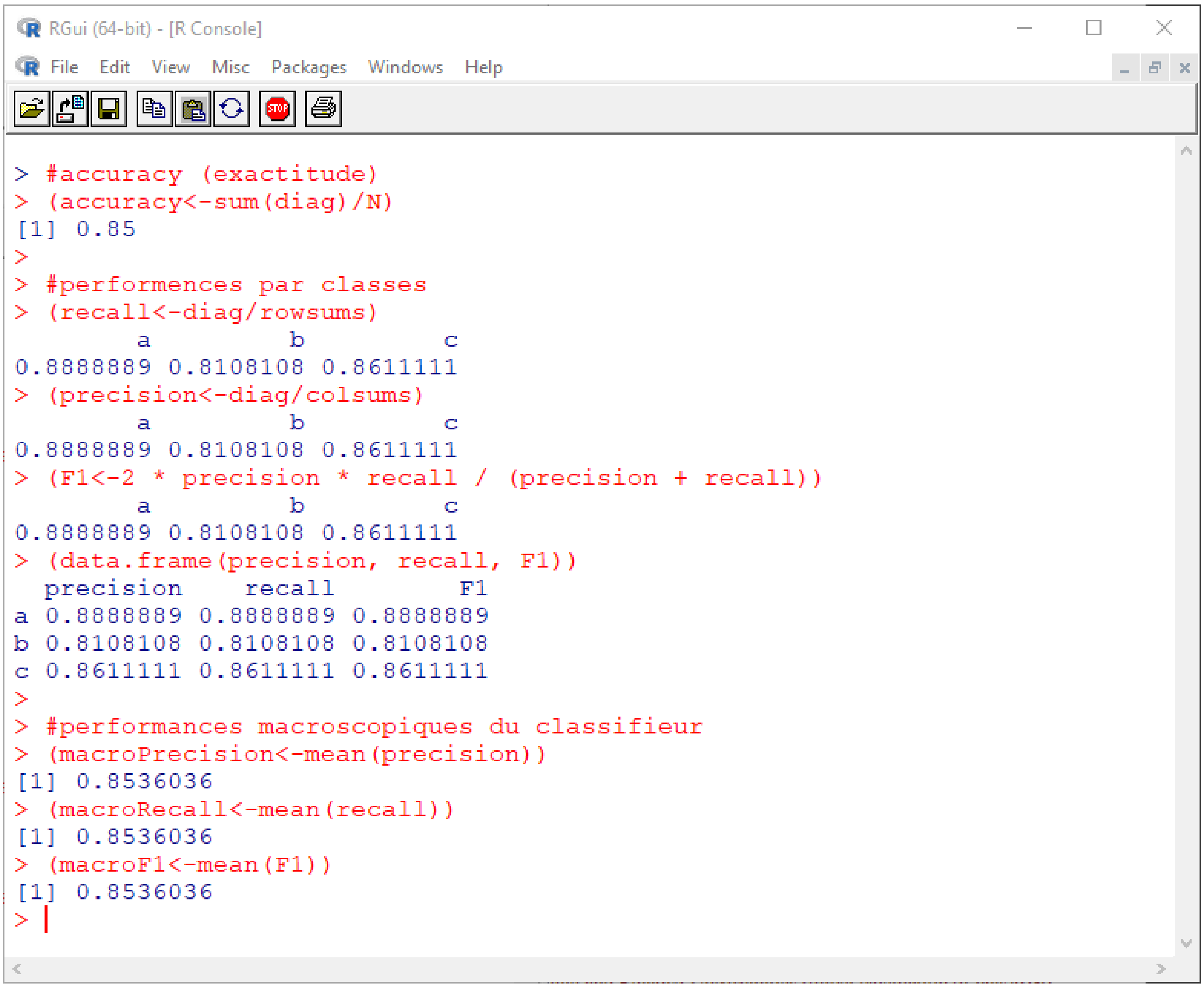
Nous avons vu dans le cours théorique quelques unes des nombres métriques de classifications. Passons ces dernières en revue en créant une matrice de confusion à la volée:



Commençons par quelques statistiques triviales:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help
> #nombre d'instances
> (N<-sum(cm))
[1] 100
> #nombre de classes
> (nc<-nrow(cm))
[1] 3
> #nombre correcte d'instances classifiées toutes classes confondues
> (diag<-diag(cm))
 a  b  c
24 30 31
> #nombre d'instances par classes
> (rowsums<-apply(cm, 1, sum))
 a  b  c
27 37 36
> #nombre de prédictions par classes
> (colsums<-apply(cm, 2, sum))
 a  b  c
27 37 36
> #distribution des instances à travers les classes
> (p<-rowsums/N)
   a    b    c
0.27 0.37 0.36
> #distribution des instances à travers les classes prédites
> (q<-colsums/N)
   a    b    c
0.27 0.37 0.36
> |
```

Maintenant calculons les indicateurs de performance classiques:



```
> #accuracy (exactitude)
> (accuracy<-sum(diag)/N)
[1] 0.85
>
> #performences par classes
> (recall<-diag/rowsums)
        a         b         c
0.8888889 0.8108108 0.8611111
> (precision<-diag/colsums)
        a         b         c
0.8888889 0.8108108 0.8611111
> (F1<-2 * precision * recall / (precision + recall))
        a         b         c
0.8888889 0.8108108 0.8611111
> (data.frame(precision, recall, F1))
  precision    recall        F1
a 0.8888889 0.8888889 0.8888889
b 0.8108108 0.8108108 0.8108108
c 0.8611111 0.8611111 0.8611111
>
> #performances macroscopiques du classifieur
> (macroPrecision<-mean(precision))
[1] 0.8536036
> (macroRecall<-mean(recall))
[1] 0.8536036
> (macroF1<-mean(F1))
[1] 0.8536036
>
```

Allons-y maintenant pour la matrice one-vs-all et ses micrométriques:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

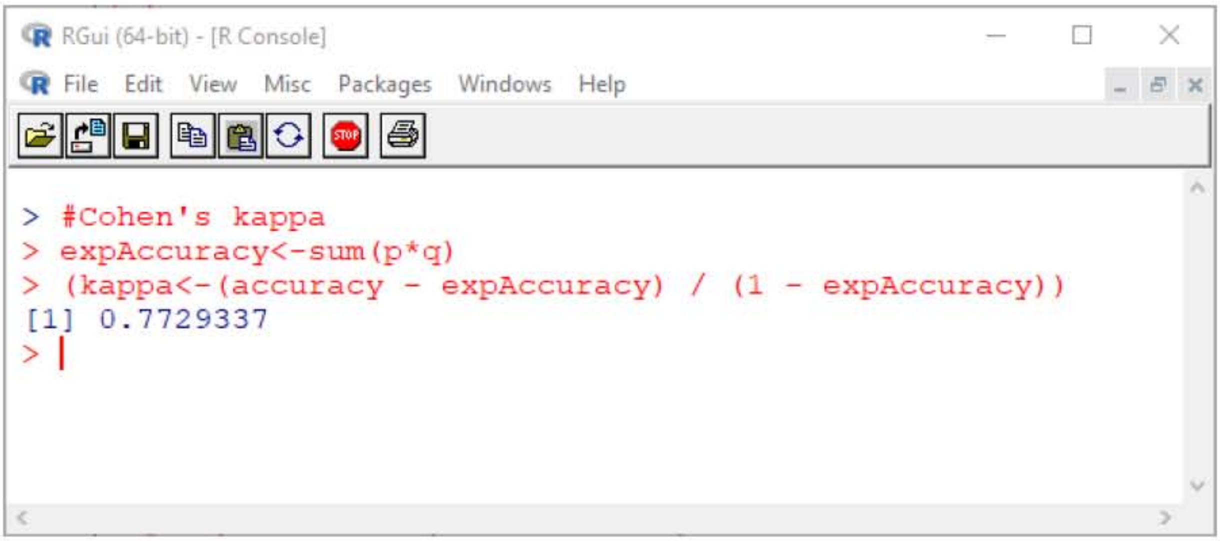
> #matrice confusion oneVsAll
> (oneVsAll<-lapply(1:nc,
+ function(i){
+             v = c(cm[i,i],
+                   rowsums[i] - cm[i,i],
+                   colsums[i] - cm[i,i],
+                   N-rowsums[i] - colsums[i] + cm[i,i]);
+                   return(matrix(v, nrow = 2, byrow = T))
+ }
+ )
+ )
[[1]]
     [,1] [,2]
[1,]   24    3
[2,]    3   70

[[2]]
     [,1] [,2]
[1,]   30    7
[2,]    7   56

[[3]]
     [,1] [,2]
[1,]   31    5
[2,]    5   59

>
> #fusion des matrices de confusion
> s<-matrix(0, nrow = 2, ncol = 2)
> for(i in 1 : nc){s = s + oneVsAll[[i]]}
> s
     [,1] [,2]
[1,]   85   15
[2,]   15  185
>
> #exactitude moyenne
> (avgAccuracy<-sum(diag(s)) / sum(s))
[1] 0.9
>
> #micro métriques moyennes
> (micro_prf<-(diag(s) / apply(s,1, sum))[1])
[1] 0.85
> 
```

Et on peut aussi calculer le kappa de Cohen:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> #Cohen's kappa
> expAccuracy<-sum(p*q)
> (kappa<-(accuracy - expAccuracy) / (1 - expAccuracy))
[1] 0.7729337
>

Exemple 364.: Calibration de modèles

R 4.1.1

Nous allons ici faire un exemple d'un concept que nous avons survolé dans le cours théorique et qui est la "calibration" d'un modèle. Malheureusement, je n'ai pas trouvé de package R qui puisse s'applique de manière générique à tous les modèles de machine learning auxquels la calibration s'impose. Le meilleur package existant à ce jour et à ma connaissance est le package **CORElearn** que nous allons utiliser ici. Il est cependant limité (vois la documentation du PDF de ce package pour plus de détails et mises à jour) à:

- Classification:
    - Forêts aléatoires avec ou sans poids (exécution en parallèle sur plusieur cœurs)
    - Arbres de décisions
    - kNN et kNN pondérés avec noyau gaussien
    - Classificateur bayésien naïf

- Régression:
    - Arbres de régression
    - Modèles linéaires avec prunnage
    - Régression locale pondérée
    - K-NN et kNN pondérés avec noyau gaussien

Nous pourrions nous concentrer sur le plus simple de tous et qui est facilement tractable à la main: le classificateur bayésien naïf! Cependant ce package ne gère que ce classificateur avec des entrées numériques et non textuelles. Donc nous nous reporterons sur le kNN.

Nous allons donc reprendre le même exemple et le même jeu de données que celui à la page Exemple 349.: Clustering DBSCAN

R 4.3.2

Nous avons u dans le cours théoriquement que le DBSCAN est un algorithem de clustering non-linéaire très simple. Nous allons voir ici comment le mettre en pratique.

Nous chargeons le jeu de données suivant:

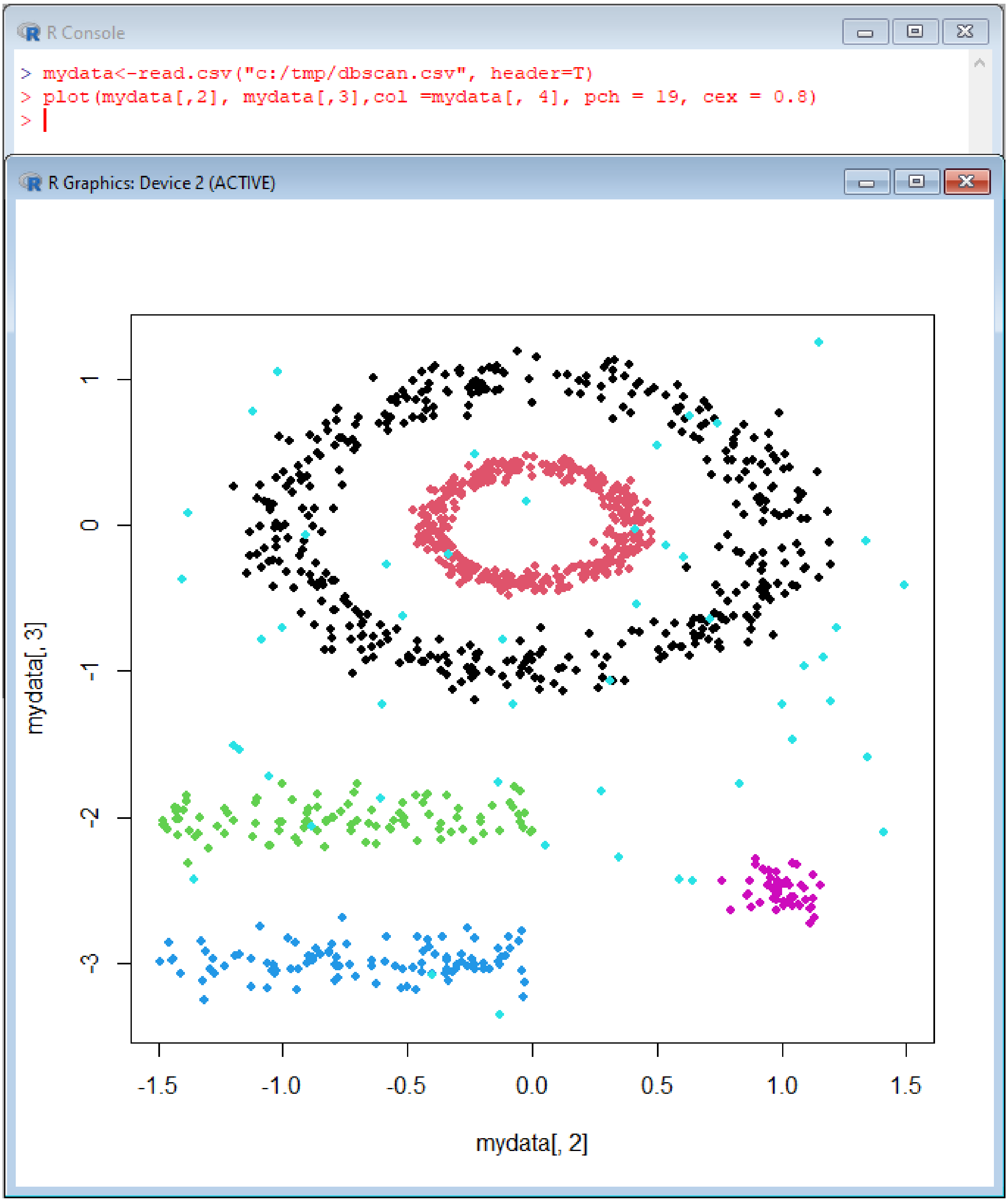
dbscan.csv - Notepad

File Edit Format View Help

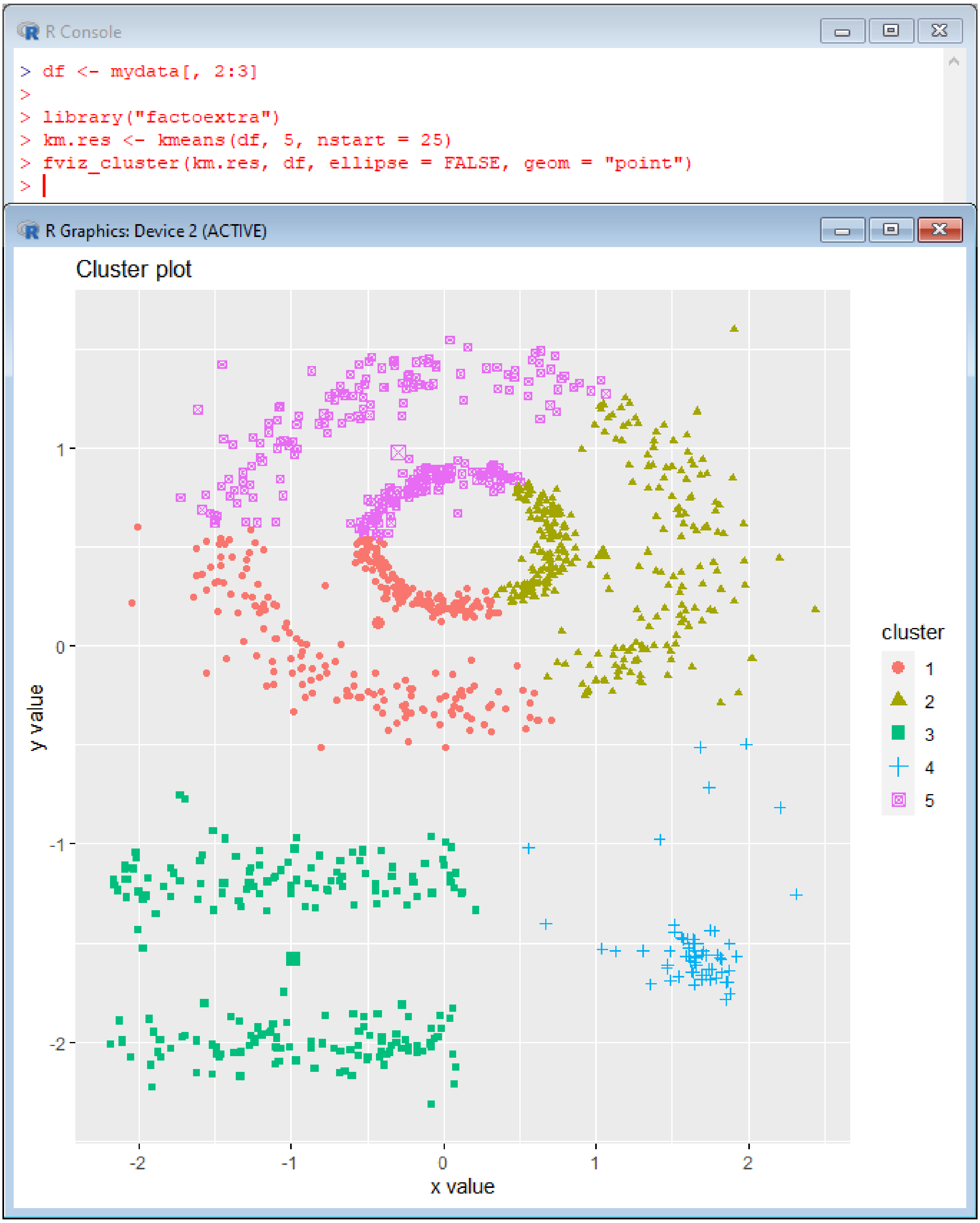
```
"ID","x","y","shape"
"1",-0.803739264931451,-0.853052609097935,1
"2",0.852850728148773,0.367618436999606,1
"3",0.92717950610565З,-0.274902437566225,1
"4",-0.752626056626365,-0.511565170496435,1
"5",0.706846224294882,0.81067919693492,1
"6",1.0346985222527,0.394655023166806,1
"7",-0.475845197699957,0.989760805249143,1
"8",-0.460301566967151,-0.858997792847955,1
"9",-0.680301544955355,-0.66149481321353,1
"10",-1.03196929988978,-0.0219935446644728,1
"11",-0.242498372239704,-1.05033275558452,1
"12",-0.892258364269624,-0.339169233215386,1
"13",-0.802034403520755,-0.77416776747668,1
"14",-0.0471955691399568,-1.01613798584755,1
"15",1.19470426661764,-0.269002460987329,1
"16",-0.221487390175705,-0.919131895891675,1
"17",-0.410475320013416,0.964479004794388,1
"18",-0.838935270223631,-0.407979035023443,1
"19",-0.889147510805223,0.614016108166731,1
"20",0.905564127260602,0.184568396547991,1
"21",-0.00579485156131487,-0.884430709419748,1
"22",-0.703810394965576,0.54749249234897,1
"23",-1.07081570617814,-0.158982746612178,1
"24",1.00723761879231,0.385175352050115,1
"25",-0.584861589344916,0.700117705614835,1
"26",-0.881140152628223,-0.0118887593850608,1
"27",0.326995042961589,0.998908556122335,1
"28",-1.04762113766461,0.118543573060775,1
"29",-0.119960489461039,-1.03541072663439,1
"30",-0.785532935689462,0.713418469247788,1
"31",0.64626878403344,-0.643552160711392,1
"32",-0.224939369705449,-0.883275717806866,1
"33",-0.719297745948344,-1.0133377180508,1
"34",-0.495498158632031,-0.919302400100962,1
"35",-0.865554333876342,-0.255682821707783,1
"36",-0.480275760487044,0.734425872156833,1
```

Ln 1, Col 4 100% Windows (CRLF) UTF-8

Voyons à quoi cela ressemble visuellement dans un premier temps:

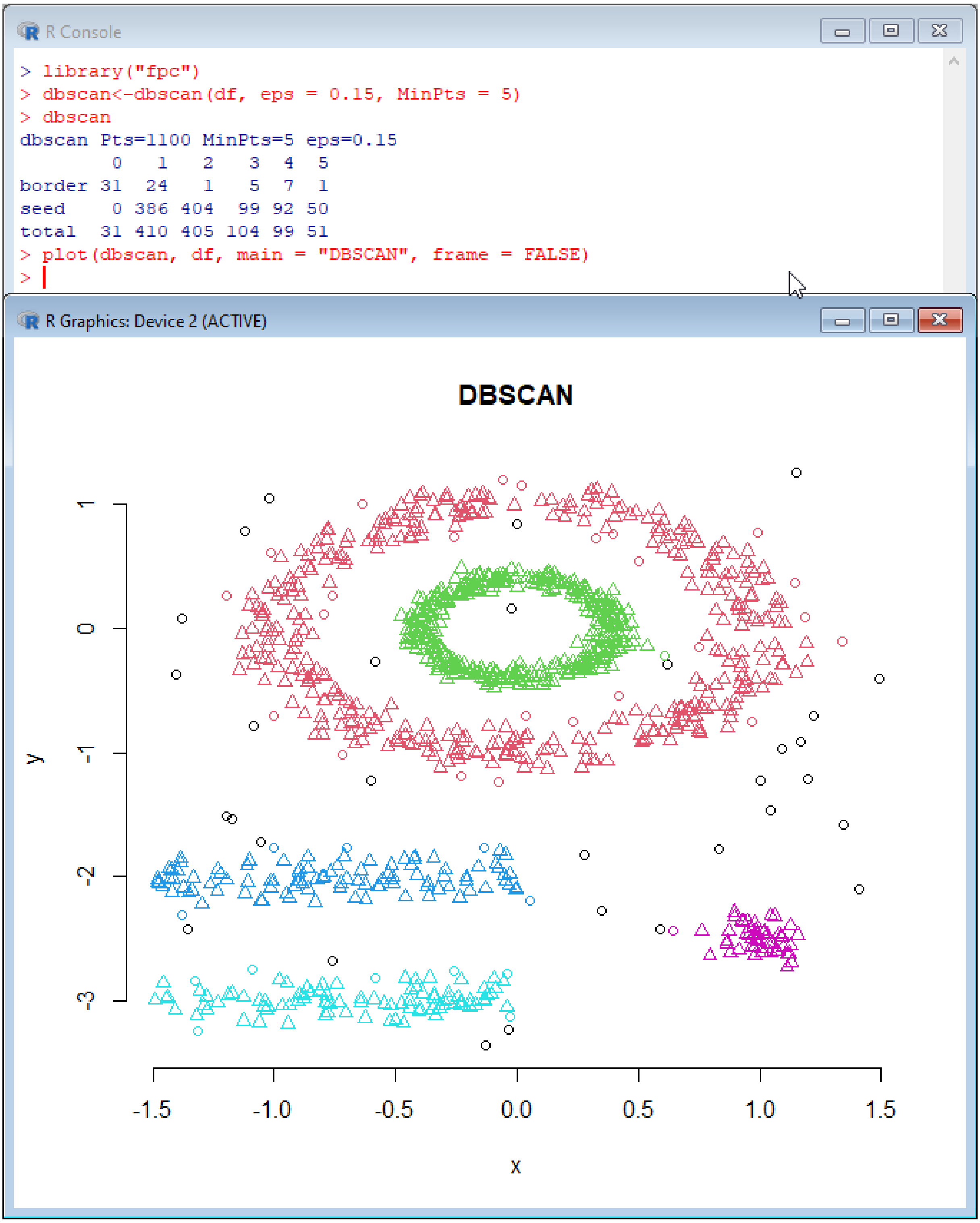


Faisons d'abord un clustering avec la méthode des k-means juste pour voir… :



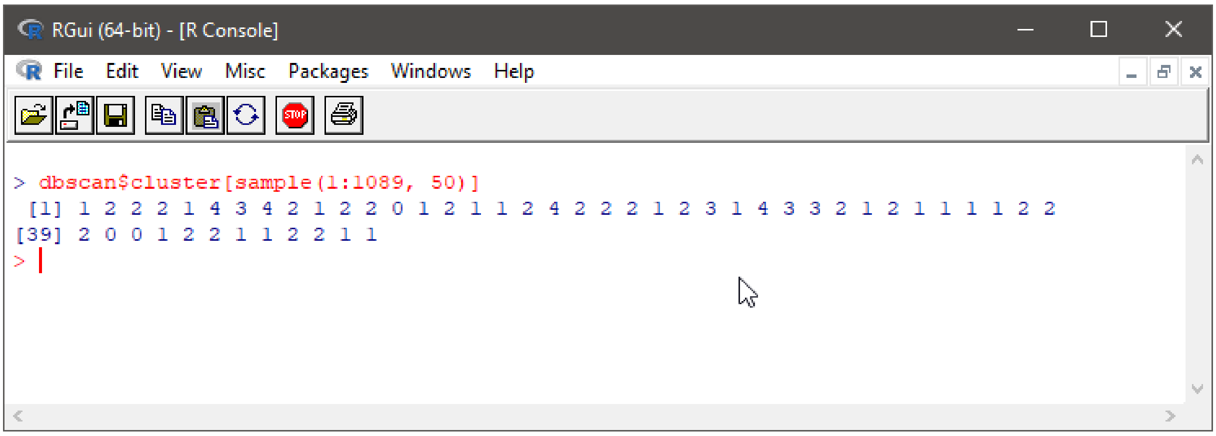
Bref vu la non-linéarité du jeu de données, nous ne sommes pas trop étonnés de la médiocrité du résultat.

Essayons maintenant avec la méthode DBSCAN:



C'est beaucoup mieux et ce sans surprises!

Nous pouvons aussi voir à quel cluster appartient quel ligne de notre jeu de données comme suit:



Exemple 350.: Classification bayésienne naïve.

Exemple 365.: Réseau de neurones de classification d'images

R 3.6.2 **x64**

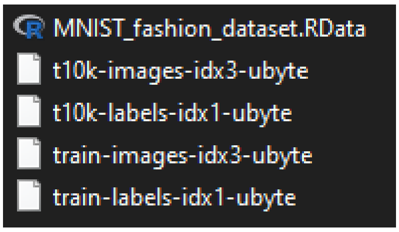
L'exemple qui suit est totalement repris du tutorial disponible ici:

https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_classification/

Le jeu de données MNIST fashion qui va être utilisé dans ce qui suit est disponible ici:

https://github.com/zalandoresearch/fashion-mnist

Ou mis dans le dossier d'exercices de votre formateur si le lien ci-dessus n'est plus fonctionnel:



Le Fashion MNIST contient 70'000 images en niveaux de gris dans 10 catégories. Les images montrent des vêtements individuels à basse résolution (28 par 28 pixels), comme on le voit ici:

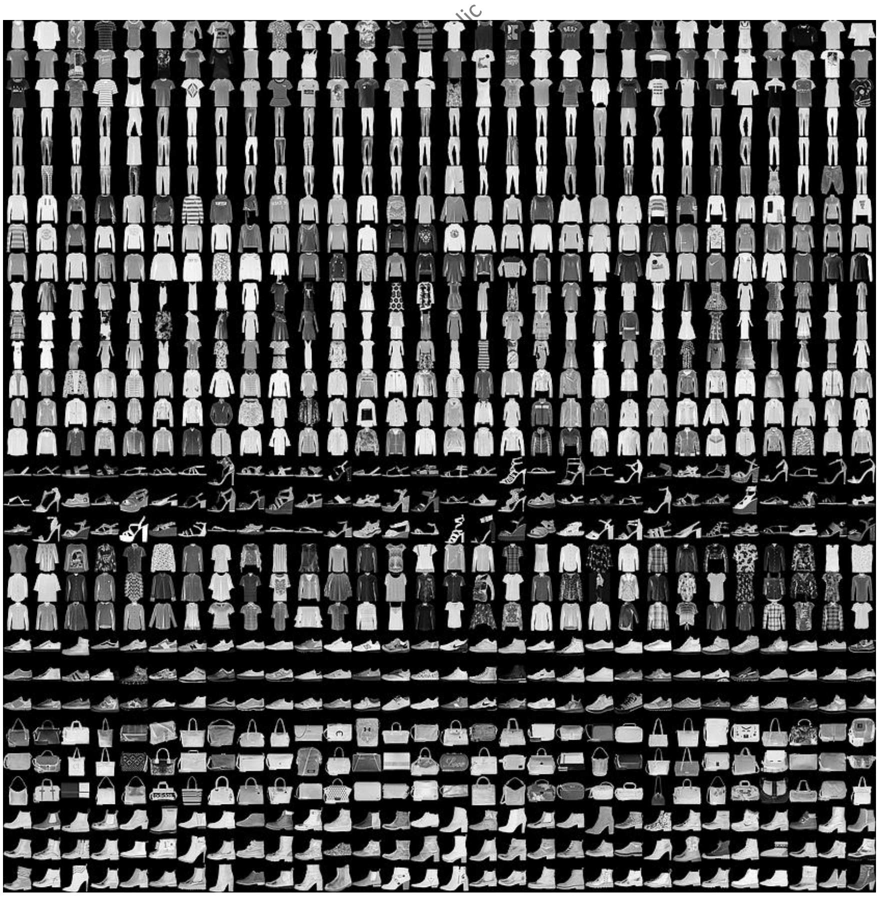
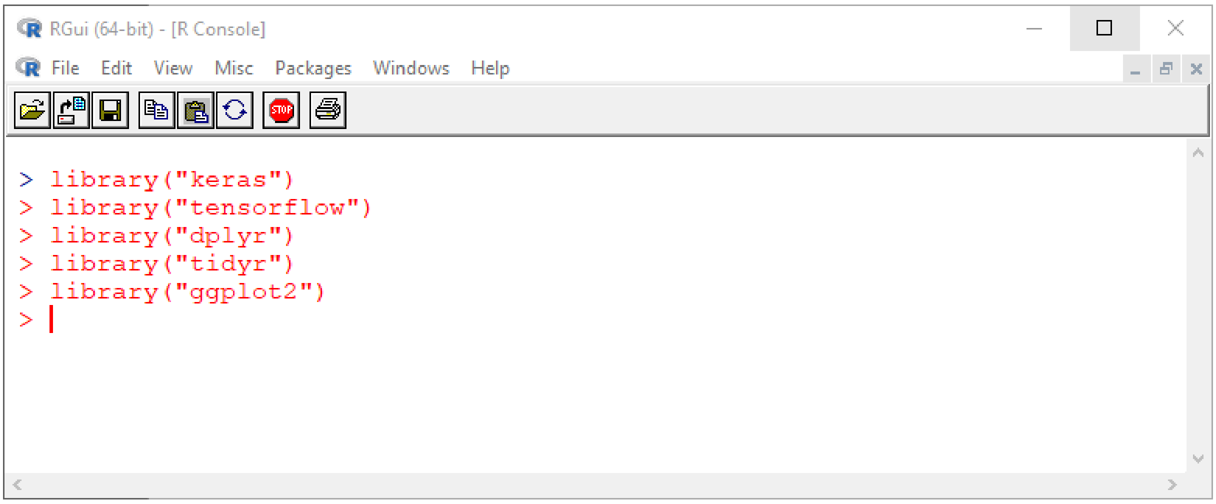


Figure 306 Fashion-MNIST samples (by Zalando, MIT License).

Pour ceux qui souhaitent apprendre à créer des jeux d'images de ce type à partir de leur propre bibliothèque, nous les renvoyons à le lecture sur la gestion de listes R (voir page 296) et du Traitement d'images dans R (voir page 3005).

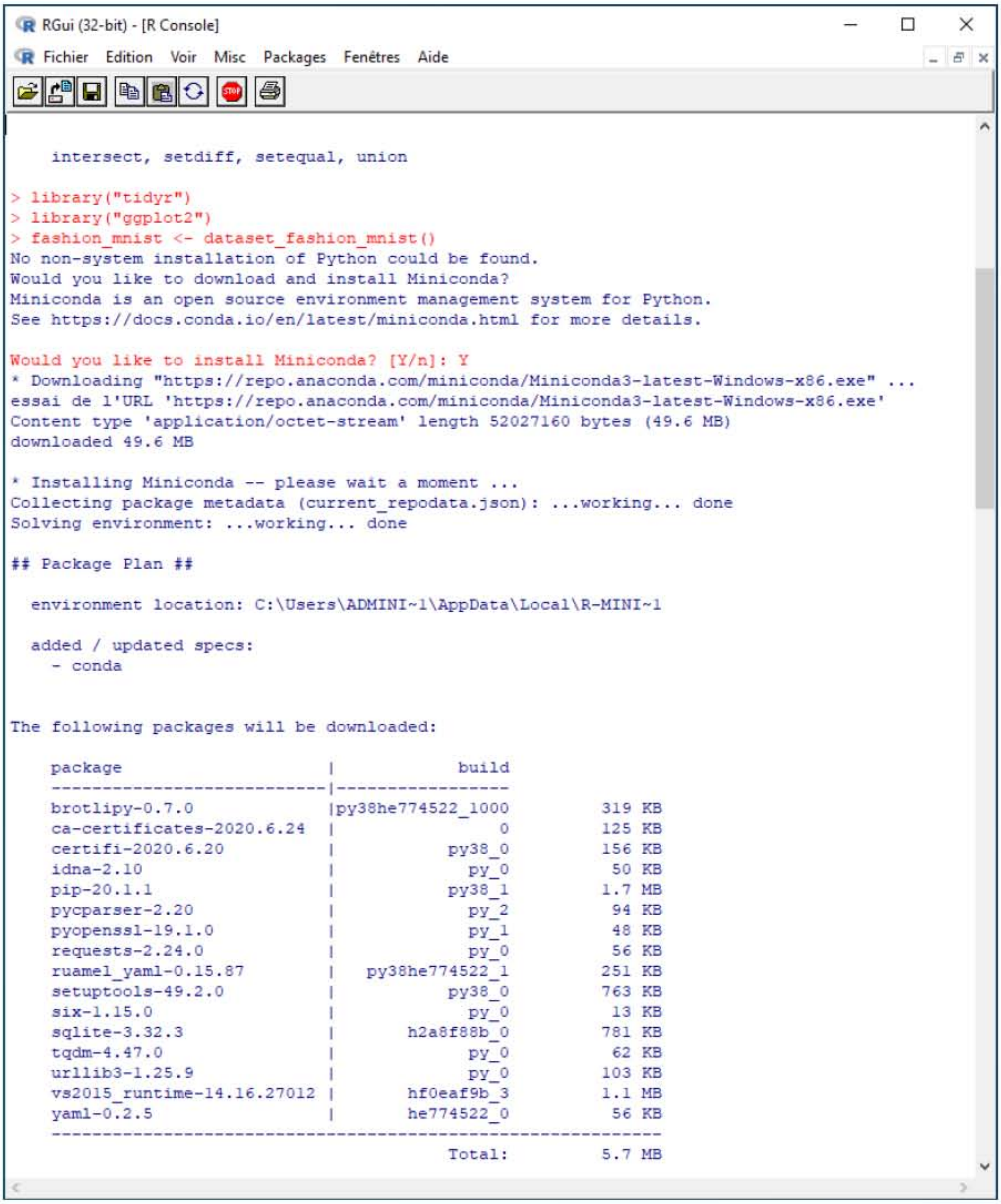
D'abord nous chargeons toutes les librairies dont nous aurons besoin:



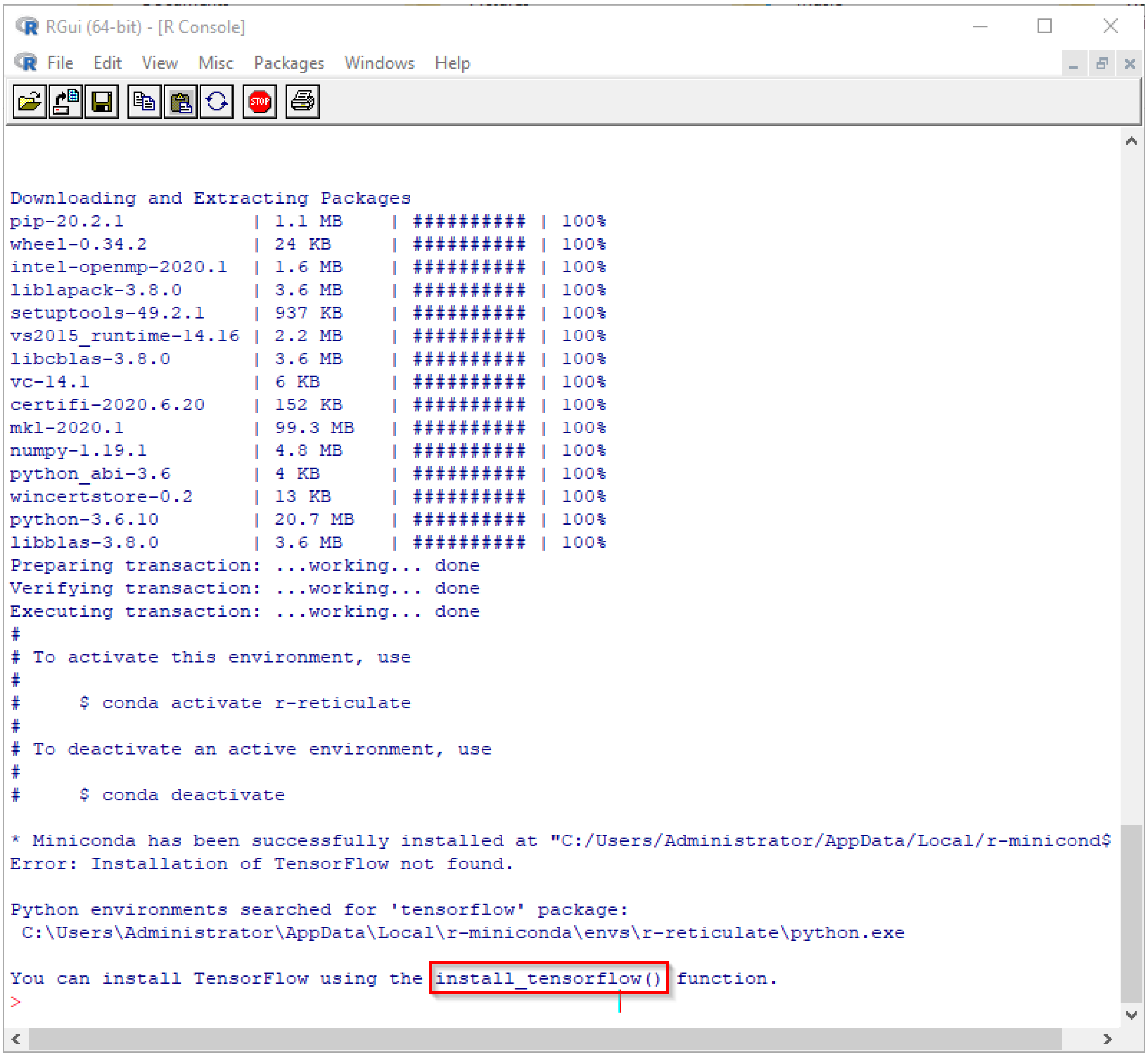
Nous chargeons le jeu de donnés directement depuis Keras avec la fonction suivante:



Si Python ou Miniconda ne sont pas installés vous aurez:


RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
intersect, setdiff, setequal, union
> library("tidyr")
> library("ggplot2")
> fashion_mnist <- dataset_fashion_mnist()
No non-system installation of Python could be found.
Would you like to download and install Miniconda?
Miniconda is an open source environment management system for Python.
See https://docs.conda.io/en/latest/miniconda.html for more details.
Would you like to install Miniconda? [Y/n]: Y
* Downloading "https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86.exe" ...
essai de l'URL 'https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86.exe'
Content type 'application/octet-stream' length 52027160 bytes (49.6 MB)
downloaded 49.6 MB
* Installing Miniconda -- please wait a moment ...
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done
Package Plan
environment location: C:\Users\ADMINI~1\AppData\Local\R-MINI~1
added / updated specs:
- conda
The following packages will be downloaded:
package	build
brotlipy-0.7.0 |py38he774522_1000 319 KB
ca-certificates-2020.6.24	0 125 KB
certifi-2020.6.20	py38_0 156 KB
idna-2.10	py_0 50 KB
pip-20.1.1	py38_1 1.7 MB
pycparser-2.20	py_2 94 KB
pyopenssl-19.1.0	py_1 48 KB
requests-2.24.0	py_0 56 KB
ruamel_yaml-0.15.87	py38he774522_1 251 KB
setuptools-49.2.0	py38_0 763 KB
six-1.15.0	py_0 13 KB
sqlite-3.32.3	h2a8f88b_0 781 KB
tqdm-4.47.0	py_0 62 KB
urllib3-1.25.9	py_0 103 KB
vs2015_runtime-14.16.27012	hf0eaf9b_3 1.1 MB
yaml-0.2.5	he774522_0 56 KB

Total: 5.7 MB



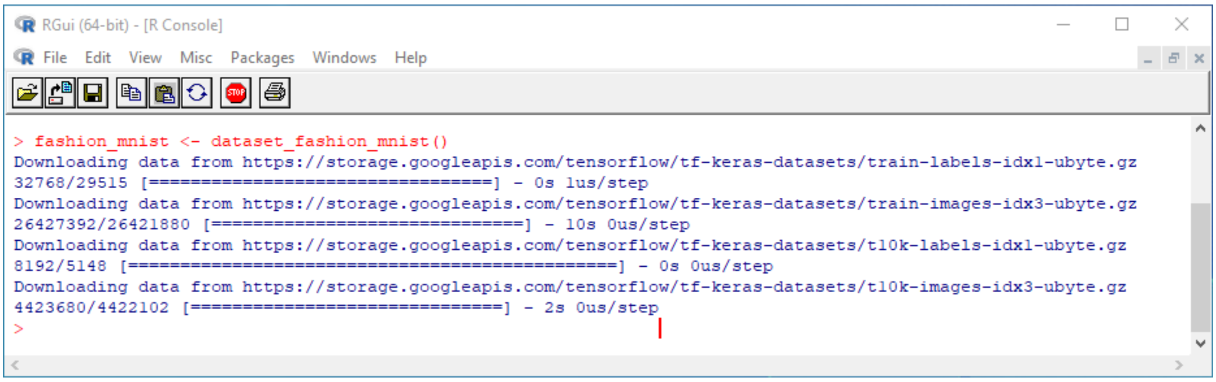
```
Downloading and Extracting Packages
pip-20.2.1           | 1.1 MB    | ########## | 100%
wheel-0.34.2         | 24 KB     | ########## | 100%
intel-openmp-2020.1  | 1.6 MB    | ########## | 100%
liblapack-3.8.0      | 3.6 MB    | ########## | 100%
setuptools-49.2.1    | 937 KB    | ########## | 100%
vs2015_runtime-14.16 | 2.2 MB    | ########## | 100%
libcblas-3.8.0       | 3.6 MB    | ########## | 100%
vc-14.1              | 6 KB      | ########## | 100%
certifi-2020.6.20    | 152 KB    | ########## | 100%
mkl-2020.1           | 99.3 MB   | ########## | 100%
numpy-1.19.1         | 4.8 MB    | ########## | 100%
python_abi-3.6       | 4 KB      | ########## | 100%
wincertstore-0.2     | 13 KB     | ########## | 100%
python-3.6.10        | 20.7 MB   | ########## | 100%
libblas-3.8.0        | 3.6 MB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done
#
# To activate this environment, use
#
#     $ conda activate r-reticulate
#
# To deactivate an active environment, use
#
#     $ conda deactivate

* Miniconda has been successfully installed at "C:/Users/Administrator/AppData/Local/r-minicond$
Error: Installation of TensorFlow not found.

Python environments searched for 'tensorflow' package:
 C:\Users\Administrator\AppData\Local\r-miniconda\envs\r-reticulate\python.exe

You can install TensorFlow using the install_tensorflow() function.
>
```

Et dès lors vous pourrez charger le jeu de données sans erreurs:



```
> fashion_mnist <- dataset_fashion_mnist()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 10s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 2s 0us/step
>
```

On voit qu'une fois Miniconda installé, il nous est demandé d'installer Tensorflow avec la fonction **install_tensorflow( )**:

```
> install_tensorflow()
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\Administrator\AppData\Local\r-miniconda\envs\r-reticulate

  added / updated specs:
    - python=3.6


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    certifi-2020.6.20          |           py36_0         156 KB
    python-3.6.10              |       h9f7ef89_2        15.9 MB
    ------------------------------------------------------------
                                           Total:        16.1 MB

The following NEW packages will be INSTALLED:

  sqlite             pkgs/main/win-64::sqlite-3.32.3-h2a8f88b_0
  zlib               pkgs/main/win-64::zlib-1.2.11-h62dcd97_4

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            conda-forge::certifi-2020.6.20-py36h9~ --> pkgs/main::certifi-2020.6.20-py$
  python             conda-forge::python-3.6.10-he025d50_1~ --> pkgs/main::python-3.6.10-h9f7ef$



Downloading and Extracting Packages
certifi-2020.6.20    | 156 KB    | ########## | 100%
python-3.6.10        | 15.9 MB   | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
```

On sépare les deux listes visibles ci-dessus et on fouille un peu leur contenu:

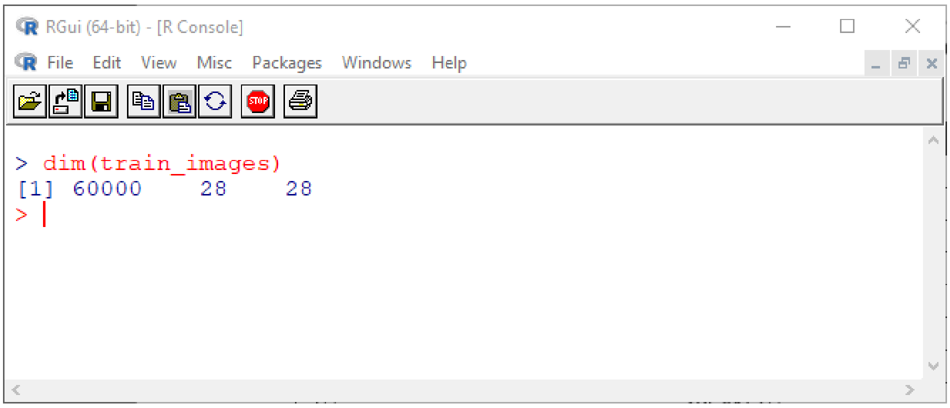
```
> #on sépare les deux listes du jeu de de données
> c(train_images, train_labels) %<-% fashion_mnist$train
> c(test_images, test_labels) %<-% fashion_mnist$test
> ls()
[1] "fashion_mnist" "test_images"   "test_labels"   "train_images"  "train_labels"
> class(test_images)
[1] "array"
> class(test_labels)
[1] "array"
> str(test_images)
 int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
> str(test_labels)
 int [1:10000(1d)] 9 2 1 1 6 1 4 6 5 7 ...
> 
```

À ce stade, nous avons quatre tableaux: Les tableaux *train_images* et *train_labels* sont l'ensemble d'apprentissage - les données que le modèle utilise pour apprendre. Le modèle est testé par rapport à l'ensemble de test: les tableaux *test_images* et *test_labels*.

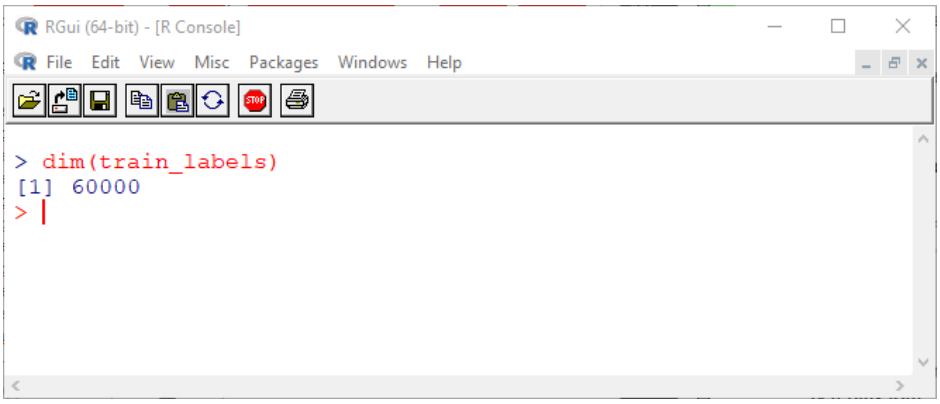
Les images sont chacune des tableaux de 28 x 28, aec des valeurs de pixels comprises entre 0 et 255. Les étiquettes sont des tableaux d'entiers, allant de 0 à 9. Ils correspondent à la classe de vêtements que l'image représente:

Digit	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

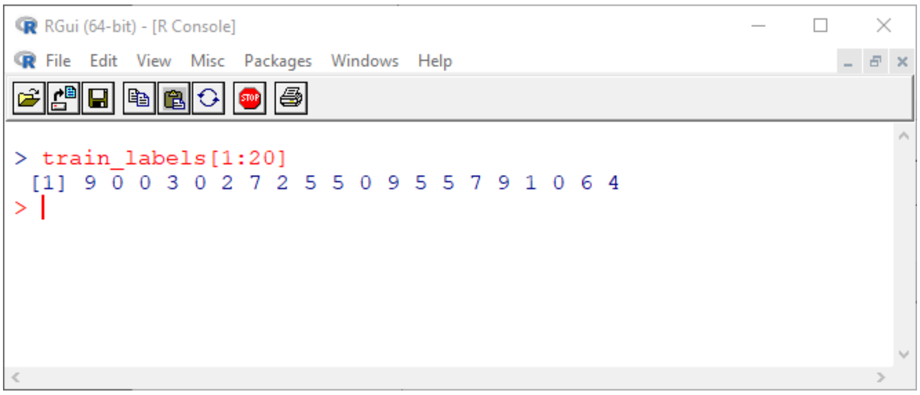
Explorons le format de l'ensemble de données avant d'entraîner le modèle. Ce qui suit montre qu'il y a 60'000 images dans le jeu d'apprentissage, chaque image étant représentée en 28 x 28 pixels:



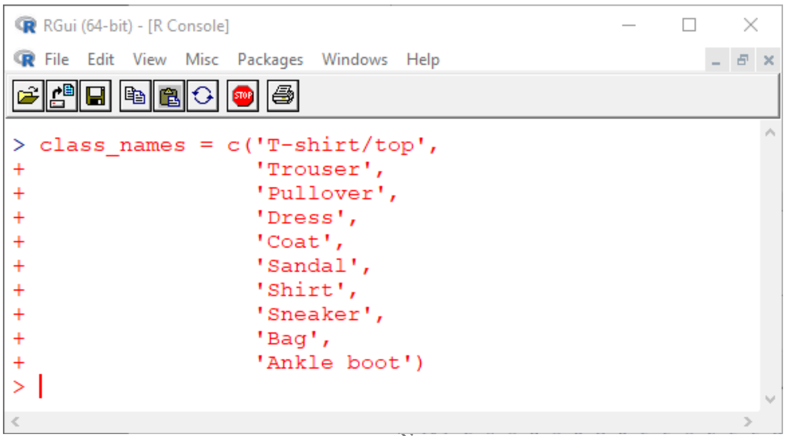
De même, il y a 60'000 étiquettes dans le jeu d'entraînement:



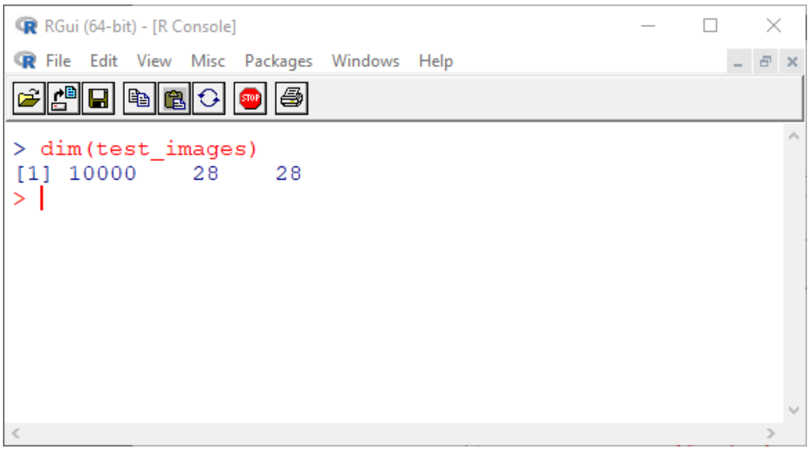
Chaque étiquette est un entier compris entre 0 et 9:



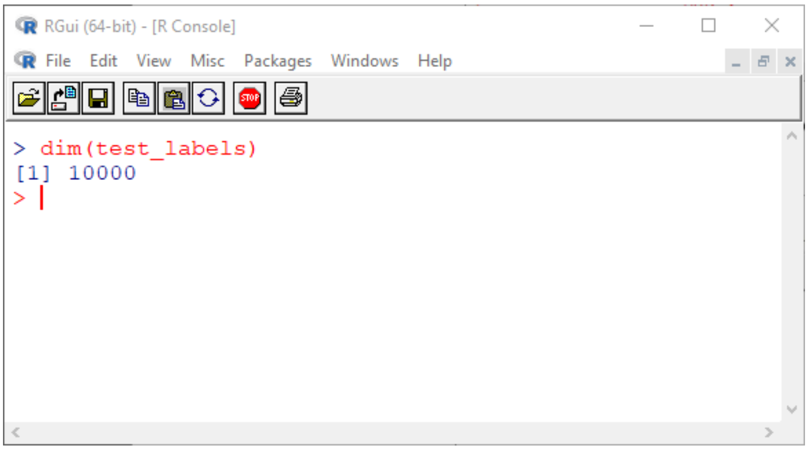
Ensuite on créée un colonne de correspondance des numéros avec les étiquettes pour plus tard:



Il y a 10'000 images dans l'ensemble de test. Encore une fois, chaque image est représentée par 28 x 28 pixels:

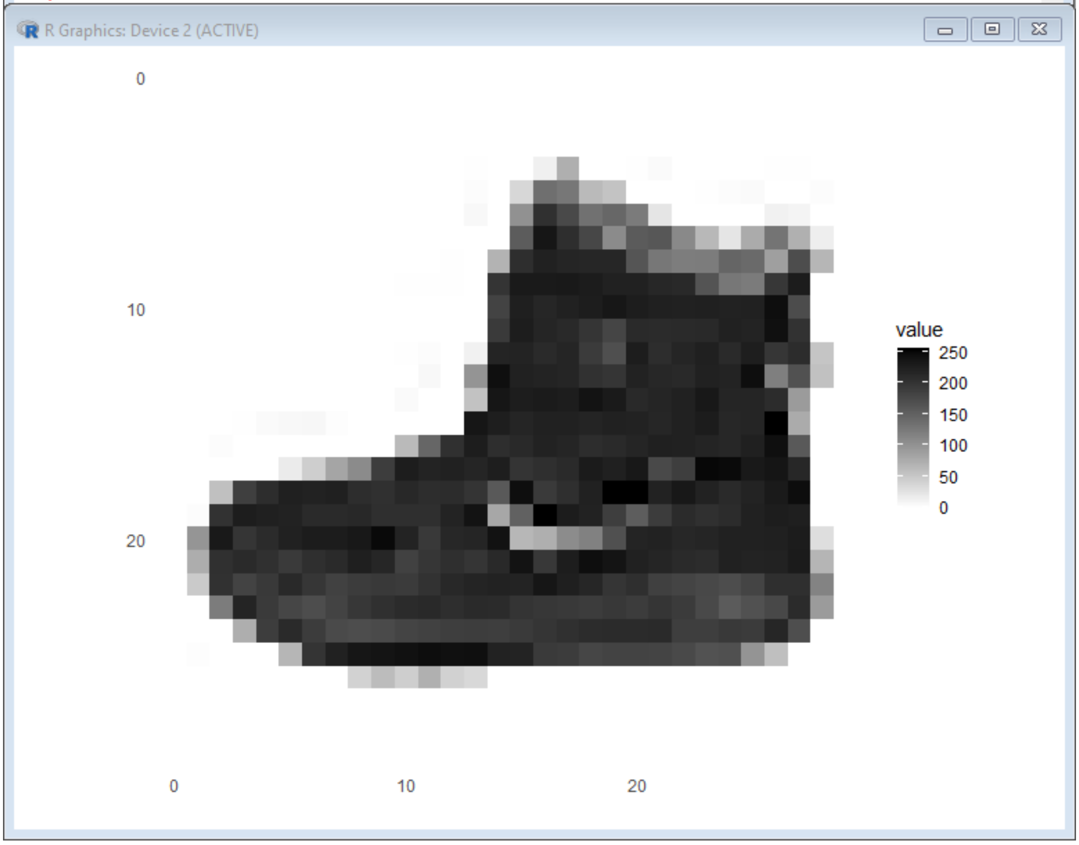


Et l'ensemble de test contient 10'000 étiquettes d'images:



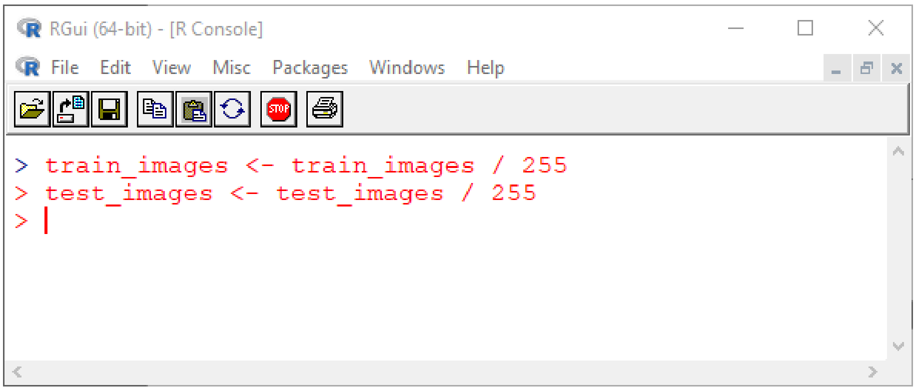
Les données doivent être prétraitées avant d'entraîner le réseau de neurones. Si nous inspectons la première image de l'ensemble d'apprentissage, nous constaterons que les valeurs de pixel sont comprises entre 0 et 255:

```
> image_1 <- as.data.frame(train_images[1, , ])
> colnames(image_1) <- seq_len(ncol(image_1))
> image_1$y <- seq_len(nrow(image_1))
> image_1 <- gather(image_1, "x", "value", -y)
> image_1$x <- as.integer(image_1$x)
>
> ggplot(image_1, aes(x = x, y = y, fill = value)) +
+   geom_tile() +
+   scale_fill_gradient(low = "white", high = "black", na.value = NA) +
+   scale_y_reverse() +
+   theme_minimal() +
+   theme(panel.grid = element_blank())   +
+   theme(aspect.ratio = 1) +
+   xlab("") +
+   ylab("")
>
```



Nous mettons ces valeurs à l'échelle de 0 à 1 avant d'alimenter le modèle de réseau neuronal. Pour cela, nous divisons simplement par 255.

Il est important que l'ensemble d'entraînement et l'ensemble de test soient prétraités de la même manière:



Nous affichons les 25 premières images de l'ensemble d'entraînement et affichons le nom de la classe sous chaque image. Nous vérifiions que les données sont au bon format et nous sommes prêts à créer et à former le réseau:

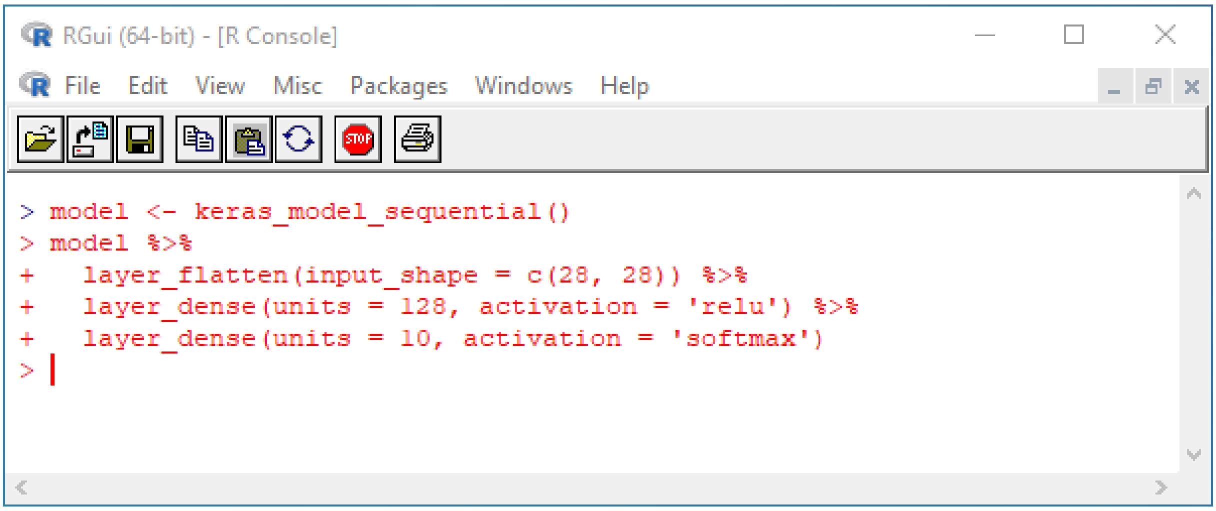


La construction du réseau neuronal nécessite la configuration des couches du modèle, puis la compilation du modèle.

Le bloc de construction de base d'un réseau neuronal est la couche. Les couches extraient des représentations des données qui y sont introduites. Et, espérons-le, ces représentations sont plus significatives pour le problème en question.

La plupart de méthodes d'apprentissage profond consistent à enchaîner des couches simples. La plupart des couches, comme *layer_dense*, ont des paramètres qui sont appris pendant l'entraînement.

Construisons le modèle:



La première couche de ce réseau, *layer_flatten*, transforme le format des images d'un array 2D (de 28 par 28 pixels), en un array 1d de 28*28 = 784 pixels. Considérez ce calque comme un dépilage de lignes de pixels dans l'image et en les alignant. Cette couche n'a pas de paramètres à apprendre; il ne reformate que les données.

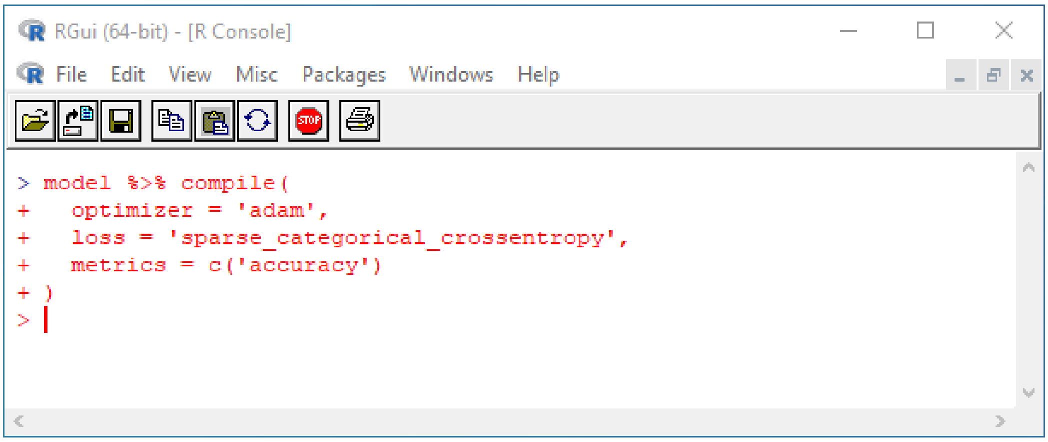
Une fois les pixels aplatis, le réseau se compose d'une séquence de deux couches denses. Ce sont des couches neuronales densément connectées ou entièrement connectées. La première couche dense a 128 nœuds (ou neurones). La deuxième (et dernière) couche est une couche softmax à 10 nœuds. Elle renvoie un tableau de 10 scores de probabilité qui totalisent 1. Chaque nœud contient un score qui indique la probabilité que l'image actuelle appartienne à l'une des 10 classes de chiffres.

Avant que le modèle ne soit prêt pour l'entraînement, il a besoin de quelques paramètres supplémentaires. Ceux-ci sont ajoutées lors de l'étape de compilation du modèle:

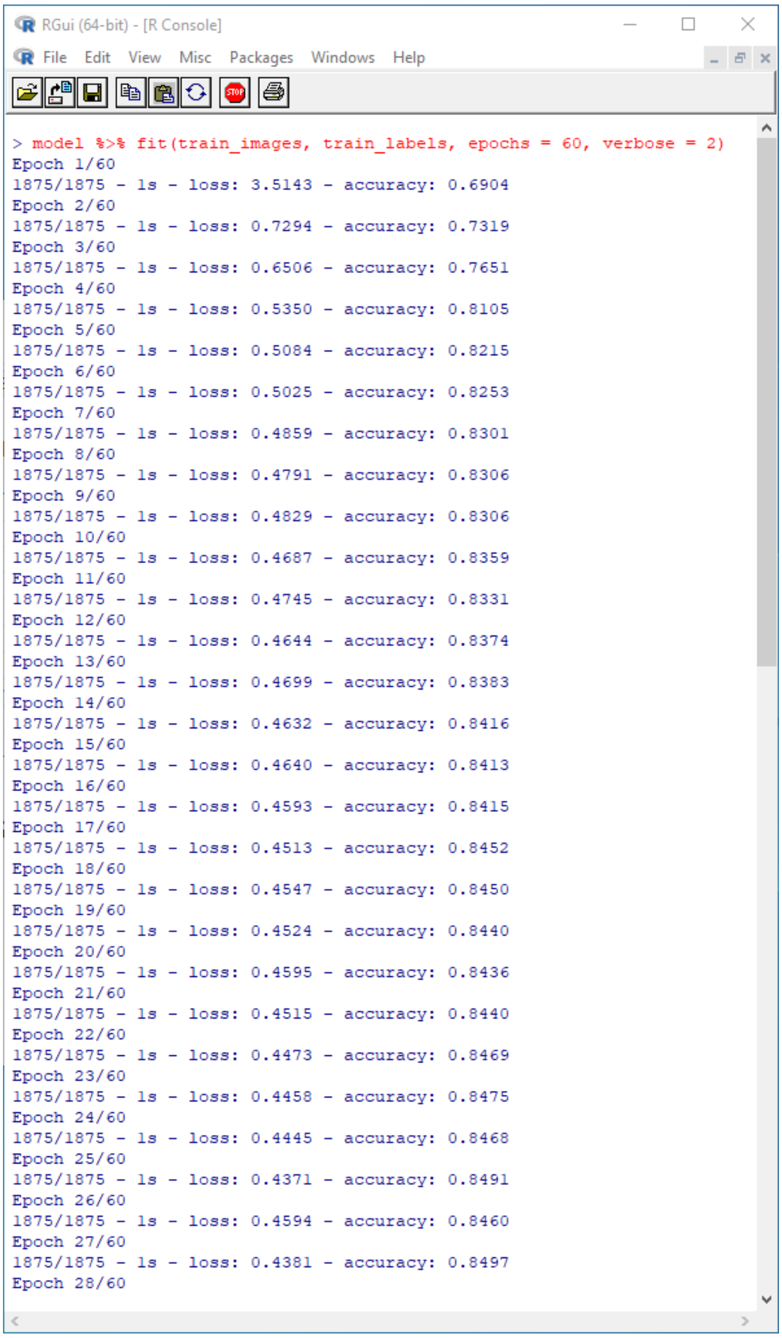
- Fonction de perte (loss function) - Cela mesure la précision du modèle pendant l'entraînement. Nous voulons minimiser cette fonction pour «orienter» le modèle dans la bonne direction.

- Optimiseur (optimizer) - C'est ainsi que le modèle est mis à jour en fonction des données qu'il voit et de sa fonction de perte.

- Mesures (metrics) - Utilisées pour surveiller les étapes de formation et de test. L'exemple suivant utilise la précision, la fraction des images correctement classées.

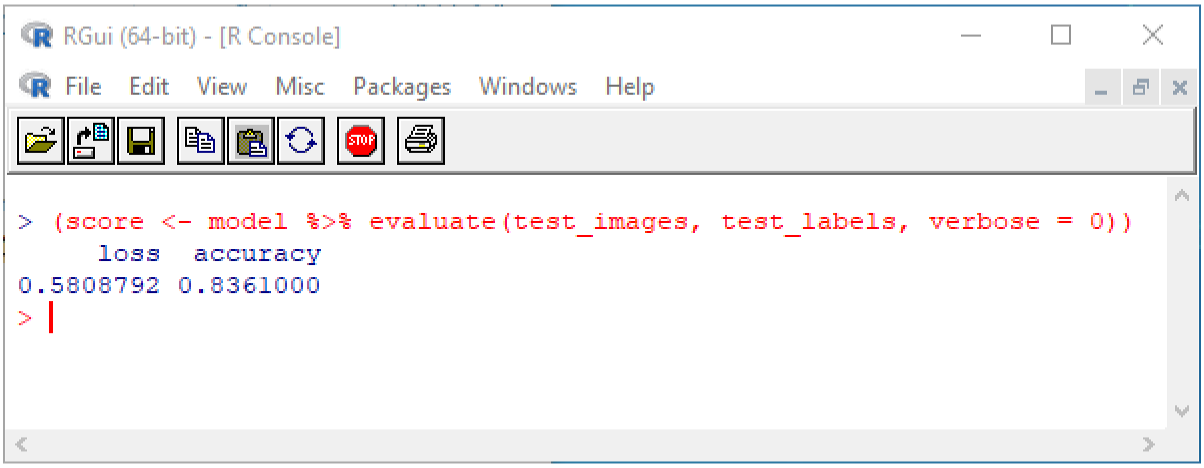


Nous lançons l'entraînement du modèle:



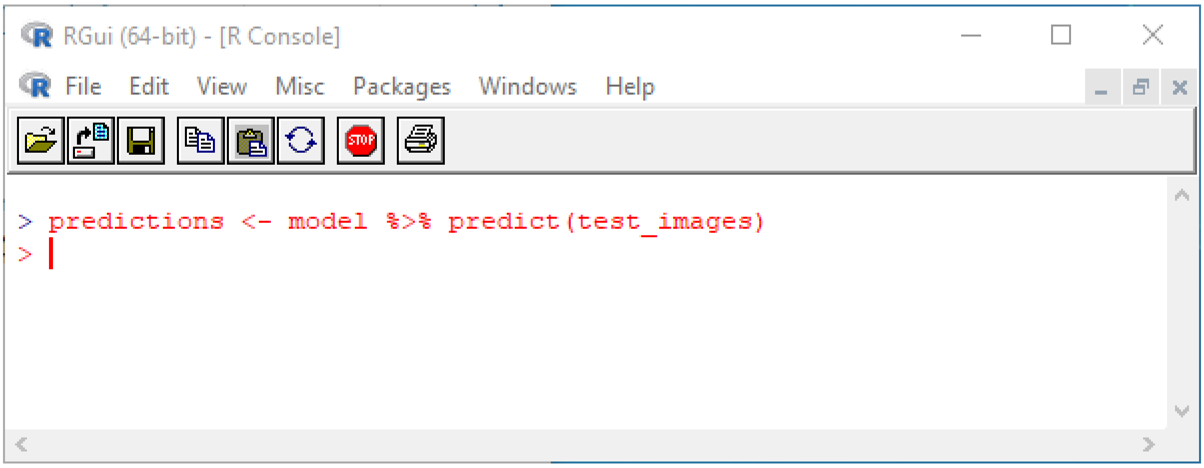
Au fur et à mesure que le modèle s'entraîne, les mesures de perte et de précision sont affichées. Ce modèle atteint une précision d'environ 0.88 (ou 88%) sur les données d'entraînement.

Ensuite, comparons les performances du modèle sur l'ensemble de données de test:

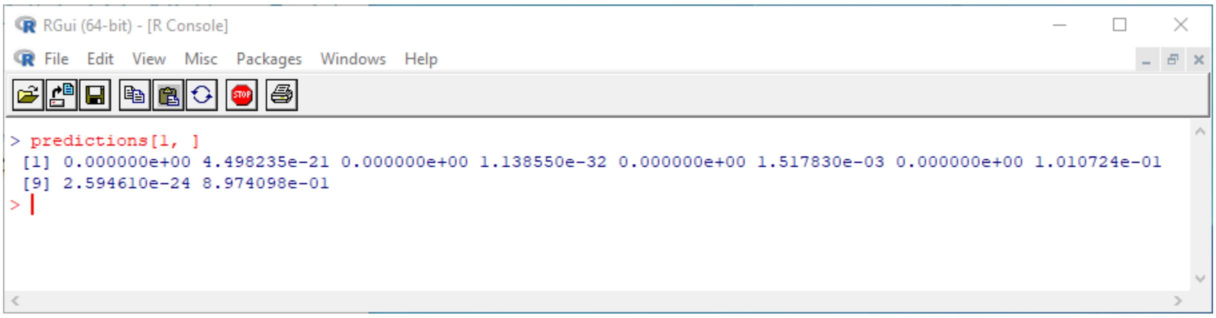


Il s'avère que la précision sur l'ensemble de données de test est un peu inférieure à la précision sur l'ensemble de données d'entraînement. Cet écart entre la précision de l'entraînement et la précision du test est un exemple de surajustement. Le surajustement se produit lorsqu'un modèle d'apprentissage automatique fonctionne moins bien sur de nouvelles données que sur leurs données d'entraînement.

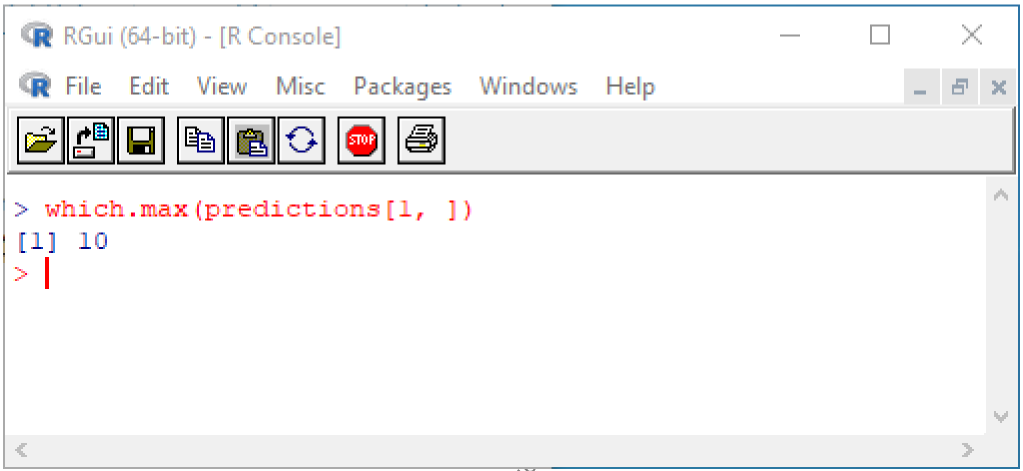
Avec le modèle entraîné, nous pouvons l'utiliser pour faire des prédictions sur certaines images:



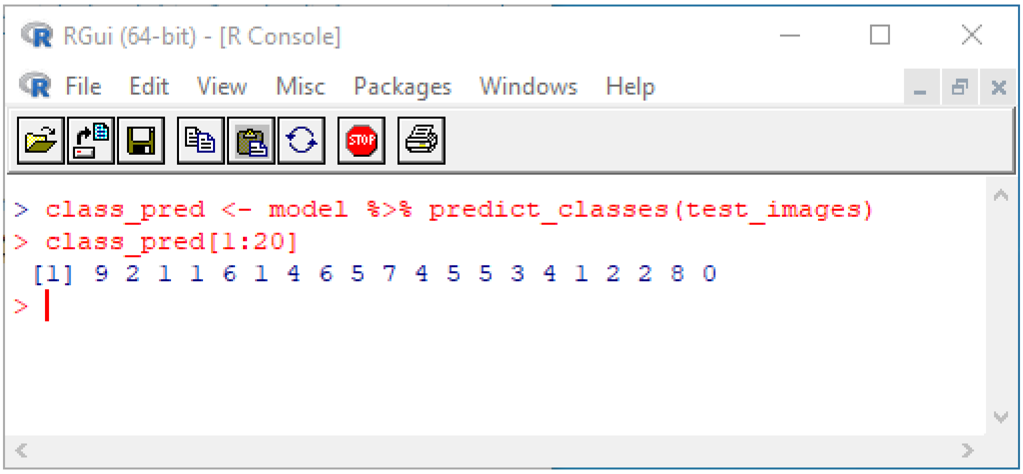
Ici, le modèle a prédit l'étiquette pour chaque image de l'ensemble de test. Jetons un coup d'œil à la première prédiction:



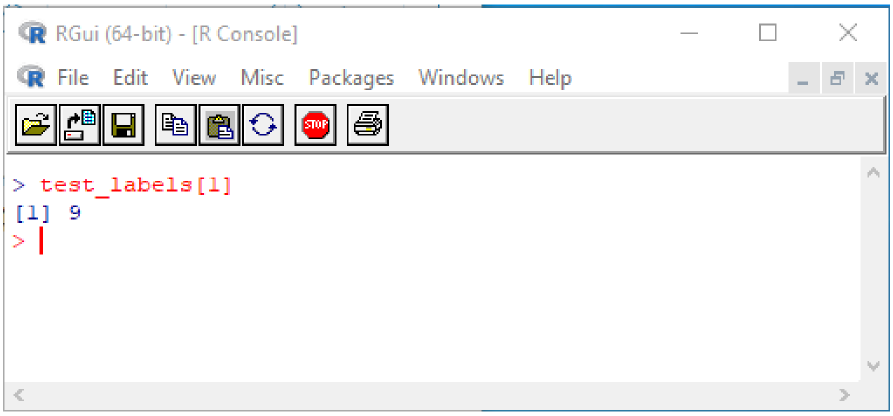
Une prédiction est un tableau de 10 nombres. Celles-ci décrivent la «confiance» du modèle que l'image correspond à chacun des 10 vêtements différents. Nous pouvons voir quelle étiquette a la valeur de confiance la plus élevée:



Alternativement, nous pouvons également obtenir directement la prédiction de classe:



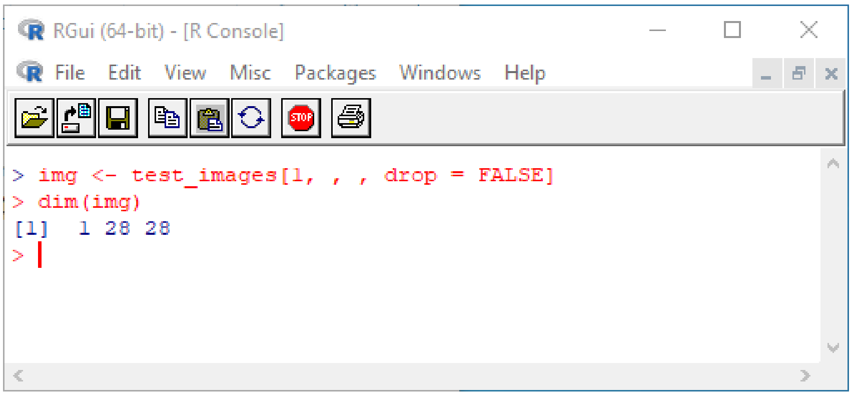
Comme les étiquettes sont labelisées à partir du nombre 0, cela signifie en fait une étiquette prédite de 9 (à trouver dans class_names [9]). Le modèle est donc le plus convaincu que la première image est une bottine. Et nous pouvons vérifier l'étiquette de test pour voir que c'est correct:



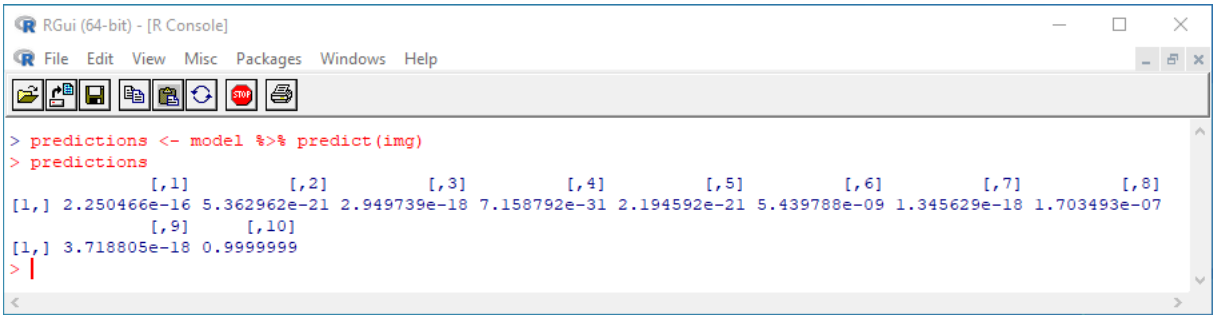
Tracons plusieurs images avec leurs prédictions. Les étiquettes de prédiction correctes sont vertes et les étiquettes de prédiction incorrectes sont rouges:



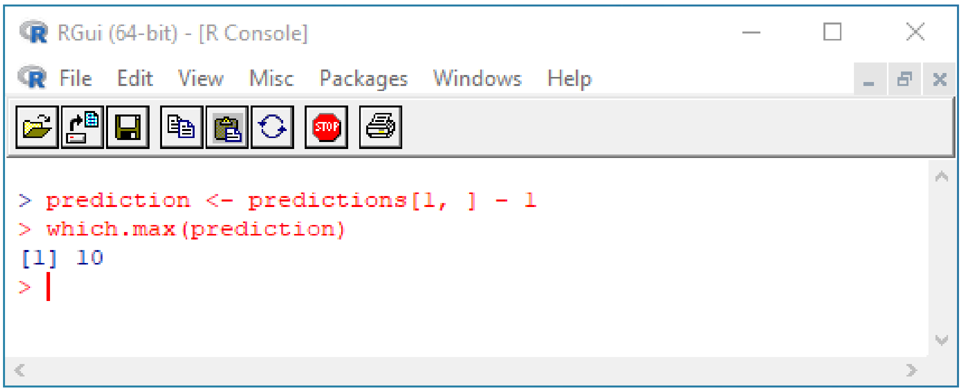
Enfin, utilisons le modèle entraîné pour faire une prédiction sur une seule image.



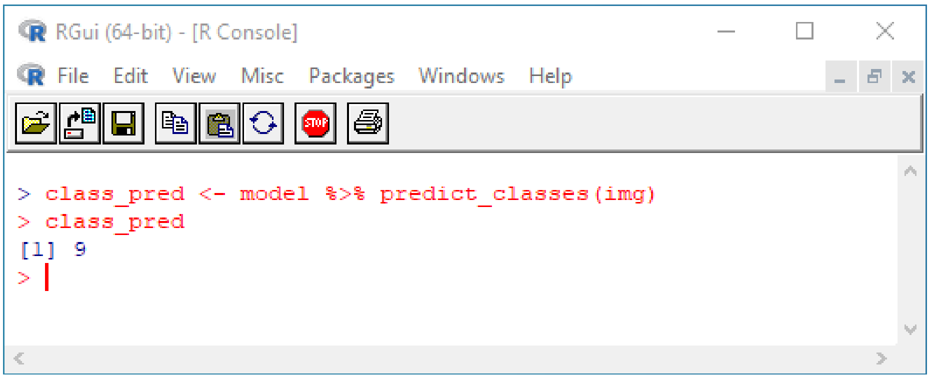
Maintenant, prédisons l'image:



La fonction **predict( )** renvoie une liste de listes, une pour chaque image du lot de données. Prenons les prédictions pour notre (seule) image dans le lot:



Ou, obtenir à nouveau directement la prédiction de classe:



Voici le script complet au besoin:

```
library("keras")
library("tensorflow")
library("dplyr")
library("tidyr")
library("ggplot2")

fashion_mnist <- dataset_fashion_mnist()

#on sépare les deux listes du jeu de de données
c(train_images, train_labels) %<-% fashion_mnist$train
c(test_images, test_labels) %<-% fashion_mnist$test


class_names = c('T-shirt/top',
                'Trouser',
                'Pullover',
                'Dress',
                'Coat',
                'Sandal',
                'Shirt',
                'Sneaker',
                'Bag',
                'Ankle boot')

dim(train_images)
dim(train_labels)
train_labels[1:20]
dim(test_images)
dim(test_labels)

image_1 <- as.data.frame(train_images[1, , ])
colnames(image_1) <- seq_len(ncol(image_1))
image_1$y <- seq_len(nrow(image_1))
image_1 <- gather(image_1, "x", "value", -y)
image_1$x <- as.integer(image_1$x)

ggplot(image_1, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black", na.value = NA) +
  scale_y_reverse() +
  theme_minimal() +
  theme(panel.grid = element_blank())   +
  theme(aspect.ratio = 1) +
  xlab("") +
  ylab("")

train_images <- train_images / 255
test_images <- test_images / 255

par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- train_images[i, , ]
  img <- t(apply(img, 2, rev))
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste(class_names[train_labels[i] + 1]))
}

model <- keras_model_sequential()
model %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  optimizer = 'adam',
```

```
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)

model %>% fit(train_images, train_labels, epochs = 60, verbose = 2)

score <- model %>% evaluate(test_images, test_labels, verbose = 0)

cat('Test loss:', score$loss, "\n")

cat('Test accuracy:', score$acc, "\n")

predictions <- model %>% predict(test_images)

predictions[1, ]

which.max(predictions[1, ])

class_pred <- model %>% predict_classes(test_images)
class_pred[1:20]

par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- test_images[i, , ]
  img <- t(apply(img, 2, rev))
  # subtract 1 as labels go from 0 to 9
  predicted_label <- which.max(predictions[i, ]) - 1
  true_label <- test_labels[i]
  if (predicted_label == true_label) {
    color <- '#008800'
  } else {
    color <- '#bb0000'
  }
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste0(class_names[predicted_label + 1], " (",
                      class_names[true_label + 1], ")"),
        col.main = color)
}

# Grab an image from the test dataset
# take care to keep the batch dimension, as this is expected by the model
img <- test_images[1, , , drop = FALSE]
dim(img)

predictions <- model %>% predict(img)
predictions

# subtract 1 as labels are 0-based
prediction <- predictions[1, ] - 1
which.max(prediction)

class_pred <- model %>% predict_classes(img)
class_pred
```

Exemple 366.: Techniques de cross-over (validation)

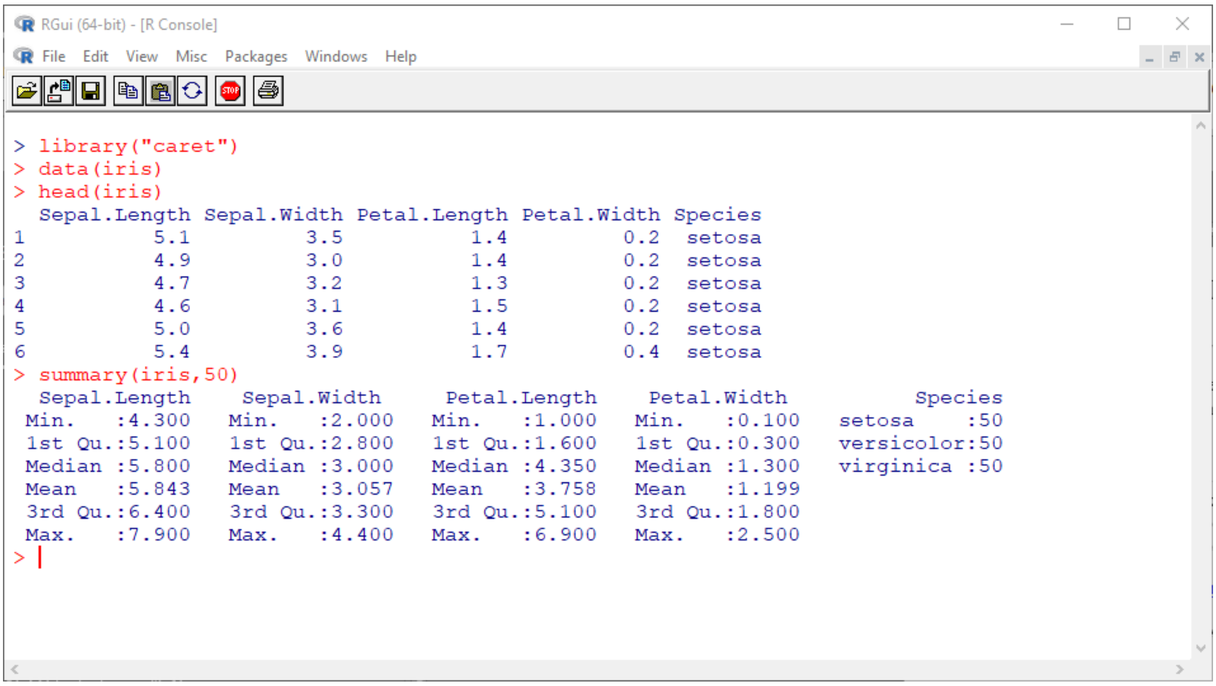
R 3.5.1

Nous allons voir ici quelques techniques de cross-over dont nous avons parlé dans le cours théorique (et pour certaines dont nous avons démontré mathématiquement les détails de méthodes de calculs).

Ici vous ne trouverez rien d'original car Jason Brownlee a écrit bien avant moi un article de blog qui présent de façon suffisamment complète (mais avec le jeu de données bateau *Iris*…) le sujet pour que je n'aie pas besoin de recréer la roue. Voici au besoin le lien vers l'article d'origine datant de septembre 2015:

https://machinelearningmastery.com/how-to-estimate-model-accuracy-in-r-using-the-caret-package/

Nous chargeons le package **caret** et le jeu de données Iris sur lequel sont basés tous les exemples:



Méthode de partition (data split)

La méthode de partition de données implique le fractionnement des données en un jeu de données d'apprentissage explicite utilisé pour préparer le modèle et un jeu de données de test non utilisé pour évaluer les performances du modèle sur des données considérées comme "invisibles".

Cela est utile lorsque vous avez un très grand ensemble de données afin que le jeu de données test puisse fournir une estimation significative des performances du modèle entraîné.

L'exemple ci-dessous divise l'ensemble de données *Iris* de sorte que 80% soit utilisé pour l'apprentissage d'un modèle bayésien naïf et 20% pour évaluer les performances du modèle en utilisant les fonction **createDataPartition( )** et **NaiveBayes( )** en particulier:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> library("klaR")
> #On définit un jeu de 80%/20% d'entraînement/test du jeu Iris
> split<-0.80
> trainIndex<-createDataPartition(iris$Species, p=split, list=FALSE)
> data_train<-iris[ trainIndex,]
> data_test<-iris[-trainIndex,]
> #On entraine un modèle naïf bayésien
> modèle<-NaiveBayes(Species~., data=data_train)
> #On fait des prédictions
> x_test<-data_test[,1:4]
> y_test<-data_test[,5]
> predictions <- predict(modèle, x_test)
> #Résumé de la performance du modèle sur la base d'une matrice de confusion
> confusionMatrix(predictions$class, y_test)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 2.963e-13

                  Kappa : 0.95
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.9000
Specificity                 1.0000            0.9500           1.0000
Pos Pred Value              1.0000            0.9091           1.0000
Neg Pred Value              1.0000            1.0000           0.9524
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3000
Detection Prevalence        0.3333            0.3667           0.3000
Balanced Accuracy           1.0000            0.9750           0.9500
> 
```

Bootstrapping

Le rééchantillonnage bootstrap comme nous le savons déjà (voir page 1969) consiste à prendre des échantillons aléatoires de l'ensemble de données (avec une re-sélection) par rapport à laquelle évaluer le modèle. Dans l'ensemble, les résultats fournissent une indication de la variance de la performance des modèles. En règle générale, un grand nombre d'itérations de rééchantillonnage sont effectuées (des milliers ou des milliers).

L'exemple suivant utilise un bootstrap avec 10 rééchantillonnages pour préparer un modèle bayésien naïf en utilisant principalement les fonctions **trainControl( )** et **train( )**:

```
> train_control<-trainControl(method="boot", number=100)
> #modèle d'entraînement
> modèle<-train(Species~., data=iris, trControl=train_control, method="nb")
Warning message:
In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 21
> predictions<-predict(modèle, x_test)
> #résumé du modèle
> modèle
Naive Bayes

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9496490  0.9236300
   TRUE      0.9521745  0.9274732

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was
 held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
> 
```


Et la performance du modèle toujours avec la matrice de confusion donne:

```
> confusionMatrix(predictions, y_test)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 2.963e-13

                  Kappa : 0.95
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.9000
Specificity                 1.0000            0.9500           1.0000
Pos Pred Value              1.0000            0.9091           1.0000
Neg Pred Value              1.0000            1.0000           0.9524
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3000
Detection Prevalence        0.3333            0.3667           0.3000
Balanced Accuracy           1.0000            0.9750           0.9500
> 
```


K-enveloppes (*K folds*)

Le processus de fractionnement peut être répété plusieurs fois sur le jeu des données d'entrainement en *K* fractions. La précision finale du modèle est prise comme la moyenne (ou autre comme la médiane!) du nombre de répétitions.

L'exemple suivant utilise une validation croisée de 10 fois avec 3 répétitions pour estimer l'algorithme de classification bayésien naïf sur le jeu de données *Iris* en utilisant principalement les fonctions **trainControl( )** et **train( )**:

```
> train_control<-trainControl(method="repeatedcv", number=10, repeats=3)
> #entraînement du modèle
> modèle<-train(Species~., data=iris, trControl=train_control, method="nb")
> #résumé du modèle
> modèle
Naive Bayes

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9533333  0.9300000
   TRUE      0.9577778  0.9366667

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was
 held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
>
```


Et la performance du modèle toujours avec la matrice de confusion donne:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> predictions<-predict(modèle, x_test)
> confusionMatrix(predictions, y_test)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 2.963e-13

                  Kappa : 0.95
 Mcnemar's Test P-Value : NA

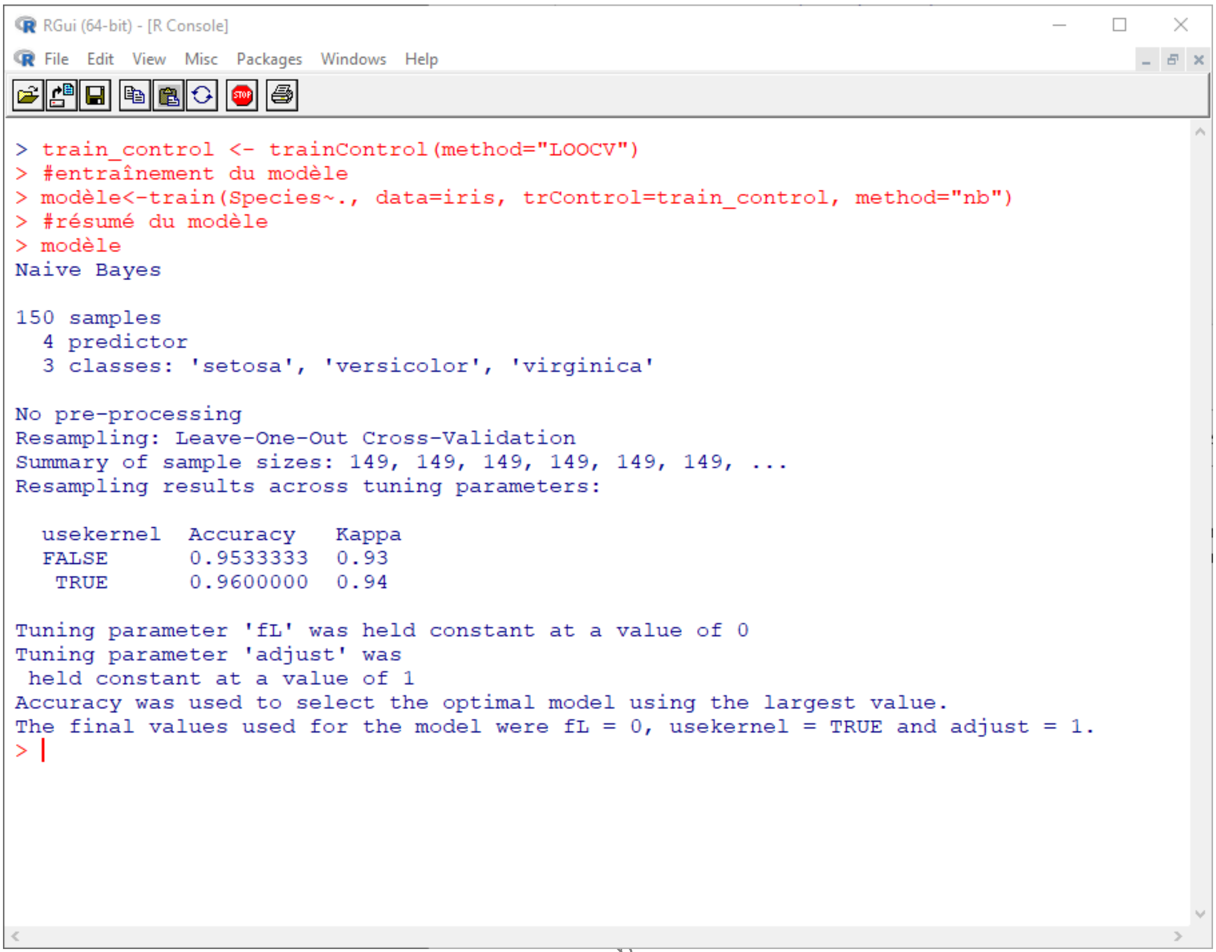
Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.9000
Specificity                 1.0000            0.9500           1.0000
Pos Pred Value              1.0000            0.9091           1.0000
Neg Pred Value              1.0000            1.0000           0.9524
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3000
Detection Prevalence        0.3333            0.3667           0.3000
Balanced Accuracy           1.0000            0.9750           0.9500
> 
```

Leave One Out Cross Validation/LOOCV (jackknife)

Dans la méthode LOOCV (Leave One Out Cross Validation), une unique instance de données est laissée de côté et un modèle est construit sur toutes les autres instances de données du jeu d'apprentissage. Ceci est répété pour toutes les instances de données (c'est du jacknife quoi… comme déjà vu à la page 1978).

L'exemple suivant montre que LOOCV permet d'estimer Naive Bayes sur l'ensemble de données *Iris* en utilisant principalement les fonctions **trainControl( )** et **train( )**:



Et la performance du modèle toujours avec la matrice de confusion donne:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> predictions<-predict(modèle, x_test)
> confusionMatrix(predictions, y_test)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         1
  virginica       0          0         9

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 2.963e-13

                  Kappa : 0.95
 Mcnemar's Test P-Value : NA

Statistics by Class:

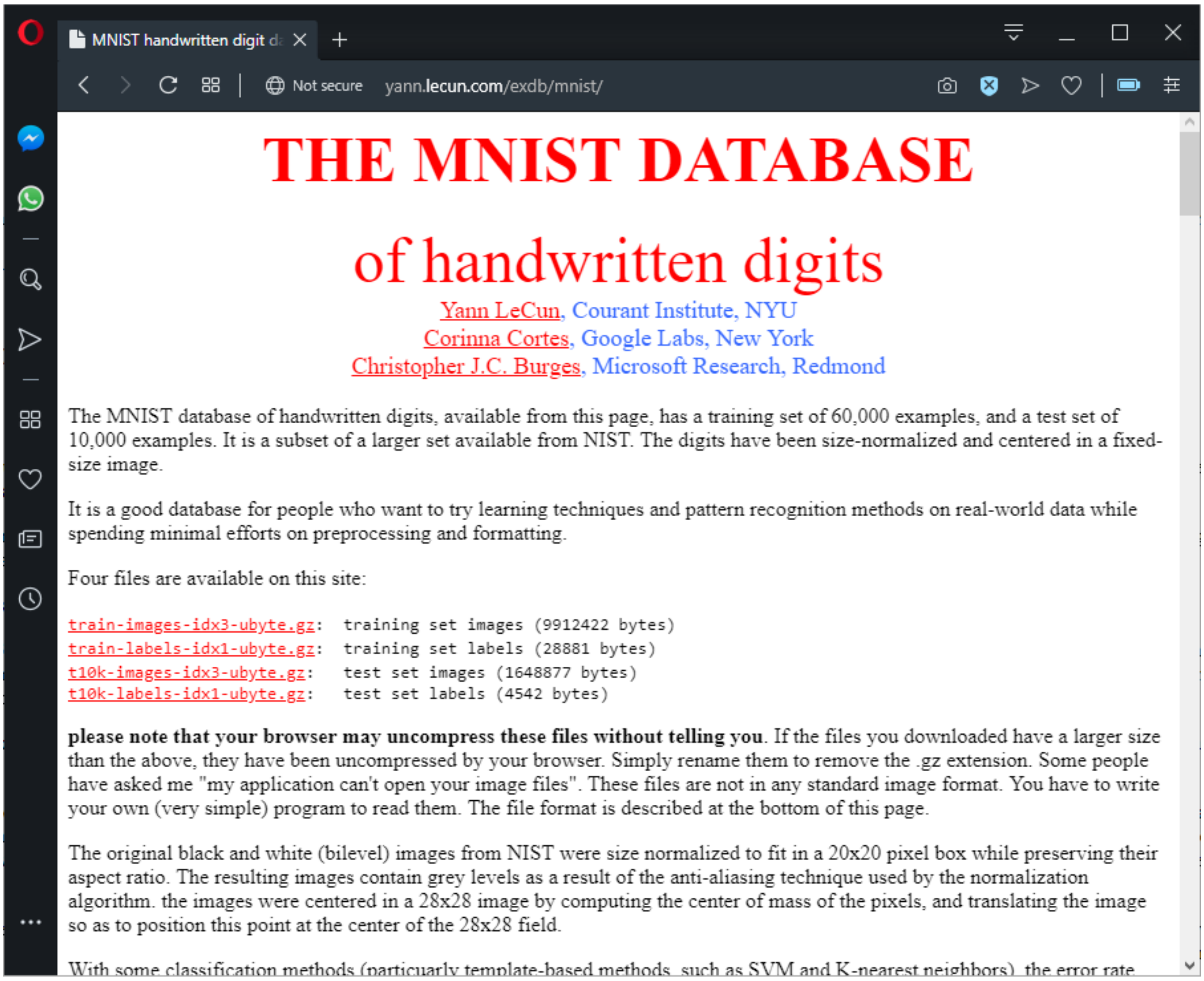
                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.9000
Specificity                 1.0000            0.9500           1.0000
Pos Pred Value              1.0000            0.9091           1.0000
Neg Pred Value              1.0000            1.0000           0.9524
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3000
Detection Prevalence        0.3333            0.3667           0.3000
Balanced Accuracy           1.0000            0.9750           0.9500
>
```

Exemple 367.: Deep Learning

R 3.5.3

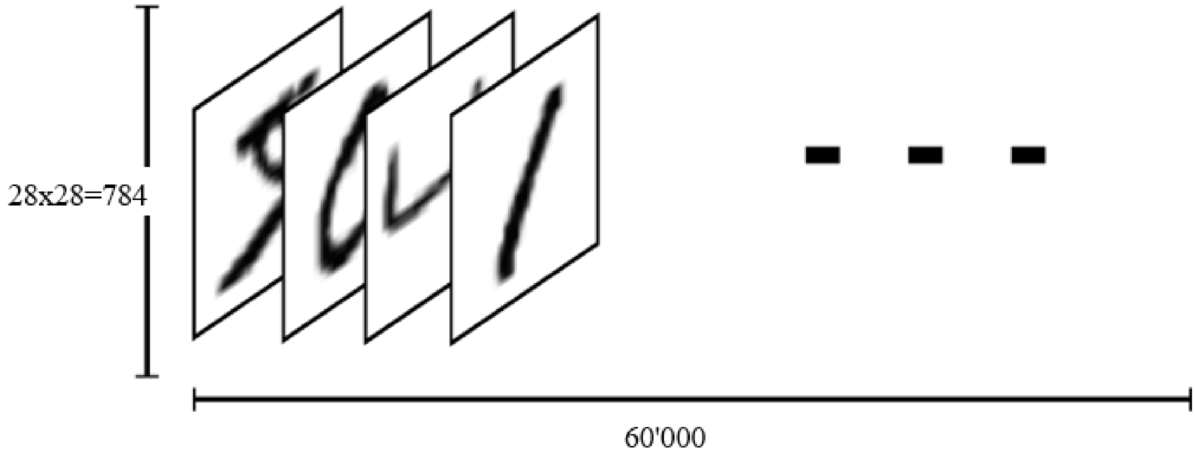
Nous allons faire un exemple de Deep Learning ici sur de l'imagerie et plus particulièrement sur un exemple et jeu de données académique fameux qu'est le MNIST (déjà rencontré plus haut):

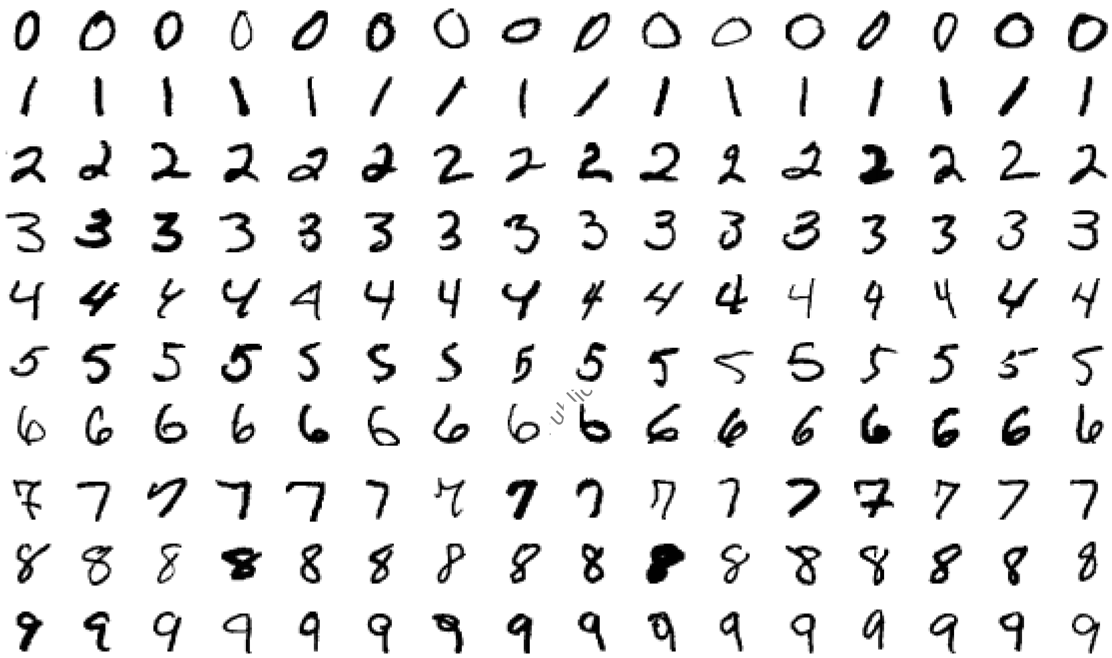
http://yann.lecun.com/exdb/mnist/



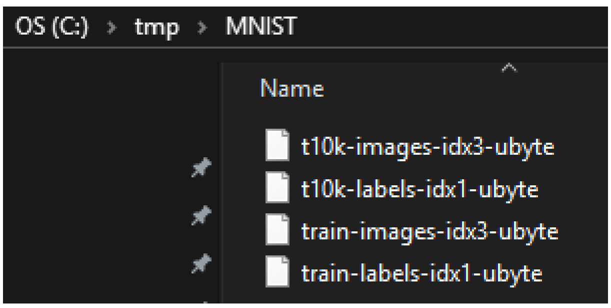
La base de données MNIST pour Mixed National Institute of Standards and Technology, est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenu un test standard. Elle regroupe 60'000 images d'apprentissage et 10'000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté (soit 28 x 28 = 784 points):





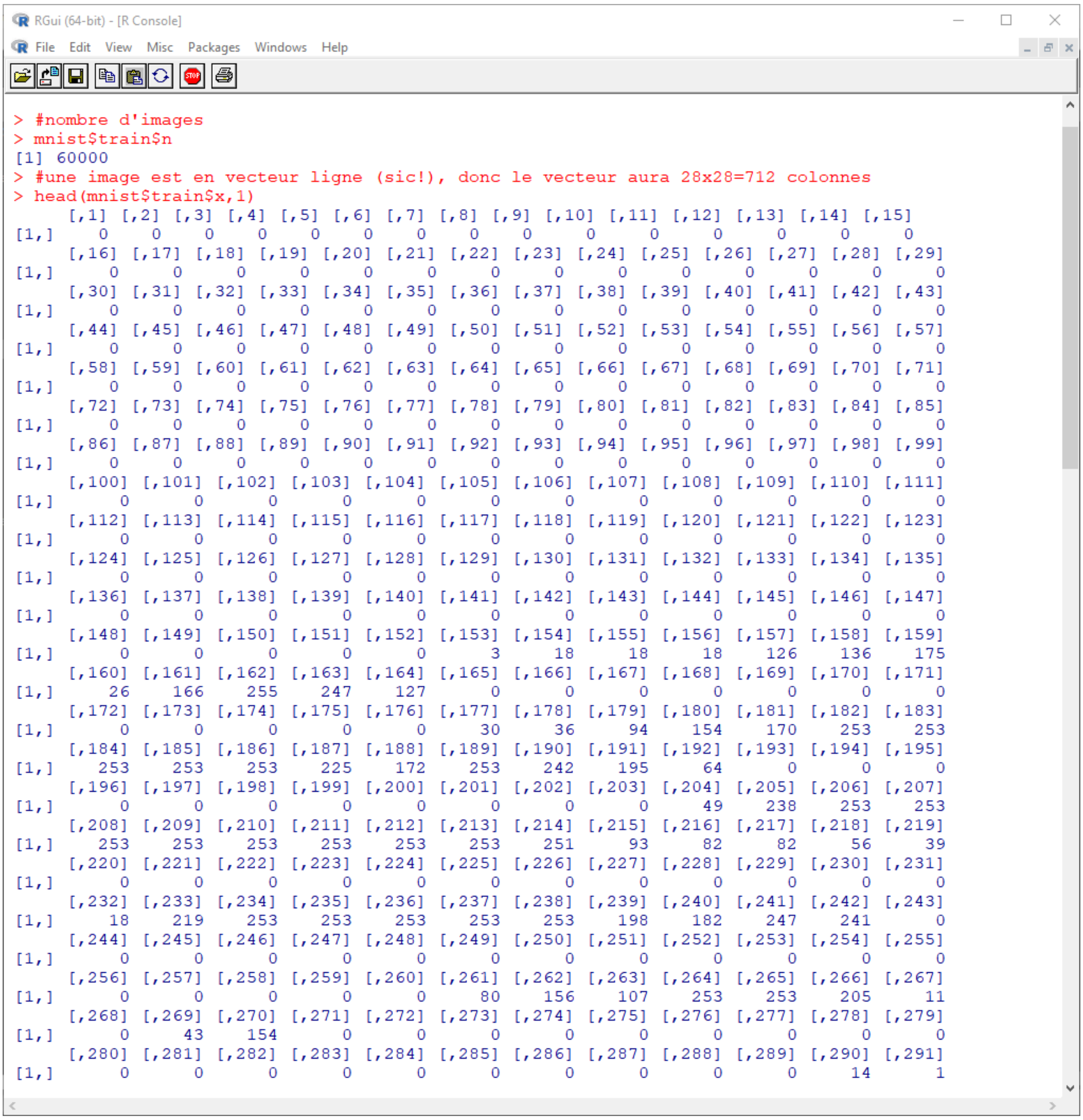
Une fois les fichiers téléchargés, nous vous conseillons de le renommer et classer comme suit:



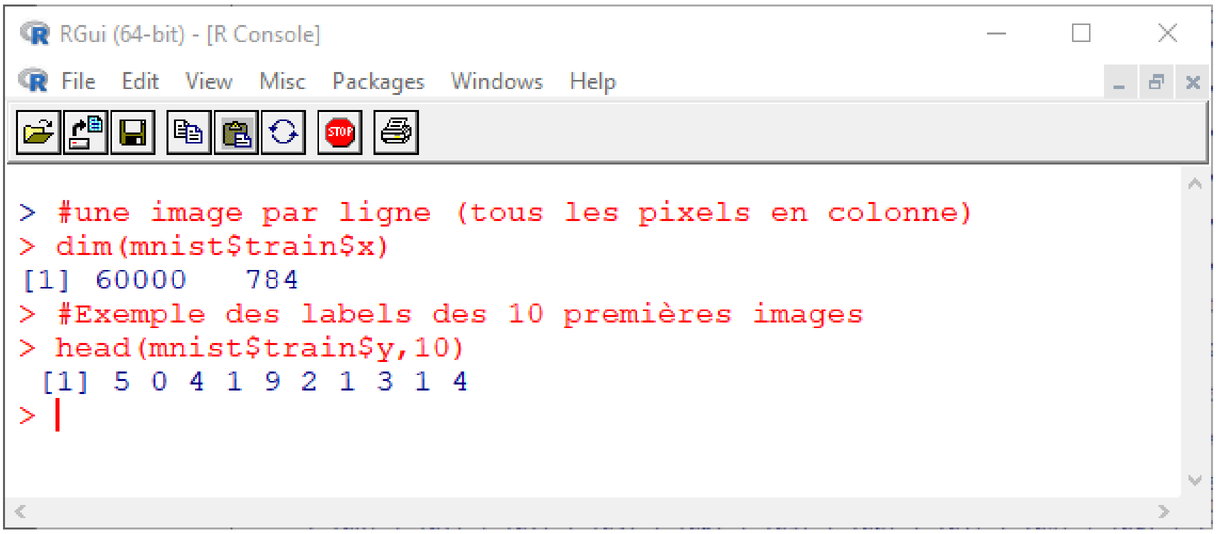
D'abord nous allons charger le package **deepnet** et utiliser sa fonction dédiée au chargement de MNIST:



Découvrons un peu le jeu de données:



Continuons à découvrir:



Maintenant avant de poursuivre, voyons si nous pouvons faire un plot de quelques-uns des digits:



Revenons maintenant au script qui va nous permettre d'entraîner le jeu de données en utilisant la fonction **dbn.dnn.train( )** du package **deepnet**:

```
> #fonction de normalisation des valeurs d'entrées entre 0 et 1
> normalize<-function(x) {
+   return (x/255) #ne pas oublier que les images sont en noir et blanc à l'origine!
+ }
> #on met en marche un timer
> ptm <- proc.time()
> #standardisation
> train_x_n<-apply(mnist$train$x, c(1,2),FUN = normalize)
> test_x_n<-apply(mnist$test$x, c(1,2),FUN = normalize)
> #entraînement
>
> dnn<-dbn.dnn.train(train_x_n, mnist$train$yy,activationfun = "sigm",
+ hidden = c(100, 70, 80), numepochs = 3, cd = 3)
begin to train dbn ......
training layer 1 rbm ...
training layer 2 rbm ...
training layer 3 rbm ...
dbn has been trained.
begin to train deep nn ......
deep nn has been trained.
> #on arrête le timer
> proc.time() - ptm
   user  system elapsed
 314.50    2.36  330.00
>
```

Et maintenant avec les fonctions **nn.test( )** et **nn.predict( )** nous pouvons tester le réseau de neurones profond et son pouvoir prédictif:


```
> #test (taux d'erreur de classification)
> err.dnn<-nn.test(dnn, test_x_n, mnist$test$yy)
> #prédiction
> dnn_predict<-nn.predict(dnn, test_x_n)
> #affichager des résultats
> err.dnn
[1] 0.0742
> #pour une image donnée (la numéro 1 dans le cas présent!), donne la
> #probabilité qu'il appartienne a une des 10 classe {0,1,2,3,4,5,6,7,9} de nombres
> dnn_predict[1,]
 [1] 2.476308e-03 7.162161e-04 5.170752e-04 1.105614e-02 2.265273e-03 9.914841e-03
 [7] 7.228431e-06 9.964496e-01 3.546970e-05 2.035836e-03
> #Affichage la classe correspondante la plus probable!
> mnist$test$y[1]
[1] 7
>
```

Exemple 368.: AutoML (Automatic Machine Learning)

R 3.4.2

Au cours des dernières années, la demande d'experts en apprentissage automatique a dépassé l'offre, malgré l'afflux de nouveaux venus sur le secteur. Pour combler cette lacune, il y a eu de grands progrès dans le développement d'un logiciel d'apprentissage automatique convivial pouvant être utilisé par des non-experts. Les premières étapes vers la simplification de l'apprentissage machine impliquaient le développement d'interfaces simples et unifiées avec divers algorithmes d'apprentissage automatique (par exemple, H2O).

Bien que H2O ait facilité l'expérimentation de l'apprentissage automatique par des non-experts, il reste encore un certain nombre de connaissances en science des données qui sont nécessaires pour produire des modèles d'apprentissage machine performants et comprendre les hypothèses sous-jacentes et les concluions. Les réseaux de neurones profonds en particulier sont notoirement difficiles à accorder pour un non-expert. Afin que les logiciels d'apprentissage automatique soient vraiment accessibles aux non-experts, nous avons conçu une interface facile à utiliser qui automatise le processus de formation d'un grand nombre de modèles candidats. AutoML peut aussi être un outil utile pour l'utilisateur avancé, en fournissant une fonction wrapper simple qui effectue un grand nombre de tâches liées à la modélisation qui nécessiteraient généralement de nombreuses lignes de code, et en libérant du temps pour se concentrer sur d'autres aspects des tâches de pipeline de data science telles que le prétraitement de données, l'ingénierie de fonctionnalités et le déploiement de modèle.

Le langage AutoML de H2O peut être utilisé pour automatiser le flux de travail d'apprentissage automatique, qui comprend l'entraînement automatique et le réglage de nombreux modèles dans une limite de temps spécifiée par l'utilisateur. L'utilisateur peut également utiliser un critère d'arrêt basé sur une mesure de performances pour le processus AutoML plutôt qu'une contrainte de temps spécifique. Les ensembles empilés seront automatiquement formés sur les modèles individuels de collecte pour produire un modèle d'ensemble très prédictif qui, dans la plupart des cas, sera le modèle le plus performant du classement AutoML. Les ensembles empilés ne sont pas encore disponibles pour les problèmes de classification multiclasse, dans ce cas, seuls les modèles singleton seront formés.

Attention!!! Au jour où j'écris ces lignes, autoML / h2o ne fait pas a priori de Machine Learning avec des textes et a ses clusters limités à 170 MB environ!!!!

D'abord, après l'avoir installée, on charge la libraire **h2o**:

```
> library(h2o)
>
> h2o.init()
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         3 minutes 45 seconds
    H2O cluster version:        3.14.0.3
    H2O cluster version age:    1 month and 29 days
    H2O cluster name:           H2O_started_from_R_Vincent_Isoz_rxp140
    H2O cluster total nodes:    1
    H2O cluster total memory:   0.81 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  4
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.4.2 (2017-09-28)

>
```

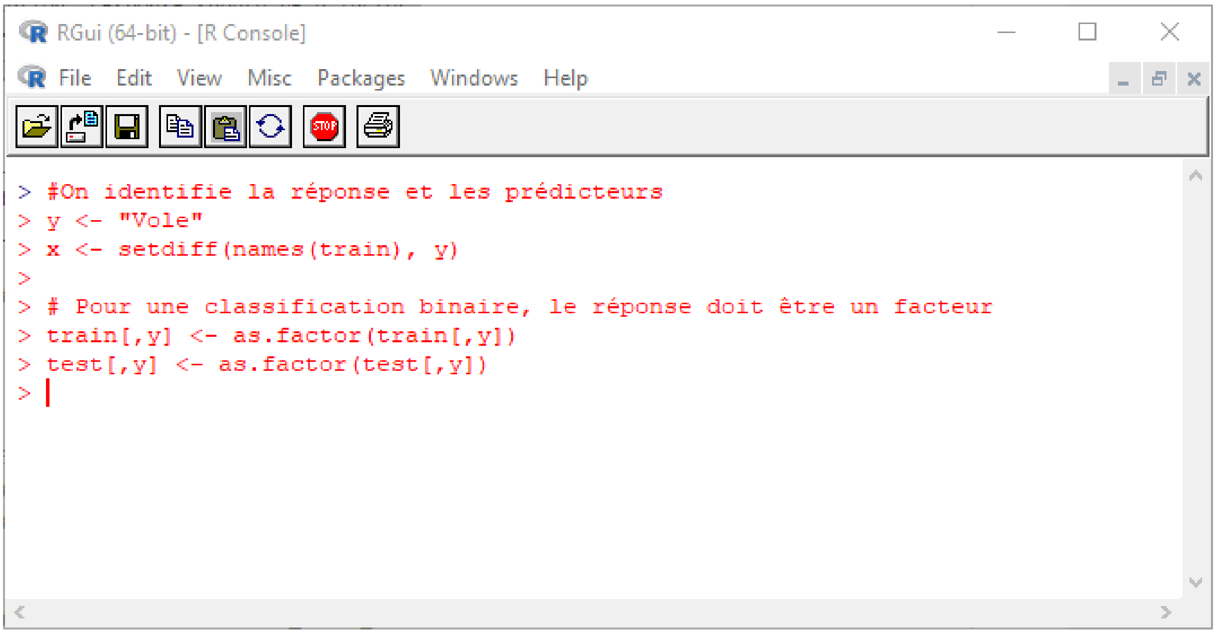
Voyons ici un exemple simple sur un jeu de données que nous connaissons bien (évitez à ce jour les accents ou caractères spéciaux dans les jeux de données!!!) que nous allons importer avec la fonction **h2o.importFile( )** :


```
> train<-h2o.importFile("c:/tmp/AutoML_Train.csv",header=T)
  |==========================================================================================| 100%
> test<-h2o.importFile("c:/tmp/AutoML_Test.csv",header=T)
  |==========================================================================================| 100%
> train
  Couleur   Type      Origine Vole
1   Rouge Sports   Domestique  Oui
2   Rouge Sports   Domestique  Oui
3   Jaune Sports   Domestique  Non
4   Jaune Sports Import<0xE9>  Oui
5   Jaune    SUV Import<0xE9>  Non
6   Jaune    SUV   Domestique  Non

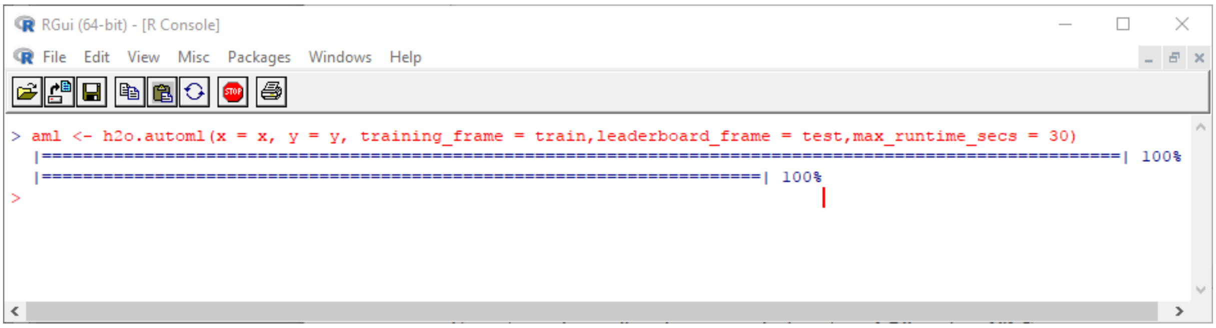
[8 rows x 4 columns]
> test
  Exemple Couleur   Type      Origine Vole
1       2   Rouge Sports   Domestique  Non
2       7   Jaune    SUV Import<0xE9>  Oui
3      11   Rouge    SUV   Domestique  Oui

[3 rows x 5 columns]
>
```

On définit quelques prérequis évidents:



Ensuite on lance l'analyse avec la fonction **h2o.automl( )** :



et on demande d'afficher tous les algorithmes leaders dans la classification:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> lb<-aml#leaderboard
> lb
An object of class "H2OAutoML"
Slot "project_name":
[1] "automl_RTMP_sid_87f8_9"

Slot "leader":
Model Details:
==============

H2OBinomialModel: glm
Model ID:  GLM_grid_0_AutoML_20171204_125456_model_0
GLM Model: summary
    family  link            regularization                                                        lambda_search
1 binomial logit Ridge ( lambda = 0.7669 ) nlambda = 30, lambda.max = 18.367, lambda.min = 0.7669, lambda.1se = 18.367
  number_of_predictors_total number_of_active_predictors number_of_iterations                  training_frame
1                          6                           6                   28 automl_training_RTMP_sid_87f8_9

Coefficients: glm coefficients
                names coefficients standardized_coefficients
1           Intercept    -0.310350                 -0.310350
2       Couleur.Jaune    -0.109929                 -0.109929
3       Couleur.Rouge     0.109856                  0.109856
4            Type.SUV    -0.203214                 -0.203214
5         Type.Sports     0.203149                  0.203149
6   Origine.Domestique     0.032087                  0.032087
7 Origine.Import<0xE9>    -0.032150                 -0.032150

H2OBinomialMetrics: glm
** Reported on training data. **

MSE:  0
RMSE:  0
LogLoss:  1
Mean Per-Class Error:  0
AUC:  1
Gini:  1
R^2:  0
Residual Deviance:  8
AIC:  22

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       Non Oui    Error  Rate
Non      3   1 0.250000  =1/4
Oui      0   3 0.000000  =0/3
Totals   3   4 0.142857  =1/7

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold    value idx
1                       max f1  0.438001 0.857143   2
2                       max f2  0.438001 0.937500   2
```

et on affichage celui qui performe le mieux (s'il y en a plusieurs!) avec tous les détails nécessaires pertinents sur plusieurs pages:

```
> aml@leader
Model Details:
==============

H2OBinomialModel: glm
Model ID:  GLM_grid_0_AutoML_20171204_125456_model_0
GLM Model: summary
    family  link                regularization                                                            lambda_search
1 binomial logit Ridge ( lambda = 0.7669 )  nlambda = 30, lambda.max = 13.367, lambda.min = 0.7669, lambda.1se = 13.367
  number_of_predictors_total number_of_active_predictors number_of_iterations                  training_frame
1                          6                           6                   20 automl_training_RTMP_sid_U7f0_9

Coefficients: glm coefficients
                   names coefficients standardized_coefficients
1             Intercept    -0.310350                 -0.310350
2         Couleur.Jaune    -0.109929                 -0.109929
3         Couleur.Rouge     0.109856                  0.109856
4              Type.SUV    -0.203214                 -0.203214
5           Type.Sports     0.203149                  0.203149
6   Origine.Domestique     0.032087                  0.032087
7 Origine.Import<0xE9>      0.032150                  0.032150

H2OBinomialMetrics: glm
** Reported on training data. **

MSE:  0
RMSE:  0
LogLoss:  1
Mean Per-Class Error:  0
AUC:  1
Gini:  1
R^2:  0
Residual Deviance:  3
AIC:  22

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       Non Oui    Error  Rate
Non      3   1 0.250000  =1/4
Oui      0   3 0.000000  =0/3
Totals   3   4 0.142857  =1/7

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold    value idx
1                       max f1  0.438001 0.857143   2
2                       max f2  0.438001 0.937500   2
3                 max f0point5  0.508685 0.909091   0
4                 max accuracy  0.508685 0.857143   0
5                max precision  0.508685 1.000000   0
6                   max recall  0.438001 1.000000   2
7              max specificity  0.508685 1.000000   0
```

```
7              max specificity  0.508685 1.000000    0
8            max absolute_mcc  0.430001 0.750000    2
9   max min_per_class_accuracy  0.430001 0.750000    2
10 max mean_per_class_accuracy  0.438001 0.875000    2

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
H2OBinomialMetrics: glm
** Reported on validation data. **

MSE:  0
RMSE:  1
LogLoss:  1
Mean Per-Class Error:  0
AUC:  1
Gini:  1
R^2:  -Infinity
Residual Deviance:  1
AIC:  15

Confusion Matrix (vertical: actual; across: predicted) for F1 optimal threshold:
       Non Oui    Error  Rate
Non      0   0           =0/0
Oui      0   1 0.000000  =0/1
Totals   0   1 0.000000   0/1

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold    value idx
1                       max f1  0.492627 1.000000   0
2                       max f2  0.492627 1.000000   0
3                 max f0point5  0.492627 1.000000   0
4                 max accuracy  0.492627 1.000000   0
5                max precision  0.492627 1.000000   0
6                   max recall  0.492627 1.000000   0
7              max specificity                     -1
8             max absolute_mcc  0.492627 0.000000   0
9   max min_per_class_accuracy                      1
10 max mean_per_class_accuracy                      1

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
H2OBinomialMetrics: glm
** Reported on cross-validation data. **
** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **

MSE:  0
RMSE:  1
LogLoss:  1
Mean Per-Class Error:  0
AUC:  0
Gini:  0
R^2:  -0
```

```
Residual Deviance:  10
AIC:  24

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       Non Oui    Error  Rate
Non      3   1 0.250000  =1/4
Oui      1   2 0.333333  =1/3
Totals   4   3 0.285714  =2/7

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold    value idx
1                       max f1  0.443016 0.666667   2
2                       max f2  0.299219 0.789474   6
3                 max f0point5  0.443016 0.666667   2
4                 max accuracy  0.443016 0.714286   2
5                max precision  0.443016 0.666667   2
6                   max recall  0.299219 1.000000   6
7              max specificity  0.559764 0.750000   0
8             max absolute_mcc  0.361909 0.471405   5
9   max min_per_class_accuracy  0.443016 0.666667   2
10 max mean_per_class_accuracy  0.443016 0.708333   2

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
Cross-Validation Metrics Summary:
                mean         sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid   cv_5_valid
accuracy         0.6 0.34641016        1.0        1.0        0.0        1.0          0.0
auc              1.0        0.0        1.0        1.0        NaN        1.0          NaN
err       0.13988757 0.19766061        0.0        0.0  0.6989561        0.0 4.8181415E-4
err_count        0.0        0.0        0.0        0.0        NaN        0.0          NaN
f0point5  0.65058094 0.30784458        1.0        1.0 0.25290442        1.0 2.3203305E-7

---
                        mean         sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid cv_5_valid
precision                0.6 0.34641016        1.0        1.0                   1.0
r2                       NaN   Infinity  0.9844948 0.26152125             -Infinity
recall                   0.6 0.34641016        1.0        1.0                   1.0
residual_deviance 0.92410195 0.74860936  0.1843583  2.2297723             2.2063792
rmse              0.23202455  0.1912578 0.06225991 0.42967394             0.6681889
specificity              0.5  0.4082483        1.0        1.0                   NaN
> 
```

Finalement on peut tester le modèle sur le jeu de test ou l'appliquer à de nouvelles données avec la fonction **h2o.predict( )** :


```
> pred<-h2o.predict(aml@leader,test)
  |==========================================================================================| 100%
> pred
  predict Non Oui
1     Oui   0   1
2     Non   1   0
3     Non   1   0

[3 rows x 3 columns]
> 
```

Forcer le deep learning

La syntaxe est fort simple, il suffit d'utiliser la fonction **h2o.deeplearning( )** :

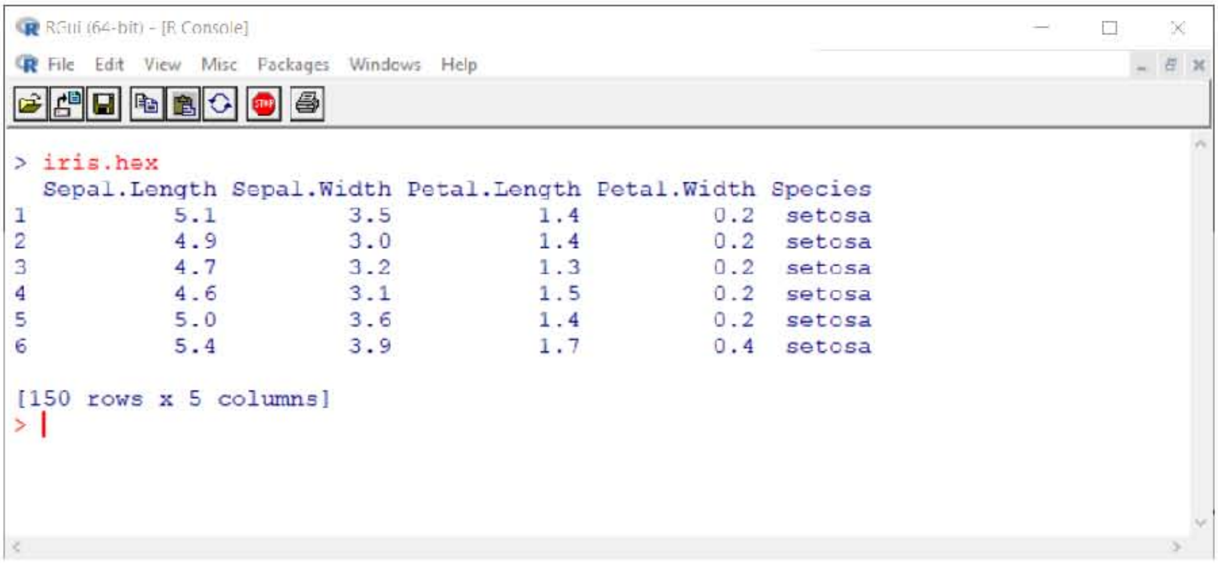
```
> library(h2o)
> h2o.init()
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         10 minutes 5 seconds
    H2O cluster timezone:       Europe/Berlin
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.18.0.8
    H2O cluster version age:    18 days
    H2O cluster name:           H2O_started_from_R_Vincent_Isoz_plf531
    H2O cluster total nodes:    1
    H2O cluster total memory:   3.36 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  4
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.5.0 (2018-04-23)

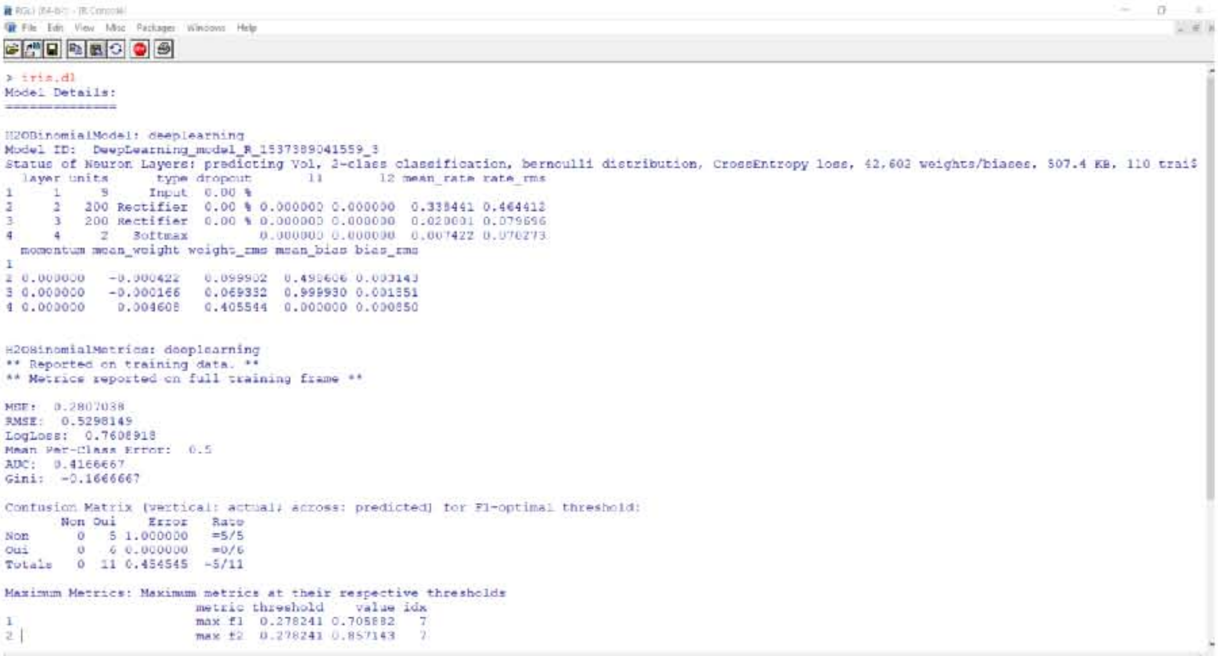
> iris.hex <- as.h2o(iris)
  |====================================| 100%
> iris.dl <- h2o.deeplearning(x = 1:4, y = 5, training_frame = iris.hex)
  |====================================| 100%
>
> h2o.predict(iris.dl, iris.hex)
  |====================================| 100%
  predict    setosa   versicolor    virginica
1  setosa 0.9999905 9.521693e-06 5.218946e-30
2  setosa 0.9998652 1.347919e-04 3.061051e-28
3  setosa 0.9999926 7.431171e-06 1.295347e-29
4  setosa 0.9999836 1.644581e-05 1.120327e-28
5  setosa 0.9999971 2.943896e-06 1.920185e-30
6  setosa 0.9999859 1.409840e-05 1.961487e-28

[150 rows x 4 columns]
```

Au besoin voici le contenu de *iris.hex* si cela peut aider:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> iris.hex
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
[150 rows x 5 columns]
>

et *iris.dl*:


> iris.dl
Model Details:
==============
H2OBinomialModel: deeplearning
Model ID: DeepLearning_model_R_1537389041559_3
Status of Neuron Layers: predicting Vol, 2-class classification, bernoulli distribution, CrossEntropy loss, 42,602 weights/biases, 507.4 KB, 110 trai$
layer units type dropout l1 l2 mean_rate rate_rms
1 1 9 Input 0.00 %
2 2 200 Rectifier 0.00 % 0.000000 0.000000 0.338441 0.464412
3 3 200 Rectifier 0.00 % 0.000000 0.000000 0.020001 0.079696
4 4 2 Softmax 0.000000 0.000000 0.007422 0.070273
momentum mean_weight weight_rms mean_bias bias_rms
1
2 0.000000 -0.000422 0.099902 0.499606 0.003143
3 0.000000 -0.000166 0.069332 0.999930 0.001851
4 0.000000 0.004608 0.405544 0.000000 0.000850
H2OBinomialMetrics: deeplearning
** Reported on training data. **
** Metrics reported on full training frame **
MSE: 0.2807038
RMSE: 0.5298149
LogLoss: 0.7608918
Mean Per-Class Error: 0.5
AUC: 0.4166667
Gini: -0.1666667
Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
Non Oui Error Rate
Non 0 5 1.000000 =5/5
Oui 0 6 0.000000 =0/6
Totals 0 11 0.454545 =5/11
Maximum Metrics: Maximum metrics at their respective thresholds
metric threshold value idx
1 max f1 0.278241 0.705882 7
2 max f2 0.278241 0.857143 7

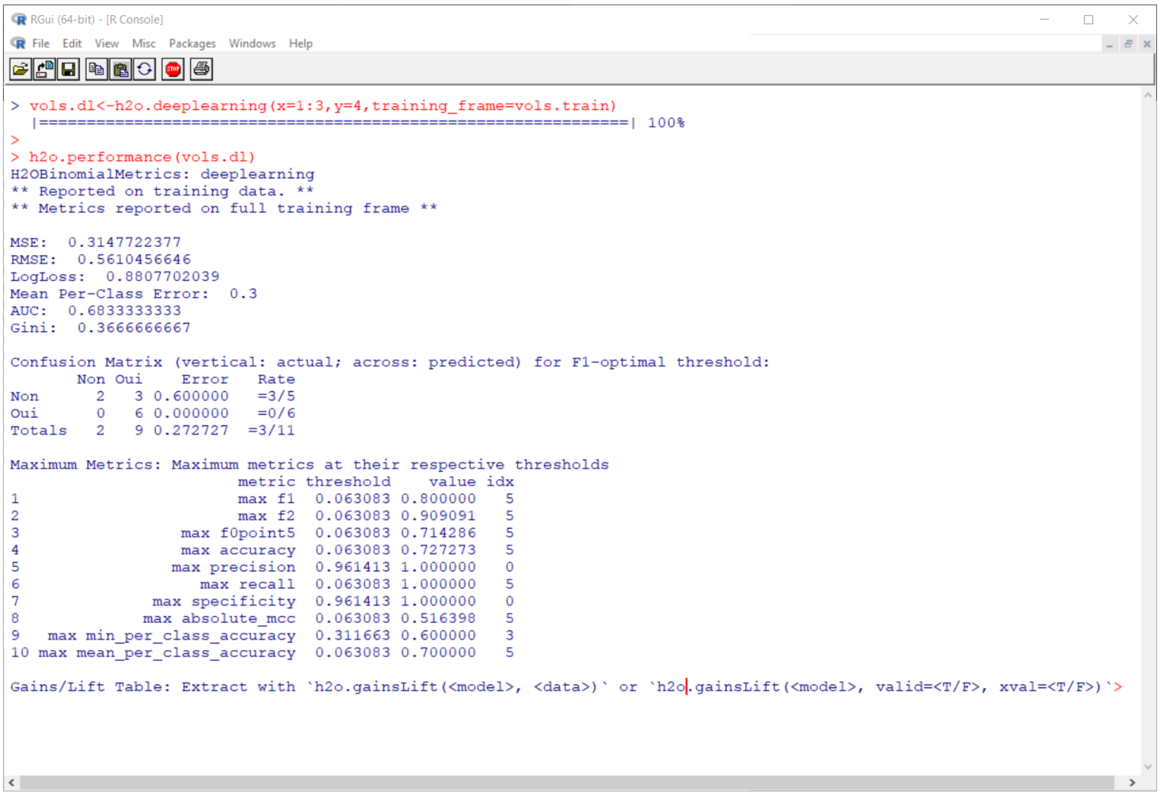
Sinon voici un deuxième exemple basé sur des données connus du cours théorique:

```
> library("h2o")
> h2o.init()
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         32 minutes 6 seconds
    H2O cluster timezone:       Europe/Berlin
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.20.0.2
    H2O cluster version age:    3 months and 6 days
    H2O cluster name:           H2O_started_from_R_Vincent_Isoz_dax044
    H2O cluster total nodes:    1
    H2O cluster total memory:   0.80 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  4
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.5.1 (2018-07-02)

>
> mydata<-read.csv("C:/tmp/DeepLearning_H2o.csv",sep=";",header=T)
> mydata
   Couleur   Type    Origine Vol
1    Rouge Sports Domestique Oui
2    Rouge Sports Domestique Non
3    Rouge Sports Domestique Oui
4    Jaune Sports Domestique Non
5    Jaune Sports     Import Oui
6    Jaune    SUV     Import Non
7    Jaune    SUV     Import Oui
8    Jaune    SUV Domestique Non
9    Rouge    SUV     Import Non
10   Rouge Sports     Import Oui
11   Rouge    SUV Domestique Oui
>
```

On l'entraîne et on regarde la performance:



```
> vols.dl<-h2o.deeplearning(x=1:3,y=4,training_frame=vols.train)
  |==============================================================| 100%
>
> h2o.performance(vols.dl)
H2OBinomialMetrics: deeplearning
** Reported on training data. **
** Metrics reported on full training frame **

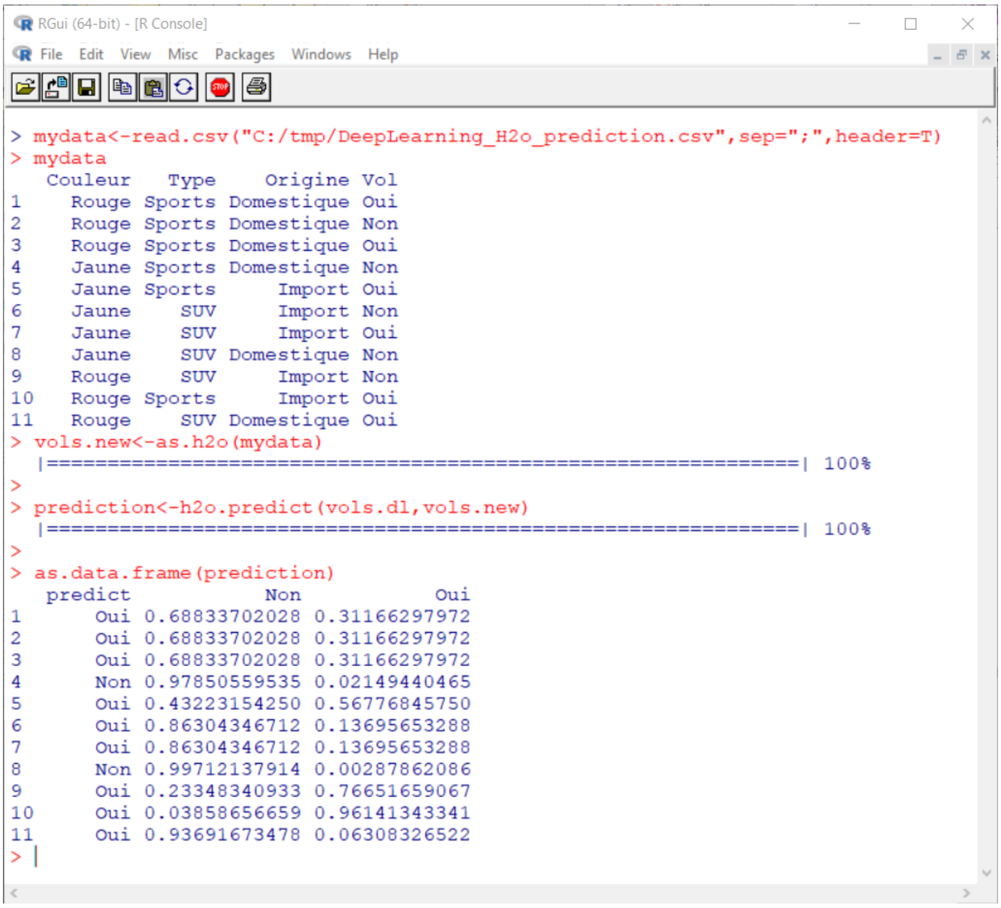
MSE:  0.3147722377
RMSE:  0.5610456646
LogLoss:  0.8807702039
Mean Per-Class Error:  0.3
AUC:  0.6833333333
Gini:  0.3666666667

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
       Non Oui    Error    Rate
Non      2   3 0.600000   =3/5
Oui      0   6 0.000000   =0/6
Totals   2   9 0.272727  =3/11

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold    value idx
1                       max f1  0.063083 0.800000   5
2                       max f2  0.063083 0.909091   5
3                 max f0point5  0.063083 0.714286   5
4                 max accuracy  0.063083 0.727273   5
5                max precision  0.961413 1.000000   0
6                   max recall  0.063083 1.000000   5
7              max specificity  0.961413 1.000000   0
8             max absolute_mcc  0.063083 0.516398   5
9   max min_per_class_accuracy  0.311663 0.600000   3
10 max mean_per_class_accuracy  0.063083 0.700000   5

Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`>
```

Et enfin on applique le modèle sur un nouveau jeu de données:



```
> mydata<-read.csv("C:/tmp/DeepLearning_H2o_prediction.csv",sep=";",header=T)
> mydata
   Couleur   Type    Origine Vol
1    Rouge Sports Domestique Oui
2    Rouge Sports Domestique Non
3    Rouge Sports Domestique Oui
4    Jaune Sports Domestique Non
5    Jaune Sports     Import Oui
6    Jaune    SUV     Import Non
7    Jaune    SUV     Import Oui
8    Jaune    SUV Domestique Non
9    Rouge    SUV     Import Non
10   Rouge Sports     Import Oui
11   Rouge    SUV Domestique Oui
> vols.new<-as.h2o(mydata)
  |==============================================================| 100%
>
> prediction<-h2o.predict(vols.dl,vols.new)
  |==============================================================| 100%
>
> as.data.frame(prediction)
   predict           Non           Oui
1      Oui 0.68833702028 0.31166297972
2      Oui 0.68833702028 0.31166297972
3      Oui 0.68833702028 0.31166297972
4      Non 0.97850559535 0.02149440465
5      Oui 0.43223154250 0.56776845750
6      Oui 0.86304346712 0.13695653288
7      Oui 0.86304346712 0.13695653288
8      Non 0.99712137914 0.00287862086
9      Oui 0.23348340933 0.76651659067
10     Oui 0.03858656659 0.96141343341
11     Oui 0.93691673478 0.06308326522
>
```

et voilà!

Exemple 369.: Traiter de gros jeux de données

R 3.0.2

Nous savons que R met tout dans la mémoire RAM par défaut, or si cette dernière est limitée par rapport aux données à analyser, outre acheter de la RAM (ce qui est compliqué dans les multinationales car prend du temps) une possibilité et d'utiliser des packages qui gèrent les données différemment (sur le disque).

Package LaF (Large ascii File)

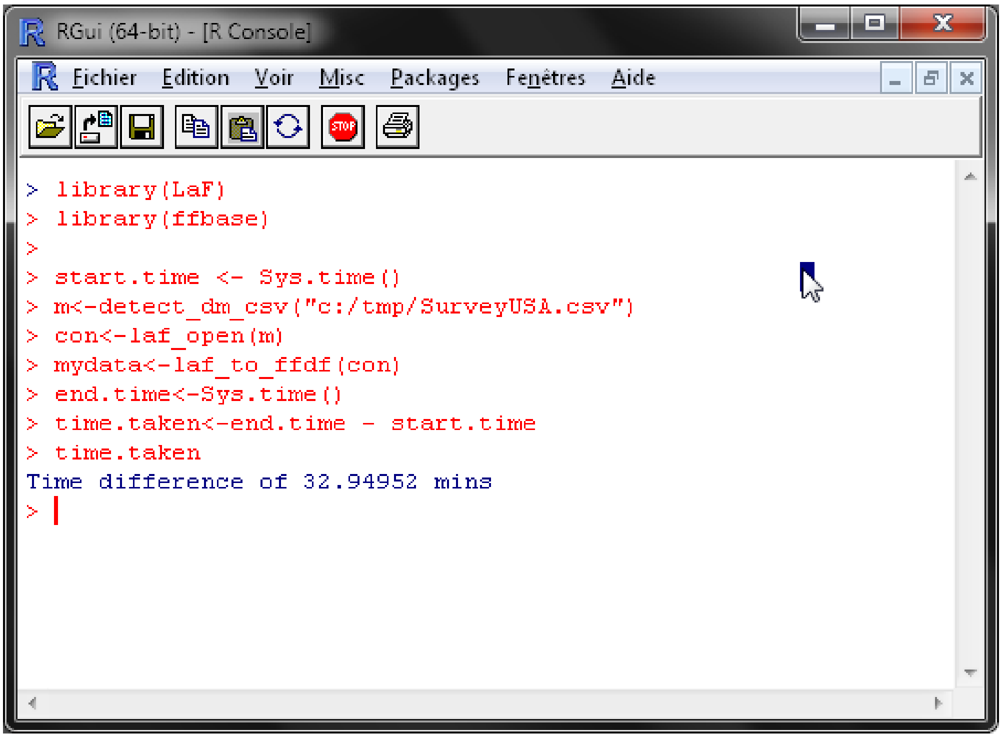
Les fonctions **detect_dm_csv( )** et **laf_open( )** du package **LaF** (Large ascii File) ont pour but premièrement de détecter le type de données des colonnes en espérant que le fichier ne soit pas trop mal structuré (c'est souvent les deux premières lignes qui déterminent le résultat final) et en second de créer la connexion au fichier avec les types de données détectés.

La fonction **laf_to_ffdf( )** du package **ff** permet de charger le fichier par morceaux sur le disque dur puisque la mémoire ne suffira pas dans le cas de l'ordinateur choisi pour la démonstration ci-dessous avec le fichier suivant:



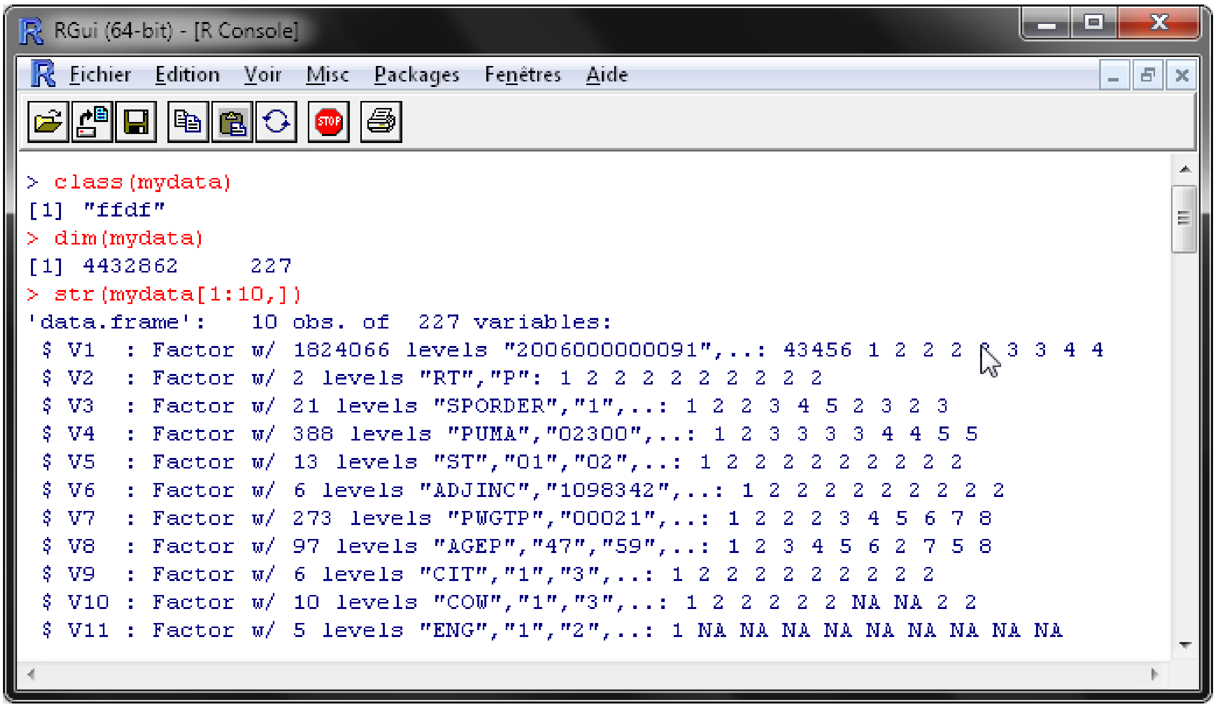
dont le contenu sera supposé inconnu.

Nous avons alors dans R (il vaut mieux ne pas travailler sur l'ordinateur pendant le traitement):



Au besoin vous pouvez écrire **alarm( )** à la fin pour avoir un signal sonore quand le calcul est terminé.

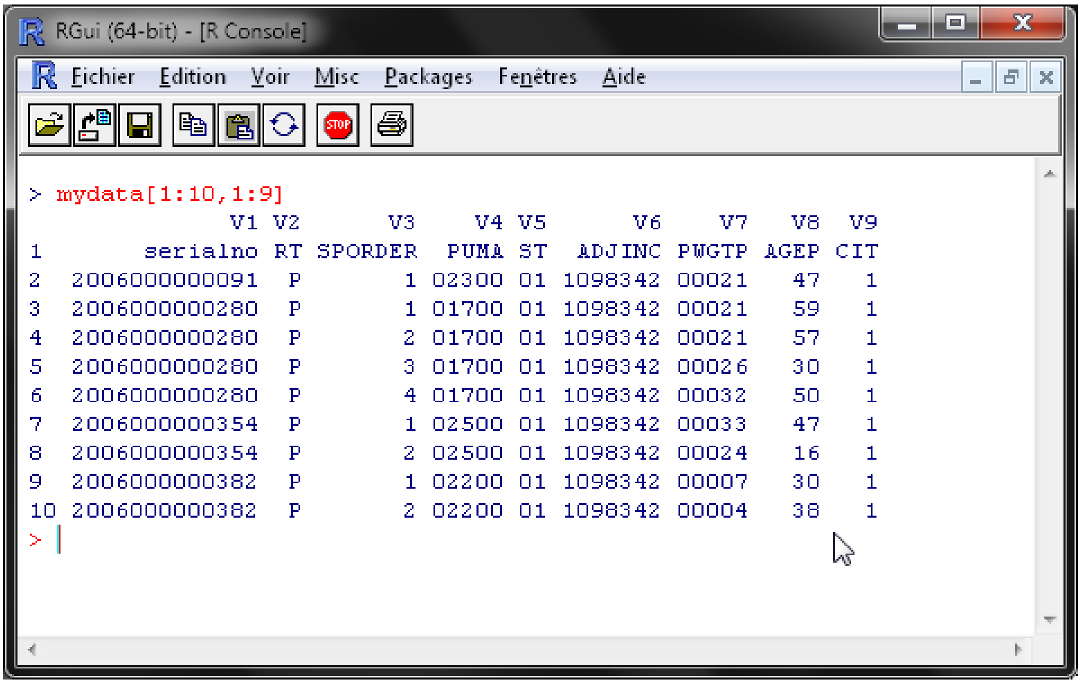
Maintenant investiguons un peu le contenu des données importées:



Nous voyons que nous avons importé environ 4.5 millions de lignes (ce n'est pas énorme) et 227 colonnes (ce qui est conséquent pour un sondage).

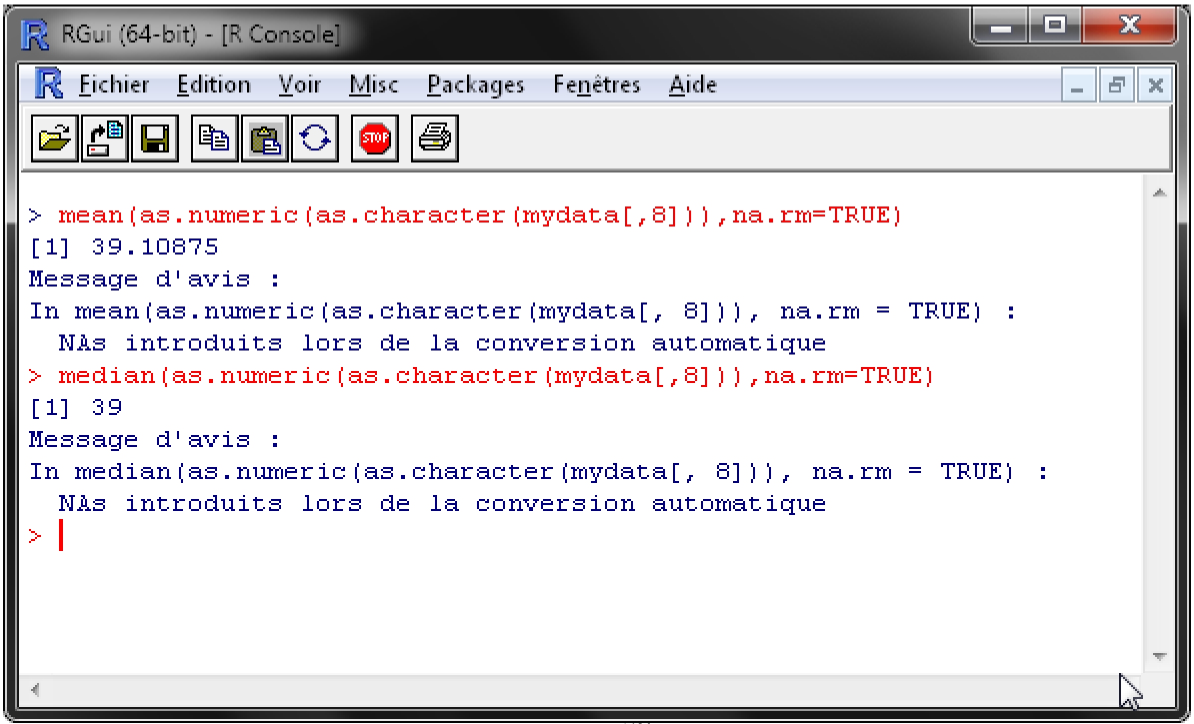
Nous voyons qu'il s'agit d'un objet de type ffdf content un data frame. Malheureusement, le fichier d'origine ne semble pas être bien structuré car la colonne *AGEP* pour "Age Population" est censée être de type Integer et tout a été identifié comme des chaînes de caractères.

Voyons les 10 premières lignes des 9 premières colonnes:



Le problème provient donc de la ligne 1 et 2 qui dans le fichier d'origine contient des textes... grrr! Malheureusement demander à la commande d'import de sauter la première ligne des V1, V2 ne marche pas (loi du maximum d'emm...).

Donc nous allons faire de la conversion classique:

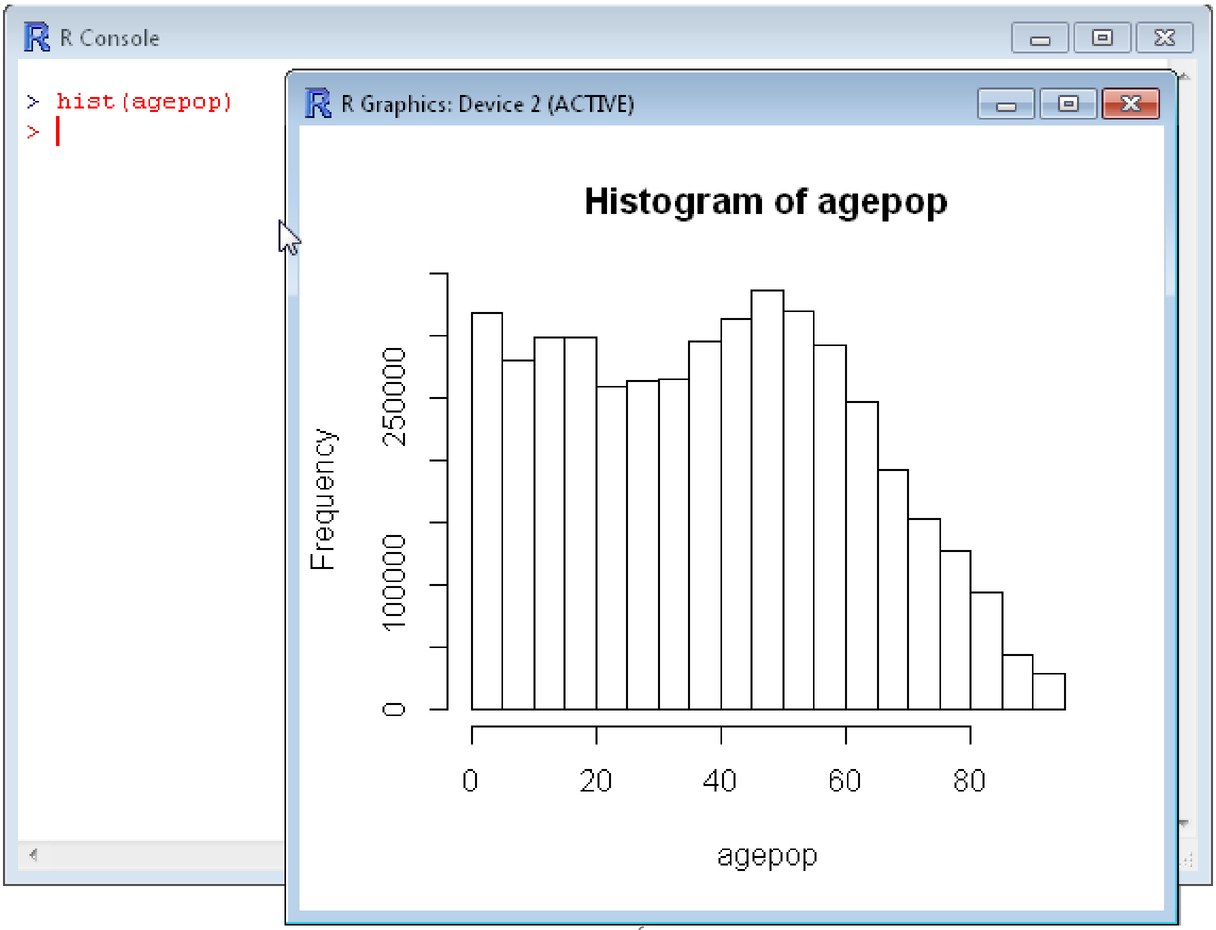


Après le principe reste le même que ce que nous avons vu jusqu'à maintenant cependant on tombe souvent sur des aberrations avec ces gros jeux de données... :

```
> agepop<-as.numeric(as.character(mydata[,8]))
Message d'avis :
NAs introduits lors de la conversion automatique
> mydata<-aggregate(agepop,by=list(agepop),FUN=length)
> mydata
   Group.1     x
1        0 50771
2        1 51785
3        2 53729
4        3 53712
5        4 53973
6        5 54729
7        6 54657
8        7 55201
9        8 56139
10       9 57030
11      10 57656
12      11 57149
13      12 59202
14      13 59742
15      14 60200
16      15 62286
17      16 63199
18      17 63253
19      18 61515
20      19 56568
21      20 53893
22      21 52410
23      22 51010
```

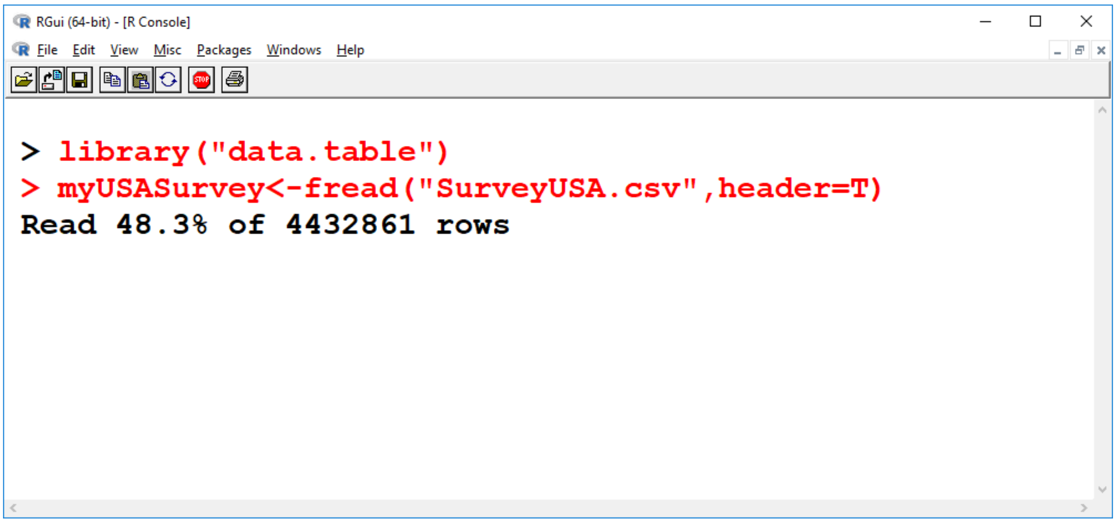
Il est effectivement difficilement imaginable qu'il y ait 50'771 personnes dont l'âge est zéro et ayant répondu...

Soit autrement vu:

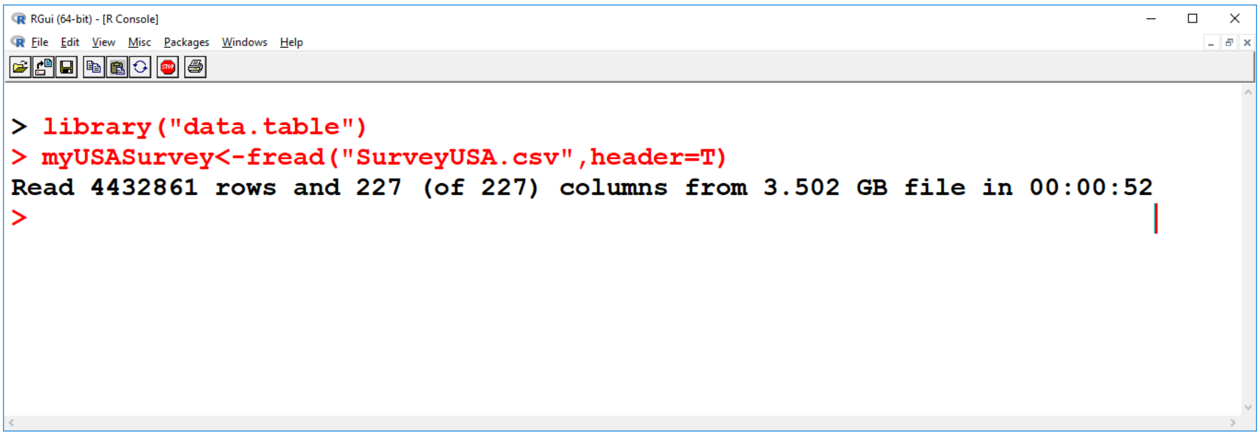

R Console
> hist(agepop)
>
R Graphics: Device 2 (ACTIVE)
Histogram of agepop
Frequency
0
100000
250000
0
20
40
60
80
agepop

Package data.table (fread)

Nous avons déjà parcouru le package **data.table**. Mais il a une fonction très utile pour les gros jeux de données, appelée **fread( )** . Elle s'utilise très simplement et affiche en plus l'avancement du chargement du fichier:



et quand l'import est terminé, nous obtenons:



Au besoin vous pouvez à nouveau écrire un **alarm( )** à la fin pour avoir un signal sonore quand le calcul est terminé.

Pour le reste…. cela se manipule comme tout autre data frame! C'est-à-dire qu'on peut importer/fusionner aussi un dossier avec plein de *.csv via les fonctions **list.files( )** et **do.call( )**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("data.table")
> files<-list.files("C:/tmp/Ventes/",pattern="*.csv",full.name = TRUE)
> files
[1] "C:/tmp/Ventes/Ventes2016.csv" "C:/tmp/Ventes/Ventes2017.csv"
> merge<-do.call(rbind,lapply(files,fread))
Read 12044587 rows and 10 (of 10) columns from 1.109 GB file in 00:00:25
Read 12044587 rows and 10 (of 10) columns from 1.109 GB file in 00:00:28
> nrow(merge)
[1] 24089174
> alarm()
>

Réduction dimensionnelle/Feature selection

Une fois les gros jeux de données chargés, on peut éventuellement éliminer certaines variables ou facteurs en utilisant les techniques empiriques de Feature selection. Nous savons déjà que l'analyse en composantes principales et la décomposition en valeurs singulières en font partie.

Analyse en composantes principales

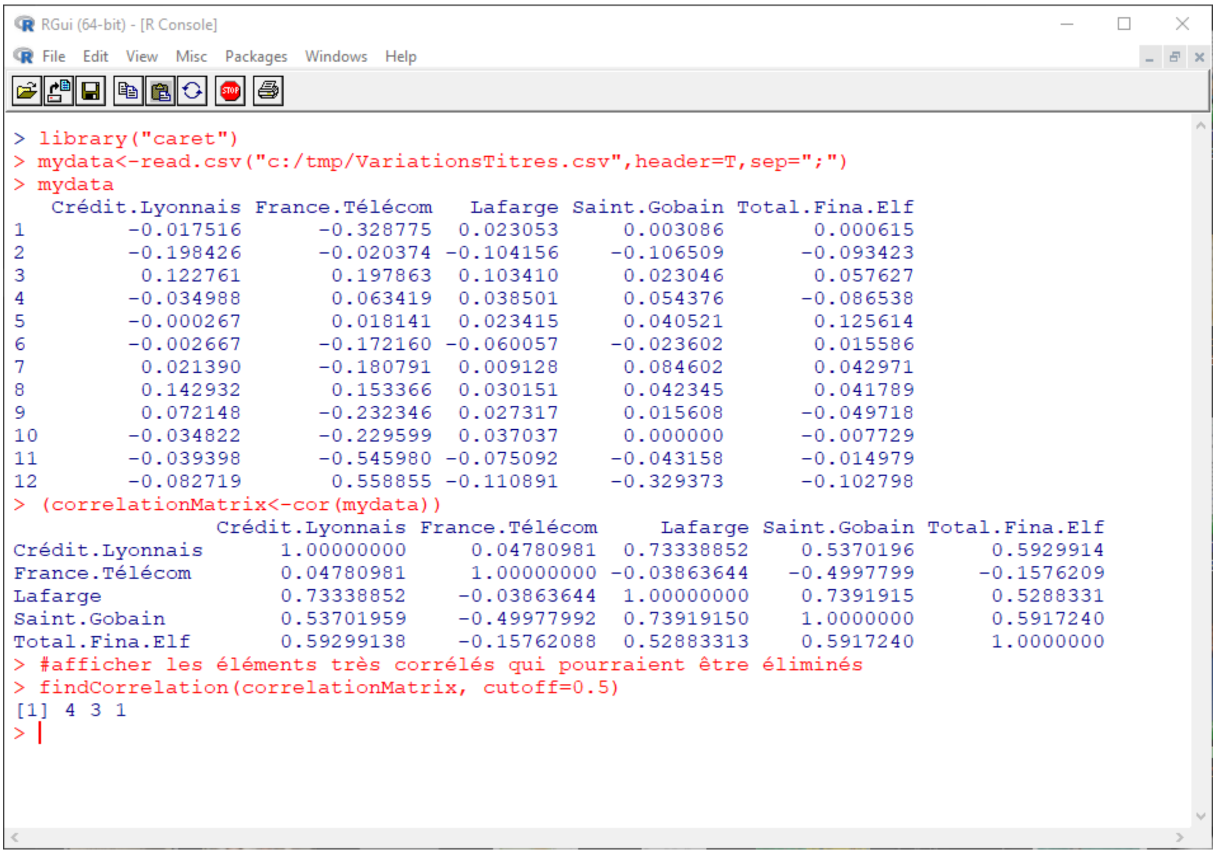
Cette technique a déjà été traitée à la page 1401.

Décomposition en valeurs singulières

Cette technique a déjà été traitée à la page 283.

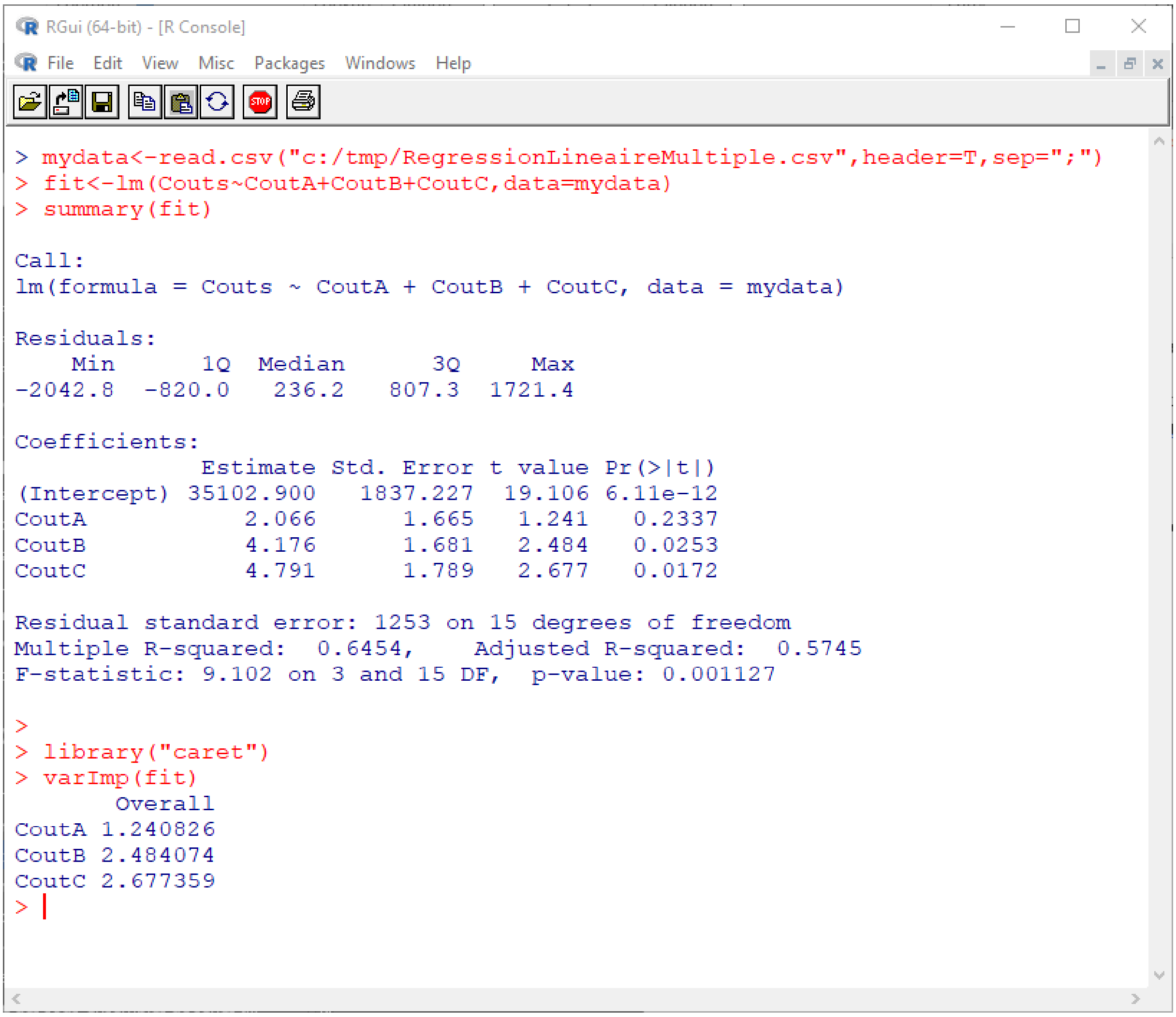
Corrélation (réduction dimensionnelle sur variables numériques)

Voyons d'abord le cas ultra-classique avec la fonction **findCorrelation( )** du package **car**:



Importance de variables (réduction dimensionnelle sur variables numériques)

Bon les techniques dans l'importance de variables sont vraiment très empiriques dès le moment où on s'éloigne des techniques de statistiques formelles. Voyons donc d'abord un cas d'application avec la régression linéaire multiple (sachant que le même principe peut s'appliquer à n'importe quel modèle de régression!) en utilisant la fonction **varImp( )** du package caret:



Résultat à comparer avec le résultat que nous avions obtenu à la page 1601.

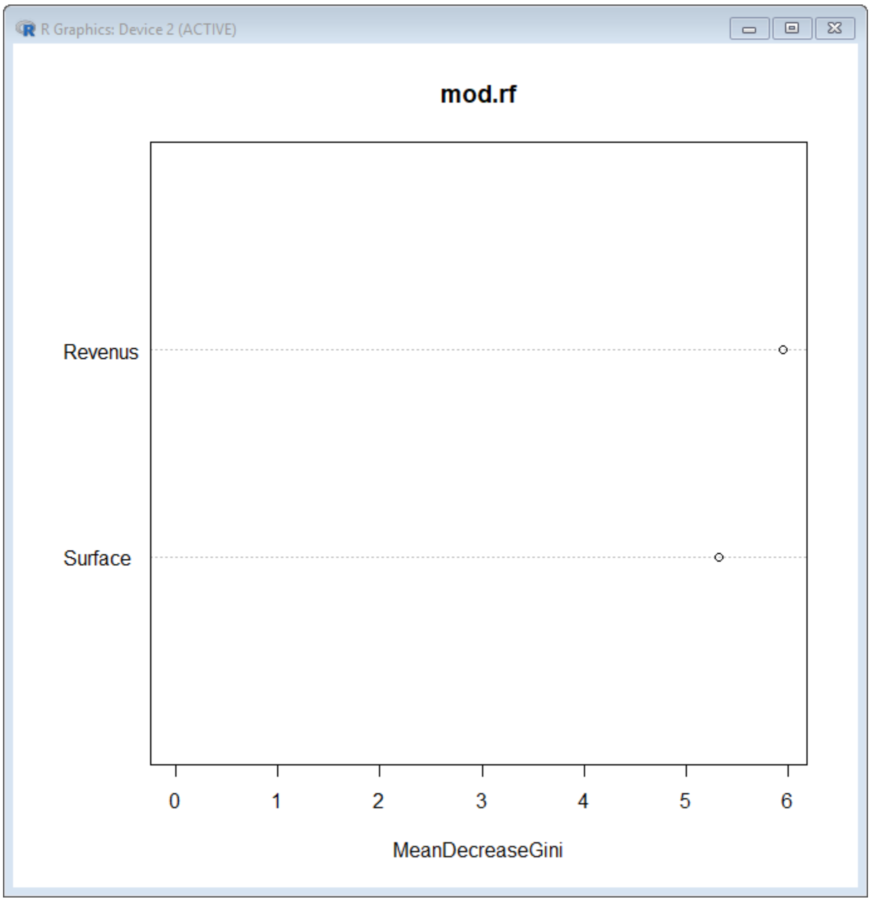
Ou encore un autre exemple reprenant le RandomForest que nous avions fait plus haut à la page 2181:

```
> library("randomForest")
> library("caret")
> mydata<-read.csv("c:/tmp/ArbreRegressionCART.csv",header=T,sep=";")
> progstat<-factor(mydata$Propriétaire,levels=1:2,labels=c("Proprio","Non Proprio"))
> mod.rf<-randomForest(progstat~Revenus+Surface,data=mydata)
> mod.rf

Call:
 randomForest(formula = progstat ~ Revenus + Surface, data = mydata)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 29.17%
Confusion matrix:
            Proprio Non Proprio class.error
Proprio           8           4   0.3333333
Non Proprio       3           9   0.2500000
> varImp(mod.rf)
         Overall
Revenus 6.258222
Surface 5.049155
> varImpPlot(mod.rf)
> 
```

Ce qui donne pour la fonction **varImpPlot( )**:



Khi-2 (réduction dimensionnelle sur variables numériques)

La méthode du khi-2 peut être utilisée pour la réduction dimensionnelle de variables catégorielles. Voyons donc un exemple en utilisant la version **32 bits de R** (car le package **FSelector**, utilise RJava en 32 bits à ce jour!):

```
> #On utilise le jeu de données HouseVotes84 qui se trouve
> #dans le package mlbench
> library("mlbench") #pour les données
> library("FSelector") #pour la méthode
> data(HouseVotes84)
> summary(HouseVotes84)
        Class           V1          V2          V3          V4          V5          V6     
 democrat  :267   n   :236   n   :192   n   :171   n   :247   n   :208   n   :152  
 republican:168   y   :187   y   :195   y   :253   y   :177   y   :212   y   :272  
                  NA's: 12   NA's: 48   NA's: 11   NA's: 11   NA's: 15   NA's: 11  
    V7          V8          V9         V10         V11         V12         V13         V14     
 n   :182   n   :178   n   :206   n   :212   n   :264   n   :233   n   :201   n   :170  
 y   :239   y   :242   y   :207   y   :216   y   :150   y   :171   y   :209   y   :248  
 NA's: 14   NA's: 15   NA's: 22   NA's:  7   NA's: 21   NA's: 31   NA's: 25   NA's: 17  
   V15         V16     
 n   :233   n   : 62  
 y   :174   y   :269  
 NA's: 28   NA's:104  
> head(HouseVotes84)
       Class   V1 V2 V3   V4   V5 V6 V7 V8 V9 V10  V11  V12 V13 V14 V15  V16
1 republican    n  y  n    y    y  y  n  n  n   y <NA>    y   y   y   n    y
2 republican    n  y  n    y    y  y  n  n  n   n    n    y   y   y   n <NA>
3   democrat <NA>  y  y <NA>    y  y  n  n  n   n    y    n   y   y   n    n
4   democrat    n  y  y    n <NA>  y  n  n  n   n    y    n   y   n   n    y
5   democrat    y  y  y    n    y  y  n  n  n   n    y <NA>   y   y   y    y
6   democrat    n  y  y    n    y  y  n  n  n   n    n    n   y   y   y    y
> 
```

Enfin, on utilise les fonctions **chi.squared( )** et **cutoff.k( )** du package **FSelector**:

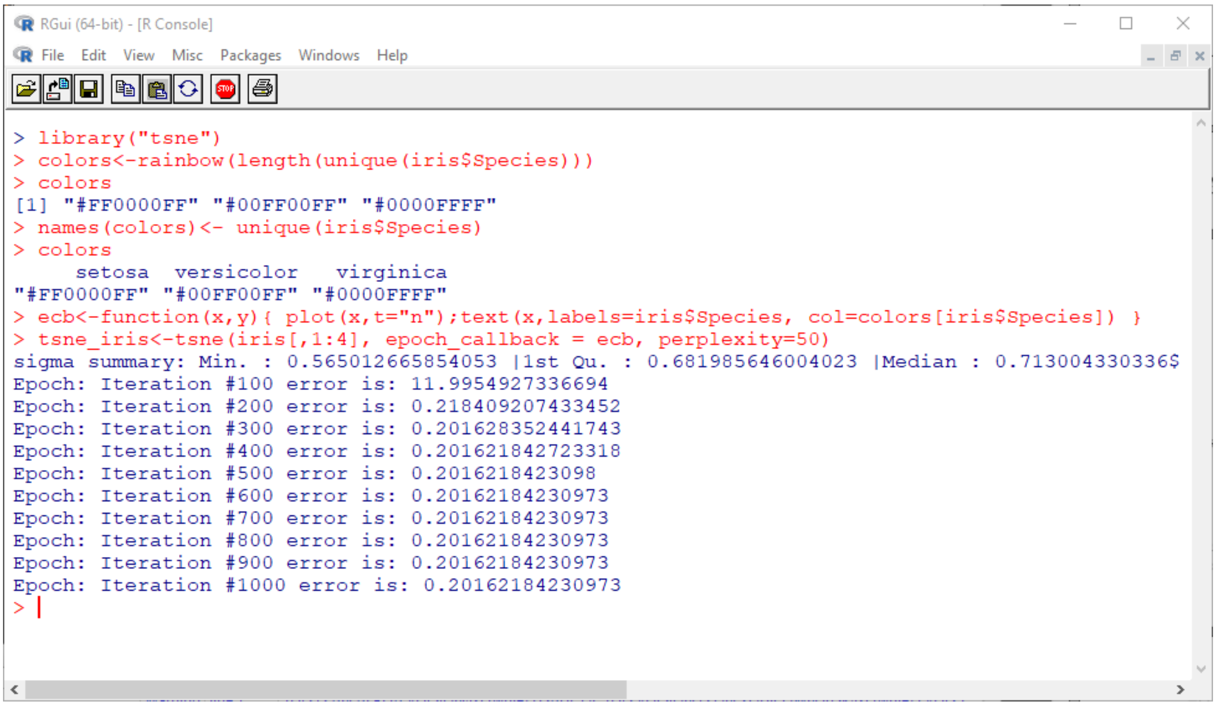
```
> #On calcule la statistique du khi-2
> poids<-chi.squared(Class~., HouseVotes84)
>
> #on affiche les résultats
> poids
    attr_importance
V1       0.409330348
V2       0.004534049
V3       0.748864321
V4       0.923255954
V5       0.718768923
V6       0.428332508
V7       0.521967369
V8       0.661876085
V9       0.629797943
V10      0.083809300
V11      0.378240781
V12      0.714922593
V13      0.555971176
V14      0.625283342
V15      0.538263037
V16      0.353273580
>
> #On sélectionne les 5 variables avec le plus grand poids
> subset<- cutoff.k(poids, 5)
> subset
[1] "V4"  "V3"  "V5"  "V12" "V8"
> #on génère la formule correspondante automatiquement
> f<-as.simple.formula(subset, "Class")
> print(f)
Class ~ V4 + V3 + V5 + V12 + V8
<environment: 0x08511ef0>
>
```

t-SNE (t-distributed stochastic neighbor embedding)

Bon voyons cette technique dont nous avons étudié relativement en détails l'idée mathématique sous-jacente dans le cours théorique (cependant nous n'avons pas fait d'application numériques à la main ni dans un autre logiciel ou tableur).

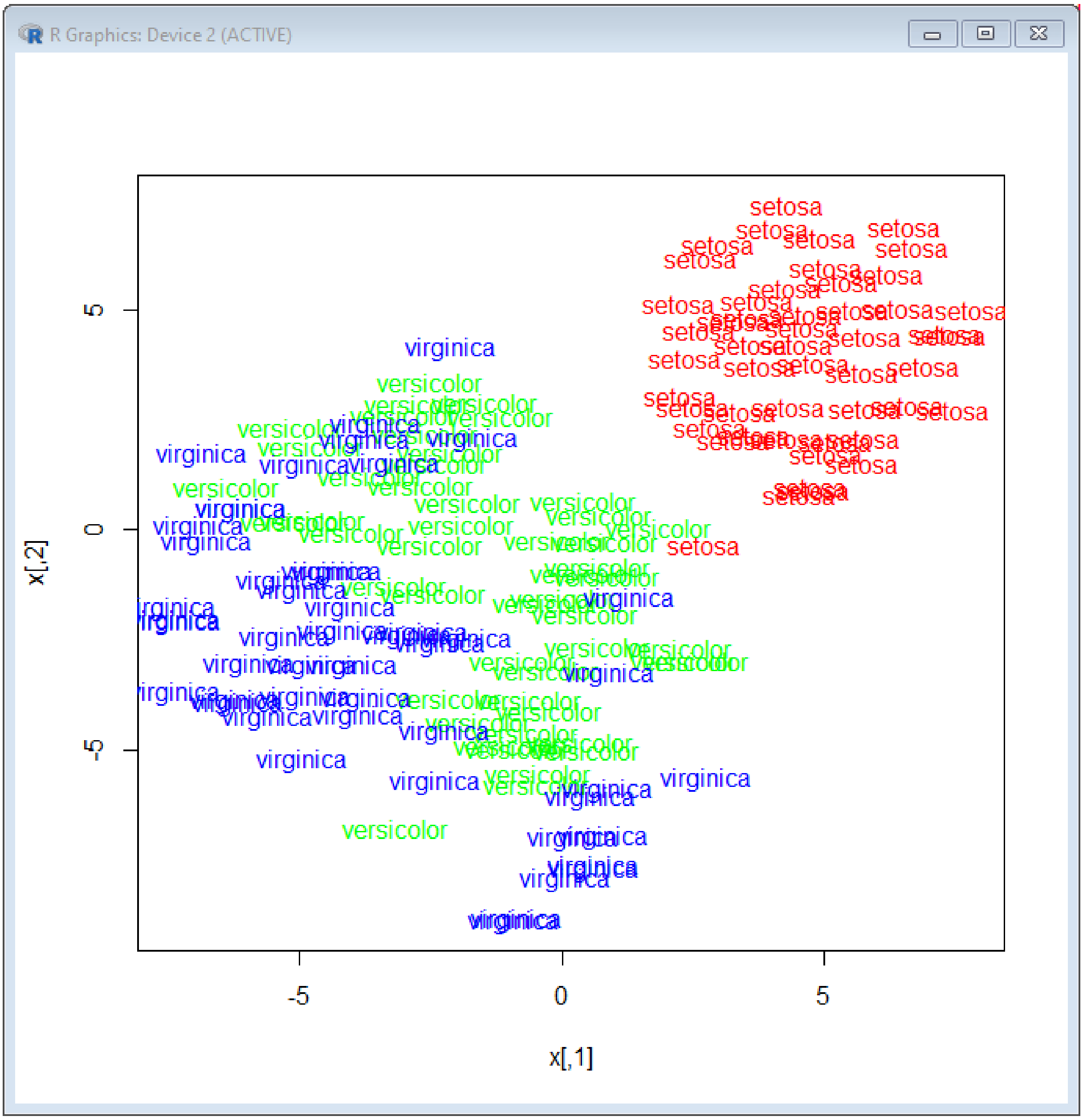
Pour faire original, nous allons simplement reproduire ici l'exemple donné dans l'aide des deux packages qui vont nous servir à illustrer cet algorithme.

D'abord commençons avec le package **tsne** et sa fonction **tsne( )**:



```
> library("tsne")
> colors<-rainbow(length(unique(iris$Species)))
> colors
[1] "#FF0000FF" "#00FF00FF" "#0000FFFF"
> names(colors)<- unique(iris$Species)
> colors
     setosa  versicolor   virginica
"#FF0000FF" "#00FF00FF" "#0000FFFF"
> ecb<-function(x,y){ plot(x,t="n");text(x,labels=iris$Species, col=colors[iris$Species]) }
> tsne_iris<-tsne(iris[,1:4], epoch_callback = ecb, perplexity=50)
sigma summary: Min. : 0.565012665854053 |1st Qu. : 0.681985646004023 |Median : 0.713004330336$
Epoch: Iteration #100 error is: 11.9954927336694
Epoch: Iteration #200 error is: 0.218409207433452
Epoch: Iteration #300 error is: 0.201628352441743
Epoch: Iteration #400 error is: 0.201621842723318
Epoch: Iteration #500 error is: 0.2016218423098
Epoch: Iteration #600 error is: 0.20162184230973
Epoch: Iteration #700 error is: 0.20162184230973
Epoch: Iteration #800 error is: 0.20162184230973
Epoch: Iteration #900 error is: 0.20162184230973
Epoch: Iteration #1000 error is: 0.20162184230973
>
```

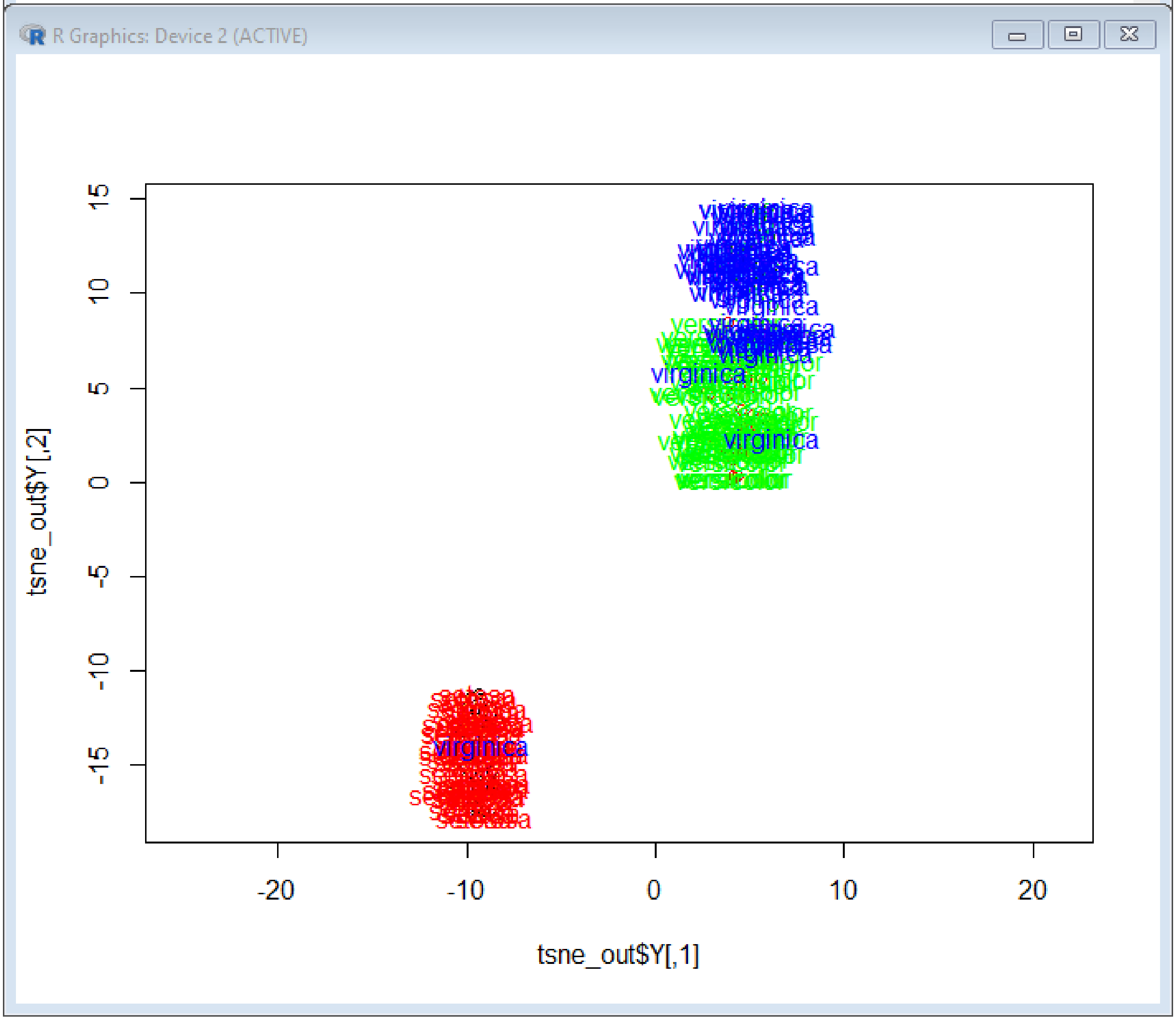
Ce qui donne graphiquement:



Ou avec le package **Rtsne** et sa fonction **Rtsne( )**:

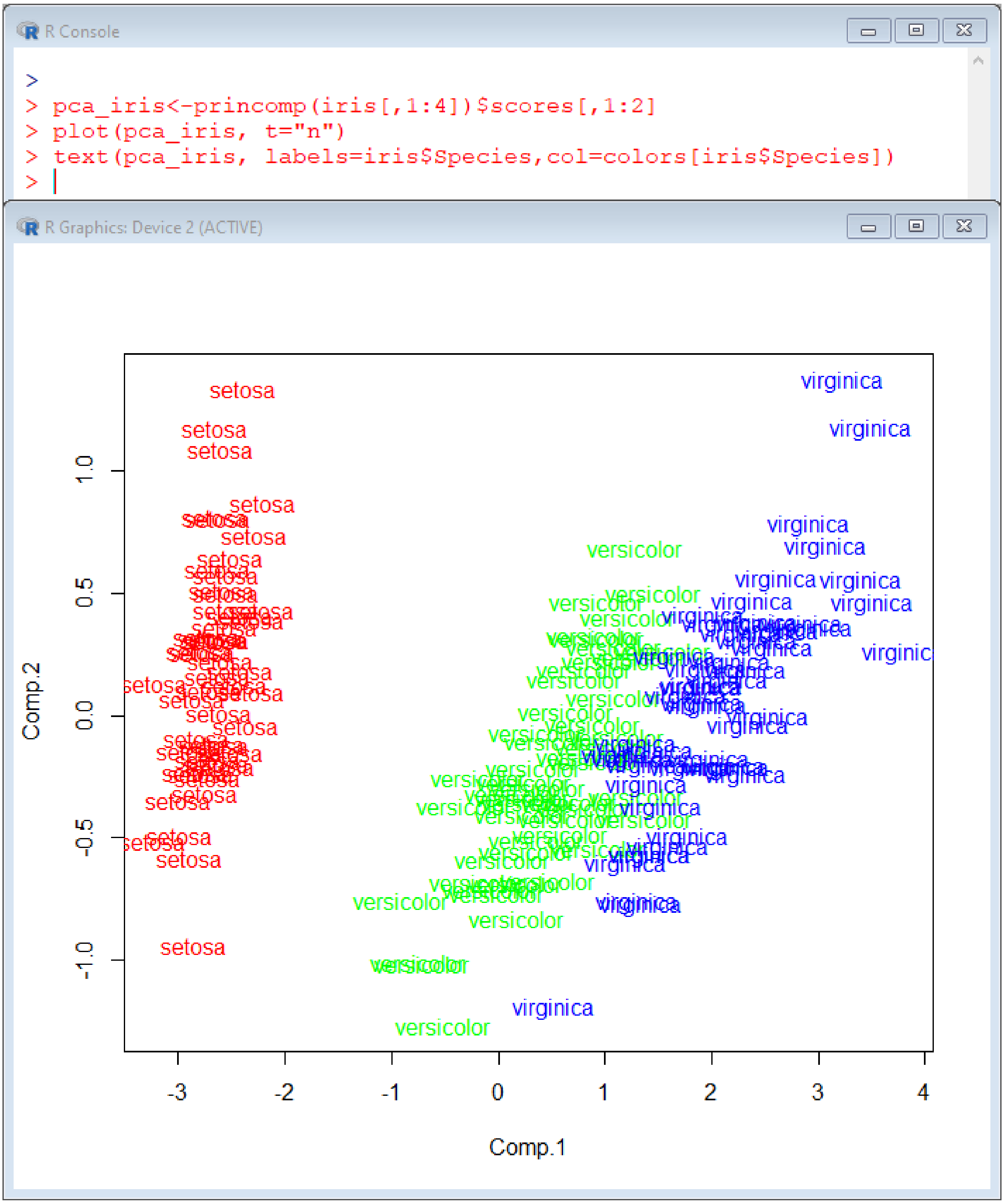
R Console

```
> library("Rtsne")
> #on supprime les doublons comme l'exige ce package
> iris_unique <- unique(iris)
> tsne_out <- Rtsne(as.matrix(iris_unique[,1:4]))
> plot(tsne_out$Y,col=iris_unique$Species,asp=1)
> text(tsne_out$Y,labels=iris$Species, col=colors[iris$Species])
>
```



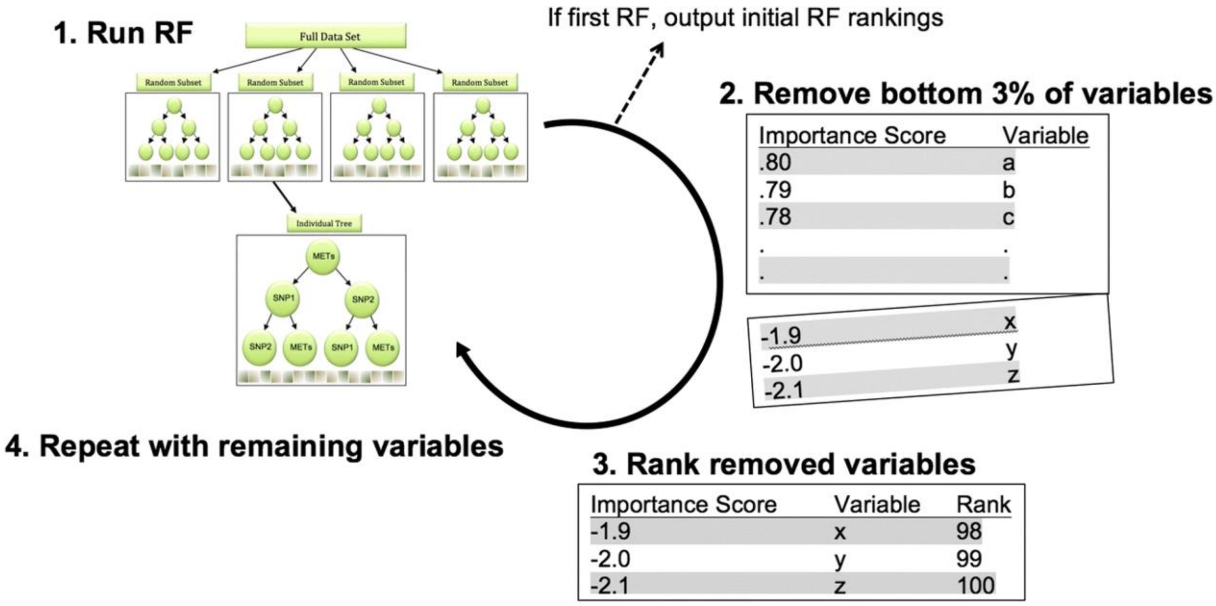
On voit comme toujours l'importance d'utiliser d'autres packages pour voir les résultats d'autres implémentations algorithmiques!

Maintenant comparons avec l'analyse en composantes principales pour le plaisir:


R Console
>
> pca_iris<-princomp(iris[,1:4])$scores[,1:2]
> plot(pca_iris, t="n")
> text(pca_iris, labels=iris$Species,col=colors[iris$Species])
>
R Graphics: Device 2 (ACTIVE)
setosa
versicolor
virginica
Comp.2
Comp.1
-1.0
-0.5
0.0
0.5
1.0
-3
-2
-1
0
1
2
3
4

Recursive Feature Elimination

L'exemple ci-dessous fournit un exemple de la méthode RFE sur l'ensemble de données *Pima Indians Diabetes*. Un algorithme de forêt aléatoire est utilisé à chaque itération pour évaluer le modèle. L'algorithme est configuré pour explorer tous les sous-ensembles possibles des attributs.



Chargeons les packages et le jeu de données utilisés pour l'exemple:

```
> library("mlbench")
> library("caret")
> # load the data
> data(PimaIndiansDiabetes) #dans le package mlbench
> head(PimaIndiansDiabetes,20)
   pregnant glucose pressure triceps insulin mass pedigree age diabetes
1         6     148       72      35       0 33.6    0.627  50      pos
2         1      85       66      29       0 26.6    0.351  31      neg
3         8     183       64       0       0 23.3    0.672  32      pos
4         1      89       66      23      94 28.1    0.167  21      neg
5         0     137       40      35     168 43.1    2.288  33      pos
6         5     116       74       0       0 25.6    0.201  30      neg
7         3      78       50      32      88 31.0    0.248  26      pos
8        10     115        0       0       0 35.3    0.134  29      neg
9         2     197       70      45     543 30.5    0.158  53      pos
10        8     125       96       0       0  0.0    0.232  54      pos
11        4     110       92       0       0 37.6    0.191  30      neg
12       10     168       74       0       0 38.0    0.537  34      pos
13       10     139       80       0       0 27.1    1.441  57      neg
14        1     189       60      23     846 30.1    0.398  59      pos
15        5     166       72      19     175 25.8    0.587  51      pos
16        7     100        0       0       0 30.0    0.484  32      pos
17        0     118       84      47     230 45.8    0.551  31      pos
18        7     107       74       0       0 29.6    0.254  31      pos
19        1     103       30      38      83 43.3    0.183  33      neg
20        1     115       70      30      96 34.6    0.529  32      pos
> 
```

Nous utilisons maintenant les fonctions **rfeControl( )** et **rfe**( ) du package **caret**:

```
> control <- rfeControl(functions=rfFuncs, method="cv", number=10)
> results <- rfe(PimaIndiansDiabetes[,1:8], PimaIndiansDiabetes[,9], sizes=c(1:8), rfeControl=control)
> print(results)

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

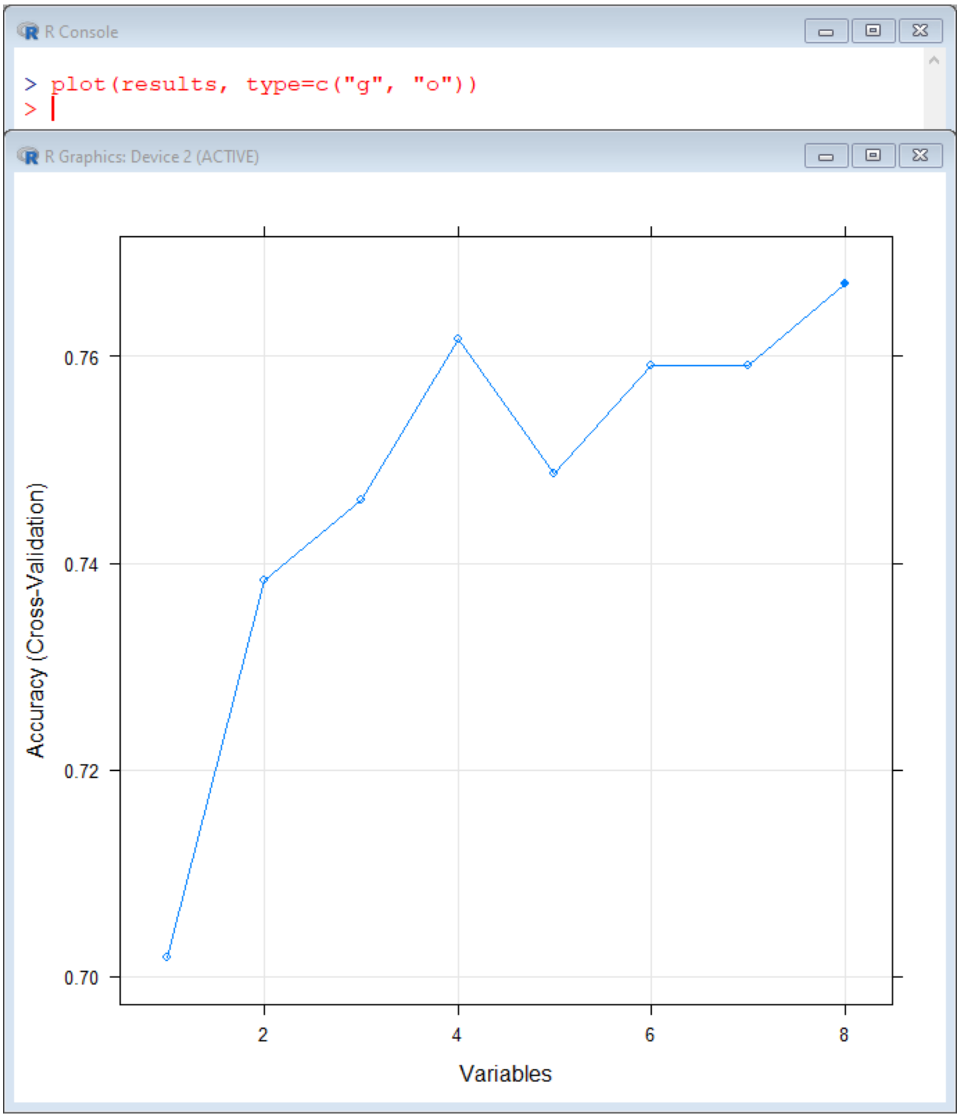
Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.7019 0.2881    0.04403 0.09645
         2   0.7383 0.4060    0.06571 0.14915
         3   0.7462 0.4234    0.04475 0.10597
         4   0.7617 0.4674    0.04732 0.10263
         5   0.7487 0.4373    0.05749 0.12710
         6   0.7592 0.4521    0.06564 0.15485
         7   0.7592 0.4544    0.06545 0.15354
         8   0.7670 0.4699    0.06333 0.14652        *

The top 5 variables (out of 8):
   glucose, mass, age, pregnant, insulin

> predictors(results)
[1] "glucose"  "mass"     "age"      "pregnant" "insulin"  "pedigree" "triceps"  "pressure"
> 
```

Nous pouvons faire un graphique:



Les 8 attributs sont sélectionnés dans cet exemple, bien que dans le graphique montrant la précision des différentes tailles de sous-ensemble d'attributs, nous pouvons voir que seulement 4 attributs donnent des résultats presque comparables.

Data Stream / Données en flux continu

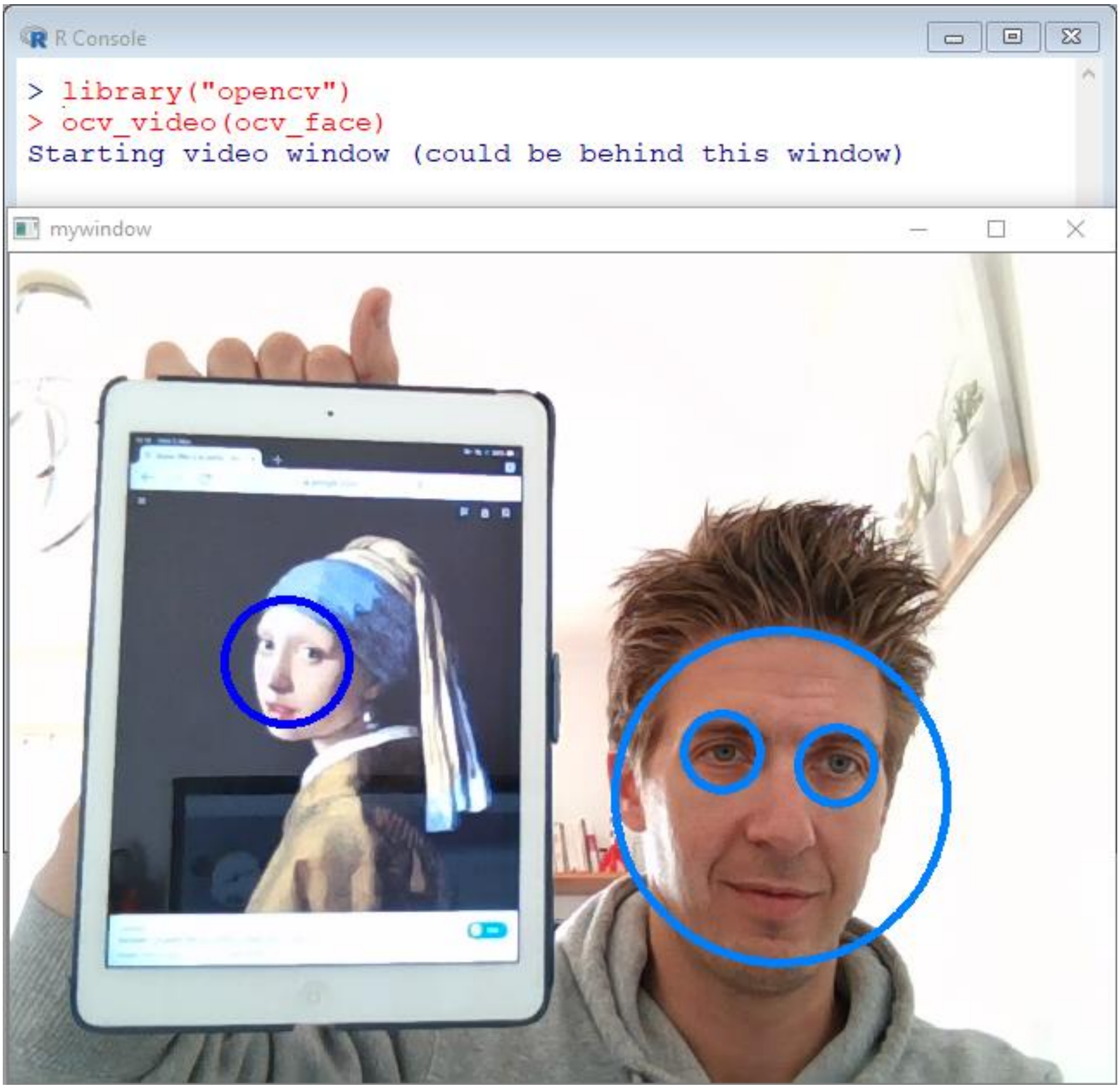
Nous avons étudié un peu les problématiques typiques des flux de données dans le cours théorique et il est apparu évident que dans la pratique, si l'on peut éviter de relancer un algorithme de ML/DL sur un data set entier juste parce qu'une petite dizaine de données est apparu alors on l'éviter pour des raisons de consommation de mémoire et de performance!

Exemple 370.: OpenCV (reconnaissance faciale)

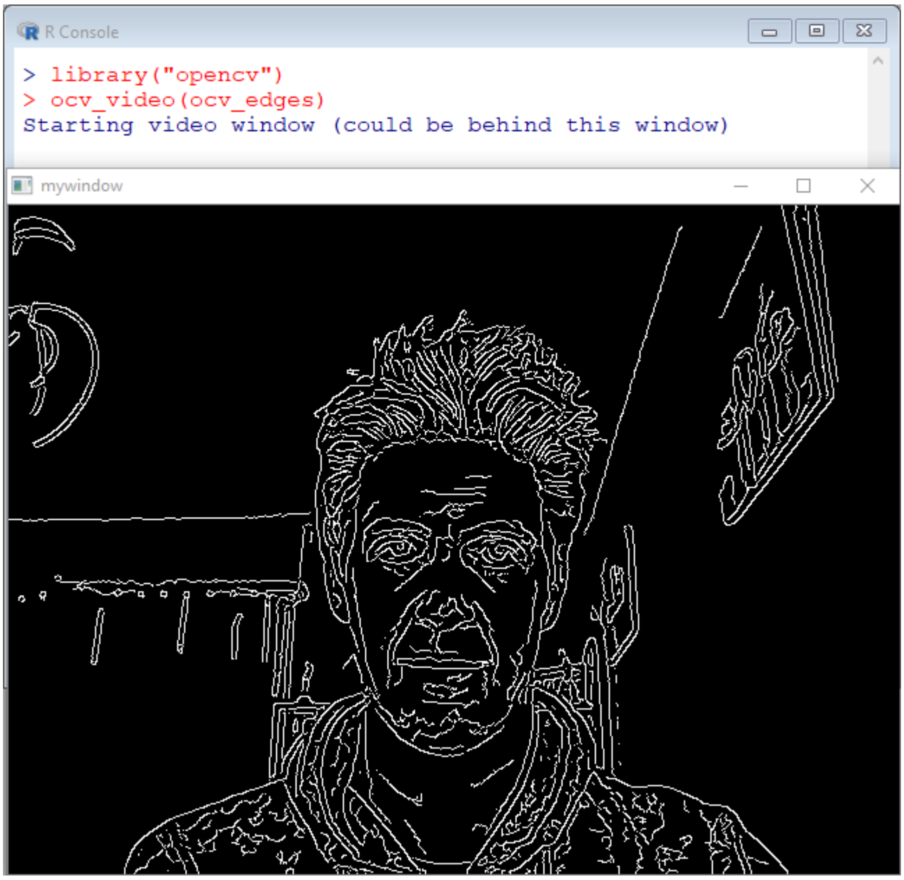
R 3.6.3

Bon le lecteur doit bien garder en tête que R est un logiciel de statistiques et son rôle n'est pas de faire de la reconnaissance vidéo. Il faudrait plus allé du côté de MATLAB et dans le pire des cas voir du côté de Python.

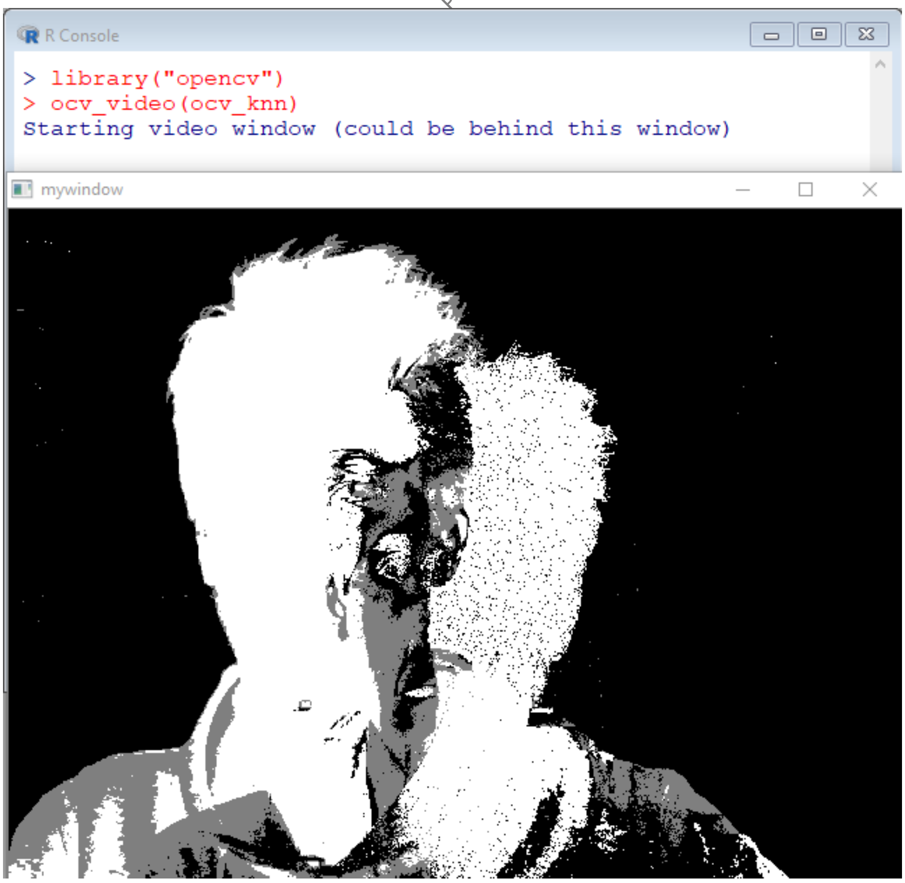
Donc voyons d'abord ce que peut faire le package **opencv** avec pour commencer la fonction **ocv_video( )** qui tente de reconnaître en temps réel un visage sur une caméra avec différents paramétres en entrée. Ci-dessous nous voyons son fonctionnement avec le paramètre *ocv_face*:



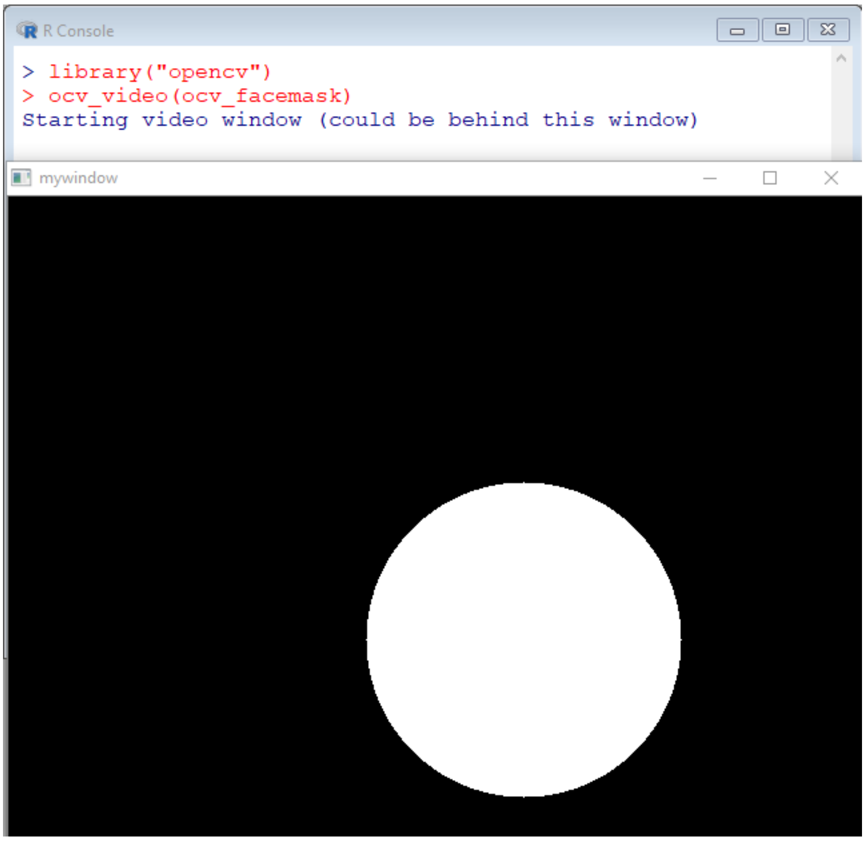
Et avec le paramètre *ocv_edges*:



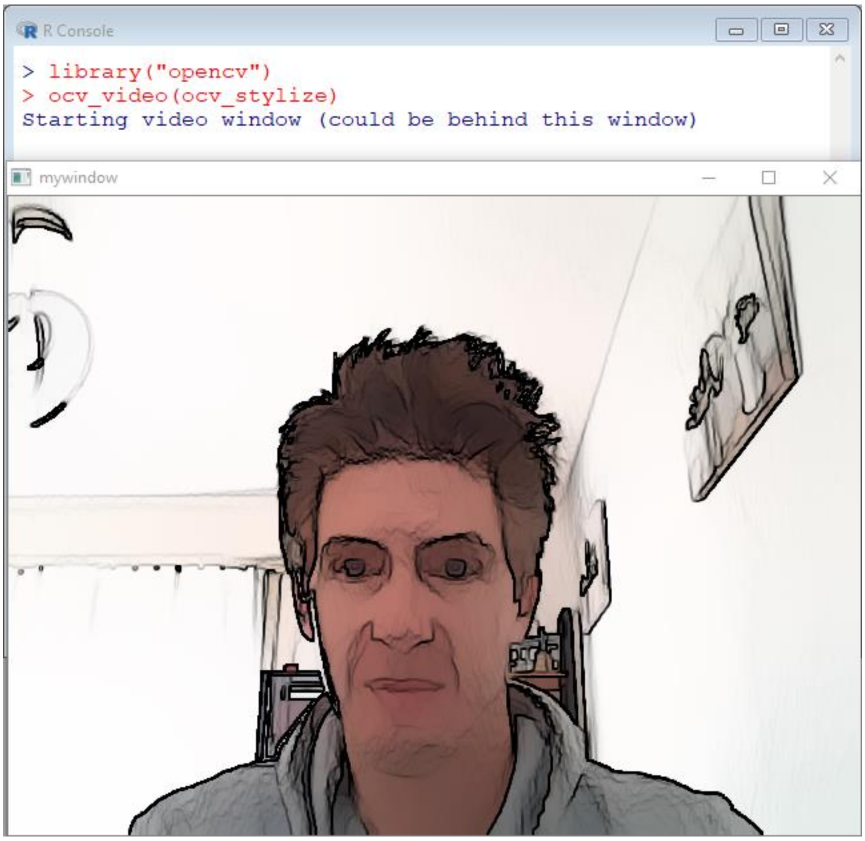
et avec le paramètre *ocv_knn*:



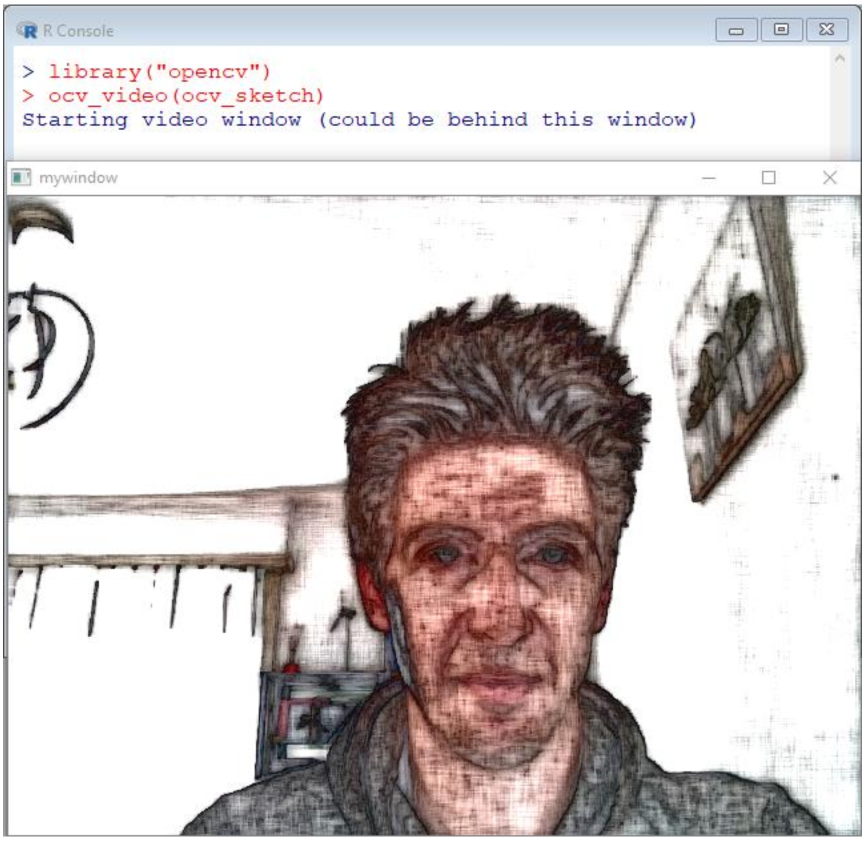
Et avec le paramètre *ocv_facemask*:



et avec le paramètre *ocv_stylize*:



et pour terminer avec le paramètre *ocv_sketch*:



Maintenant faisons de la reconnaissance faciale sur une photo. Considérons la photo suivante:



Nous écrivons alors le script suivant:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help
> library("opencv")
> faces<-ocv_read("c:/tmp/Solvay_Conference_1927.jpg")
> #trouver les visages
> faces <- ocv_face(faces)
> #mettre un masque sur les visages
> facemask <- ocv_facemask(faces)
> #afficher la liste des visages trouvés et leurs coordonnées
> attr(facemask,'faces')
   radius    x   y
1      35 1111 386
2      32  458 204
3      34 1288 256
4      36  420 424
5      33 1878 184
6      38  632 402
7      32 1430 268
8      32  920 270
9      30 1756 256
10     36 1518 396
11     32 1574 258
12     36  223 403
13     41 1747 406
14     36 1316 380
15     39 1944 411
16     34 1110 296
17     40 1862 466
18     32  616 254
19     40 1468 460
20     40  864 400
21     42 1626 458
22     38 1250 462
23     36  740 474
24     34  734 308
25     36  293 474
26     36 1958 258
27     42  984 482
28     43  515 485
29     36 2026 492
30     36  290 528
> #enregistrer le résultat pour visualisation
> ocv_write(faces , 'Solvay_Conference_1927_Output.jpg')
[1] "C:\\tmp\\Solvay_Conference_1927_Output.jpg"
> 
```

Ce qui donne comme résultat:



colourized by pastincolour.com

SOLVAY CONFERENCE 1927

A. PICARD E. HENRIOT P. EHRENFEST Ed. HERSEN Th. DE DONDER E. SCHRÖDINGER E. VERSCHAFFELT W.PAULI W. HEISENBERG R.H FOWLER L. BRILLOUIN

P. DEBYE M. KNUDSEN W.L. BRAGG H.A. KRAMERS P.A.M. DIRAC A.H. COMPTON L. de BROGLIE M. BORN N. BOHR

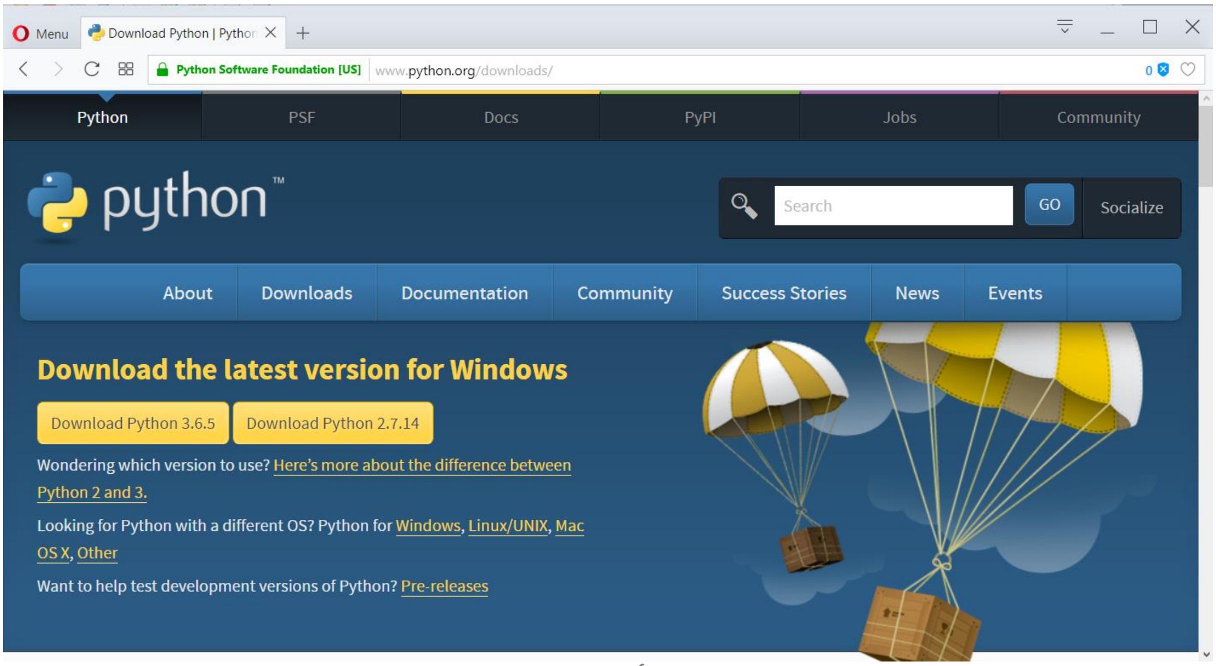
I. LANGMUIR M. PLANCK Mme CURIE H.A.LORENTZ A. EINSTEIN P. LANGEVIN Ch.E. GUYE C.T.R. WILSON O.W. RICHARDSON

Absents : Sir W.H. BRAGG, H. DESLANDRES et E. VAN AUBEL

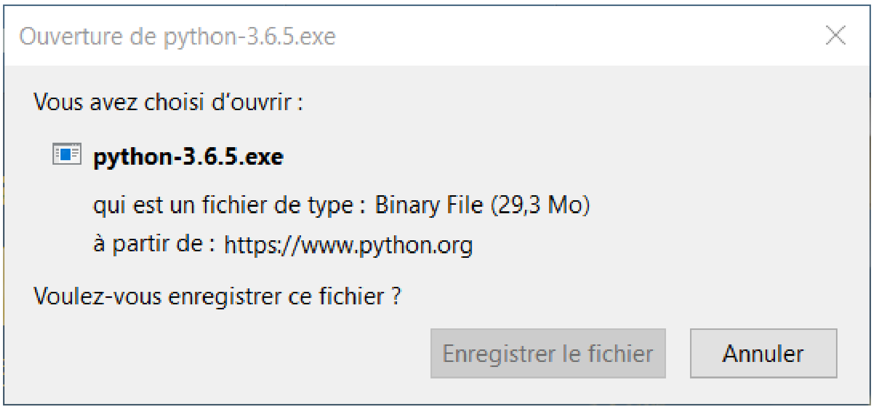
Exemple 371.: TensorFlow

R 3.4.3

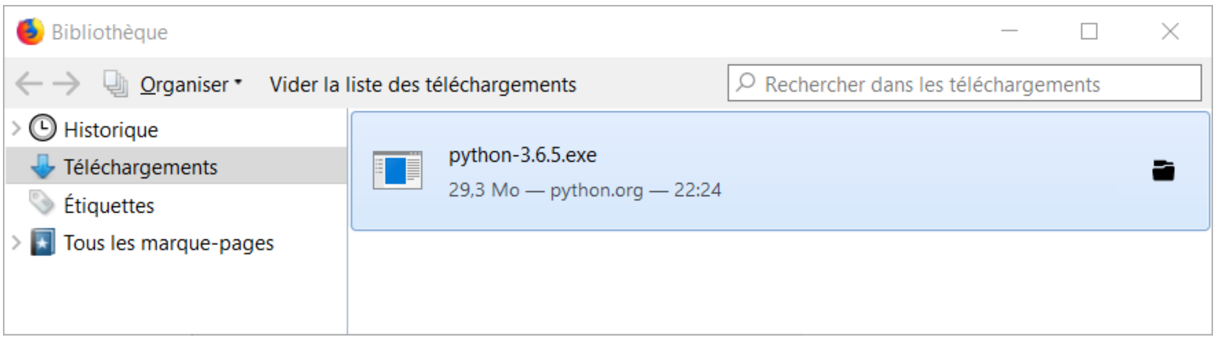
Pour utiliser TensorFlow de Google, nous allons devoir d'abord installer Python. Pour cela nous ouvrons notre navigateur pour aller à l'adresse suivante:



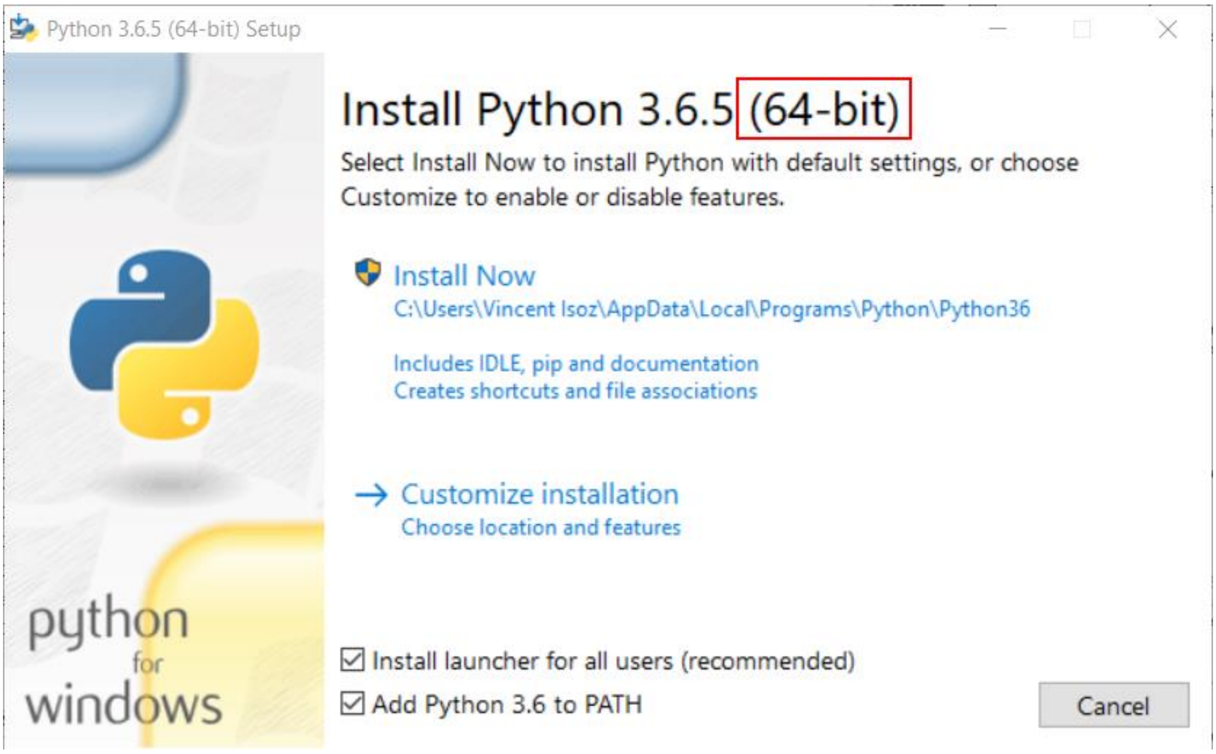
Nous cliquons sur le bouton Download Python 3.6.5:



Nous cliquons sur **Enregistrer le fichier**:

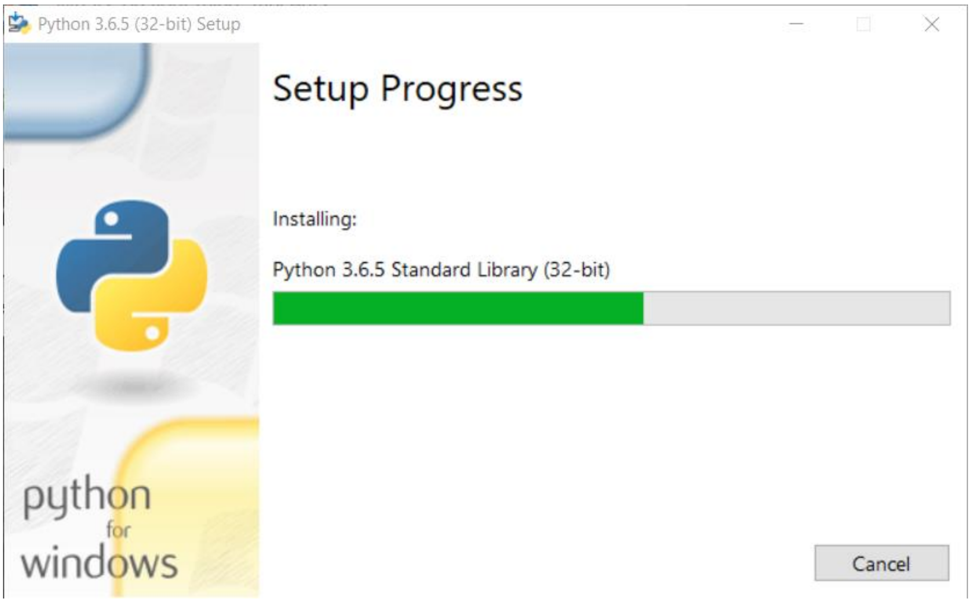


Une fois le téléchargement terminé, nous lançons l'exécutable:

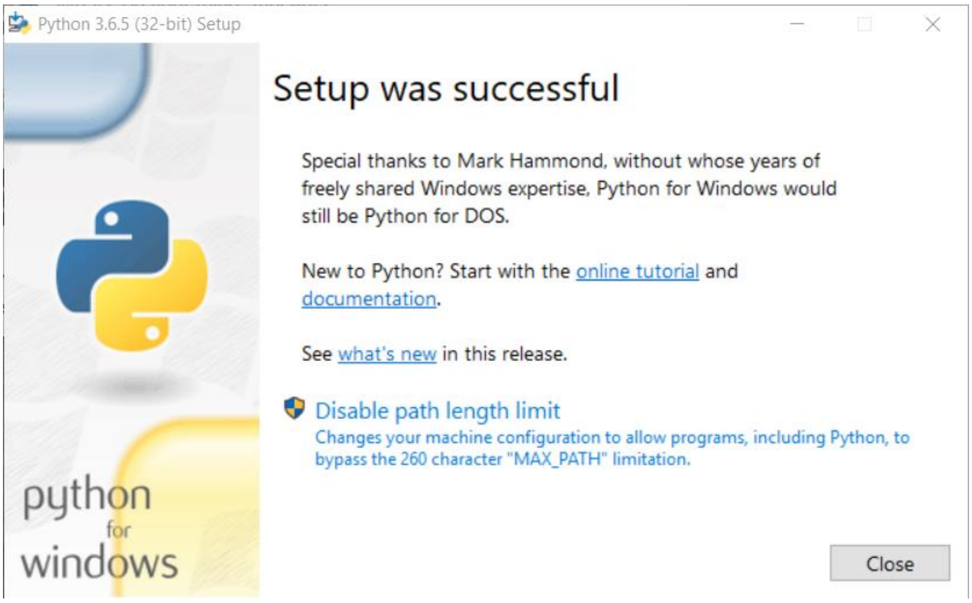


Attention à bien installer la version 64 bit pour un système x64 et la 32bit pour un système x86!!!!

Nous validons par **Install Now**:

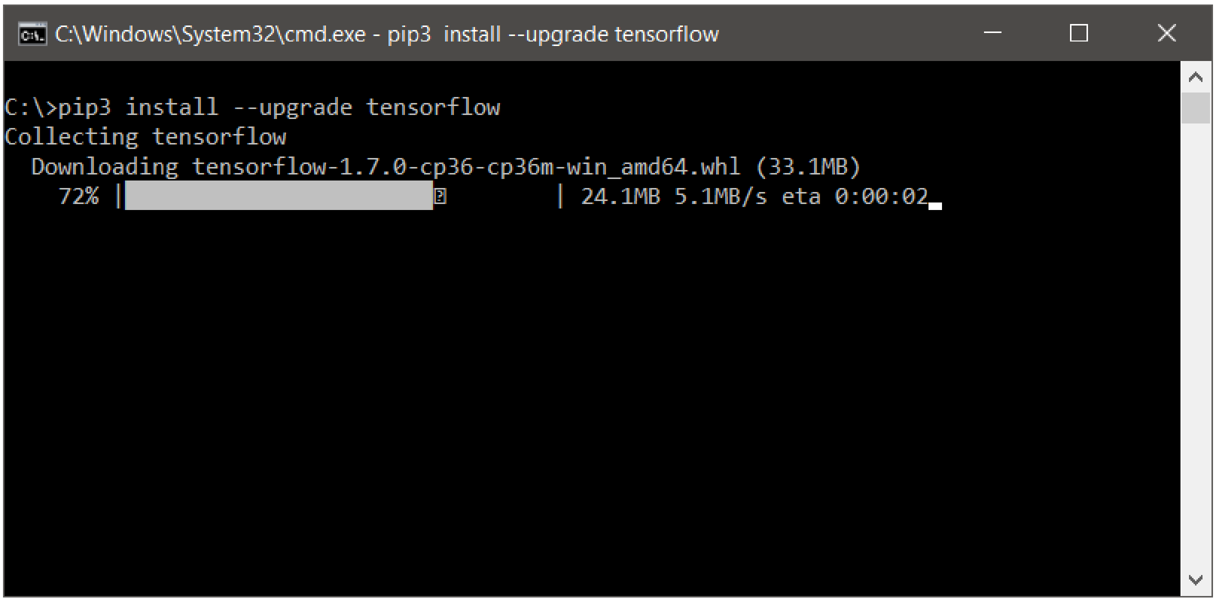


Une fois l'installation terminée:

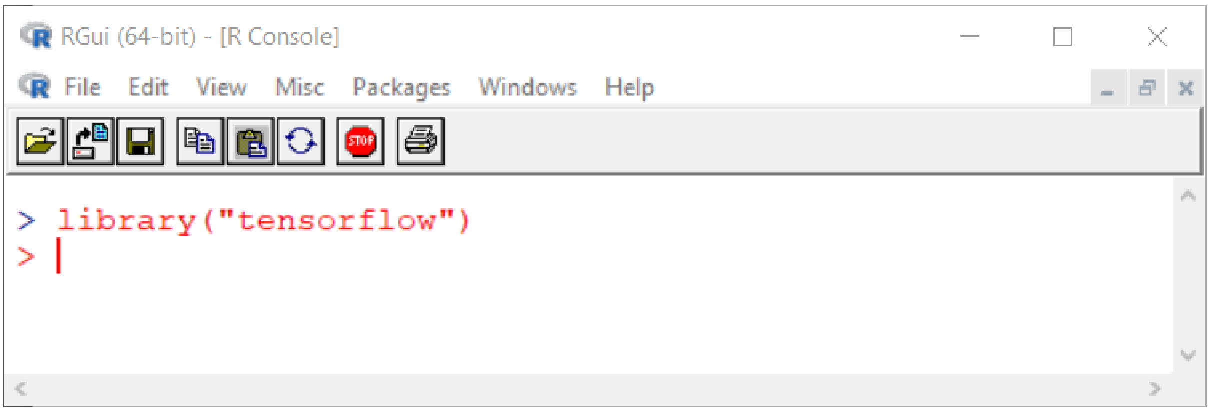


Nous vous conseillons de redémarrer l'ordinateur avant de poursuivre!

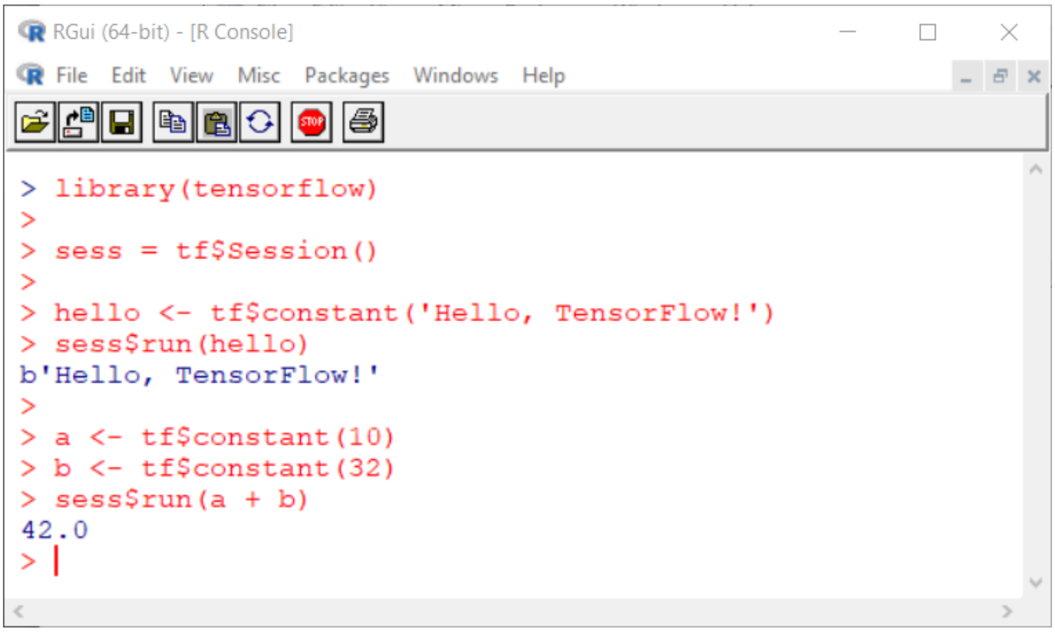
Une fois ceci fait, aux dernières nouvelles…, et dans le shell MS-DOS de Windows, tapez:



Ensuite, nous pouvons installer tensorflow dans R et le lancer sans messages d'erreur:



Et on texte avec un exemple simpliste repris du site officiel:



Et pour la suite, reportez-vous à l'adresse suivante:

https://tensorflow.rstudio.com/learn/examples.html

Personnellement (subjectivement), je trouve ce langage immonde donc je ne détaillerai pas plus!

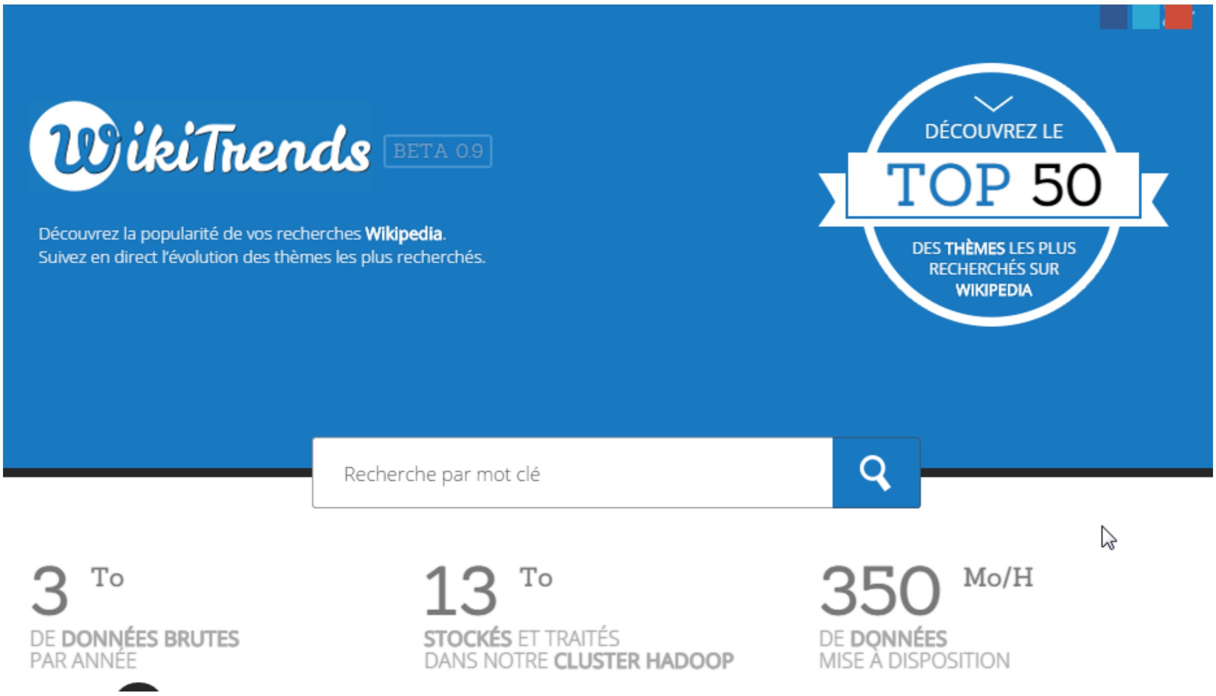
Exemple 372.: Hadoop (Cloudera)/R Studio Server

R 3.2.0

Lorsque l'on traite des Téraoctets de données il devient difficile de travail avec un unique serveur. Une solution consiste alors à utiliser un cluster et pourquoi ne pas utiliser celui d'Apache: Hadoop. Wikitrends est un exemple d'utilisation de Hadoop analysant en temps réel plus de 13 Téraoctets de données. La page officielle est ici:

http://www.wikitrends.eu/#/

Elle nous donne les informations suivantes en ce début 2015:



Le reste viendra…

25. Séries temporelles (séries chronologiques)

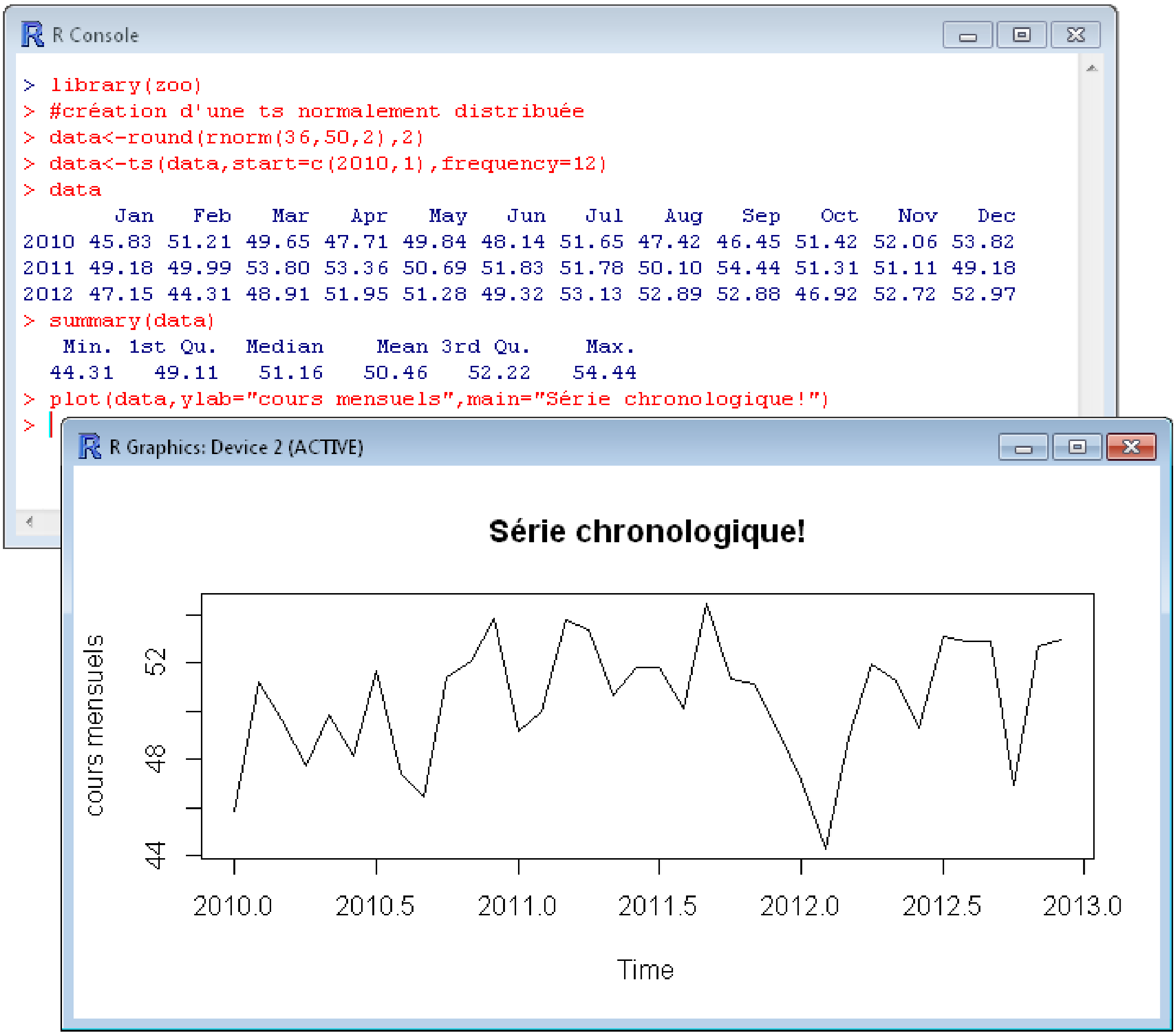
Il s'agit ici de présenter l'ensemble des techniques empiriques d'analyse de séries chronologiques et de projections dont les détails mathématiques ont été vus dans le cours théorique.

Dans la pratique n'oubliez pas de faire du back-testing!!!

Exemple 373.: Génération d'une série temporelle à partir de données brutes

R 3.0.2

On utilise la commande **ts( )** du package **zoo** pour générer rapidement une série de dates sur une base périodique donnée (frequency=12 donnera des mois, frequency=4 donnera des trimestres et ainsi de suite):

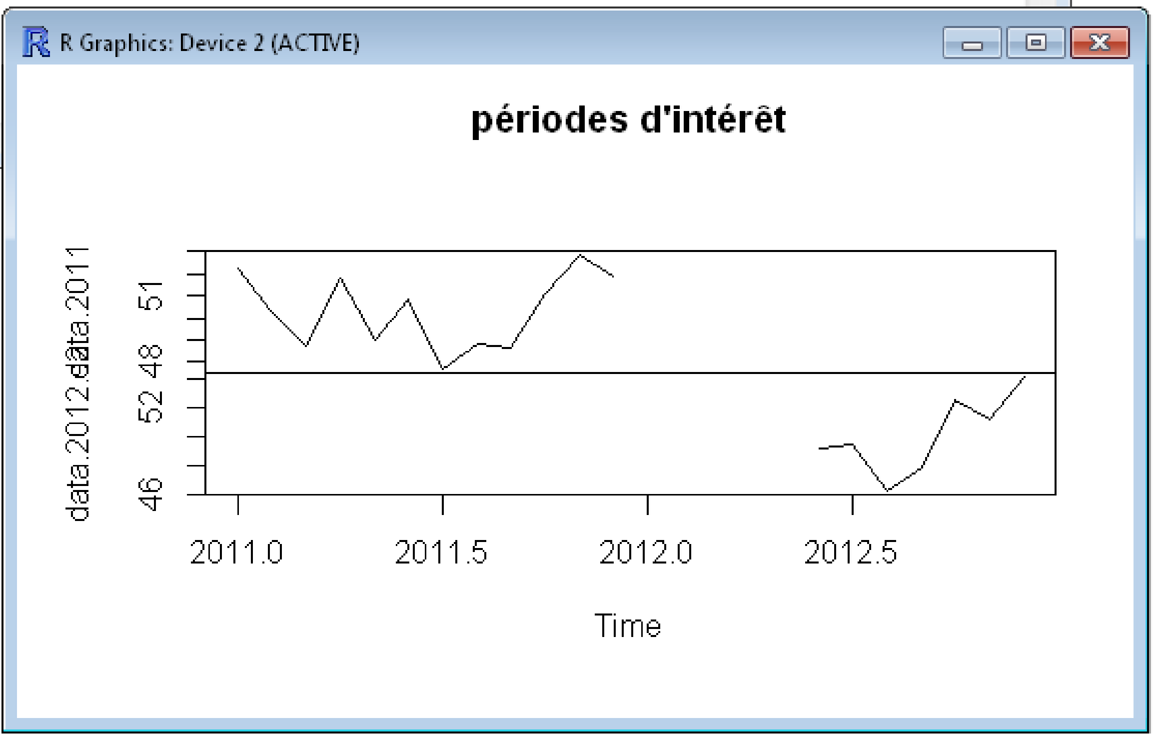


Exemple 374.: Extraire des sous-ensembles de séries temporelles

R 3.0.2

Il est possible d'extraire des sous-parties en spécifiant la date de début et de fin de la période de la série qui nous intéresse avec la commande **window( )** du package **zoo**:

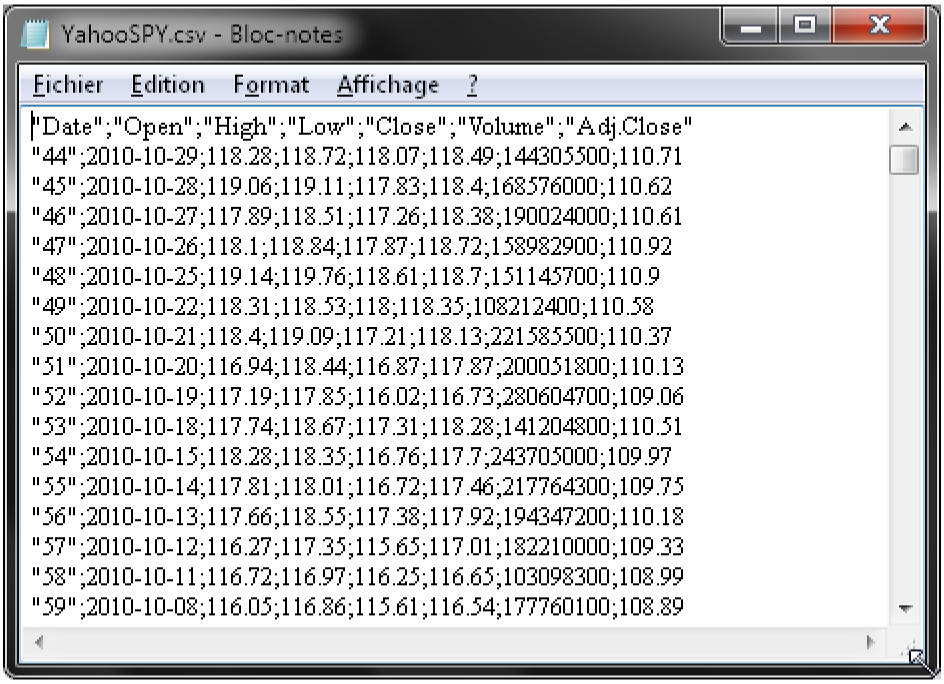
```
> library(zoo)
> data<-round(rnorm(36,50,2),2)
> data<-ts(data,start=c(2010,1),frequency=12)
> #extraire une année entière
> data.2011<-window(data,start=c(2011,1),end=c(2011,12))
> #extraire un semestre
> data.2012.s2<-window(data,start=c(2012,6),end=c(2012,12))
> plot(cbind(data.2011,data.2012.s2),main="périodes d'intérêt")
>
```



Exemple 375.: Lire des données temporelles

R 3.0.2

Ce qui va suivre ne devrait normalement pas nécessiter d'explications théoriques. Il va s'agir avec la package zoo et la commande **read.zoo( )** de traiter un fichier *.csv qui suppose que la première colonne est des dates:




```
"Date";"Open";"High";"Low";"Close";"Volume";"Adj.Close"
"44";2010-10-29;118.28;118.72;118.07;118.49;144305500;110.71
"45";2010-10-28;119.06;119.11;117.83;118.4;168576000;110.62
"46";2010-10-27;117.89;118.51;117.26;118.38;190024000;110.61
"47";2010-10-26;118.1;118.84;117.87;118.72;158982900;110.92
"48";2010-10-25;119.14;119.76;118.61;118.7;151145700;110.9
"49";2010-10-22;118.31;118.53;118;118.35;108212400;110.58
"50";2010-10-21;118.4;119.09;117.21;118.13;221585500;110.37
"51";2010-10-20;116.94;118.44;116.87;117.87;200051800;110.13
"52";2010-10-19;117.19;117.85;116.02;116.73;280604700;109.06
"53";2010-10-18;117.74;118.67;117.31;118.28;141204800;110.51
"54";2010-10-15;118.28;118.35;116.76;117.7;243705000;109.97
"55";2010-10-14;117.81;118.01;116.72;117.46;217764300;109.75
"56";2010-10-13;117.66;118.55;117.38;117.92;194347200;110.18
"57";2010-10-12;116.27;117.35;115.65;117.01;182210000;109.33
"58";2010-10-11;116.72;116.97;116.25;116.65;103098300;108.99
"59";2010-10-08;116.05;116.86;115.61;116.54;177760100;108.89
```

Nous avons alors:

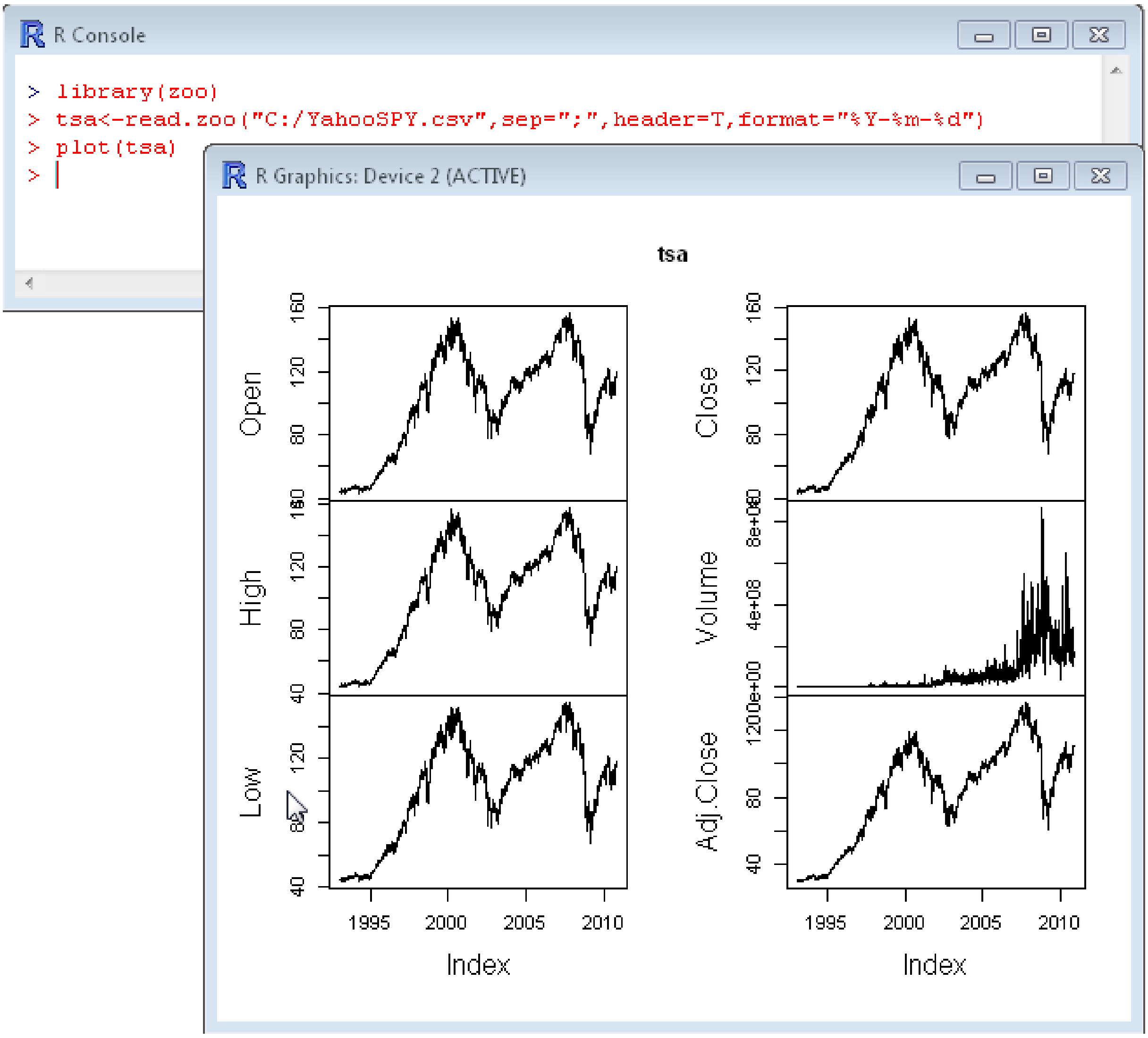
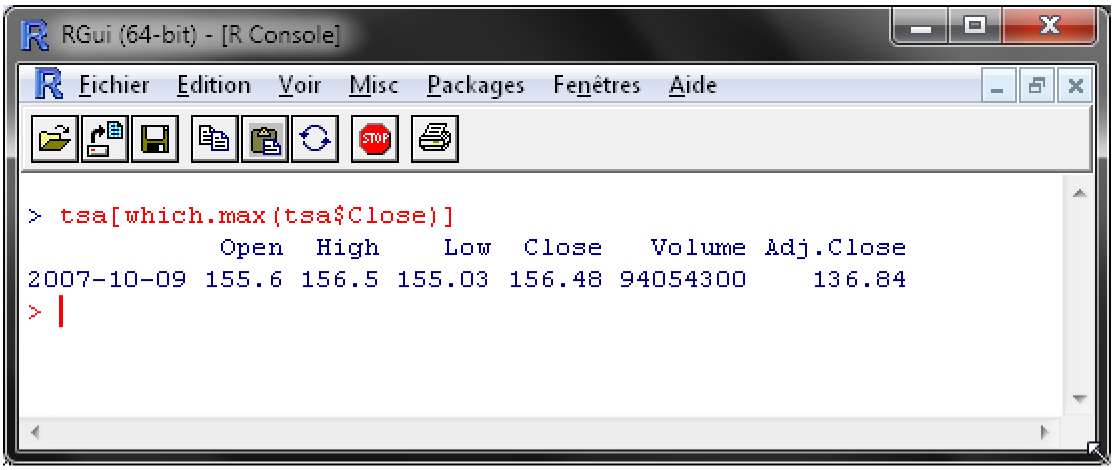
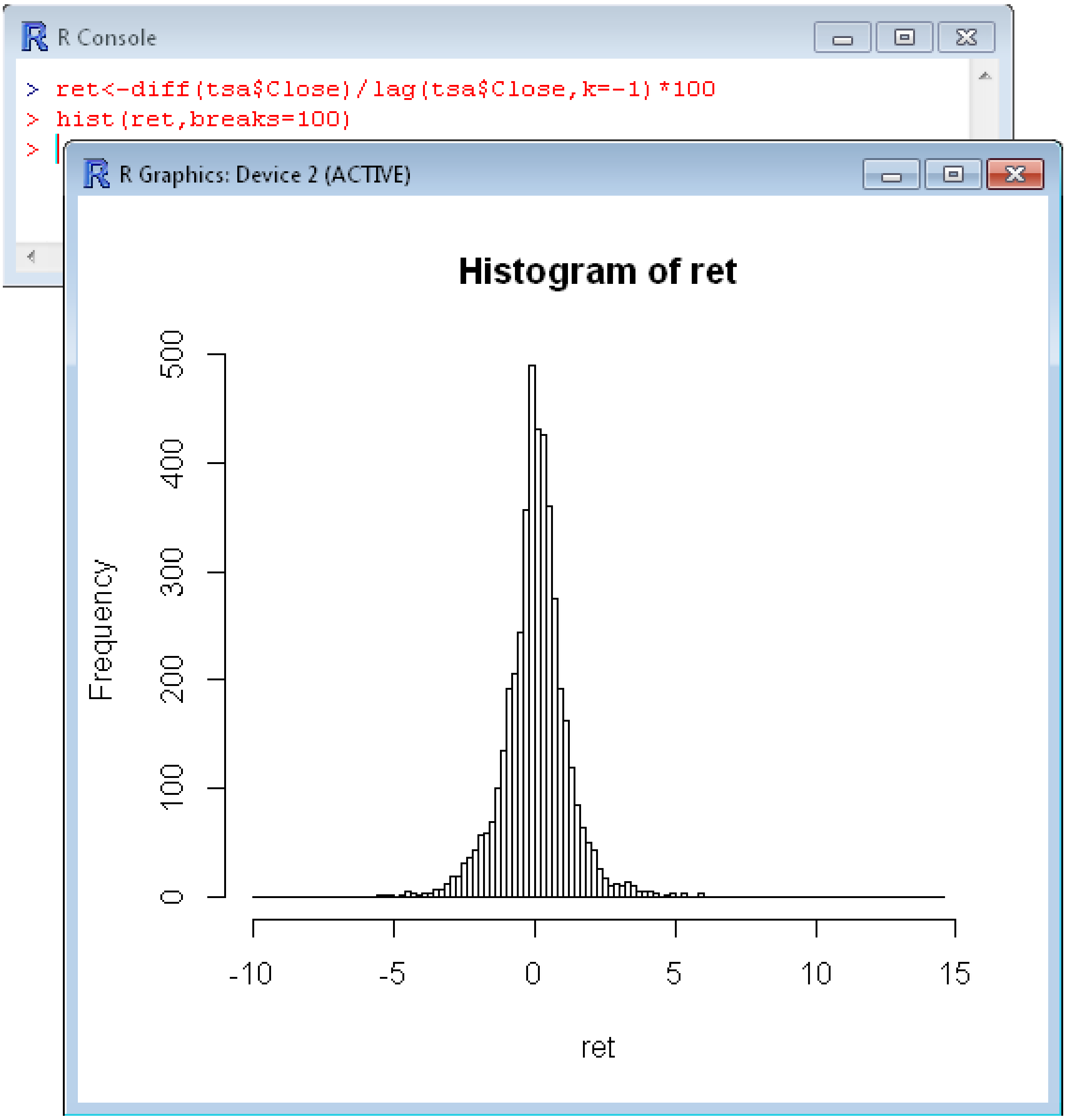


Figure 307 Graph de séries temporelle de variables multiples

On peut s'amuser à obtenir le jour où le prix de l'action a été le plus élevé avec les techniques de filtre habituelles de R:



Ensuite, nous pouvons mesurer et afficher la répartition statistique du rendement:



Nous pouvons rechercher le maximum dans une certaine fenêtre de temps et calculer basiquement la VaR historique:

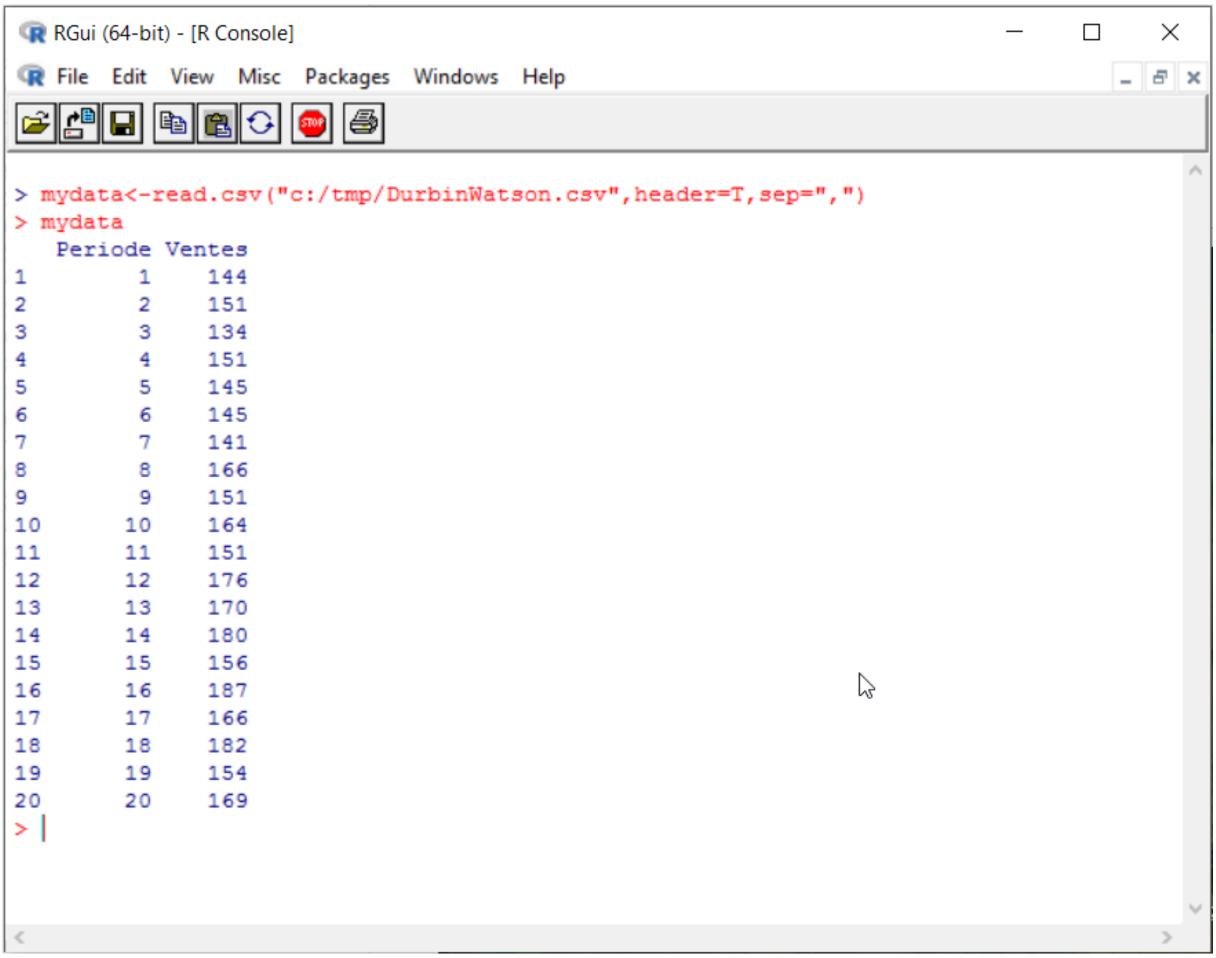
```
> ret_2010<-window(ret,start='2010-01-01',end='2010-12-31')
> ret_2010[which.max(ret_2010)]
2010-05-10
  4.404099
> #VaR historique
> quantile(ret_2010,probs=0.01)
       1%
-3.302642
>
```

Exemple 376.: Test de Durbin-Watson

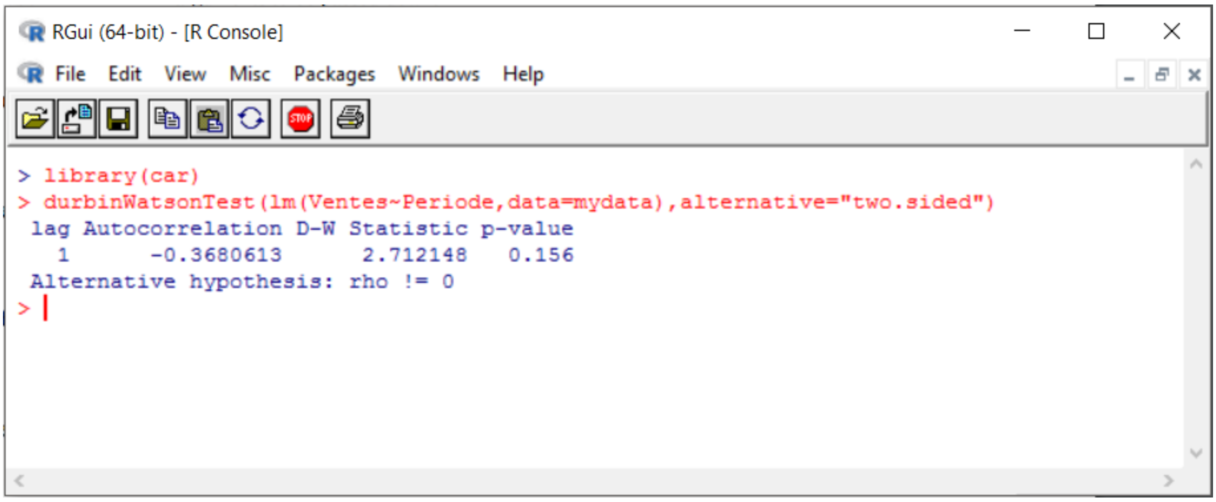
R 3.3.2

Nous avons introduit dans le cours théorique la statistique de Durbin-Watson. Nous n'avons pas fait de calculs à la main ou dans le tableur Microsoft Excel car c'est vraiment trop laborieux. Notre seul propose ici va être de voir comment Minitab affiche la statistique et comparer cette dernière à R.

Nous chargeons pour cela le fichier *DurbinWatson.csv*:

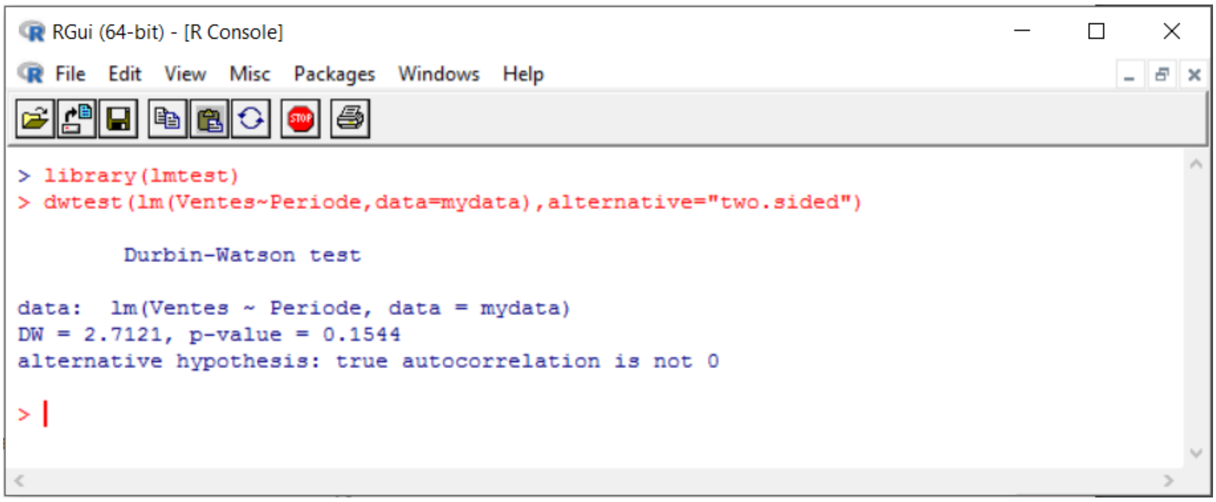


Nous utilisons ensuite la fonction **durbinWatsonTest( )** du package **car**:



Donc nous avons la même valeur DW que Minitab à la différence que nous avons beaucoup plus d'informations utiles et que nous savons à quoi nous en tenir! De plus nous avons de nombreux paramètres de calculs avec R qui n'existent pas avec Minitab!!!

Nous pouvons aussi comparer avec la fonction **dwtest( )** du package **lmtest**:



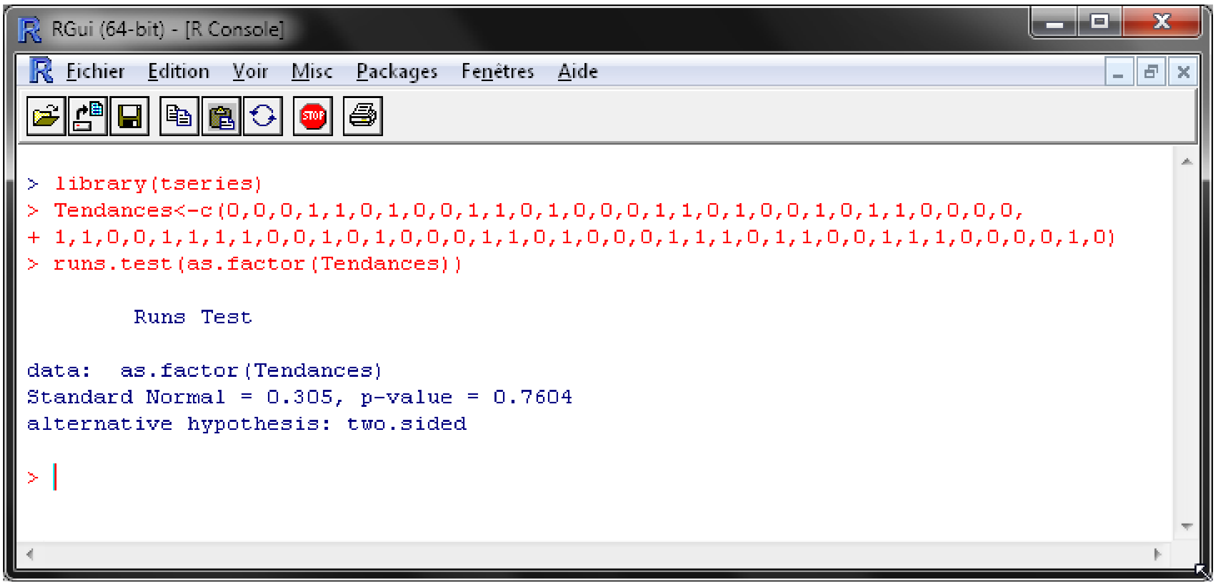
Exemple 377.: Test des suites (test de Walf-Wolfowitz)

R 3.0.2

Comme à l'habitude le but va être de vérifier que le logiciel utilise les résultats démontrés dans le cours théorique avec un exemple implicitement lié à une série temporelle (les "0" étant des "baisses" et les "1" étant des valeurs "haussières"). La question étant de savoir si les séquences peuvent être considérées comme aléatoires ou non statistiquement parlant.

Nous devrions aussi retrouver les calculs faits à la main et à l'aide de MS Excel.

Nous allons utiliser la commande **runs.test( )** du package **tseries**:



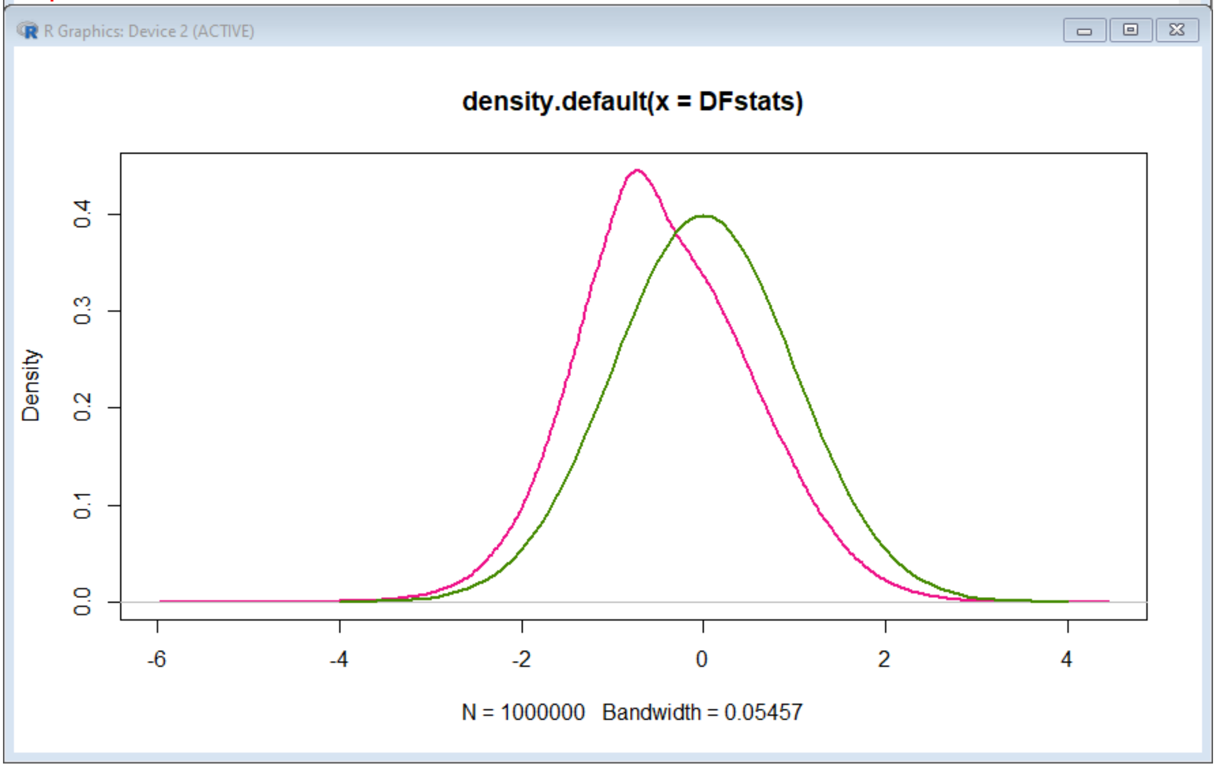
Bien que nous retrouvions les calculs faits dans le cours théorique, il est dommage que R ne renvoie pas plus d'informations... Je préfère Minitab à ce niveau là et ce même si la conclusion reste la même!

Exemple 378.: Test simple de Dickey-Fuller

R 3.6.0

Dans le cours théorique nous avons démontré non sans mal la loi asymptotique de la distribution du test simple de DF pour le cas sans constante et sans drift. Avant de voir comment utiliser ce dernier, voyons comment en reproduire les valeurs critiques grâce au code par simulation de Monte Carlo fournit par le professeur Hanck Christoph:

```
> T <- 20
> reps <- 1000000
>
> DFstats <- rep(NA,reps)
>
> for (i in 1:reps){
+   u <- rnorm(T)
+   W <- 1/sqrt(T)*cumsum(u)
+   DFstats[i] <- (W[T]^2-1)/(2*sqrt(mean(W^2)))
+ }
>
> (CriticalValues <- sort(DFstats)[c(0.01,0.05,0.1)*reps])
[1] -2.582982 -1.926346 -1.600071
>
> #plot et comparaison avec la loi normale
> plot(density(DFstats),lwd=2,col=c("deeppink2"))
> xax <- seq(-4,4,by=.1)
> lines(xax,dnorm(xax),lwd=2,col=c("chartreuse4"))
>
```

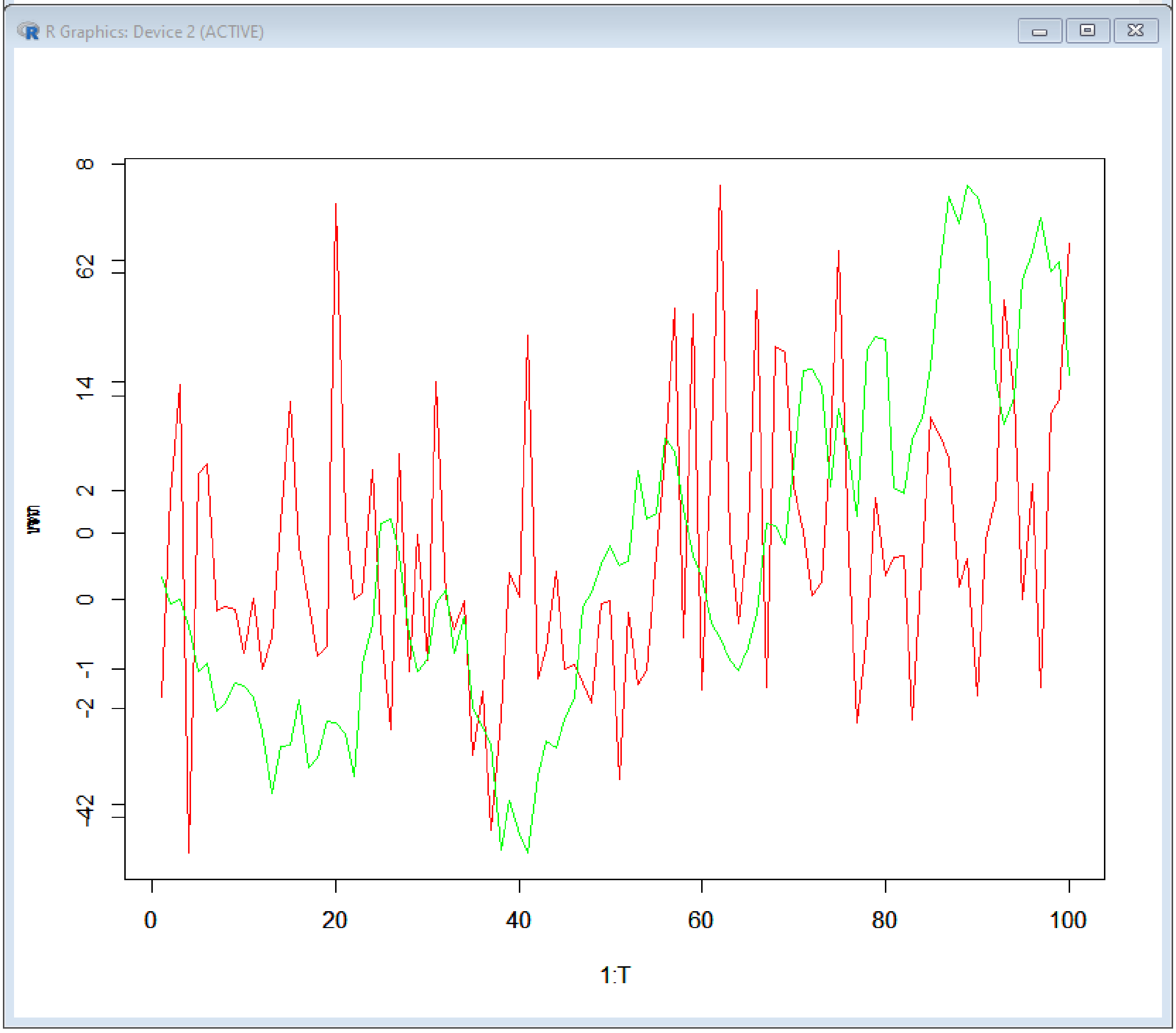


On retombe bien donc sur les valeurs données dans les tables!

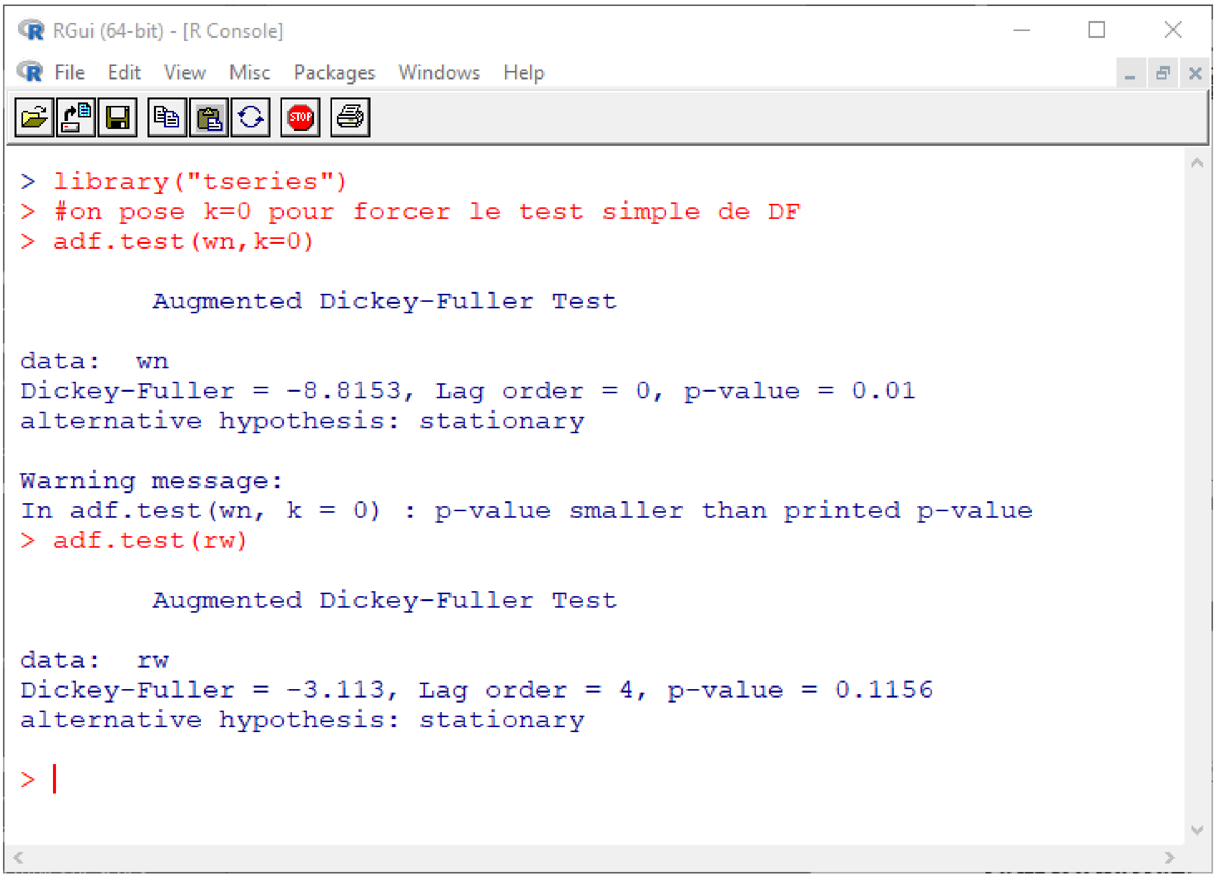
Maintenant pour la suite, construisons deux processus stochastiques de type bruit blanc, un stationnaire et l'autre qui ne l'est pas:

R Console

```
> set.seed(1234)
> T <- 100
> #processus de Wiener stationnaire
> wn <- rnorm(T)
> #processu de Wiener non-stationnaire
> rw <- cumsum(rnorm(T))
> plot(1:T, wn , type = "l", col="red")
> par(new=TRUE)
> plot(1:T, rw , type = "l", col="green")
>
```



Maintenant voyons d'abord le test simple de Dickey-Fuller avec le package **tseries** et sa fonction **adf.test( )**:



Et avec le package **urca** et sa fonction **ur.df( )** d'abord sur le bruit blanc stationnaire:

```
> summary(ur.df(wn, type="none", lags=0))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5183 -0.8053 -0.3356  0.4996  2.5490

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1  -0.8409     0.1013    -8.3 5.77e-13

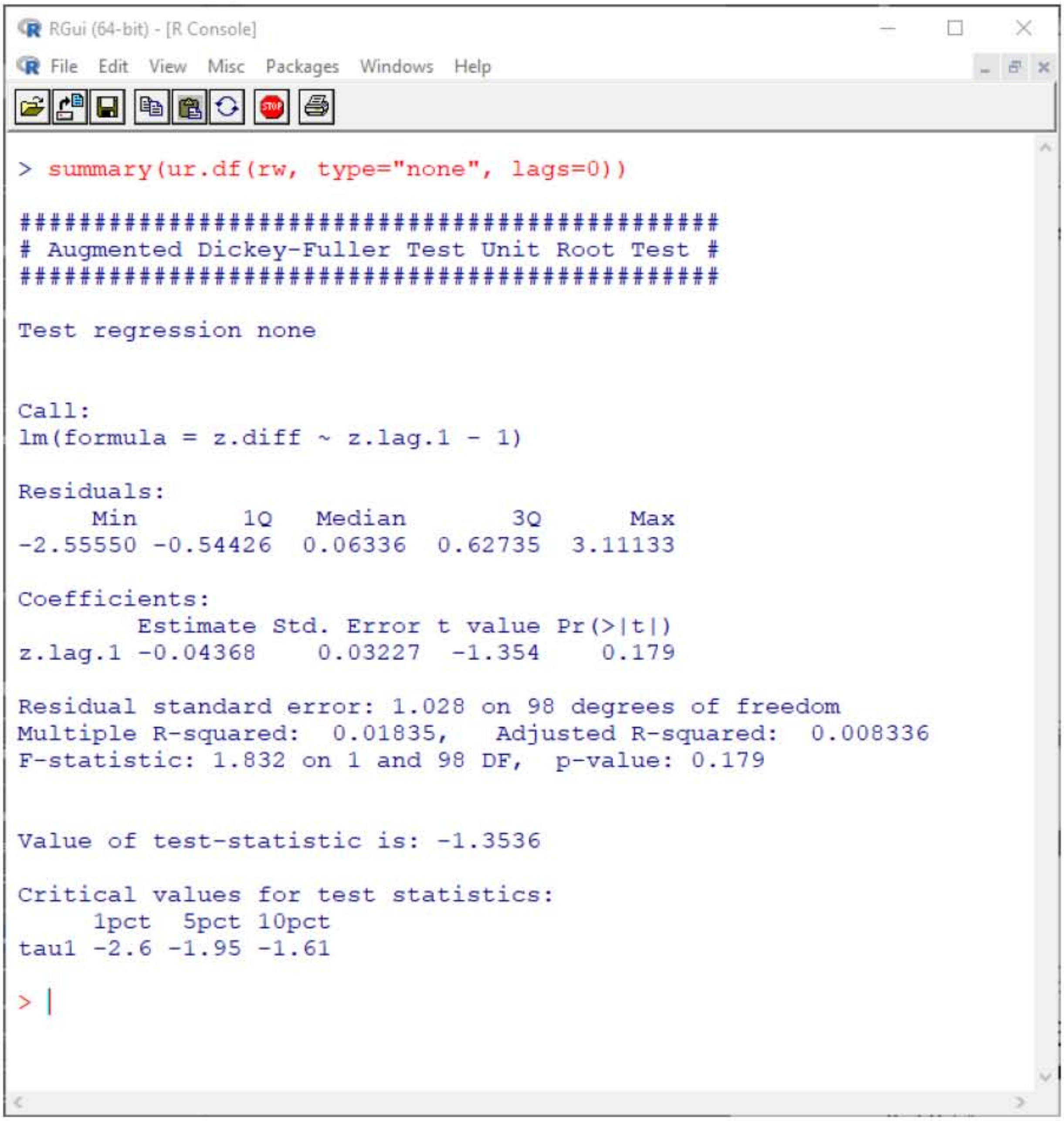
Residual standard error: 1.002 on 98 degrees of freedom
Multiple R-squared:  0.4128,    Adjusted R-squared:  0.4068
F-statistic:  68.9 on 1 and 98 DF,  p-value: 5.775e-13


Value of test-statistic is: -8.3004

Critical values for test statistics:
     1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61

>
```

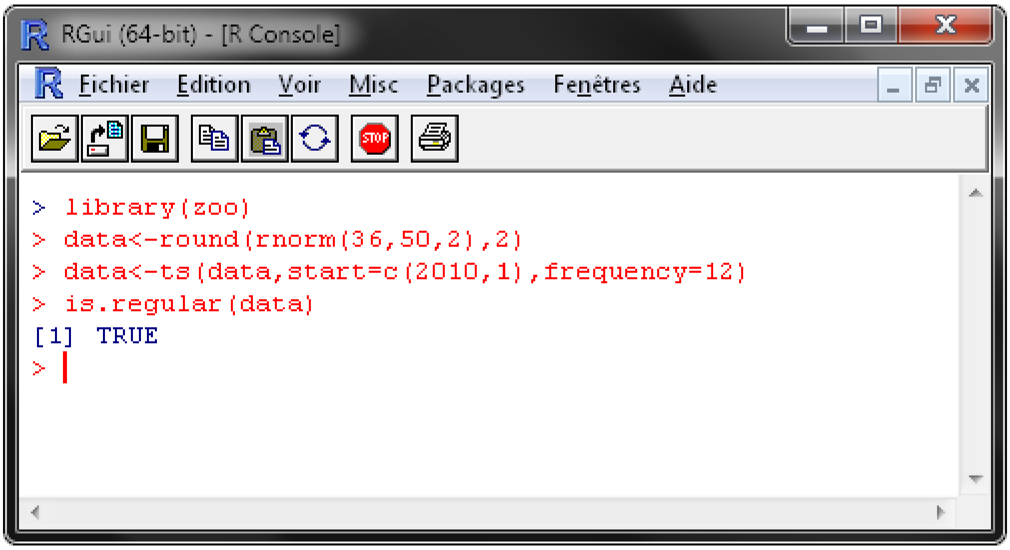
Et sur le bruit non-stationnaire:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> summary(ur.df(rw, type="none", lags=0))
###
Augmented Dickey-Fuller Test Unit Root Test
###
Test regression none
Call:
lm(formula = z.diff ~ z.lag.1 - 1)
Residuals:
Min 1Q Median 3Q Max
-2.55550 -0.54426 0.06336 0.62735 3.11133
Coefficients:
Estimate Std. Error t value Pr(>|t|)
z.lag.1 -0.04368 0.03227 -1.354 0.179
Residual standard error: 1.028 on 98 degrees of freedom
Multiple R-squared: 0.01835, Adjusted R-squared: 0.008336
F-statistic: 1.832 on 1 and 98 DF, p-value: 0.179
Value of test-statistic is: -1.3536
Critical values for test statistics:
1pct 5pct 10pct
tau1 -2.6 -1.95 -1.61
>

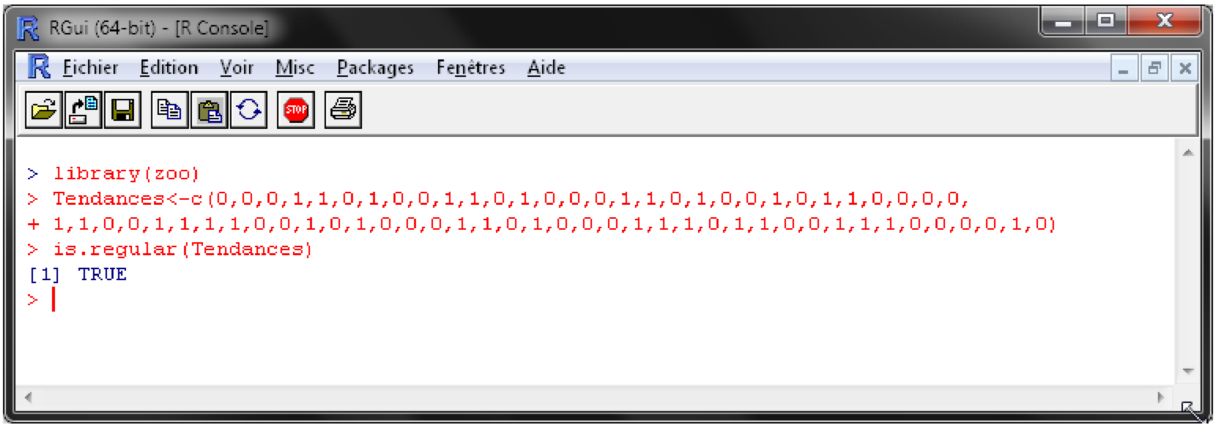
Exemple 379.: Contrôle de la stationnarité d'une série temporelle

R 3.0.2

Nous utilisons pour cela la commande **is.regular** du package **zoo**:



et aussi bien évidemment avec la suite de hausses/baisses:

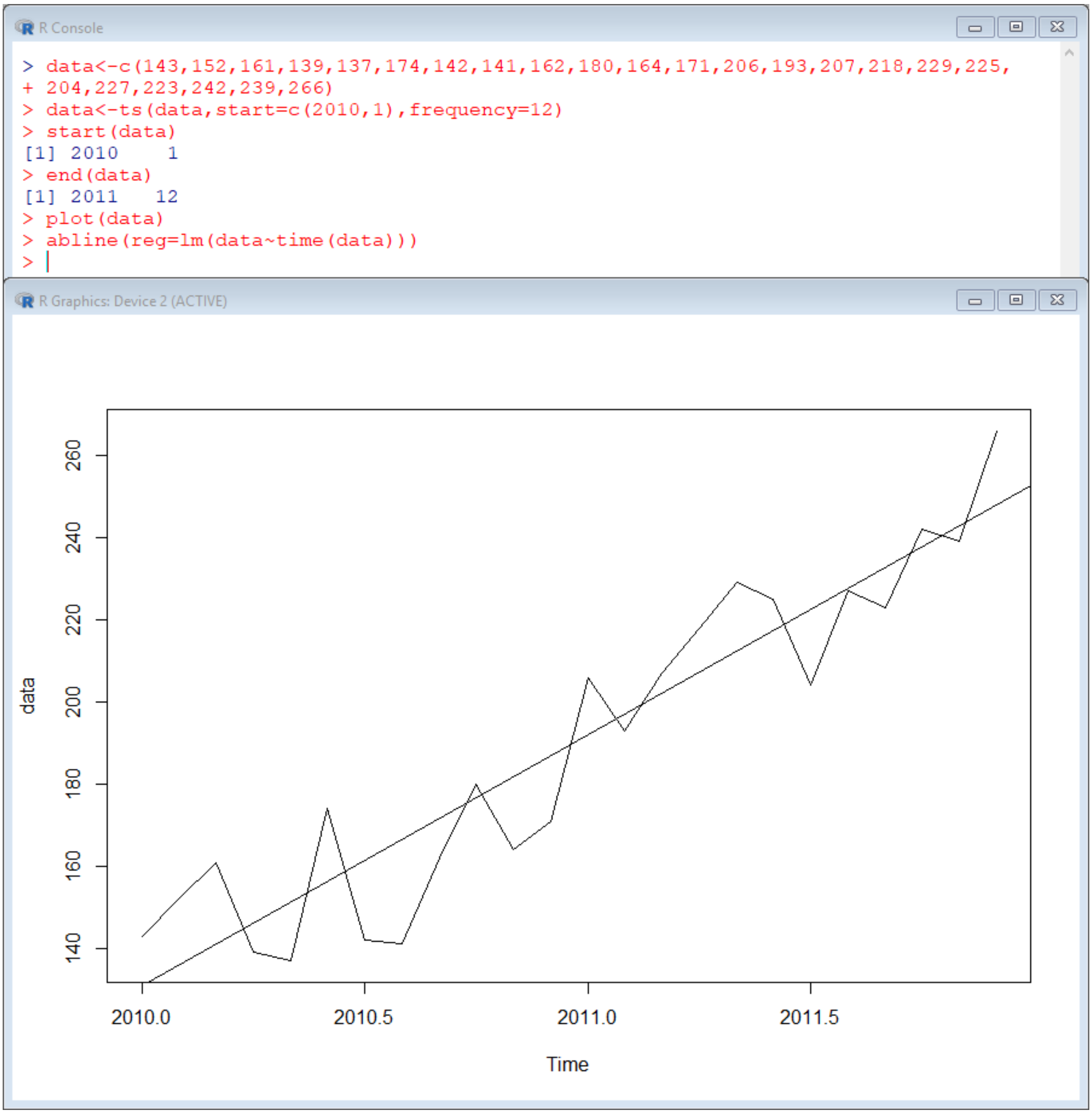


Exemple 380.: Régression linéaire avec dates

R 3.5.1

C'est une question classique de ceux qui travaillent avec Microsoft Excel. Comment faire une régression linéaire d'une série temporelle.

Eh bien (hormis l'aspect esthétique du graph déjà traité antérieurement avec les couleurs, affichage de la corrélation et de l'équation), il suffit d'écrire:



Exemples 381.: Modèles ETS (Error Trend Seasonality)

R 3.0.2

Décomposition d'une série temporelle par modèle additif et multiplicatif

Nous allons ici reprendre le même exemple que dans le cours théorique en utilisant la commande **decompose( )** du package **zoo**:

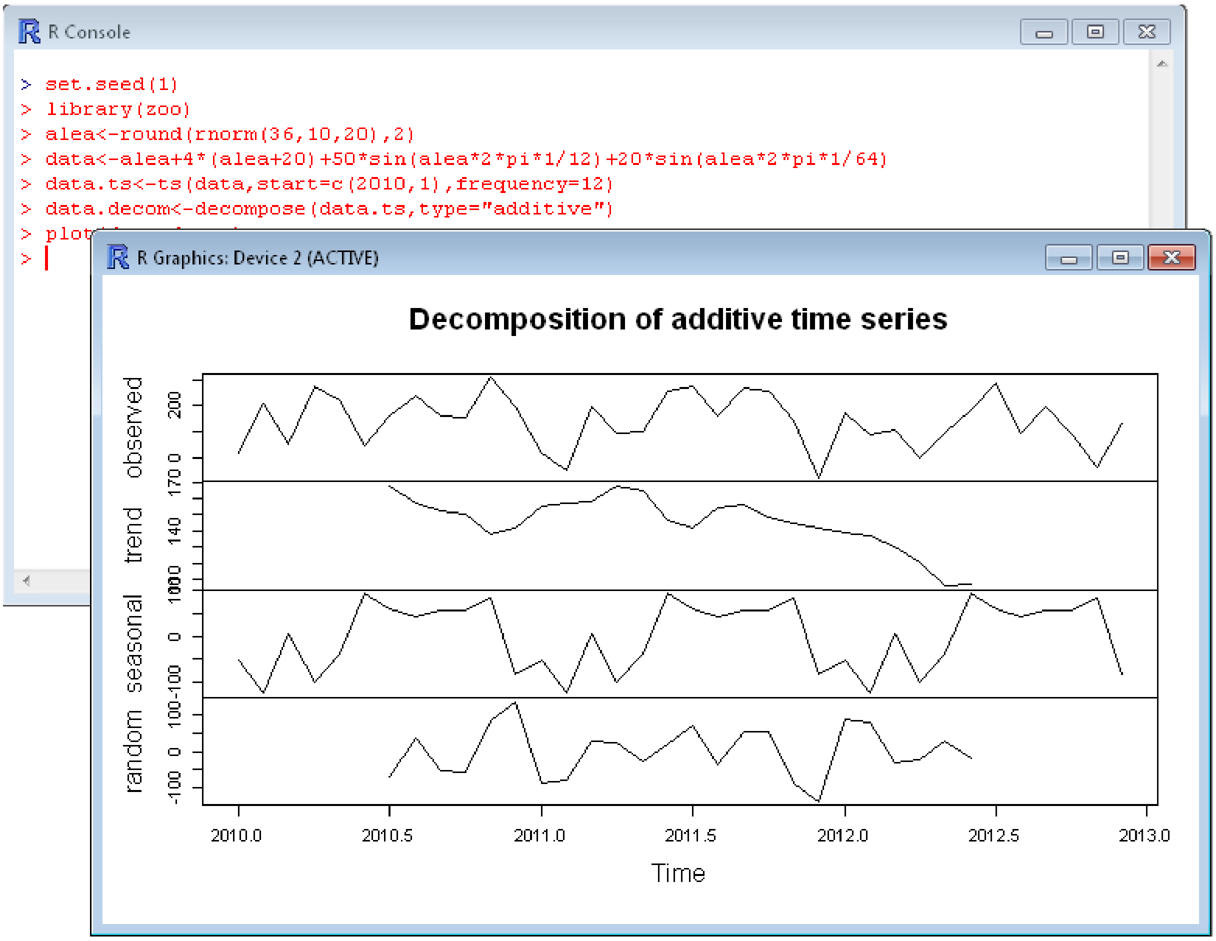
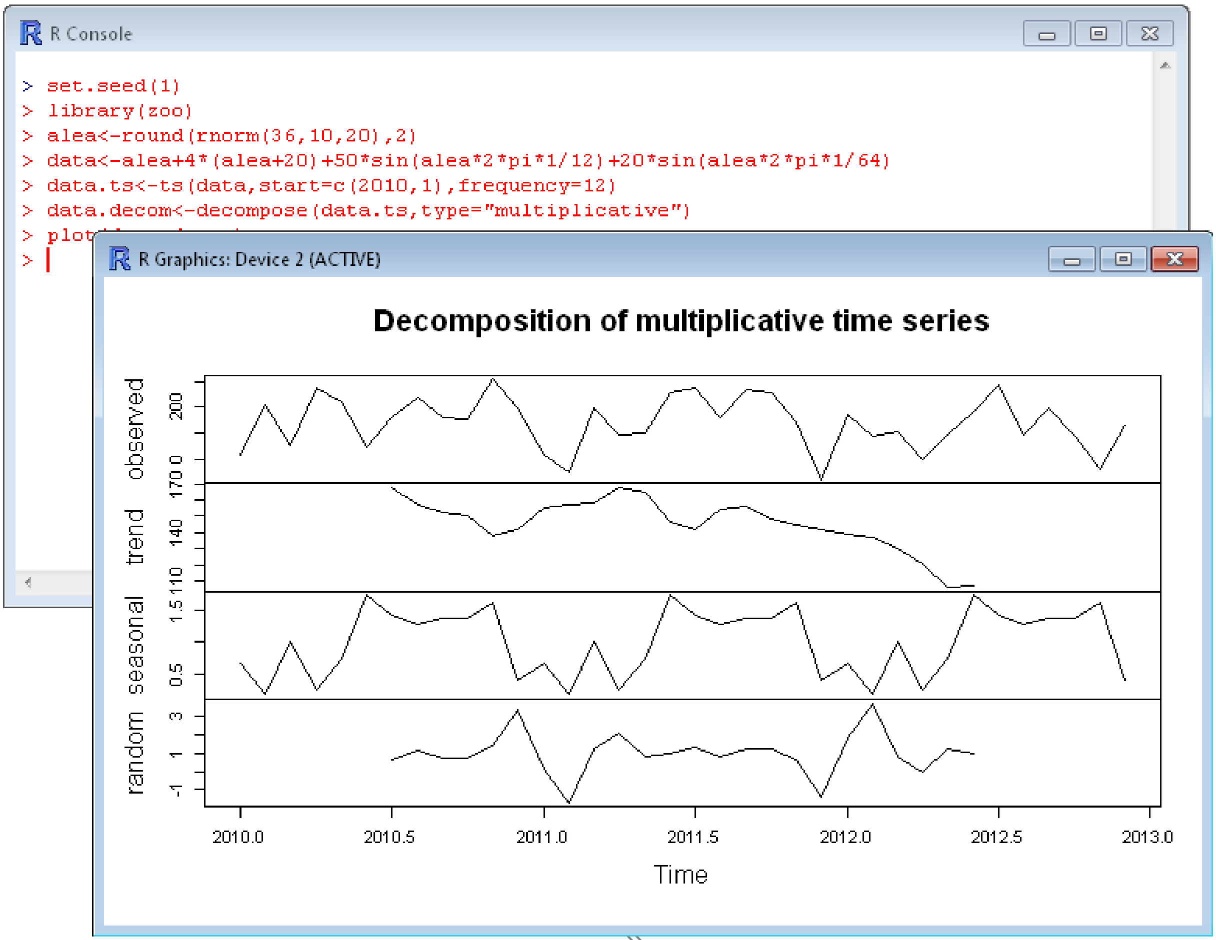
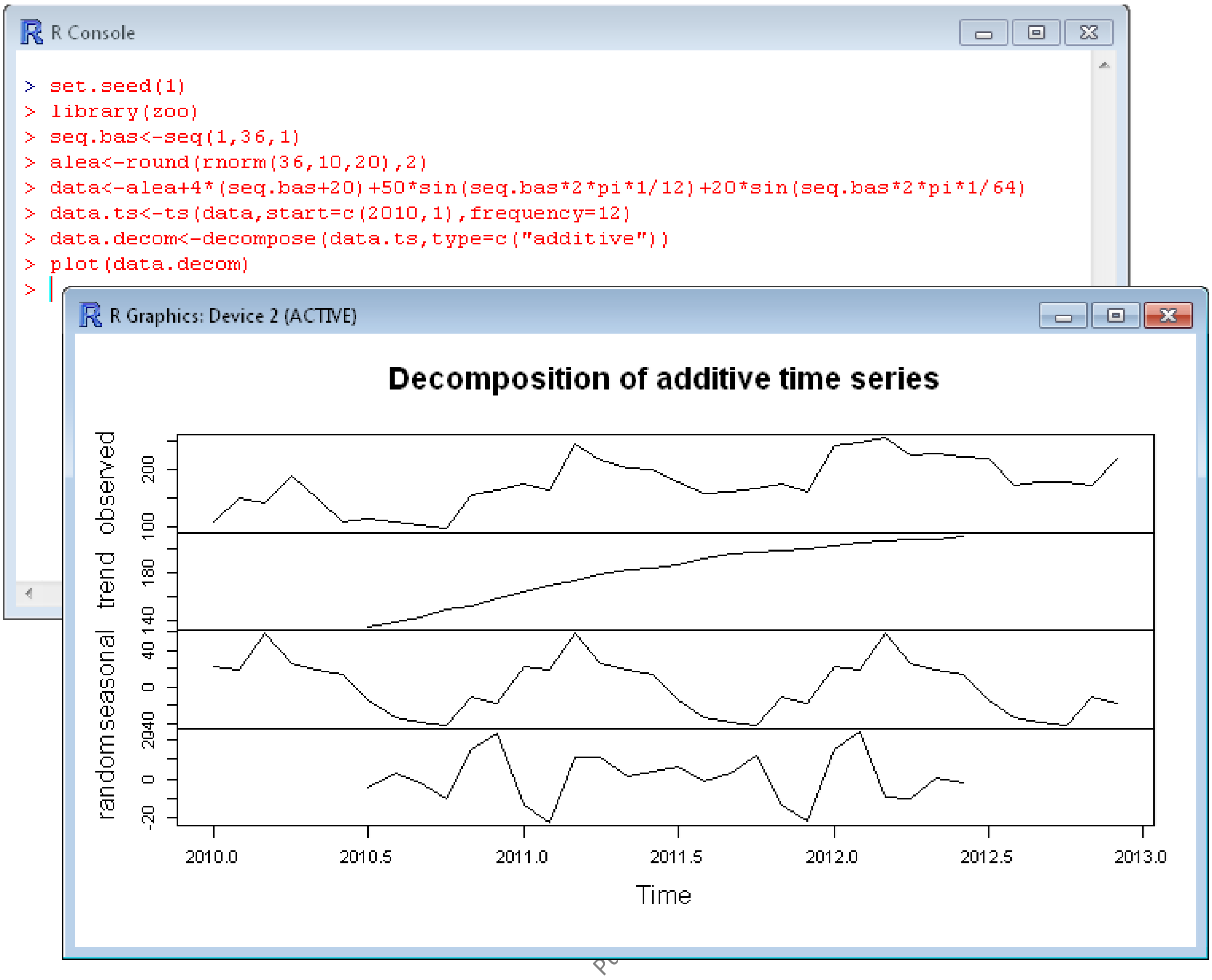


Figure 308 Graph de décomposition RSTO d'une série temporelle

Nous n'avons pas étudié dans le cours théorique comment décomposer une série avec un modèle multiplicatif mais voyons quand même comment faire (avec exactement les mêmes données):

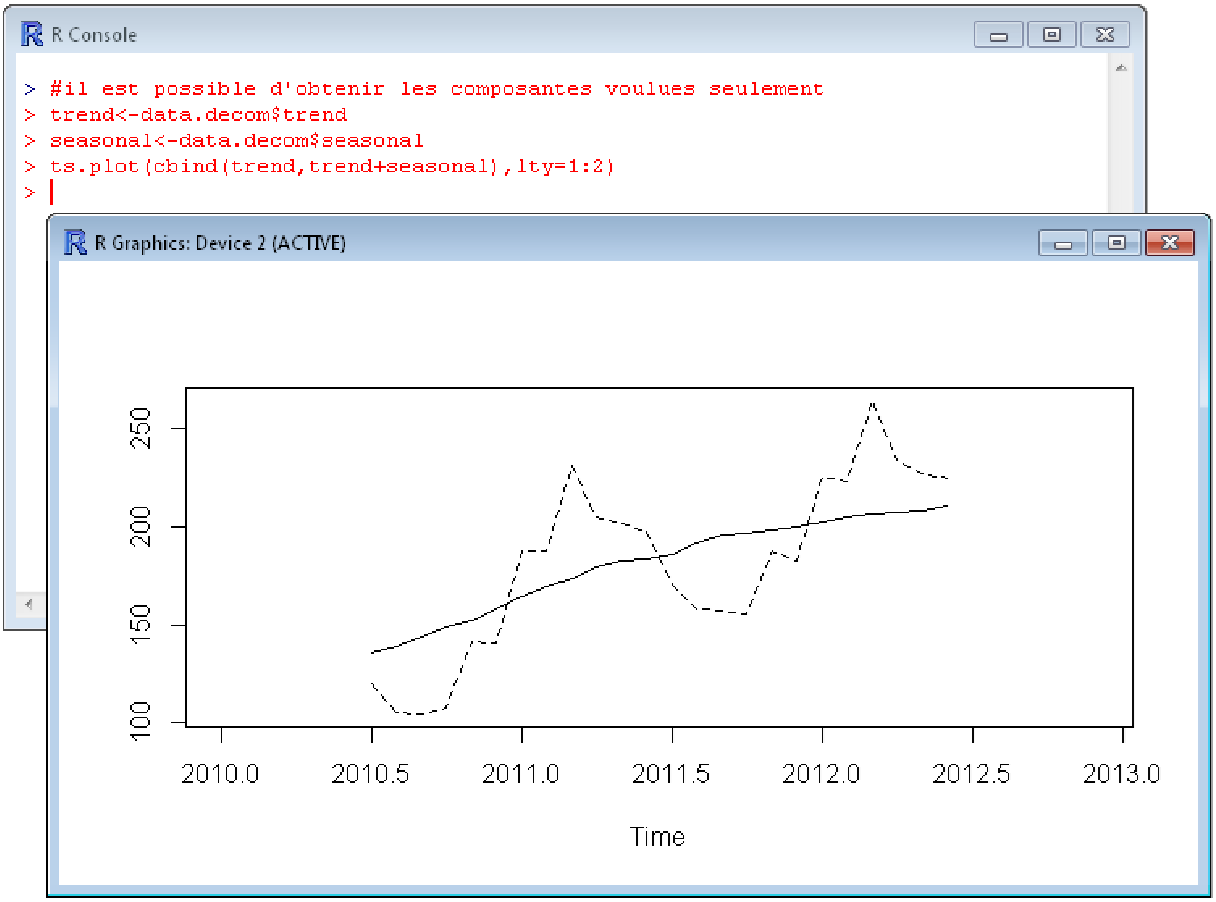


Faisons à nouveau la même chose mais avec un cas trivial pour voir comment les algorithmes de décomposition performent en additif:



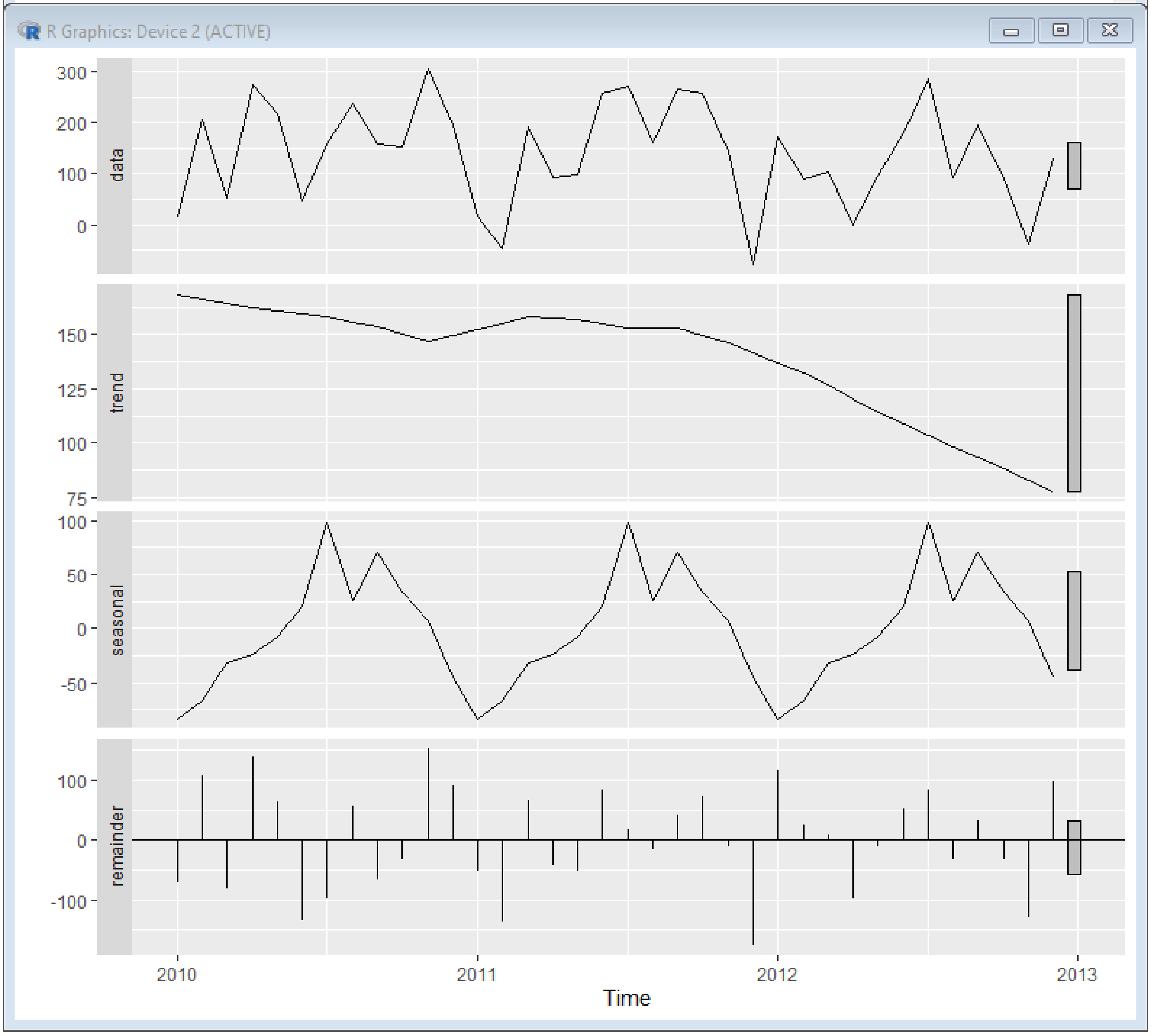
Comme quoi un modèle reste un modèle...

Avec la commande **ts.plot( )** du package **zoo**, nous pouvons plotter que les composantes et les mélanges qui nous intéressent:



Sinon avec le package **forecast** et les fonctions **autplot( )** et **stl( )** nous pouvons obtenir:

```
R Console
> set.seed(1)
> library("forecast")
> alea<-round(rnorm(36,10,20),2)
> data<-alea+4*(alea+20)+50*sin(alea*2*pi*1/12)+20*sin(alea*2*pi*1/64)
> data.ts<-ts(data,start=c(2010,1),frequency=12)
> autoplot(stl(data.ts,s.window="periodic"))
>
```



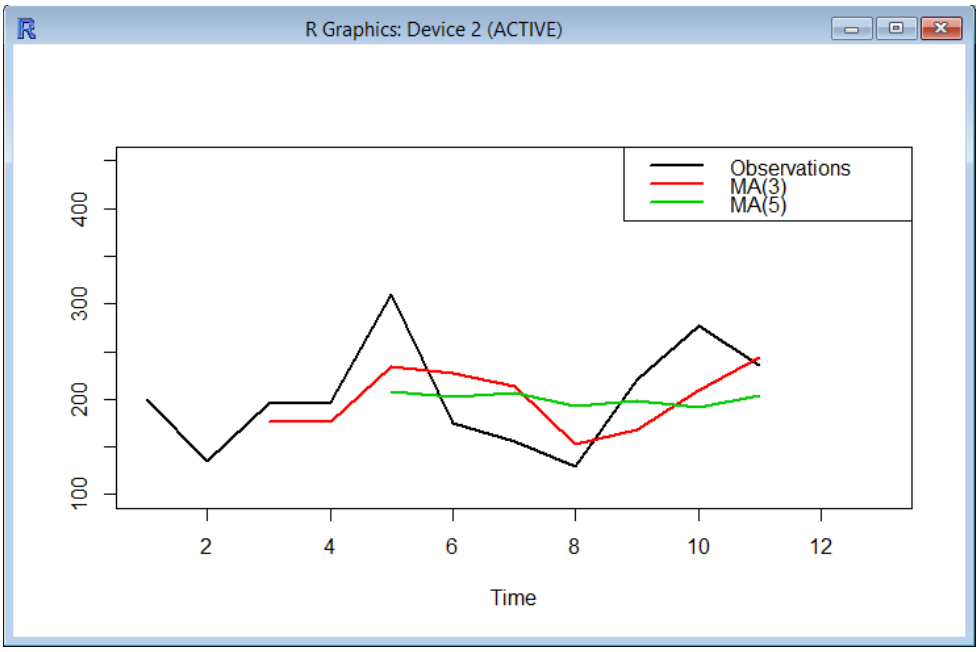
Lissage par moyenne mobile simple

Nous allons ici vérifier que nous obtenons bien les mêmes résultats que ceux obtenus dans le cours théorique et donc avec les mêmes données brutes et les comparer accessoirement avec ce que nous avons obtenu dans Microsoft Excel et Minitab.

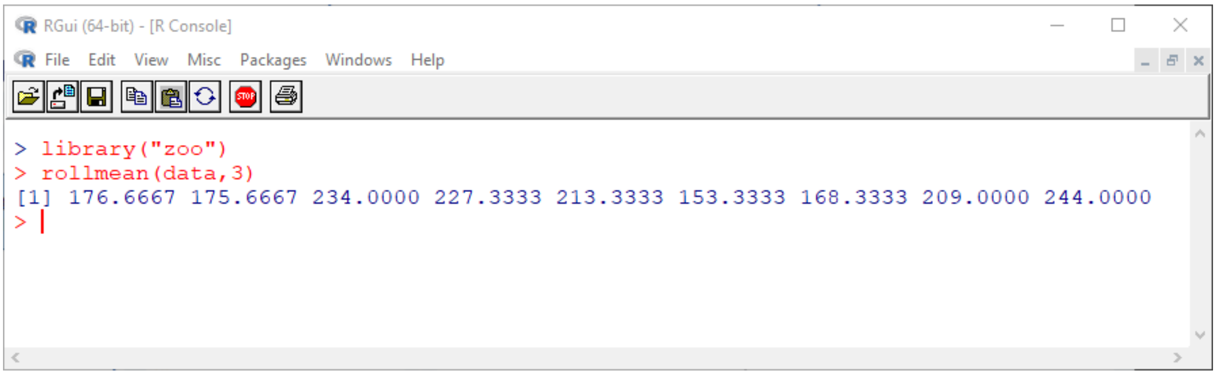
Nous allons utiliser pour cela la commande **filter( )** du package **zoo** et les mêmes données que dans le cours théorique (données visibles ci-dessous dans la colonne *data*)

```
> data<-c(200,135,195,197,310,175,155,130,220,277,235)
> #choix du nombre de périodes k à utiliser dans le calcul de la moyenne
> data.ma3<-filter(data,sides=1,c(rep(1,3))/3)
> data.ma5<-filter(data,sides=1,c(rep(1,5))/5)
> cbind(data,data.ma3,data.ma5)
Time Series:
Start = 1
End = 11
Frequency = 1
   data data.ma3 data.ma5
 1  200       NA       NA
 2  135       NA       NA
 3  195 176.6667       NA
 4  197 175.6667       NA
 5  310 234.0000    207.4
 6  175 227.3333    202.4
 7  155 213.3333    206.4
 8  130 153.3333    193.4
 9  220 168.3333    198.0
10  277 209.0000    191.4
11  235 244.0000    203.4
> ts.plot(data,data.ma3,data.ma5,col=1:3,lwd=2,xlim=c(1,13),ylim=c(100,450))
> legend("topright",lty=c(1,1,1),lwd=c(2,2,2),col=1:3,legend=c("Observations","MA(3)","MA(5)"))
>
```

Donc tout est parfaitement conforme. Au niveau graphique cela donne donc:

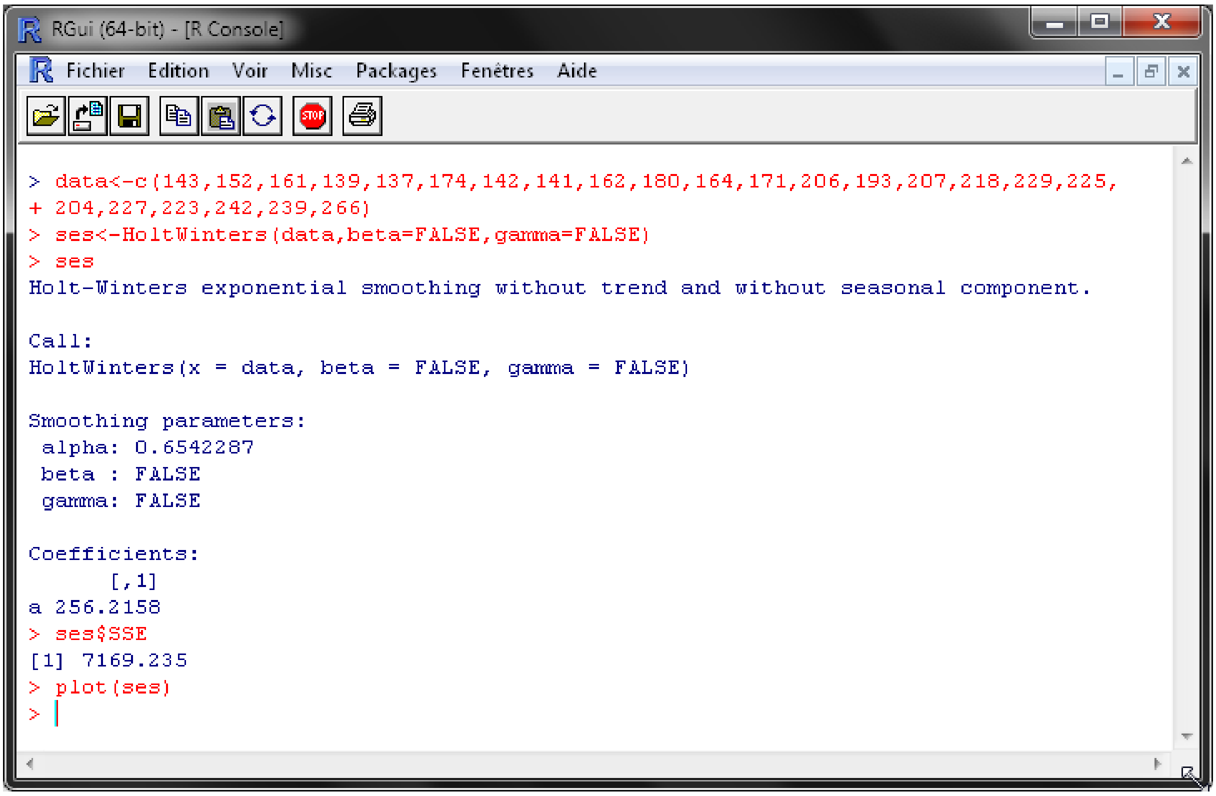


Ou avec le package **zoo** et sa fonction **rollmean( )**:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("zoo")
> rollmean(data,3)
[1] 176.6667 175.6667 234.0000 227.3333 213.3333 153.3333 168.3333 209.0000 244.0000
>

Lissage exponentiel simple

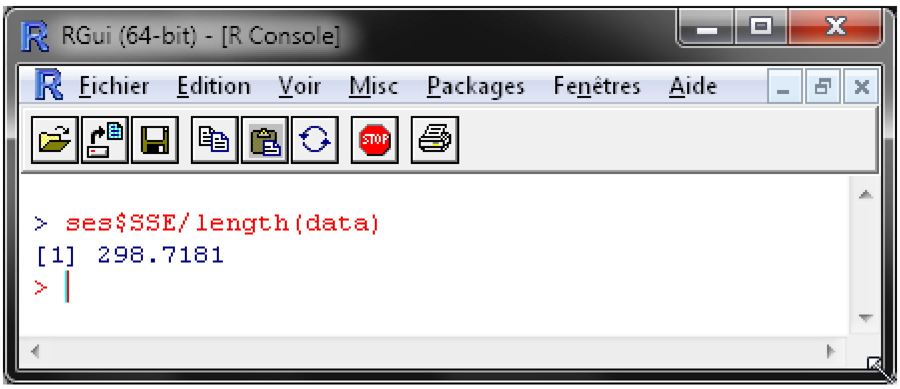
Nous utilisons les mêmes données que dans le cours théorique nous laissons R trouver la meilleure valeur du paramètre du modèle en utilisant la commande **HoltWinters( )** native de R:



Nous retrouvons donc la même valeur du paramètre alpha que dans le cours théorique (où nous avions fait les calculs avec MS Excel) et dans le cours Minitab.

Pour la valeur de 256.2158 à quelques pourcents près non significatifs c'est à peu près conforme au cous théorique et à Minitab. Mais attention à ces petites différences lorsque l'on gère des projets dont les ordres de grandeur est le milliard de $...

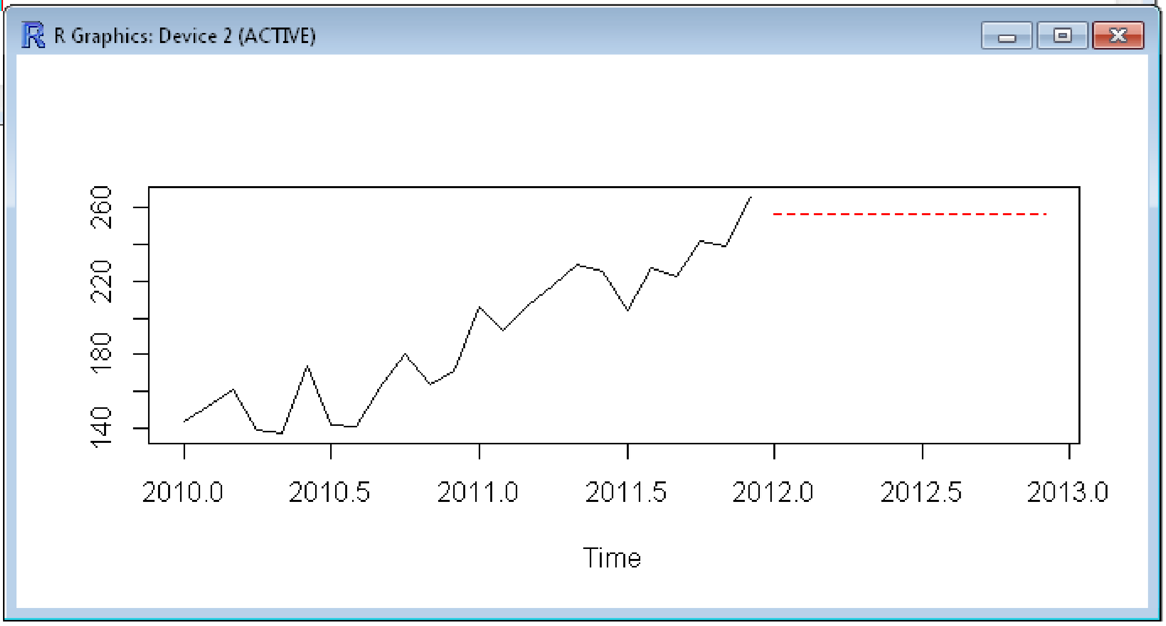
Par contre il est dommage que R ne nous indique par les valeurs classiques d'erreurs que sont les MAD, MSD et MAPE. Mais nous pouvons retrouver le MSD avec le calcul simple suivant:



Donc à un pourcentage non significatif près, nous retrouvons la même que dans le cours théorique et qu'avec Minitab.

Nous pouvons utiliser la commande native **predict( )** pour faire des projections. Ici nous allons faire un exemple aberrant allant au-delà d'une période:

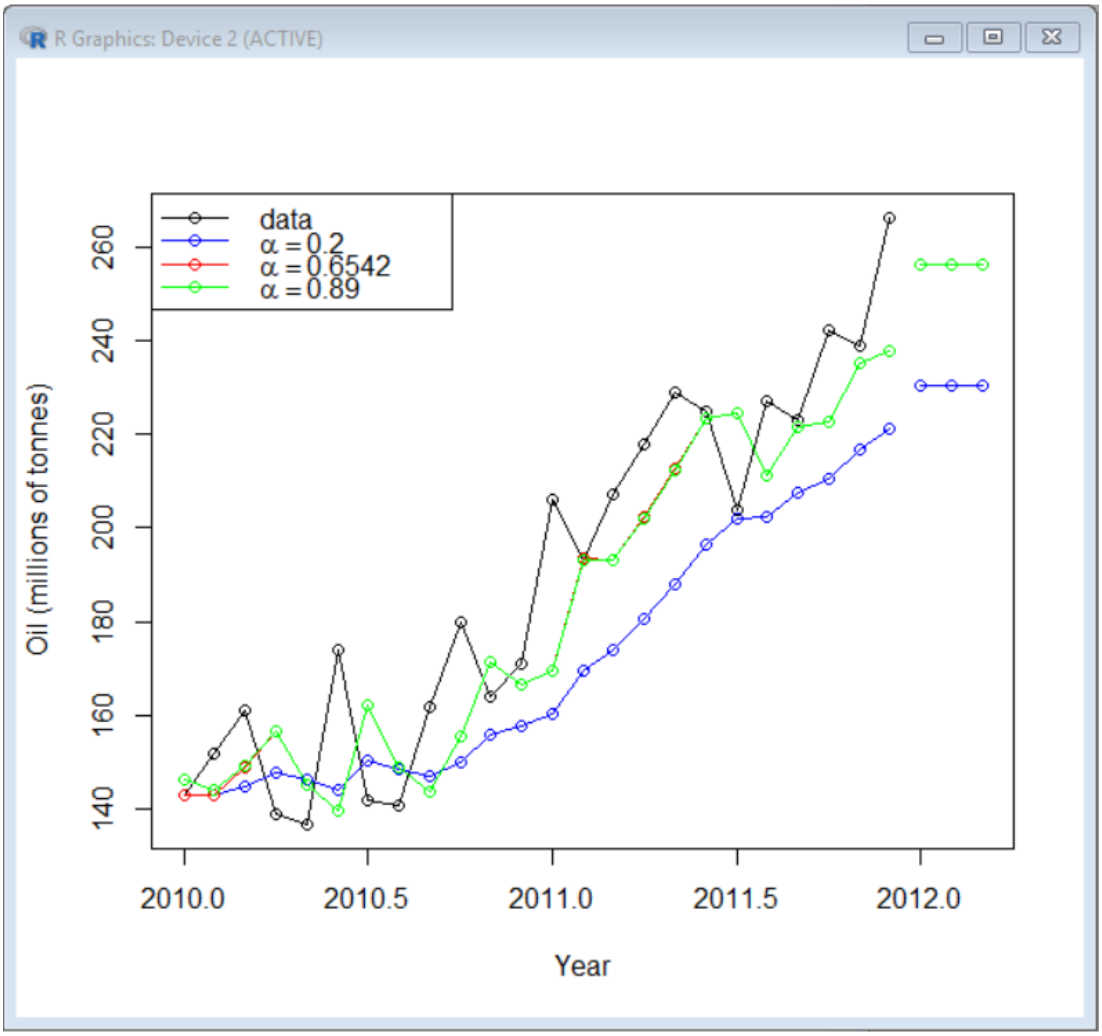
```
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ 204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=12)
> ses<-HoltWinters(data,beta=FALSE,gamma=FALSE)
> data.predict<-predict(ses,n.ahead=12)
> ts.plot(data,data.predict,lty=1:2,col=c("black","red"))
>
```



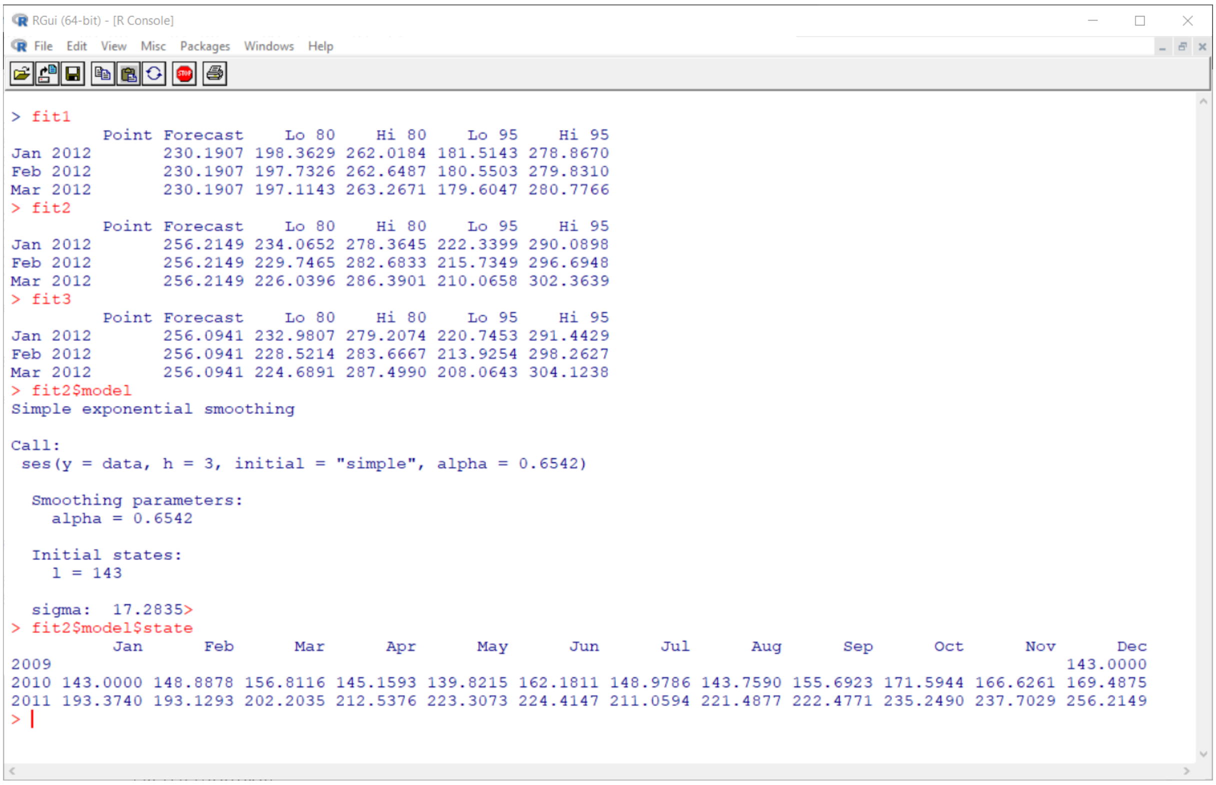
On peut faire mieux avec la fonction **ses( )** et la package **fpp**:

```
> library("fpp")
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ 204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=12)
> fit1 <- ses(data, alpha=0.2, initial="simple", h=3)
> fit2 <- ses(data, alpha=0.6542, initial="simple", h=3)
> fit3 <- ses(data, h=3)
> plot(fit1,PI=F, ylab="Oil (millions of tonnes)",
+   xlab="Year", main="", fcol="white", type="o")
> lines(fitted(fit1), col="blue", type="o")
> lines(fitted(fit2), col="red", type="o")
> lines(fitted(fit3), col="green", type="o")
> lines(fit1$mean, col="blue", type="o")
> lines(fit2$mean, col="red", type="o")
> lines(fit3$mean, col="green", type="o")
> legend("topleft",lty=1, col=c(1,"blue","red","green"),
+   c("data", expression(alpha == 0.2), expression(alpha == 0.6542),
+   expression(alpha == 0.89)),pch=1)
>
```

Ce qui donne:

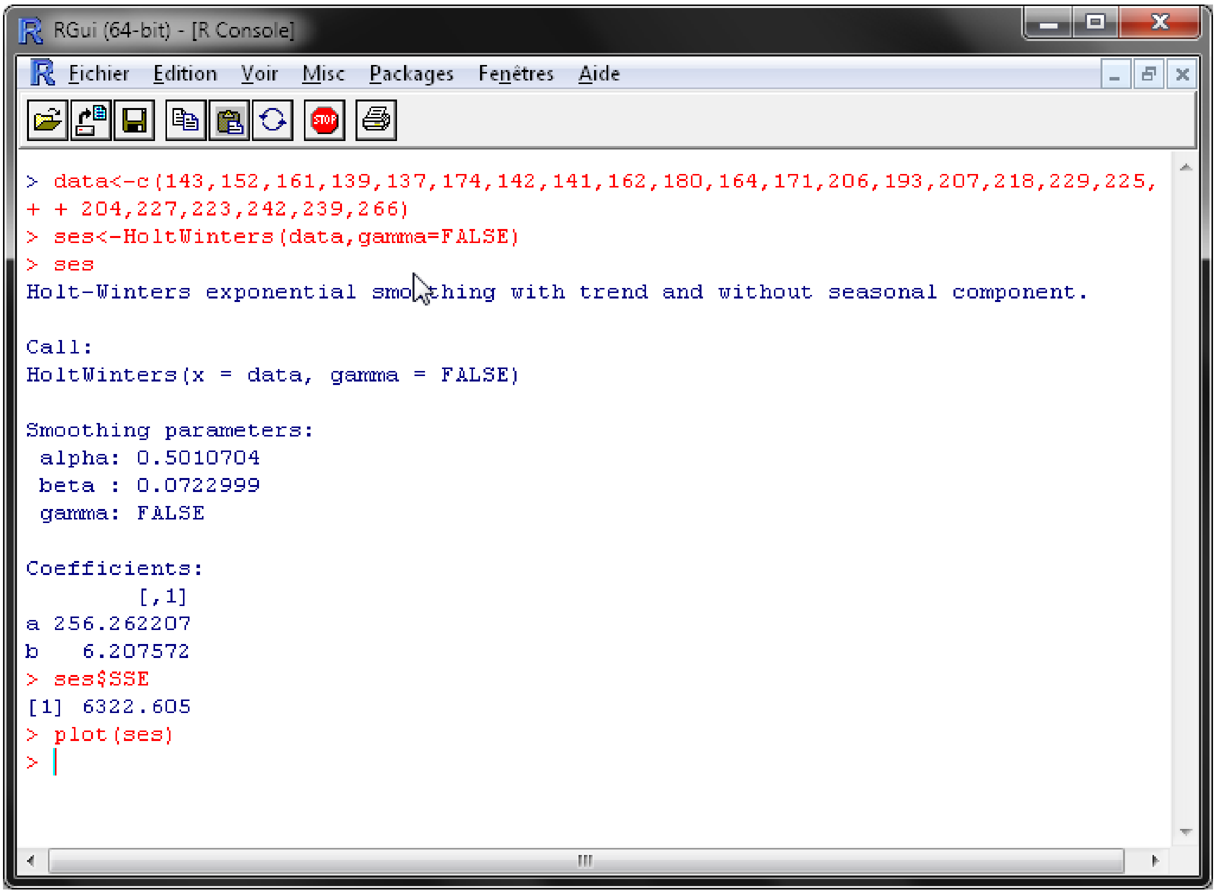


On peut obtenir des détails sur les séries:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> fit1
Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
Jan 2012 230.1907 198.3629 262.0184 181.5143 278.8670
Feb 2012 230.1907 197.7326 262.6487 180.5503 279.8310
Mar 2012 230.1907 197.1143 263.2671 179.6047 280.7766
> fit2
Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
Jan 2012 256.2149 234.0652 278.3645 222.3399 290.0898
Feb 2012 256.2149 229.7465 282.6833 215.7349 296.6948
Mar 2012 256.2149 226.0396 286.3901 210.0658 302.3639
> fit3
Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
Jan 2012 256.0941 232.9807 279.2074 220.7453 291.4429
Feb 2012 256.0941 228.5214 283.6667 213.9254 298.2627
Mar 2012 256.0941 224.6891 287.4990 208.0643 304.1238
> fit2$model
Simple exponential smoothing
Call:
ses(y = data, h = 3, initial = "simple", alpha = 0.6542)
Smoothing parameters:
alpha = 0.6542
Initial states:
l = 143
sigma: 17.2835>
> fit2$model$state
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2009 143.0000
2010 143.0000 148.8878 156.8116 145.1593 139.8215 162.1811 148.9786 143.7590 155.6923 171.5944 166.6261 169.4875
2011 193.3740 193.1293 202.2035 212.5376 223.3073 224.4147 211.0594 221.4877 222.4771 235.2490 237.7029 256.2149
>

Lissage exponentiel double selon Holt (modèle additif)

Nous utilisons les mêmes données que dans le cours théorique nous laissons R trouver la meilleure valeur des paramètres du modèle en utilisant la commande **HoltWinters( )** native de R:



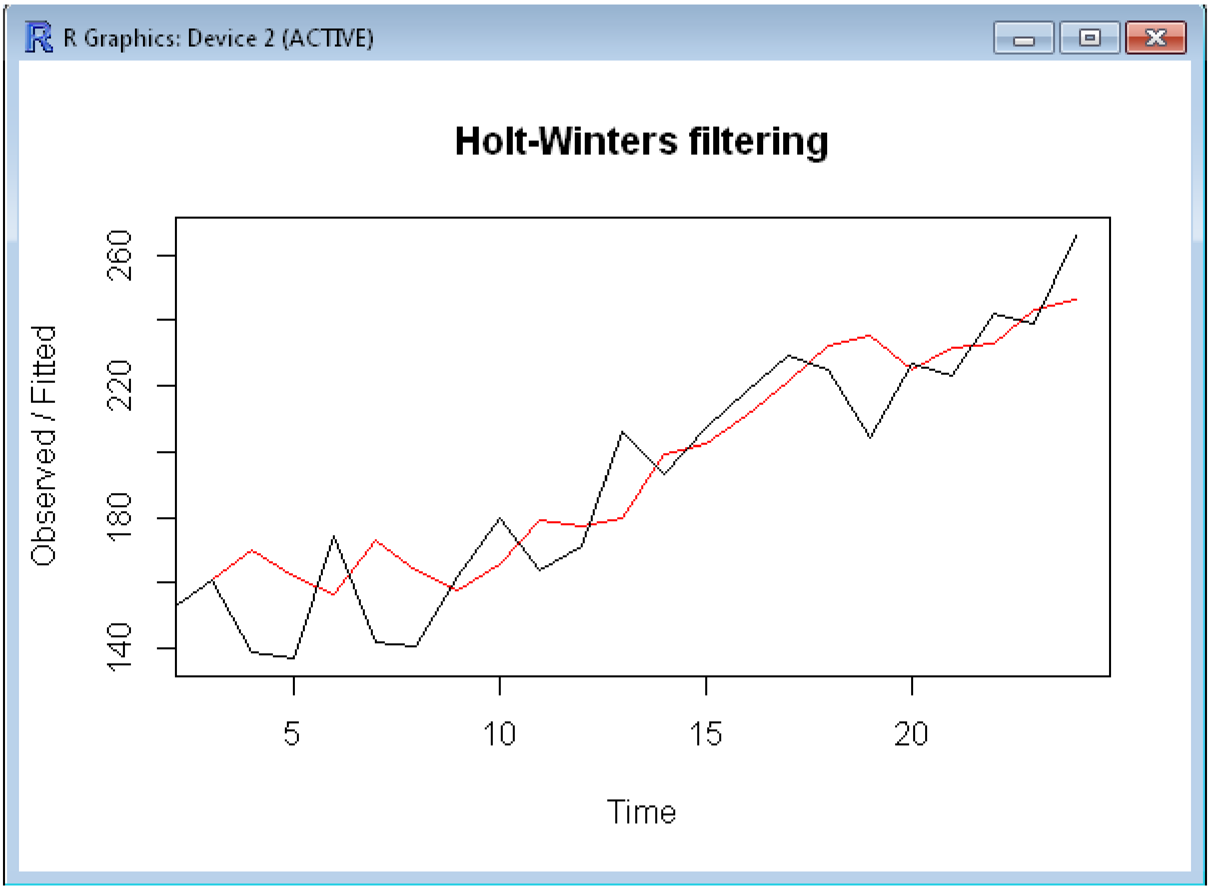
Rappelons que ce modèle est sensible au choix de la méthode de calcul de départ.

Ainsi les deux coefficients sont très différents (+ de 40%) de ceux calculés à la main dans le cours théorique[38]. Par rapport à Minitab[39], la différence est de l'ordre de 5%.

Évidemment le meilleur choix est celui qui minimise le SSE. Ainsi dans le cours théorique, nous avions obtenu un SSE d'un peu plus de 10'000... et avec Minitab de l'ordre de 6430... donc R fait mieux si l'on peut dire...

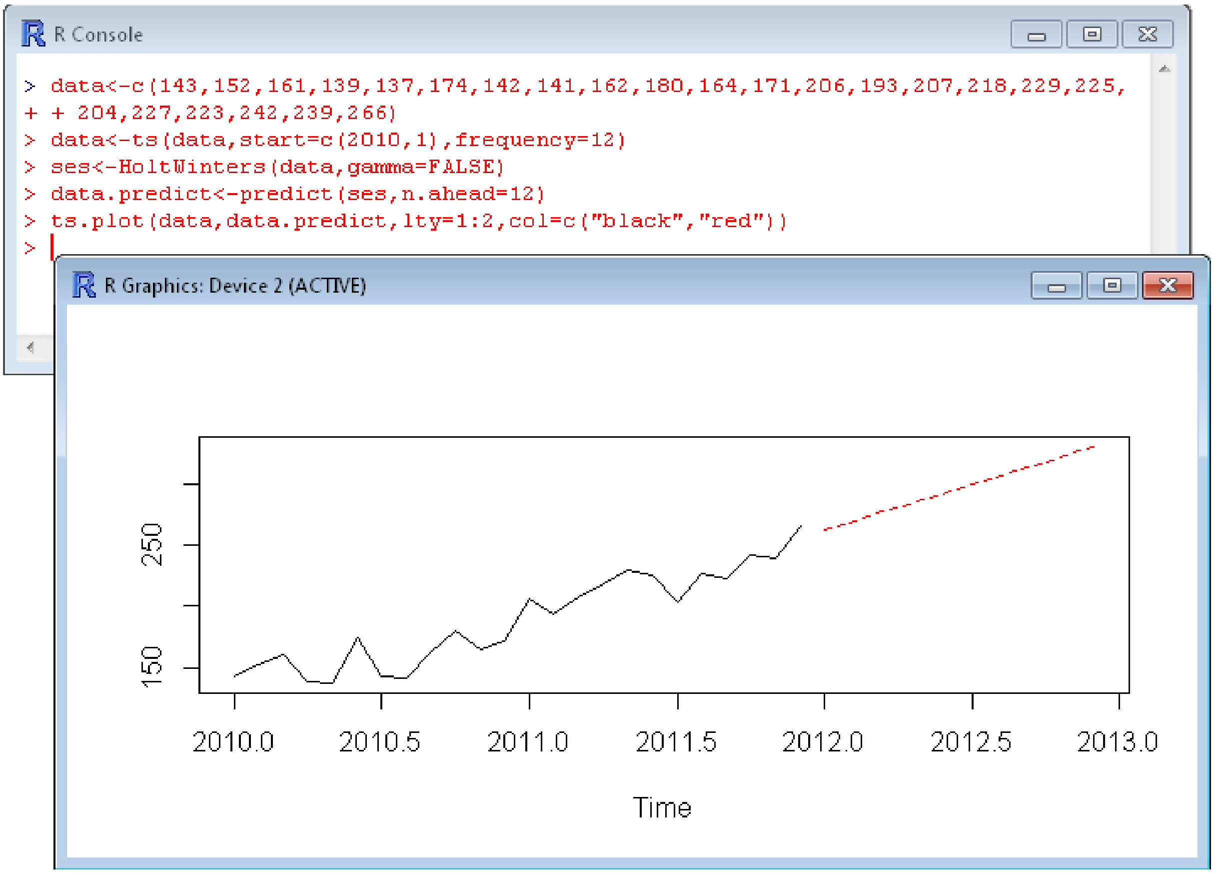
Le graphique donne quant à lui:

[38] Respectivement 0.463 et 0.041
[39] Respectivement 0.4759 et 0.0977



Je n'ai par contre pas trouvé comment faire un lissage exponentiel double selon la méthode de Brown.

Nous pouvons utiliser la commande native **predict( )** pour faire des projections. Ici nous allons faire un exemple aberrant allant au-delà d'une période:



Étudions également un cas très instructeur. La différence entre la fonction **HoltWinters( )** et la fonction **ets( )** qui pourtant tous deux utilisent en théorie le même modèle additif de Holt. Voyons donc cela:

```
> library(forecast)
> serie<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,
+ 193,207,218,229,225,204,227,223,242,239,266)
> ets(ts(serie))
ETS(A,A,N)

Call:
 ets(y = ts(serie))

  Smoothing parameters:
    alpha = 0.3072
    beta  = 1e-04

  Initial states:
    l = 141.3296
    b = 4.7128

  sigma:  16.1923

     AIC     AICc      BIC
215.5552 218.8886 221.4455
>
```

Comme nous pouvons le voir les résultats sont très différents!!!!! Pourquoi donc?

Eh bien les raisons sont bien expliquées ici (auteur d'un livre de 360 pages sur le lissage exponentiel):

https://robjhyndman.com/hyndsight/estimation2/

> I have a question about the ets() function in R, which I am trying to use for Holt-Winters exponential smoothing. My problem is that I am getting very different estimates of the alpha, beta and gamma parameters using ets() compared to HoltWinters(), and I can't figure out why.

This is a common question, so I thought the answer might be of sufficient interest to post here.

There are several issues involved.

1. `HoltWinters()` and `ets()` are optimizing different criterion. `HoltWinters()` is using heuristic values for the initial states and then estimating the smoothing parameters by optimizing the MSE. `ets()` is estimating both the initial states and smoothing parameters by optimizing the likelihood function (which is only equivalent to optimizing the MSE for the linear additive models).

2. The two functions use different optimization routines and different starting values. That wouldn't matter if the surfaces being optimized were smooth, but they are not. Because the MSE and likelihood surfaces are both fairly bumpy, it is easy to find a local optimum. The only way to avoid this problem is to use a much slower computational method such as PSO.

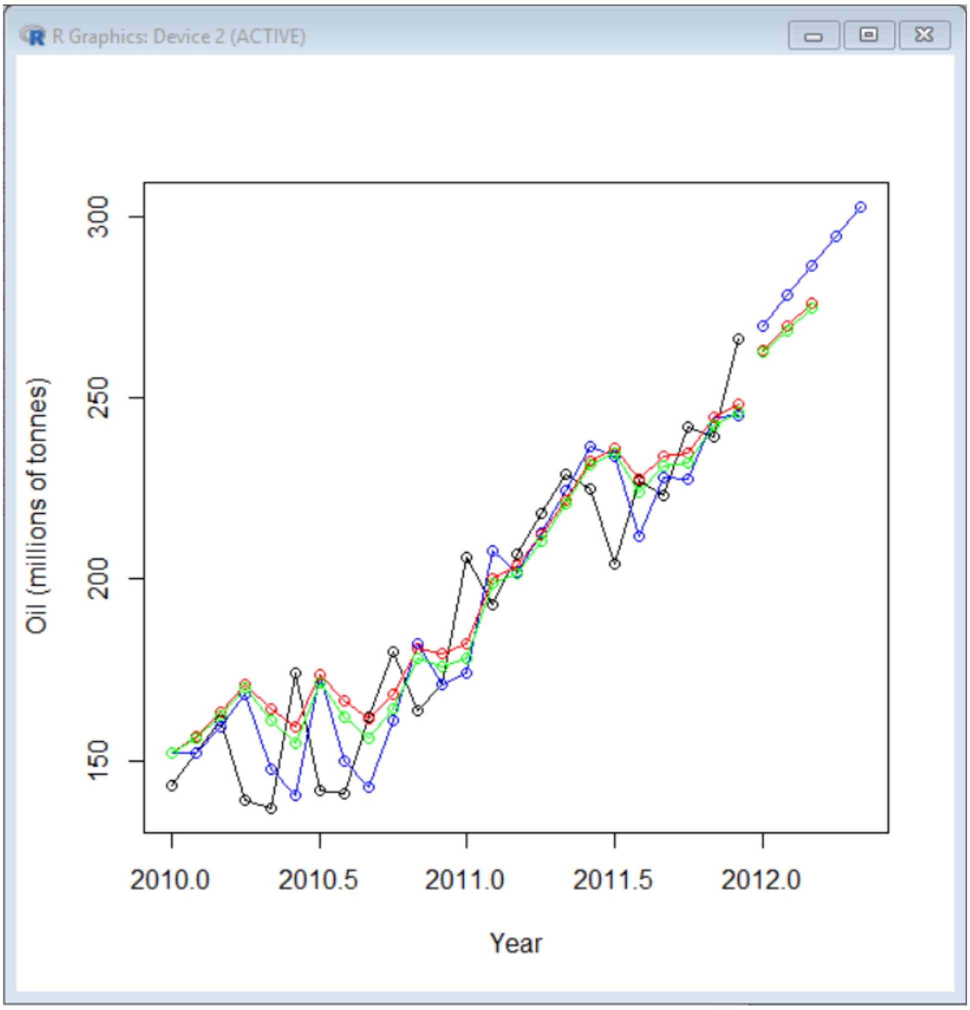
3. `ets()` searches over a restricted parameter space to ensure the resulting model is forecastable. `HoltWinters()` ignores this issue (it was written before the problem was even discovered). See this paper for details (equivalently chapter 10 of my exponential smoothing book).

I have experimented with many different choices of the starting values for the initial values and smoothing parameters, and what is implemented in `ets()` seems about as good as is possible without using a much slower optimization routine. Where there is a difference between `ets()` and `HoltWinters()`, the results from `ets()` are usually more reliable.

Sinon on a encore une autre approche possible, toujours avec le package **fpp** mais avec la fonction **holt( )** :

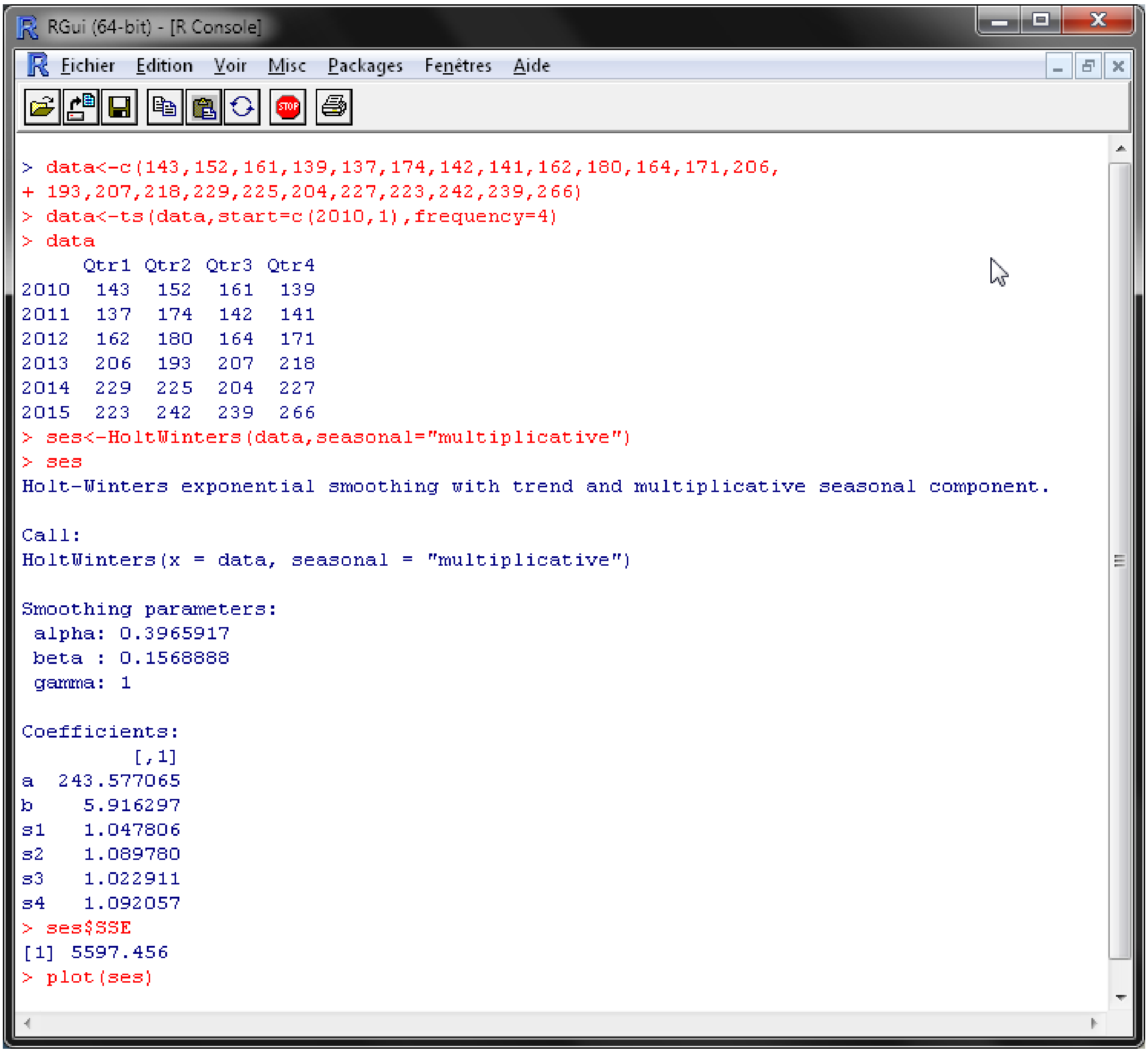
```
> library("fpp")
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ 204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=12)
> fit1 <- holt(data, alpha=0.8, beta=0.2, initial="simple", h=5)
> fit2 <- holt(data, alpha=0.463, beta=0.041, initial="simple", h=3)
> fit3 <- holt(data, initial="simple", h=3)
> plot(fit1,PI=F, ylab="Oil (millions of tonnes)",
+   xlab="Year", main="", fcol="white", type="o")
> lines(fitted(fit1), col="blue", type="o")
> lines(fitted(fit2), col="red", type="o")
> lines(fitted(fit3), col="green", type="o")
> lines(fit1$mean, col="blue", type="o")
> lines(fit2$mean, col="red", type="o")
> lines(fit3$mean, col="green", type="o")
> |
```

Ce qui donne visuellement:



Lissage exponentiel triple selon Holt et Winters (modèle multiplicatif)

Nous utilisons les mêmes données que dans le cours théorique et nous laissons R trouver la meilleure valeur des paramètres du modèle toujours en utilisant la commande **HoltWinters( )** native de R:



```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,
+ 193,207,218,229,225,204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=4)
> data
     Qtr1 Qtr2 Qtr3 Qtr4
2010  143  152  161  139
2011  137  174  142  141
2012  162  180  164  171
2013  206  193  207  218
2014  229  225  204  227
2015  223  242  239  266
> ses<-HoltWinters(data,seasonal="multiplicative")
> ses
Holt-Winters exponential smoothing with trend and multiplicative seasonal component.

Call:
HoltWinters(x = data, seasonal = "multiplicative")

Smoothing parameters:
 alpha: 0.3965917
 beta : 0.1568888
 gamma: 1

Coefficients:
         [,1]
a  243.577065
b    5.916297
s1   1.047806
s2   1.089780
s3   1.022911
s4   1.092057
> ses$SSE
[1] 5597.456
> plot(ses)
```

Nous ne pouvons pas ici comparer avec Minitab puisque ce dernier n'est pas capable de trouver les 3 paramètres optimaux. Par contre par rapport faits à la main dans le cours théorique il n'y a aucune comparaison possible... relativement différent pour les paramètres de lissage où nous avions obtenu dans le cours théorique:

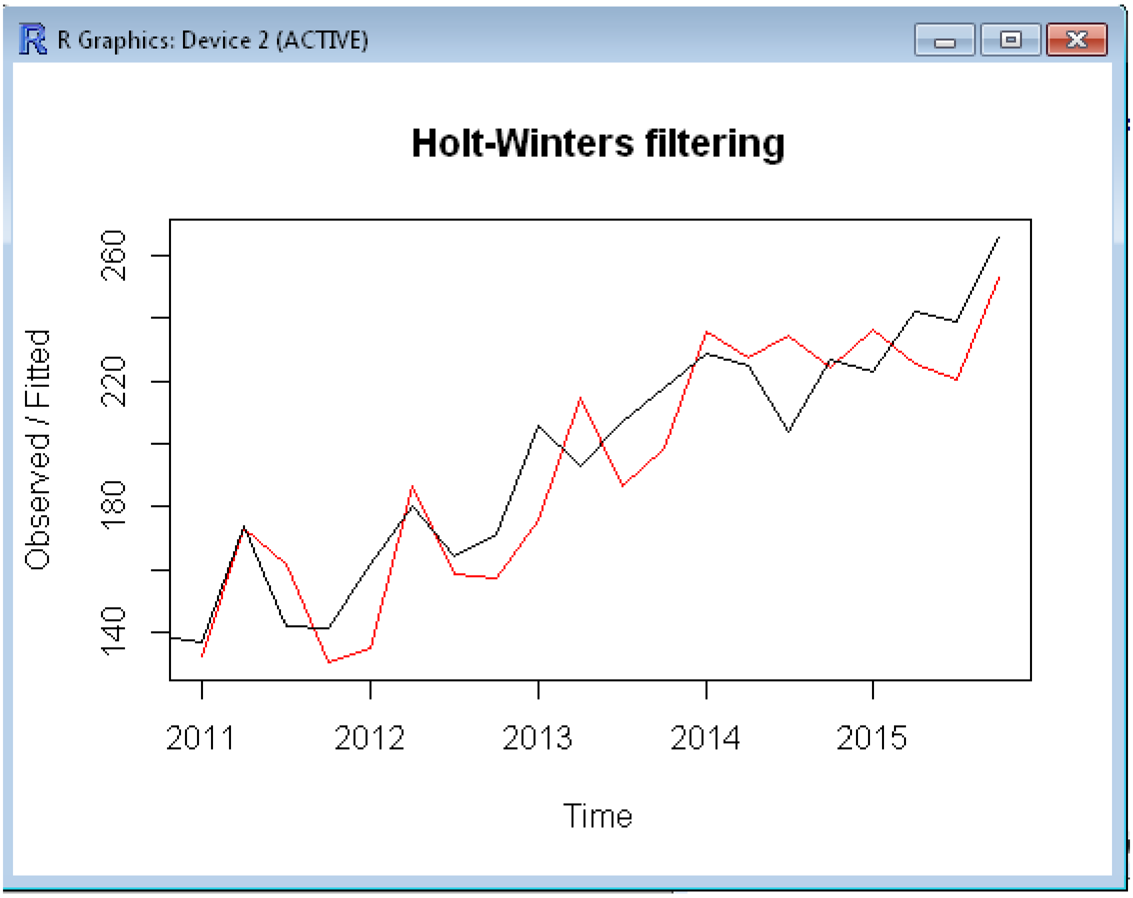
$$\alpha = 0.329, \beta = 1.06, \gamma = 1$$

et:

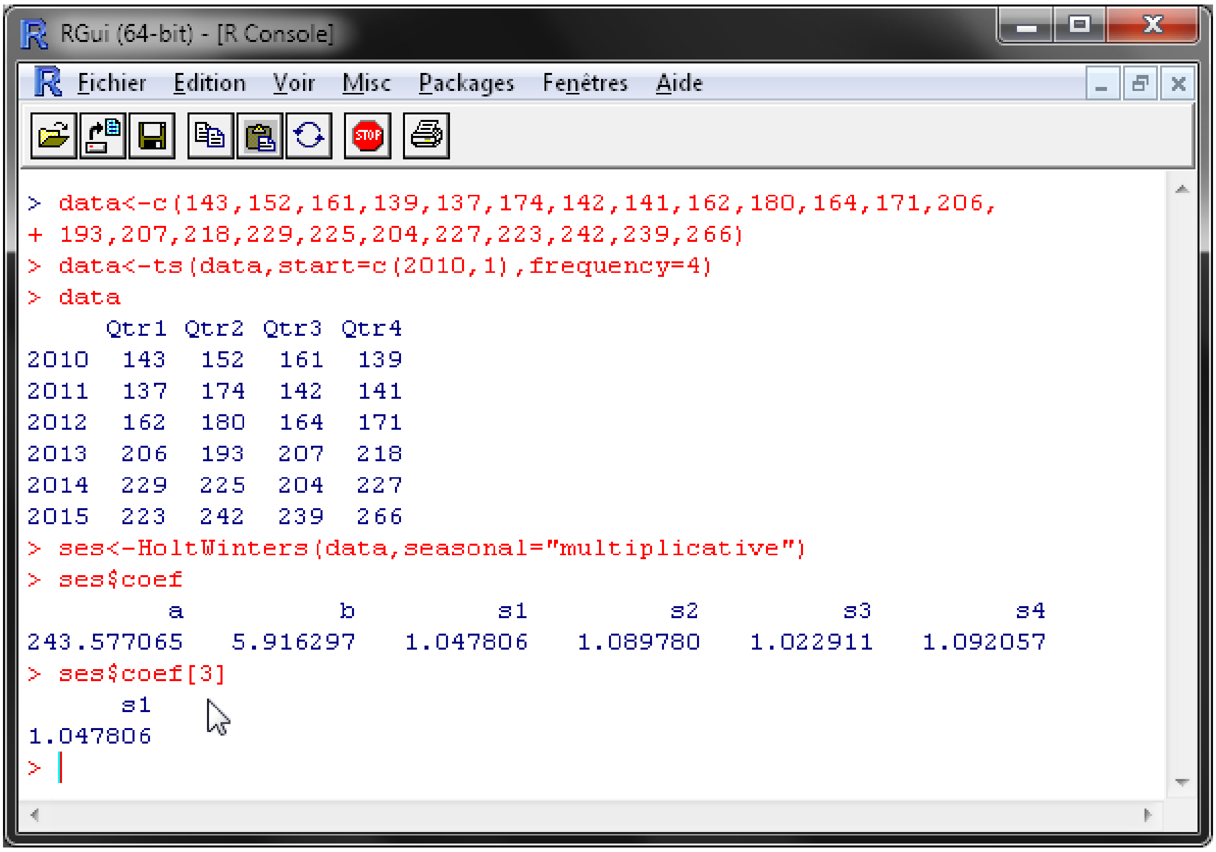
$$a = 249.38$$
$$b = 5.58$$
$$s_1 = 0.961$$
$$s_2 = 1.022$$
$$s_3 = 1.082$$
$$s_4 = 0.934$$

Mais comme nous l'avons mentionné dans le cours théorique, ce modèle est aussi sensible à la manière de calculer les toutes premières valeurs et comme il existe de nombreuses règles empiriques, c'est la raison pour laquelle vous aurez peu de chance de trouver deux logiciels qui donnent le même résultat pour ce modèle.

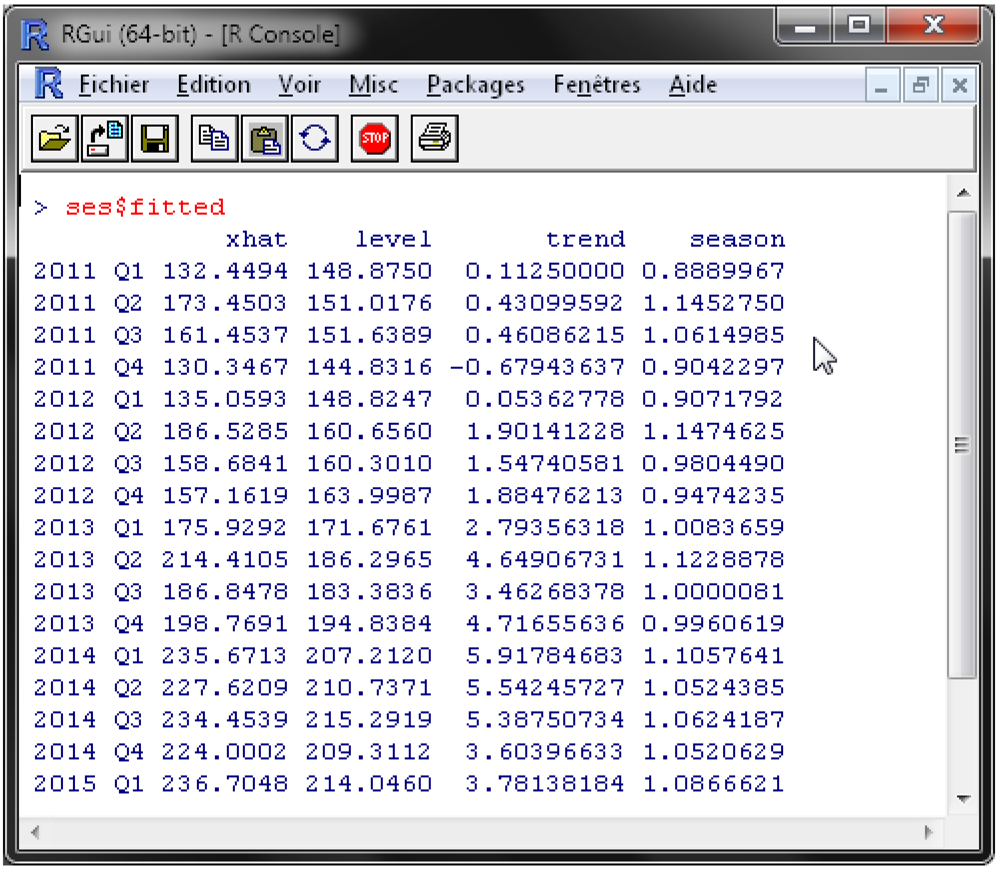
Ceci étant dit, le plot nous donne quant à lui:

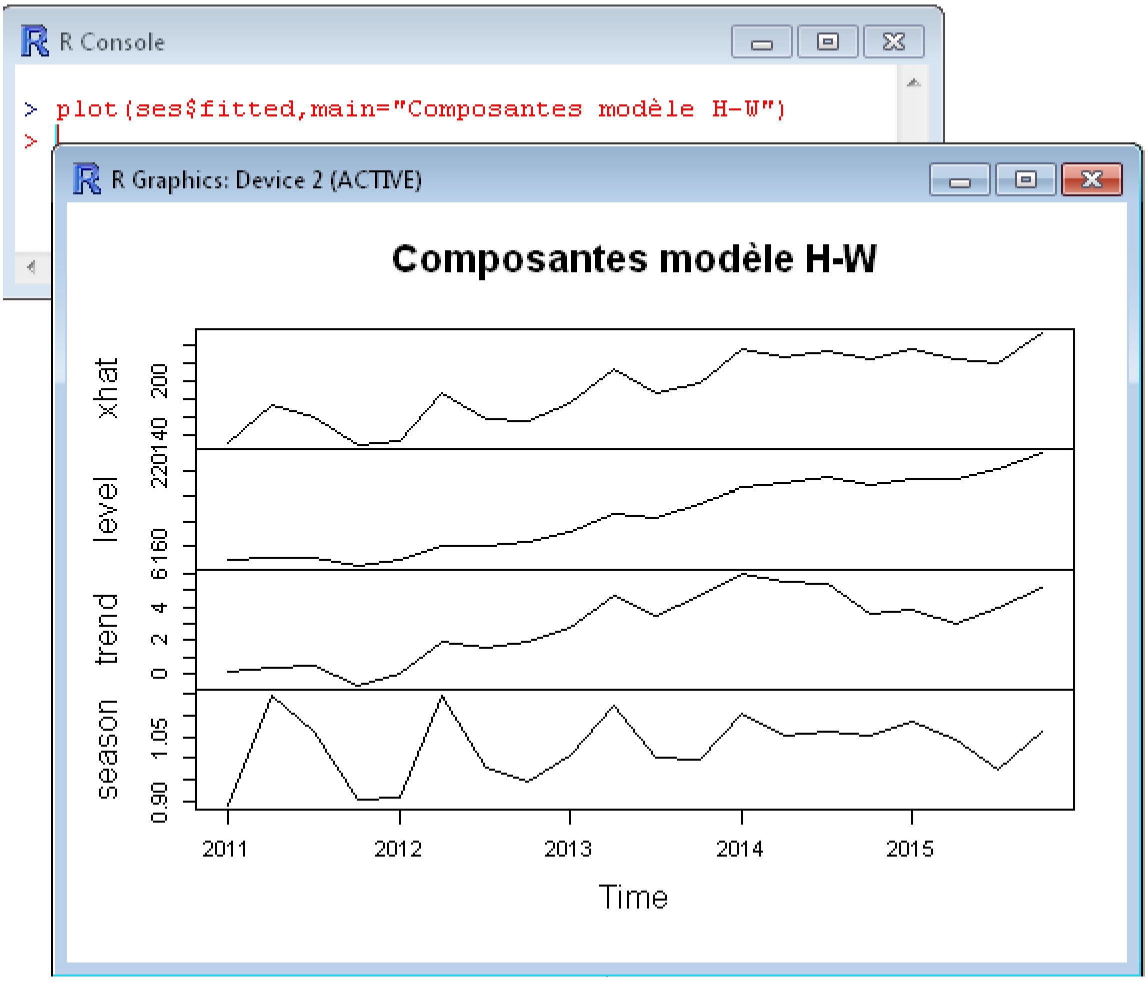


Nous pouvons obtenir si besoin est (très utile dans certains cas) que les coefficients du modèle ou même un seul coefficient:

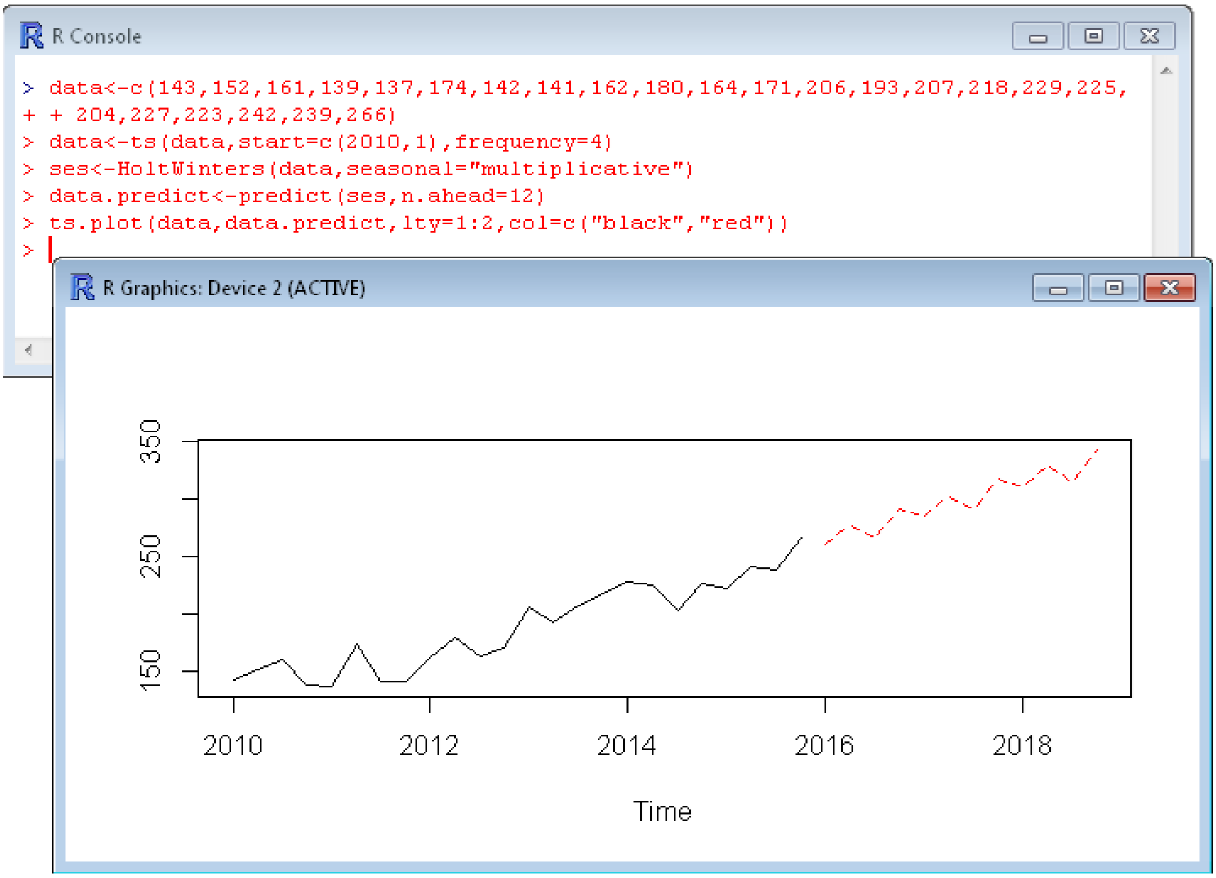


ou obtenir que les valeurs lissées des différentes composantes du modèle tel que calculé dans le cours théorique:





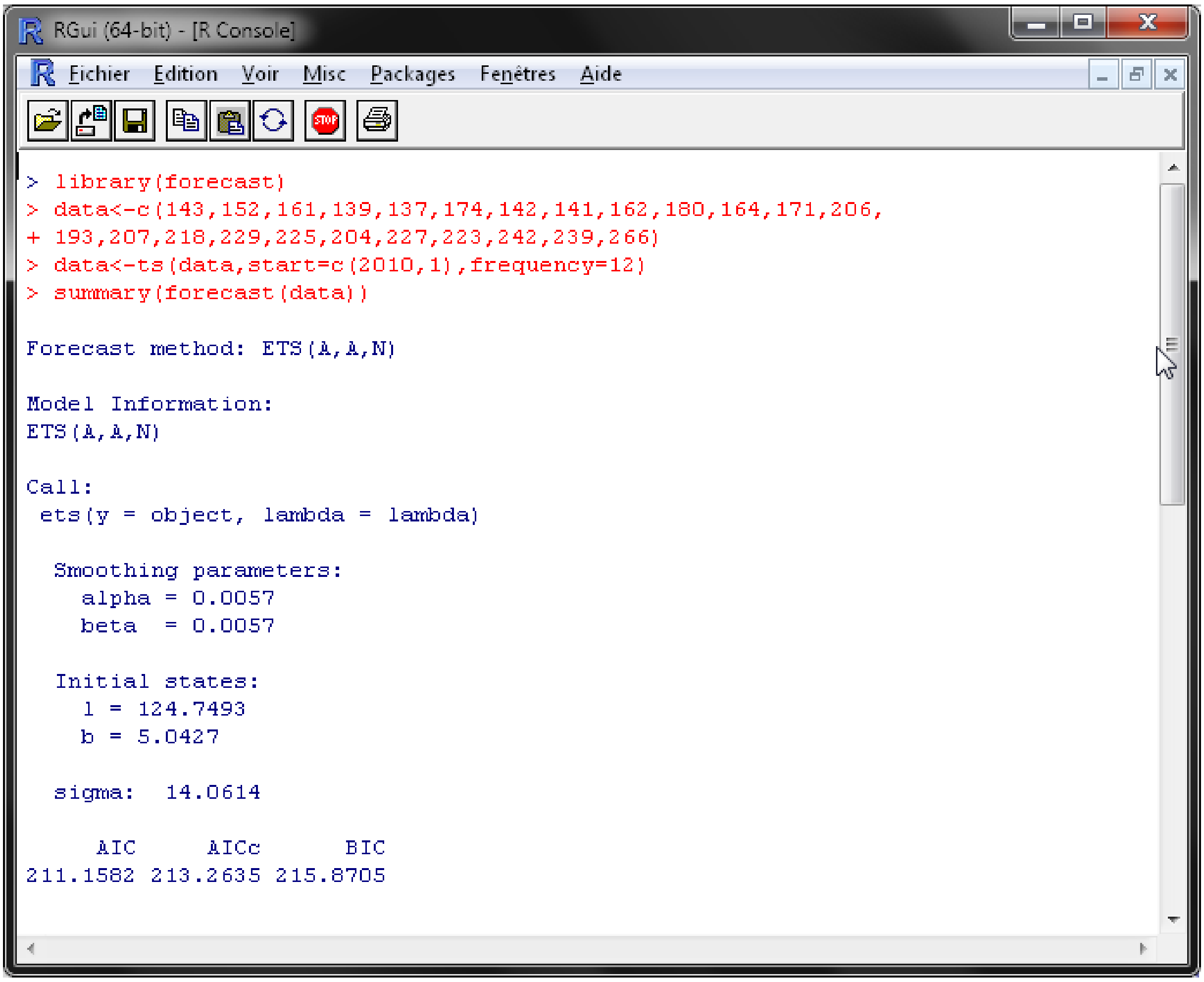
Nous pouvons utiliser la commande native **predict( )** pour faire des projections. Ici nous allons faire un exemple aberrant allant au-delà d'une période:


R Console
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ + 204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=4)
> ses<-HoltWinters(data,seasonal="multiplicative")
> data.predict<-predict(ses,n.ahead=12)
> ts.plot(data,data.predict,lty=1:2,col=c("black","red"))
R Graphics: Device 2 (ACTIVE)
150
250
350
2010
2012
2014
2016
2018
Time

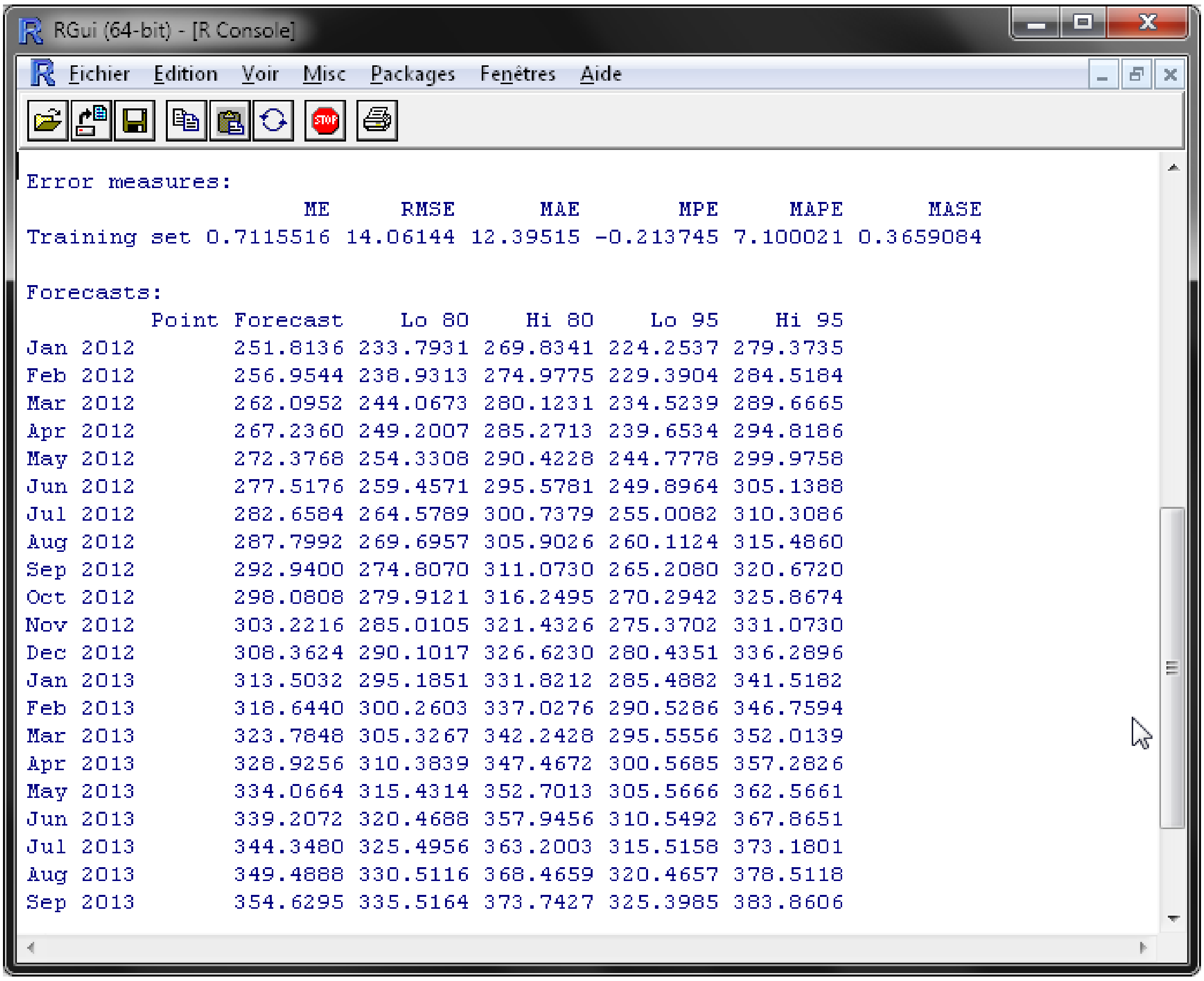
Meilleur modèle prévisionnel au sens des erreurs

Dans le package **forecast**, la commande **forecast( )** va effectuer une prévision en se comparant les erreurs d'une trentaine de modèles et afficher la projection du meilleur modèle considéré.

Voyons cela avec les données utilisées dans le cours théorique:



Donc le meilleur modèle est un modèle ETS(A,A,N) ce qui signifierait que c'est un modèle additif de Holt-Winters avec erreurs additives. La suite donne:



```
Error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.7115516 14.06144 12.39515 -0.213745 7.100021 0.3659084

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jan 2012       251.8136 233.7931 269.8341 224.2537 279.3735
Feb 2012       256.9544 238.9313 274.9775 229.3904 284.5184
Mar 2012       262.0952 244.0673 280.1231 234.5239 289.6665
Apr 2012       267.2360 249.2007 285.2713 239.6534 294.8186
May 2012       272.3768 254.3308 290.4228 244.7778 299.9758
Jun 2012       277.5176 259.4571 295.5781 249.8964 305.1388
Jul 2012       282.6584 264.5789 300.7379 255.0082 310.3086
Aug 2012       287.7992 269.6957 305.9026 260.1124 315.4860
Sep 2012       292.9400 274.8070 311.0730 265.2080 320.6720
Oct 2012       298.0808 279.9121 316.2495 270.2942 325.8674
Nov 2012       303.2216 285.0105 321.4326 275.3702 331.0730
Dec 2012       308.3624 290.1017 326.6230 280.4351 336.2896
Jan 2013       313.5032 295.1851 331.8212 285.4882 341.5182
Feb 2013       318.6440 300.2603 337.0276 290.5286 346.7594
Mar 2013       323.7848 305.3267 342.2428 295.5556 352.0139
Apr 2013       328.9256 310.3839 347.4672 300.5685 357.2826
May 2013       334.0664 315.4314 352.7013 305.5666 362.5661
Jun 2013       339.2072 320.4688 357.9456 310.5492 367.8651
Jul 2013       344.3480 325.4956 363.2003 315.5158 373.1801
Aug 2013       349.4888 330.5116 368.4659 320.4657 378.5118
Sep 2013       354.6295 335.5164 373.7427 325.3985 383.8606
```

Donc ici nous avons un nombre d'informations très intéressantes avec les indicateurs d'erreurs de mesures que nous avons vu dans le cours théorique. Graphiquement le meilleur modèle donne:

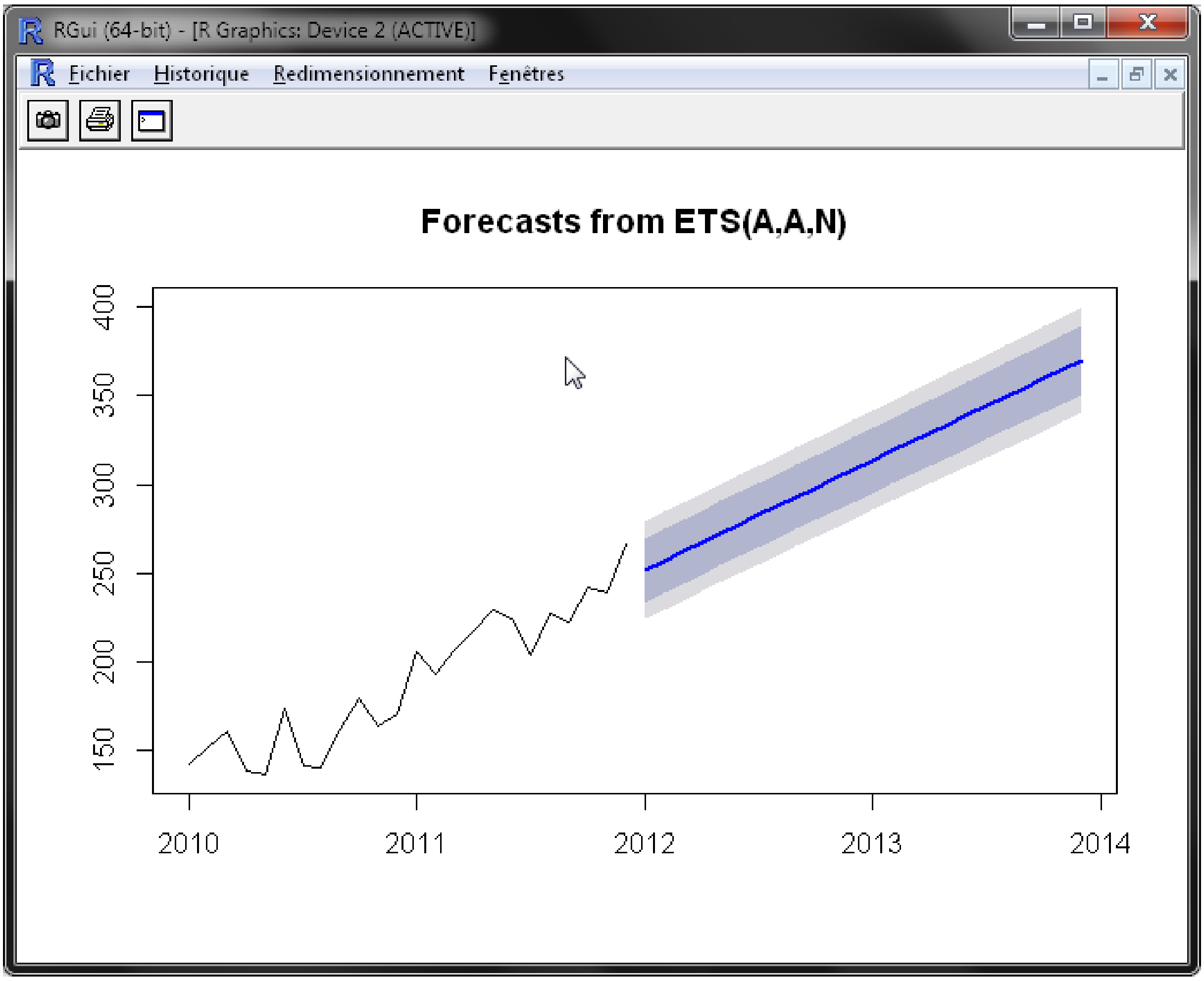
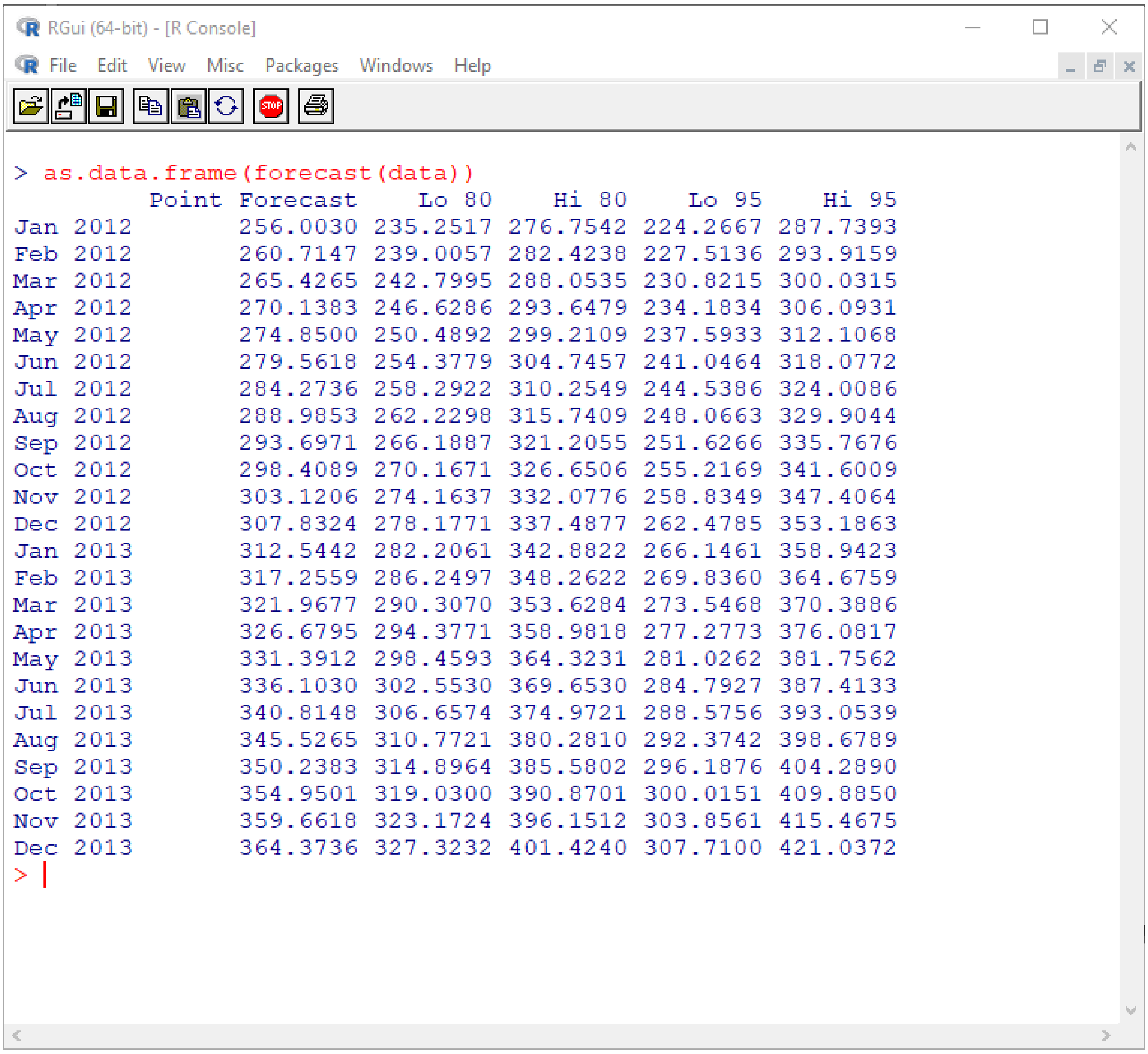


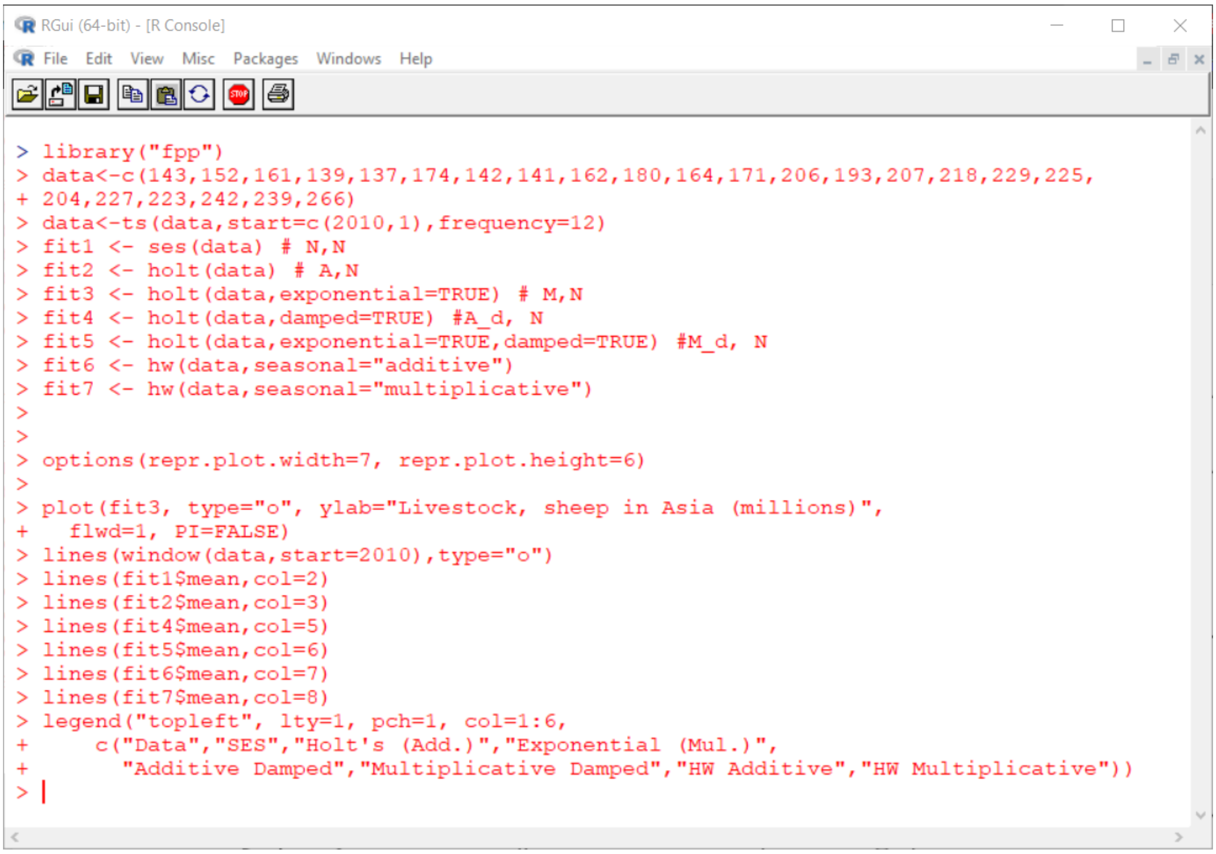
Figure 309 Graph de projection d'une série ETS

À prendre évidemment avec des pincettes et toujours avec le même danger de lecture aberrant que l'on retrouve dans toute la littérature spécialisée et que nous avons déjà mentionné dans le cours théorique.

On peut extraire les données de la projection en tant que data.frame facilement:



Toujours avec le package **fpp** et les fonctions **ses( )** , **holt( )** et **hw( )** on peut compiler un peu toutes les méthodes vues ci-dessus (et d'autres que nous avons juste mentionné dans le cours théorique!):



```
> library("fpp")
> data<-c(143,152,161,139,137,174,142,141,162,180,164,171,206,193,207,218,229,225,
+ 204,227,223,242,239,266)
> data<-ts(data,start=c(2010,1),frequency=12)
> fit1 <- ses(data) # N,N
> fit2 <- holt(data) # A,N
> fit3 <- holt(data,exponential=TRUE) # M,N
> fit4 <- holt(data,damped=TRUE) #A_d, N
> fit5 <- holt(data,exponential=TRUE,damped=TRUE) #M_d, N
> fit6 <- hw(data,seasonal="additive")
> fit7 <- hw(data,seasonal="multiplicative")
>
>
> options(repr.plot.width=7, repr.plot.height=6)
>
> plot(fit3, type="o", ylab="Livestock, sheep in Asia (millions)",
+   flwd=1, PI=FALSE)
> lines(window(data,start=2010),type="o")
> lines(fit1$mean,col=2)
> lines(fit2$mean,col=3)
> lines(fit4$mean,col=5)
> lines(fit5$mean,col=6)
> lines(fit6$mean,col=7)
> lines(fit7$mean,col=8)
> legend("topleft", lty=1, pch=1, col=1:6,
+     c("Data","SES","Holt's (Add.)","Exponential (Mul.)",
+       "Additive Damped","Multiplicative Damped","HW Additive","HW Multiplicative"))
>
```

Ce qui donne visuellement:

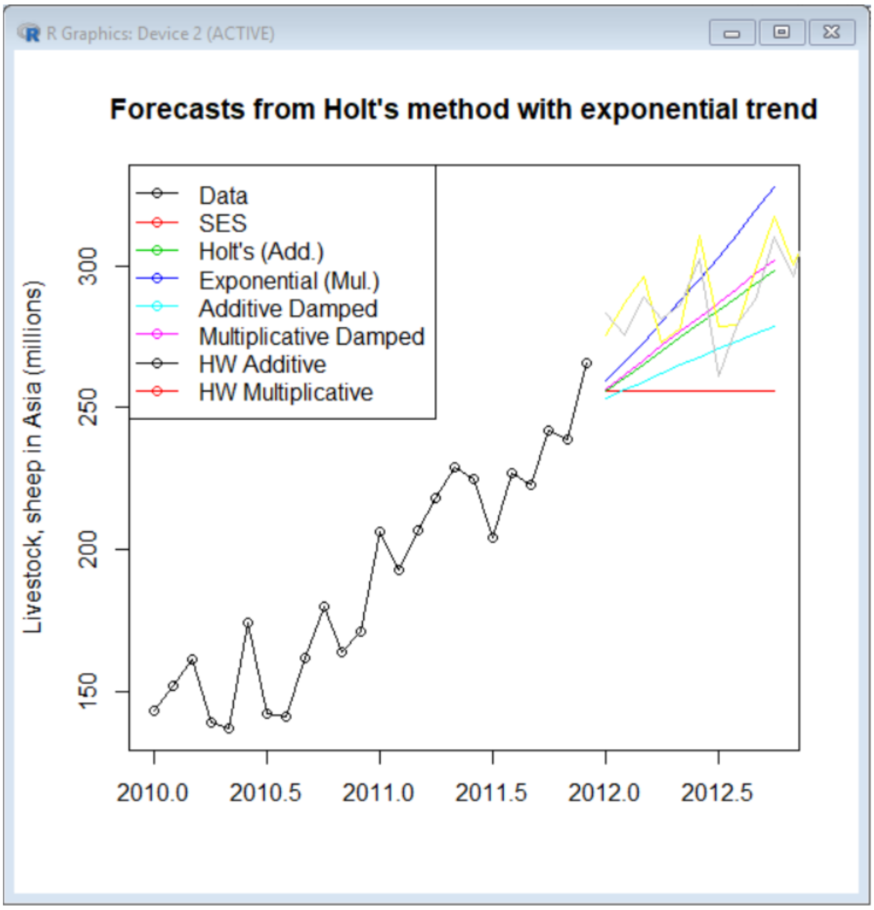


Figure 310 Graph de compilation de modèles ETS

Modèle de demandes intermittentes de Croston

Nous avons présenté dans le cours théorique le modèle de Croston qui est un classique dans les cursus de Master en Supply Chain. Nous avons vu que ce modèle était très naïf… Comme il n'est pas stochastique, il ne permet pas d'avoir des intervalles de confiance.

Voyons comment l'utiliser avec le package **tsintermittent** et sa fonction **crost( )**.

Importons d'abord les données (qui ne sont pas les mêmes que dans le cours théorique car ce package implémente une version de Croston qui n'est pas la version naive/vanilla):

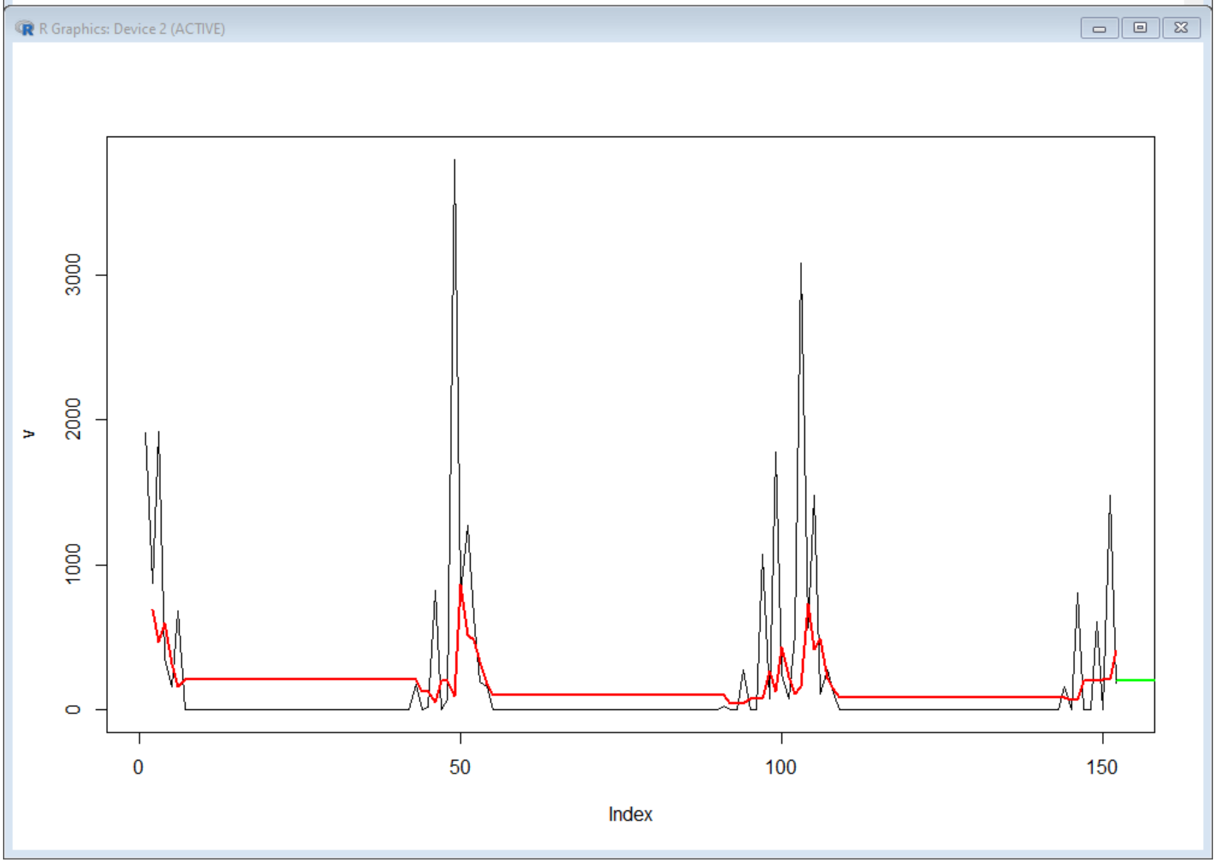
```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("tsintermittent")
> v<-read.csv("c:/tmp/Croston.csv",header=T)
> v<-v$Ventes #pour le graphique
> t(v)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18]
[1,] 1910  874 1920  350  160  685    0    0    0     0     0     0     0     0     0     0     0     0
     [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
[1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
[1,]     0     0     0     0     0     0     0   176     0    16   826     0    66  3798   800  1274   638
     [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69]
[1,]   192   160     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
[1,]     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100] [,101] [,102]
[1,]     0     0     0     0    28     0     0   276     0     0  1072    80  1776    240     80    528
     [,103] [,104] [,105] [,106] [,107] [,108] [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116]
[1,]   3081    566   1483    112    272    120      0      0      0      0      0      0      0      0
     [,117] [,118] [,119] [,120] [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128] [,129] [,130]
[1,]      0      0      0      0      0      0      0      0      0      0      0      0      0      0
     [,131] [,132] [,133] [,134] [,135] [,136] [,137] [,138] [,139] [,140] [,141] [,142] [,143] [,144]
[1,]      0      0      0      0      0      0      0      0      0      0      0      0      0    160
     [,145] [,146] [,147] [,148] [,149] [,150] [,151] [,152]
[1,]      0    808      0      0    608      0   1480    184
> |
```

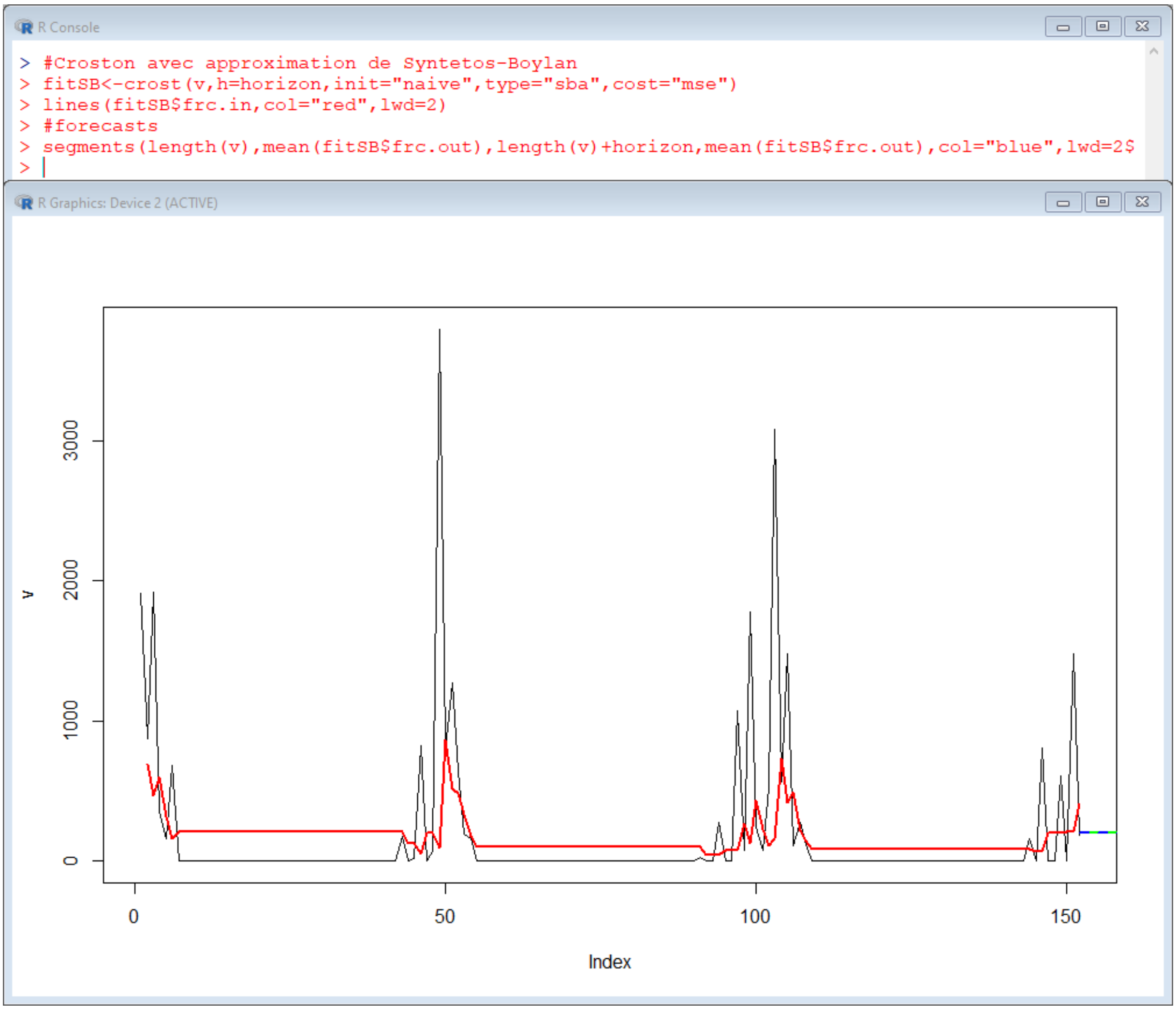
On peut maintenant exécuter le modèle et en afficher une version graphique:

R Console

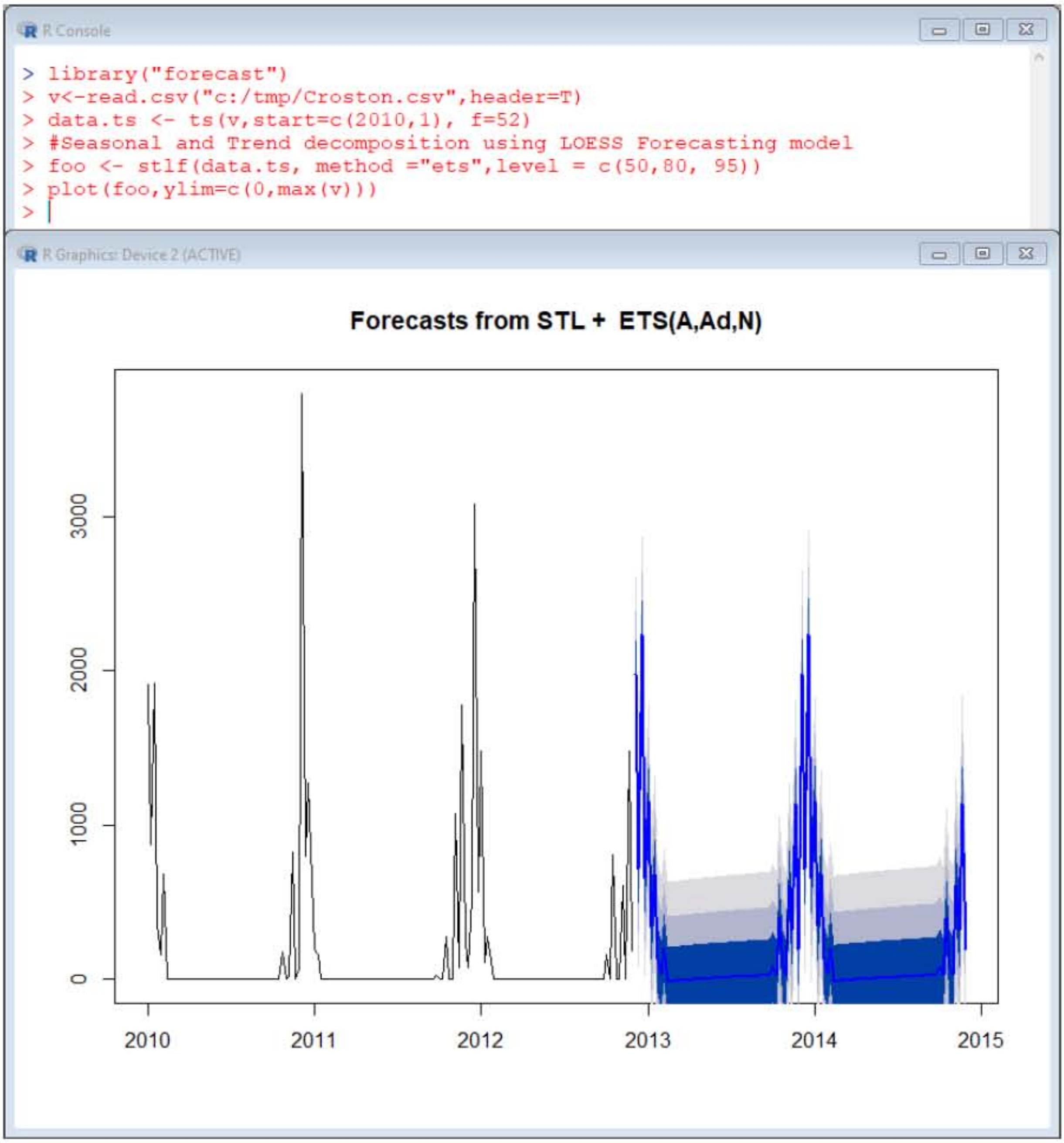
```
> #Croston original (forecast sur 10 périodes)
> horizon=10
> fitC<-crost(v,h=horizon,init="naive",type="croston",cost="mse")
> plot(v,type="l")
> lines(fitC$frc.in,col="red",lwd=2)
> #forecasts
> segments(length(v),mean(fitC$frc.out),length(v)+horizon,mean(fitC$frc.out),col="green",lwd=2)
> |
```



Nous pouvons également rajouter (en bleu traitillé) la variante de Syntetos-Boylon que nous avons vu dans le cours théorique (ce qui change de toute façon peu le résultat dans le cas présent…):



Bon faut être honnête ce n'est pas vraiment convaincant comme résultat (on s'y attendait!). Il vaut peut-être mieux passer par une projection ETS avec un peu de LOESS… (c'est de l'ingénierie statistique qui vaut ce qui vaut hein!). Donc voyons cela avec le package **forecast** et sa fonction **stlf( )**:


R Console
> library("forecast")
> v<-read.csv("c:/tmp/Croston.csv",header=T)
> data.ts <- ts(v,start=c(2010,1), f=52)
> #Seasonal and Trend decomposition using LOESS Forecasting model
> foo <- stlf(data.ts, method ="ets",level = c(50,80, 95))
> plot(foo,ylim=c(0,max(v)))
>
R Graphics: Device 2 (ACTIVE)
Forecasts from STL + ETS(A,Ad,N)
0
1000
2000
3000
2010
2011
2012
2013
2014
2015

Exemples 382.: Comparaisons STL, ETS et TSLM

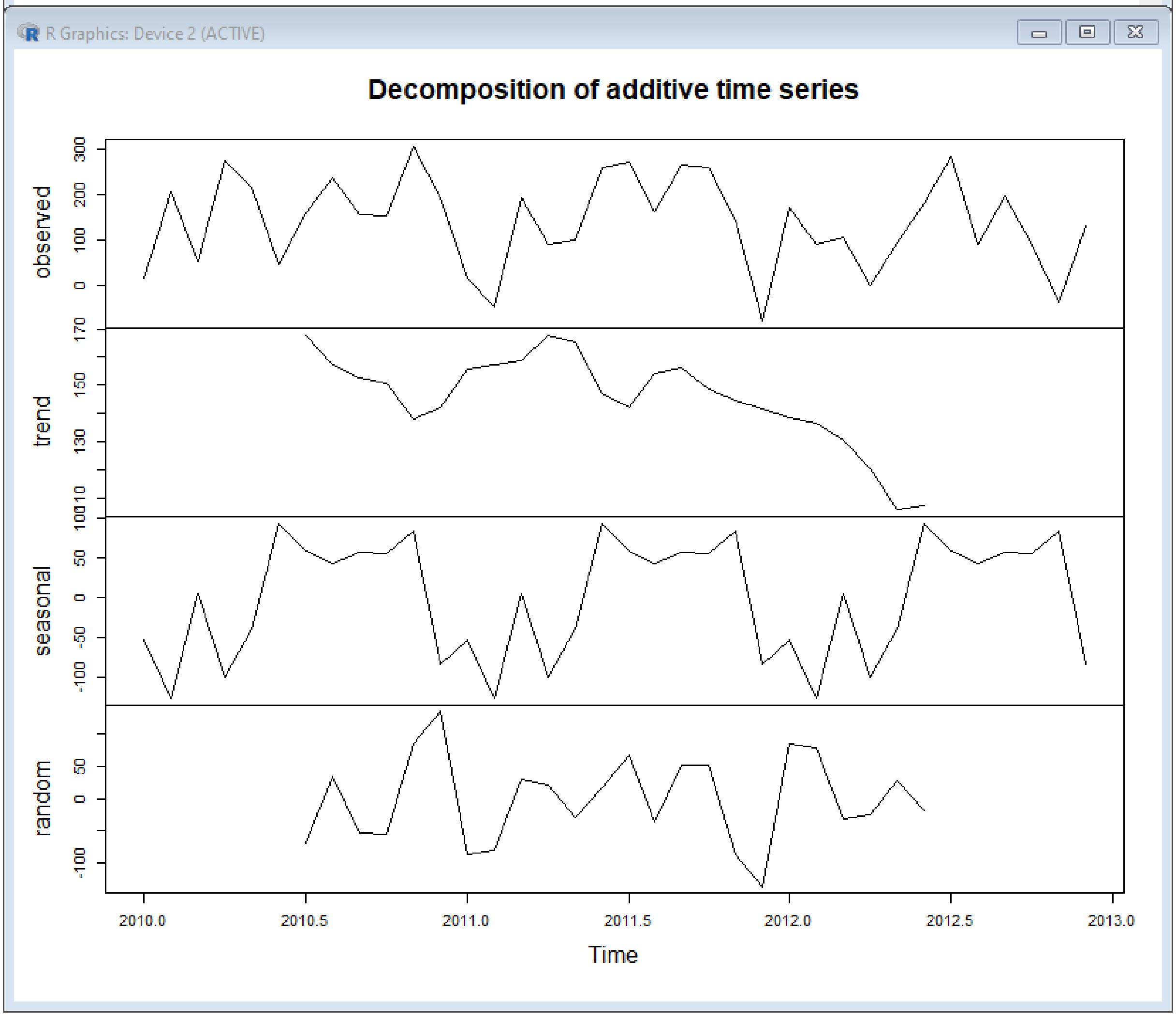
R 3.6.3

Comparons ici les trois méthodes qui nous connues:

- STL (Seasonal Trend LOESS)
- ETS (Error Trend Seasonal)
- TSLM (Trend Seasonal Linear Model)

D'abord générons une série aléatoire et affichons-là:

```
> set.seed(1)
> alea<-round(rnorm(36,10,20),2)
> data<-alea+4*(alea+20)+50*sin(alea*2*pi*1/12)+20*sin(alea*2*pi*1/64)
>
>
> library("forecast")
> data.ts<-ts(data,start=c(2010,1),frequency=12)
> plot(decompose(data.ts))
>
```

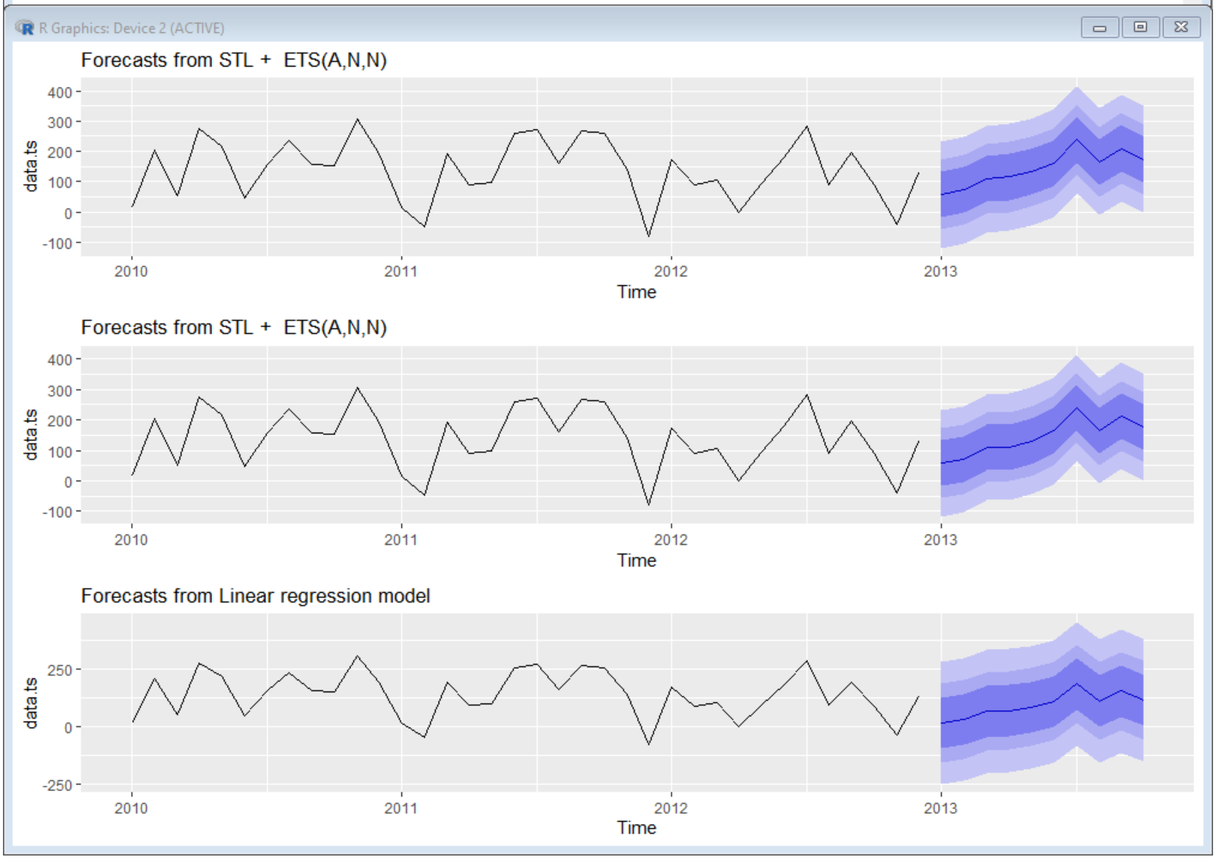


Maintenant comparison numériquement les différences en utilisant respectivement les fonctions **str( )**, **stlf( )** et **tslm( )**:

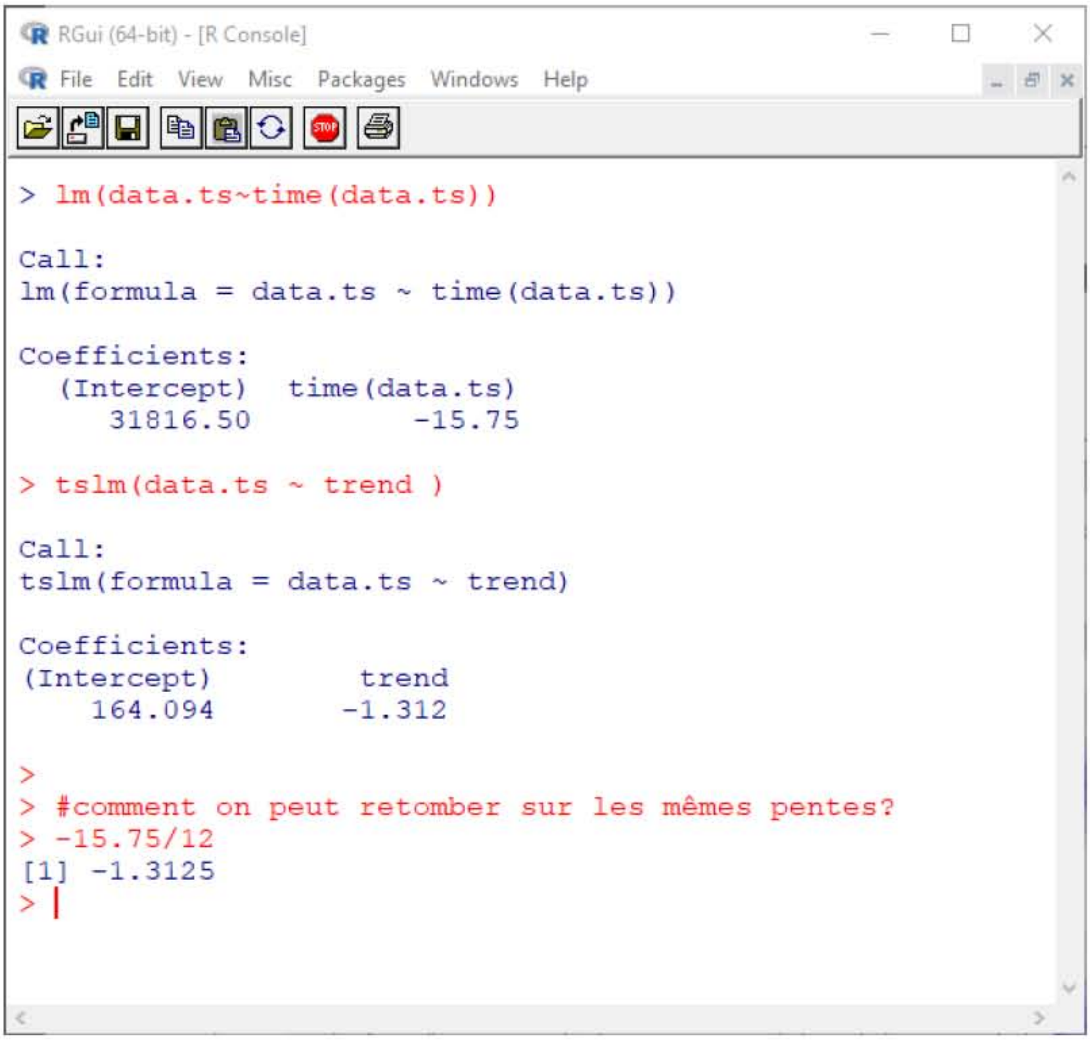
```
> #première approche STL (Seasonal Trend LOESS)
> forecast(stl(data.ts,s.window="periodic"),method="ets",level=c(60,80, 95),h=10)
         Point Forecast      Lo 60    Hi 60       Lo 80    Hi 80       Lo 95    Hi 95
Jan 2013       56.38825 -19.278710 132.0552 -58.8311688 171.6077 -119.824644 232.6011
Feb 2013       73.35512  -2.311839 149.0221 -41.8642982 188.5745 -102.857773 249.5680
Mar 2013      108.54132  32.874364 184.2083  -6.6780957 223.7607  -67.671571 284.7542
Apr 2013      115.51974  39.852779 191.1867   0.3003195 230.7392  -60.693156 291.7326
May 2013      132.61353  56.946575 208.2805  17.3941145 247.8330  -43.599361 308.8264
Jun 2013      160.79500  85.128039 236.4620  45.5755784 276.0144  -15.417898 337.0079
Jul 2013      238.32727 162.660315 313.9942 123.1078544 353.5467   62.114378 414.5402
Aug 2013      165.48245  89.815490 241.1494  50.2630297 280.7019  -10.730447 341.6953
Sep 2013      210.13656 134.469597 285.8035  94.9171363 325.3560   33.923659 386.3495
Oct 2013      174.02163  98.354673 249.6886  58.8022121 289.2411   -2.191265 350.2345
>
> #deuxième approche ETS (Error Trend Seasonal)
> stlf(data.ts, method ="ets",level=c(60,80, 95),h=10)
         Point Forecast      Lo 60    Hi 60      Lo 80    Hi 80       Lo 95    Hi 95
Jan 2013       57.49560 -17.319251 132.3104 -56.426299 171.4175 -116.732908 231.7241
Feb 2013       70.91653  -3.898318 145.7314 -43.005366 184.8384 -103.311976 245.1450
Mar 2013      110.22072  35.405872 185.0356  -3.701176 224.1426  -64.007786 284.4492
Apr 2013      111.59091  36.776060 186.4058  -2.330988 225.5128  -62.637599 285.8194
May 2013      130.74136  55.926508 205.5562  16.819459 244.6633  -43.487152 304.9699
Jun 2013      163.48888  88.674029 238.3037  49.566980 277.4108  -10.739631 337.7174
Jul 2013      240.76779 165.952943 315.5826 126.845894 354.6897   66.539282 414.9963
Aug 2013      165.27361  90.458755 240.0885  51.351706 279.1955   -8.954906 339.5021
Sep 2013      213.44924 138.634389 288.2641  99.527340 327.3711   39.220728 387.6778
Oct 2013      175.57416 100.759310 250.3890  61.652260 289.4961    1.345648 349.8027
>
> #troisième TSLM (Trend Seasonal Linear Model)
> forecast(tslm(data.ts ~ trend + season),level=c(60,80, 95),h=10)
         Point Forecast       Lo 60    Hi 60      Lo 80    Hi 80      Lo 95    Hi 95
Jan 2013       14.98316 -95.9492471 125.9156 -155.70588 185.6722 -252.62413 282.5904
Feb 2013       30.84385 -80.0885549 141.7762 -139.84519 201.5329 -236.76344 298.4511
Mar 2013       64.92373 -46.0086758 175.8561 -105.76531 235.6128 -202.68356 332.5310
Apr 2013       69.63122 -41.3011780 180.5636 -101.05781 240.3203 -197.97606 337.2385
May 2013       84.45405 -26.4783504 195.3865  -86.23499 255.1431 -183.15324 352.0613
Jun 2013      110.44419  -0.4882095 221.3766  -60.24484 281.1332 -157.16310 378.0515
Jul 2013      185.78525  74.8528434 296.7176   15.09621 356.4743  -81.82204 453.3925
Aug 2013      111.21461   0.2822123 222.1470  -59.47442 281.9037 -156.39267 378.8219
Sep 2013      154.14287  43.2104636 265.0753  -16.54617 324.8319 -113.46442 421.7502
Oct 2013      114.60794   3.6755344 225.5403  -56.08110 285.2970 -152.99935 382.2152
>
```

Soit graphiquement:

```
> library("egg")
> p1<-autoplot(forecast(stl(data.ts,s.window="periodic"),method="ets",level=c(60,80, 95),h=10))
> p2<-autoplot(stlf(data.ts, method ="ets",level=c(60,80, 95),h=10))
> p3<-autoplot(forecast(tslm(data.ts ~ trend + season),level=c(60,80, 95),h=10))
> grid.arrange(p1, p2, p3, ncol = 1)
>
```



Maintenant répodons à une question fréquente entre le TSLM et la simple régression:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> lm(data.ts~time(data.ts))
Call:
lm(formula = data.ts ~ time(data.ts))
Coefficients:
(Intercept) time(data.ts)
31816.50 -15.75
> tslm(data.ts ~ trend)
Call:
tslm(formula = data.ts ~ trend)
Coefficients:
(Intercept) trend
164.094 -1.312
>
> #comment on peut retomber sur les mêmes pentes?
> -15.75/12
[1] -1.3125
>

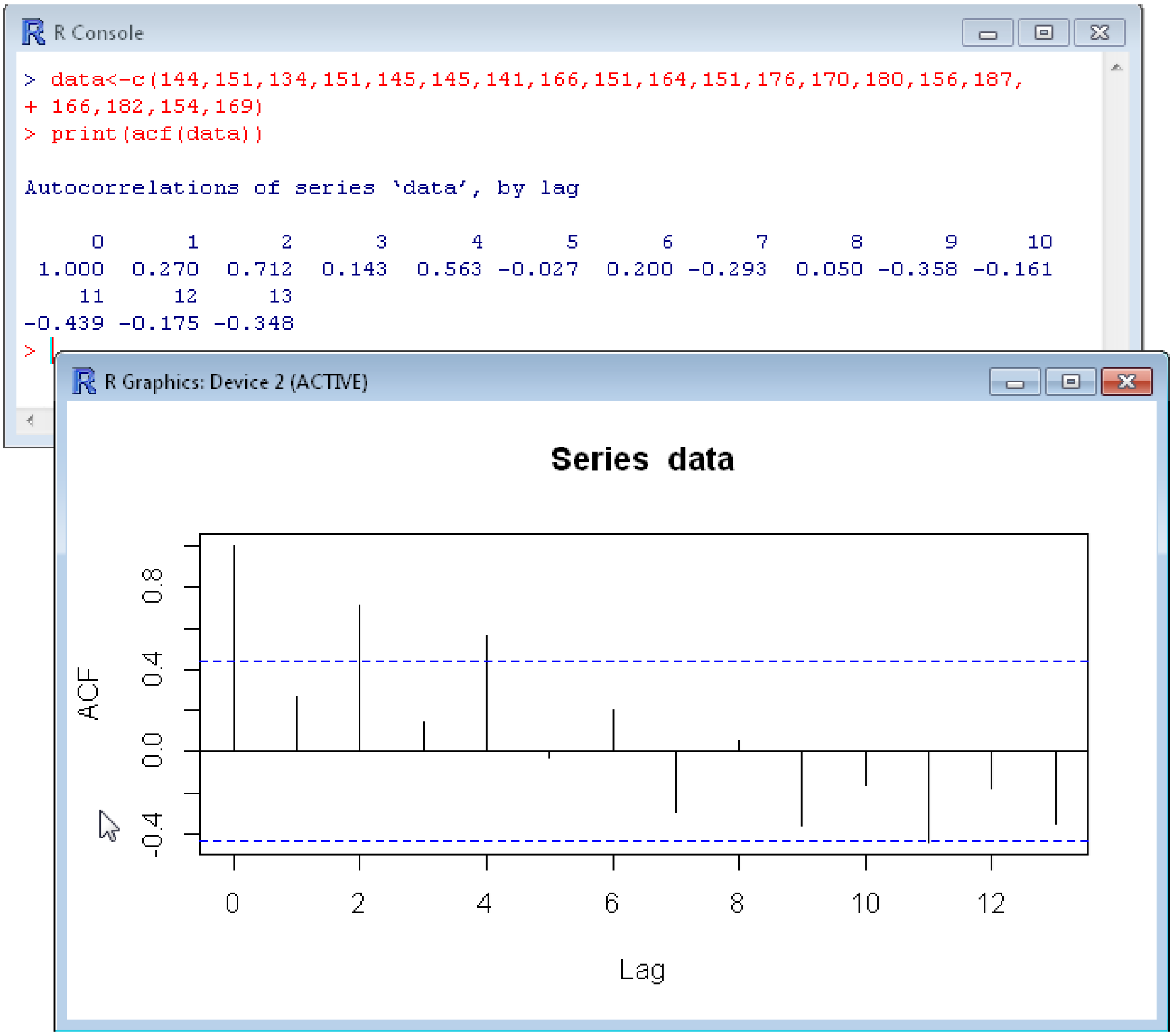
Exemples 383.: Modèles univariés AR(p), ARIMA(p,d,q) et ARCH(p,q)

R 3.0.2

Coefficients d'autocorrélation

Nous allons ici calculer les coefficients d'autocorrélation avec les mêmes données que dans le cours théorique après avoir donné la définition (intuitive) de l'autocorrélation.

Nous utilisons alors la commande **acf( )** native de R pour obtenir:

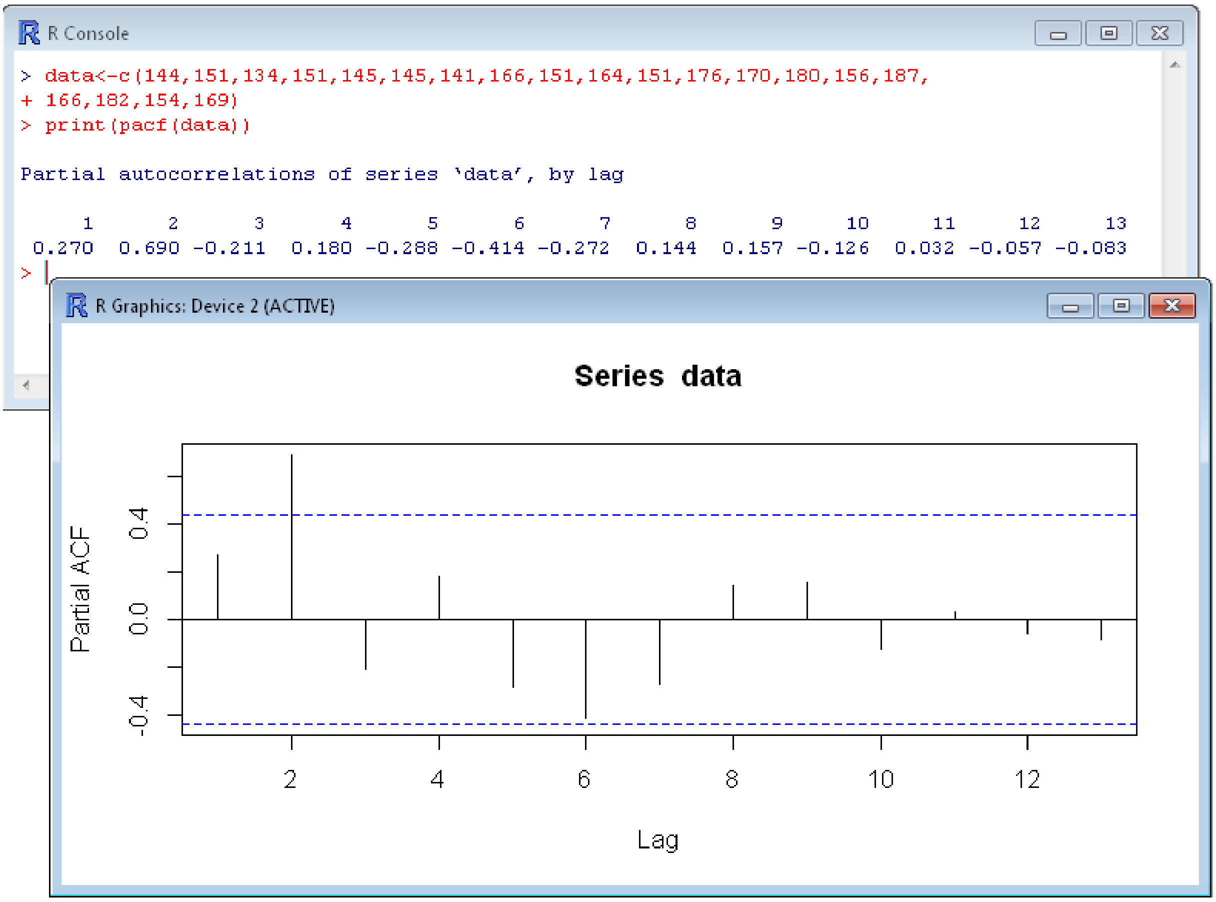


Nous retrouvons donc exactement les mêmes valeurs que celles calculées à la main et que dans le cours Minitab!

Coefficients d'autocorrélation partielle

Nous allons ici calculer les coefficients d'autocorrélation avec les mêmes données que dans le cours théorique après avoir donné la définition (intuitive) de l'autocorrélation.

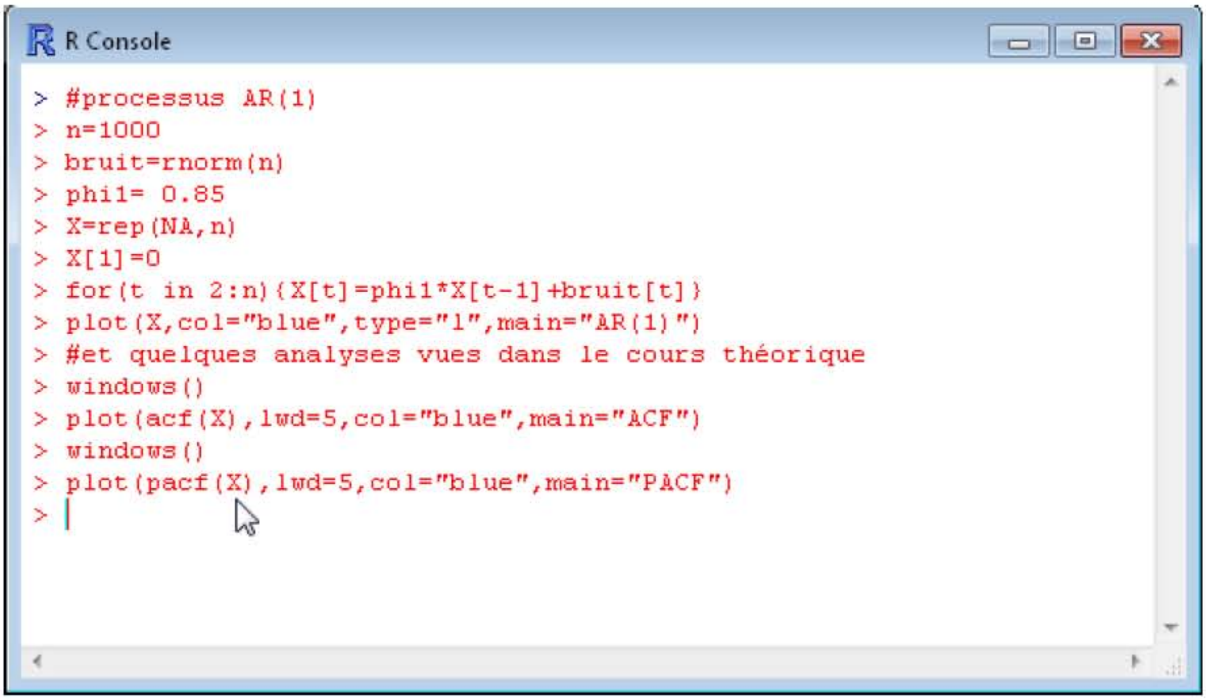
Nous utilisons alors la commande **acf( )** native de R pour obtenir:

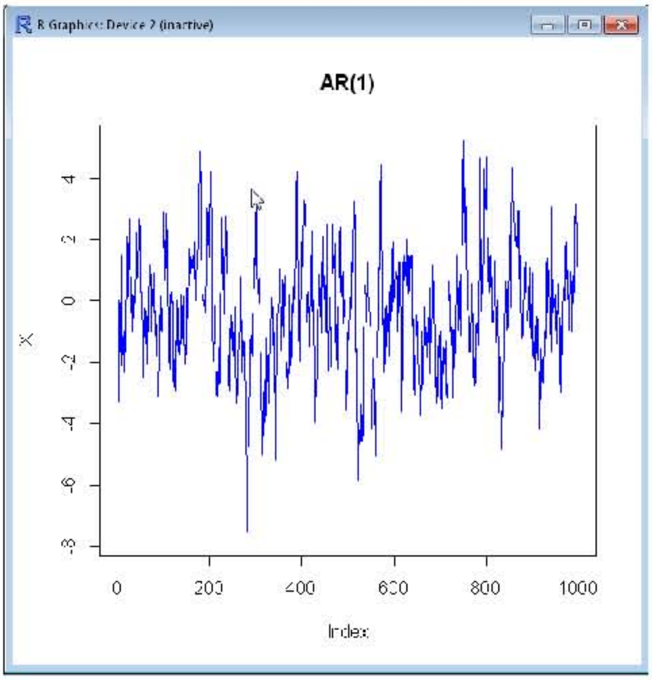


Nous retrouvons donc exactement les mêmes valeurs que celles calculées à la main et que dans le cours Minitab!

Simulation d'un processus AR(1)

Juste une petite mise en bouche...





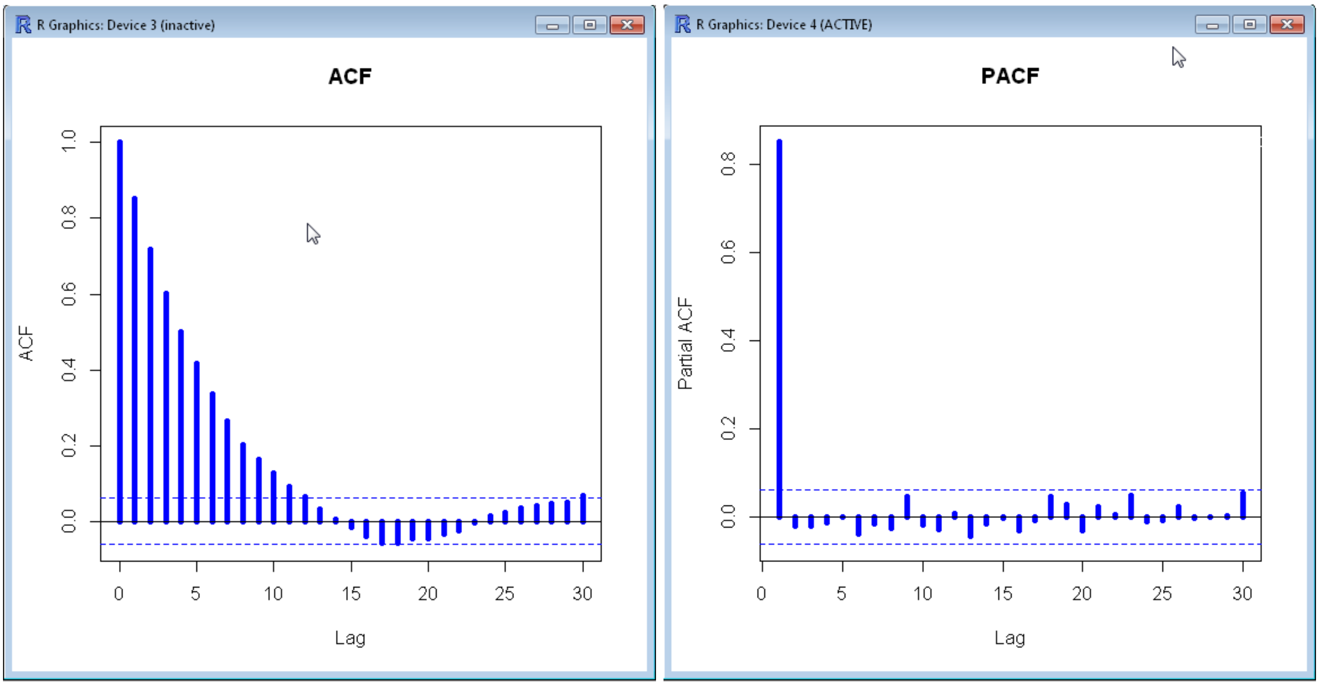
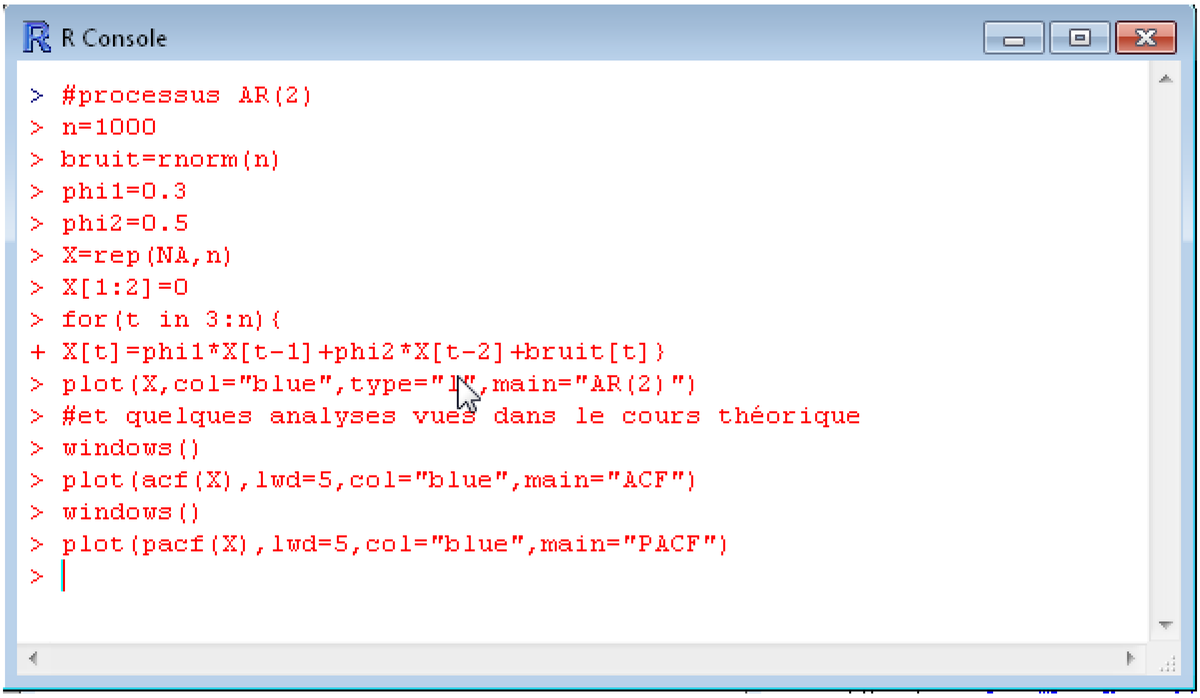


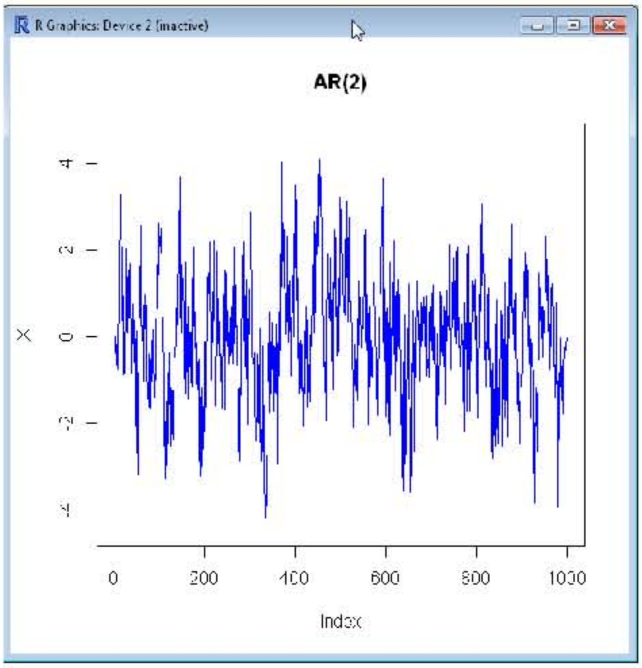
Figure 311 Graph ACF et PACF de séries temporelles

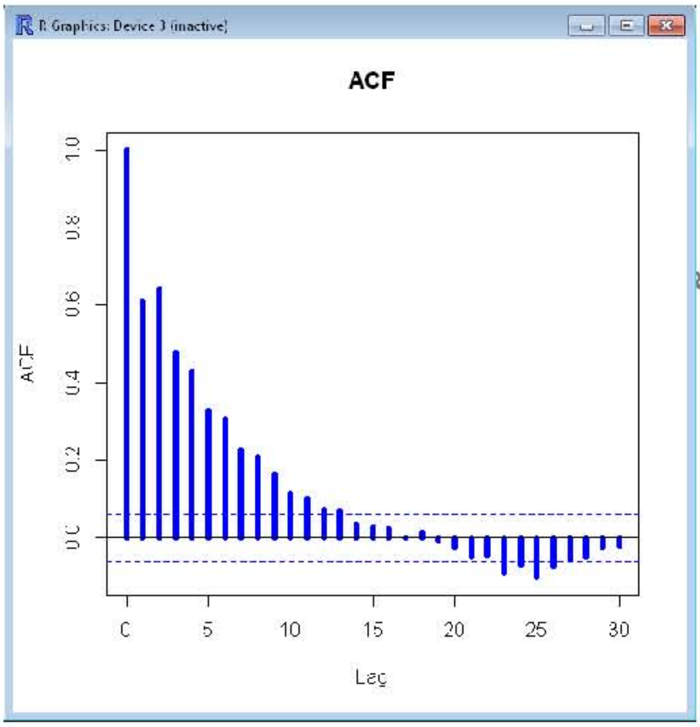
Simulation d'un processus AR(2)

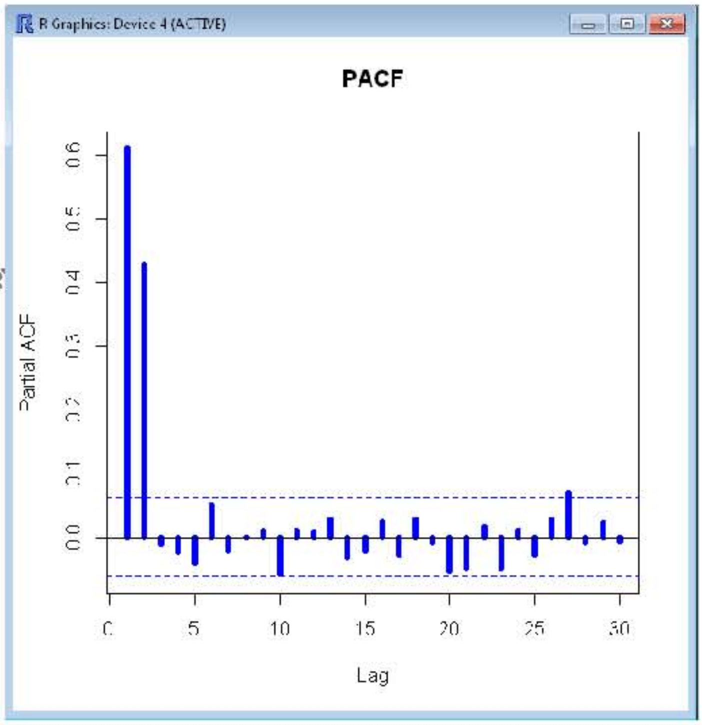
Toujours dans la même idée pour montrer comment le principe de l'exemple précédent est extensible:



```
> #processus AR(2)
> n=1000
> bruit=rnorm(n)
> phi1=0.3
> phi2=0.5
> X=rep(NA,n)
> X[1:2]=0
> for(t in 3:n){
+ X[t]=phi1*X[t-1]+phi2*X[t-2]+bruit[t]}
> plot(X,col="blue",type="l",main="AR(2)")
> #et quelques analyses vues dans le cours théorique
> windows()
> plot(acf(X),lwd=5,col="blue",main="ACF")
> windows()
> plot(pacf(X),lwd=5,col="blue",main="PACF")
>
```

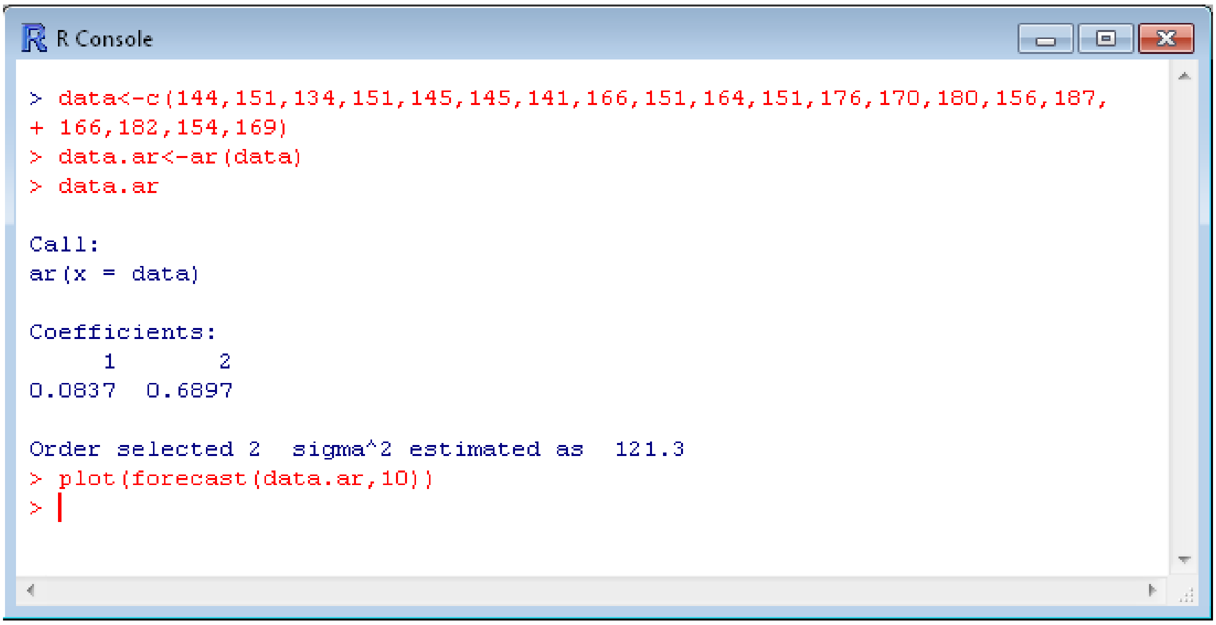

R Graphics: Device 2 (inactive)
AR(2)
X
Index
0
200
400
600
800
1000


R Graphics: Device 3 (inactive)
ACF
ACF
Lag


R Graphics: Device 4 (ACTIVE)
PACF
Partial ACF
Lag

Ajuster un modèle AR(p)/ARIMA(p,0,0)

Commençons par le modèle AR le plus simple:



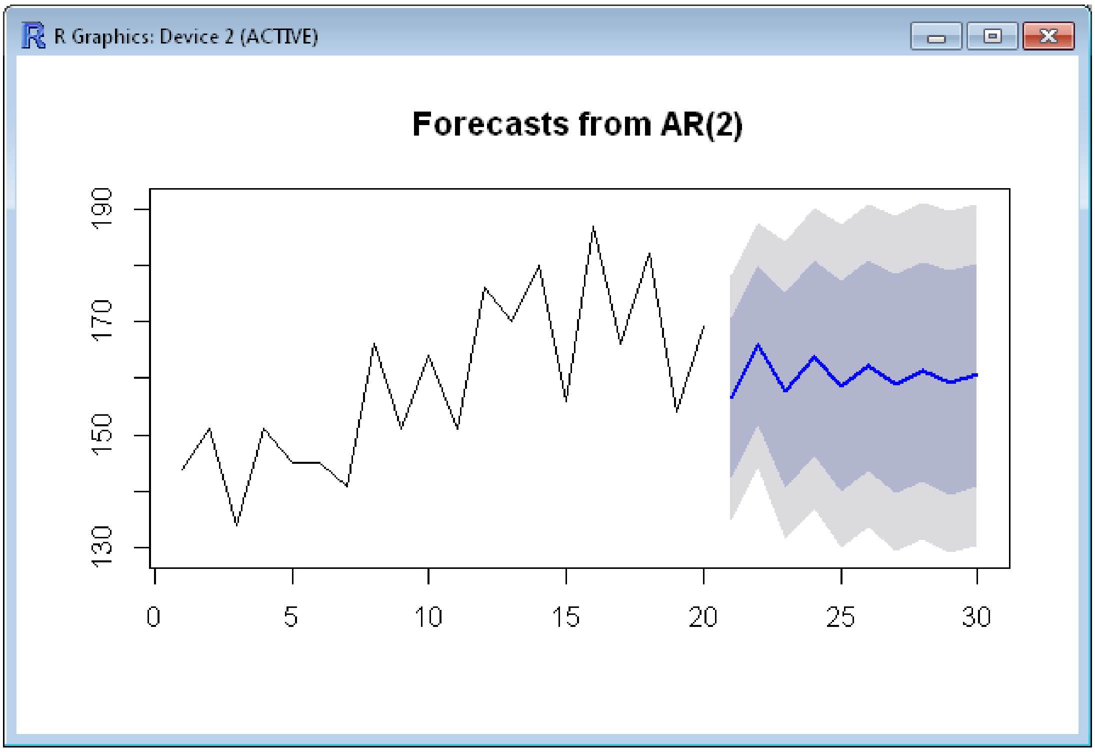
```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.ar<-ar(data)
> data.ar

Call:
ar(x = data)

Coefficients:
     1       2
0.0837  0.6897

Order selected 2  sigma^2 estimated as  121.3
> plot(forecast(data.ar,10))
>
```

Donc selon R le meilleur modèle semble être un modèle AR(2). Graphiquement cela donne:



et pour les résidus:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,0,0))
> data.arima$residuals
Time Series:
Start = 1
End = 20
Frequency = 1
 [1] -15.15  -8.15 -25.15  -8.15 -14.15 -14.15 -18.15   6.85  -8.15   4.85
[11]  -8.15  16.85  10.85  20.85  -3.15  27.85   6.85  22.85  -5.15   9.85
> sum(data.arima$residuals^2)
[1] 4242.55
> 
```

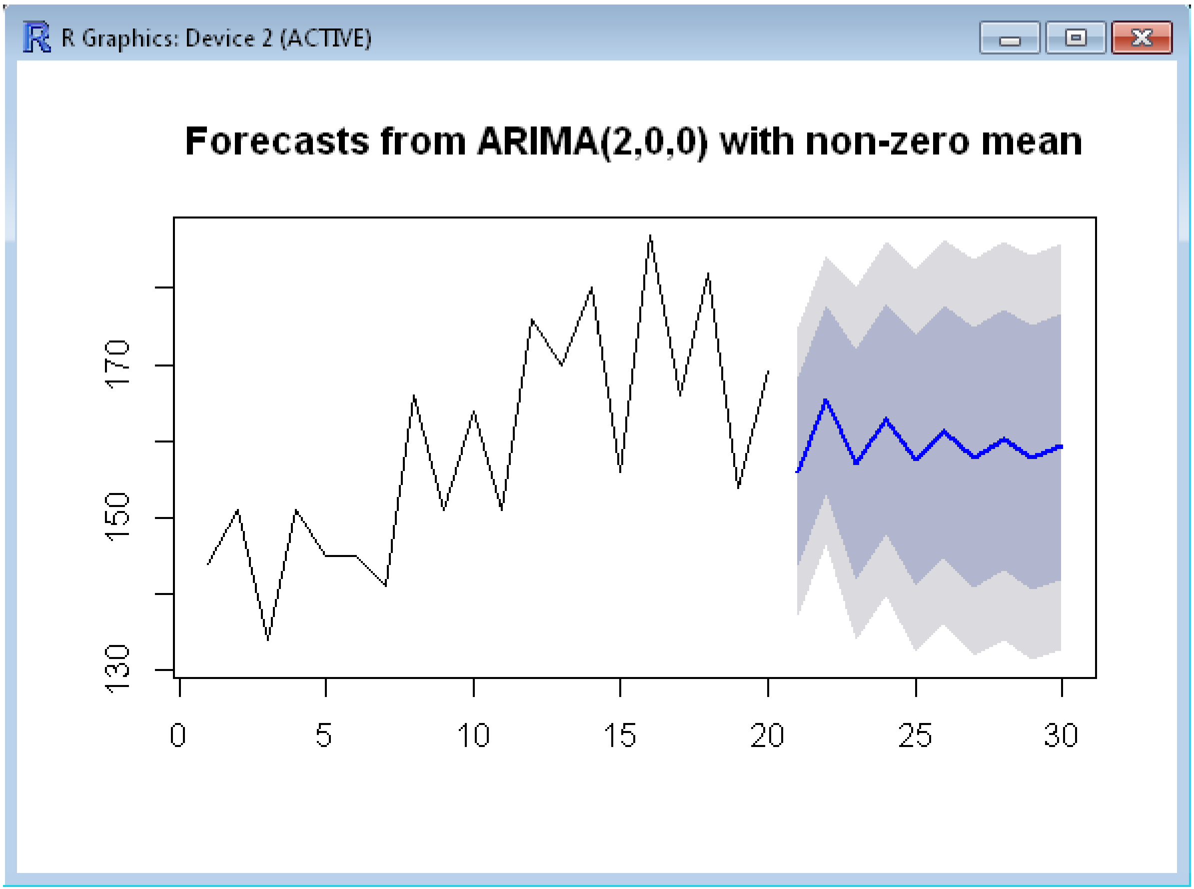
L'équivalent d'un AR(2) est bien évidemment à un ARIMA(2,0,0) et en général un AR(p) a un équivalent ARIMA(p,d,q). Voyons cela:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(2,0,0))
> data.arima
Series: data
ARIMA(2,0,0) with non-zero mean

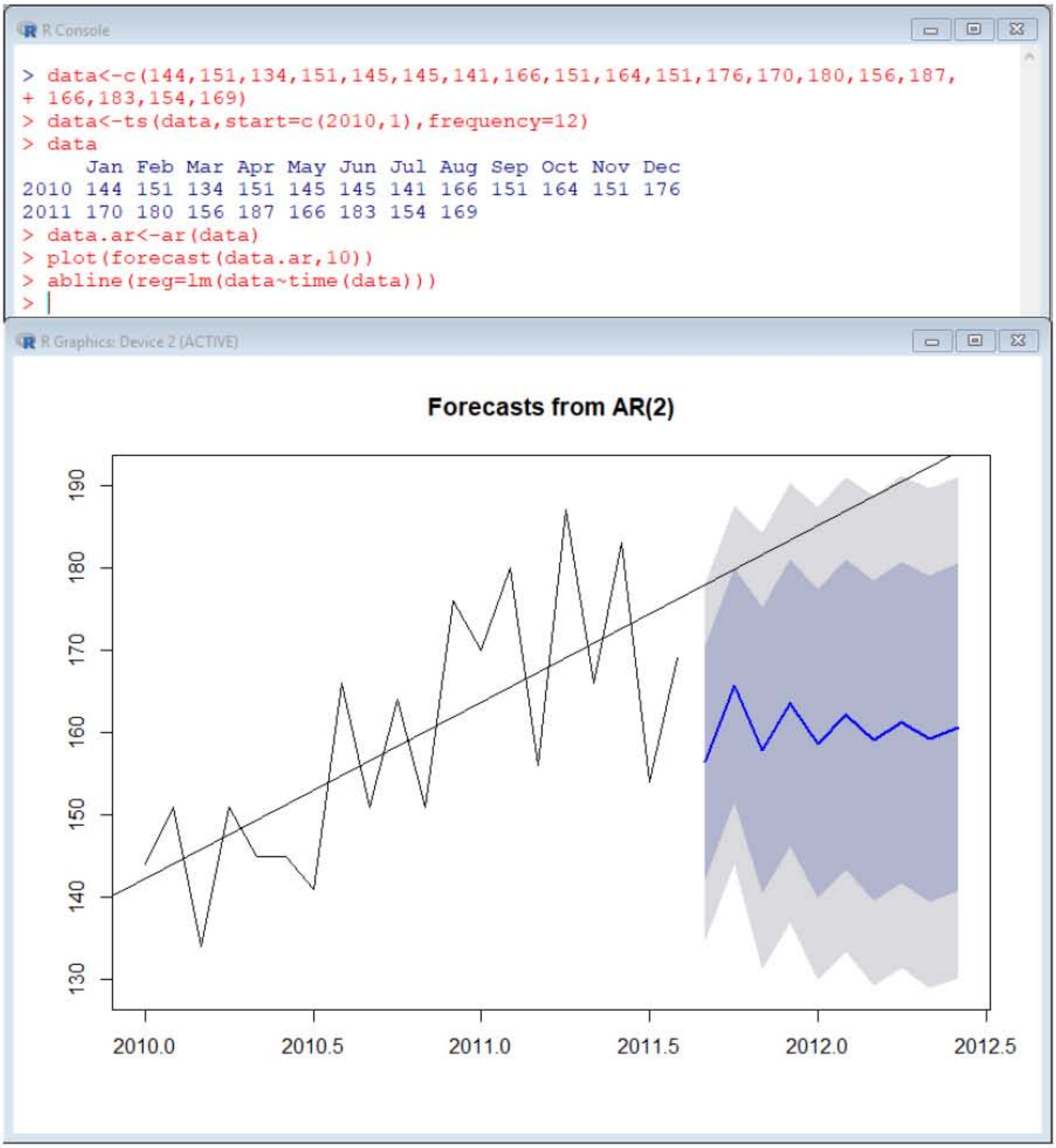
Coefficients:
         ar1     ar2  intercept
      0.0788  0.6974   157.3461
s.e.  0.1413  0.1429     7.5098

sigma^2 estimated as 92.18:  log likelihood=-74.32
AIC=156.63   AICc=159.3   BIC=160.62
> library(forecast)
> plot(forecast(data.arima,10))
> 
```

Les coefficients sont quand même significativement différents (environ 10%). Au niveau graphique, cela nous donne quelque chose par contre de très semblable:

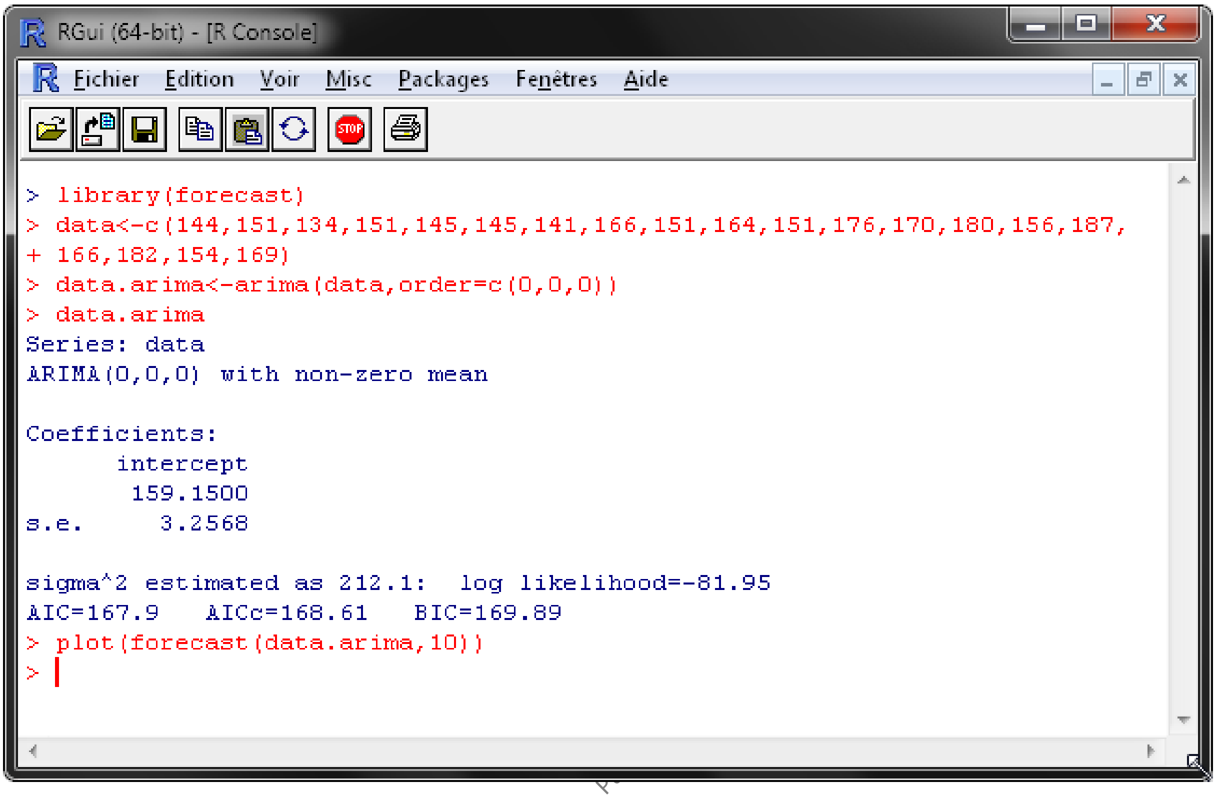


Enfin, suite à la demande d'un client, si on veut comparer ce modèle avec une simple régression linéaire, on utilisera:

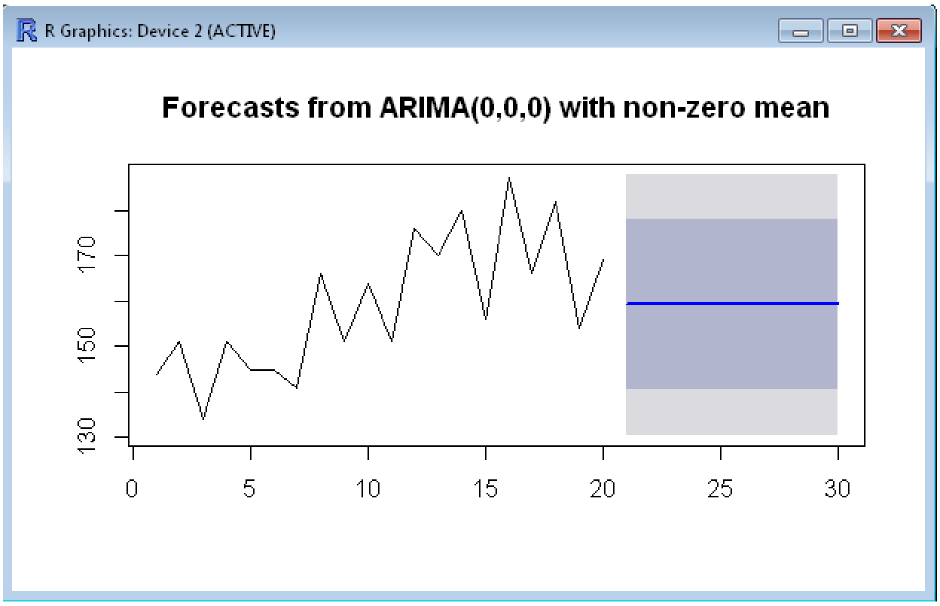

R Console
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,183,154,169)
> data<-ts(data,start=c(2010,1),frequency=12)
> data
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2010 144 151 134 151 145 145 141 166 151 164 151 176
2011 170 180 156 187 166 183 154 169
> data.ar<-ar(data)
> plot(forecast(data.ar,10))
> abline(reg=lm(data~time(data)))
>
R Graphics: Device 2 (ACTIVE)
Forecasts from AR(2)
130
140
150
160
170
180
190
2010.0
2010.5
2011.0
2011.5
2012.0
2012.5

Ajuster un modèle ARIMA(0,0,0)/Moyenne mobile

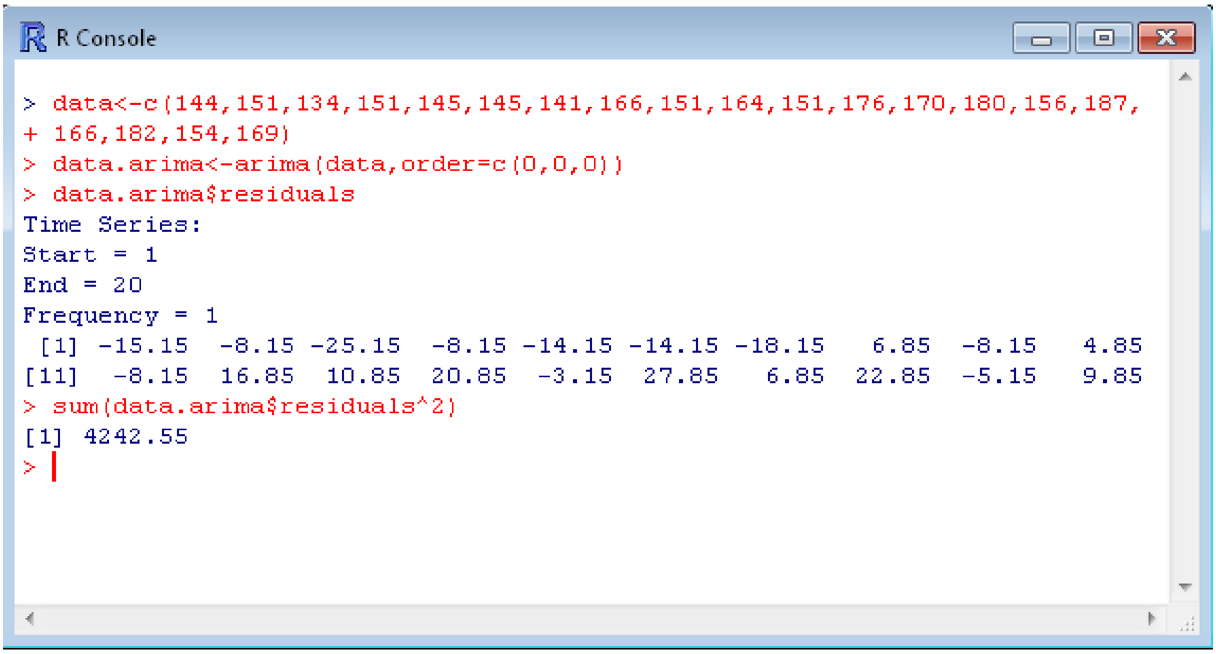
Maintenant passons au modèle par le modèle ARIMA(0,0,0) donc de type moyenne simple. Cela nous donne:



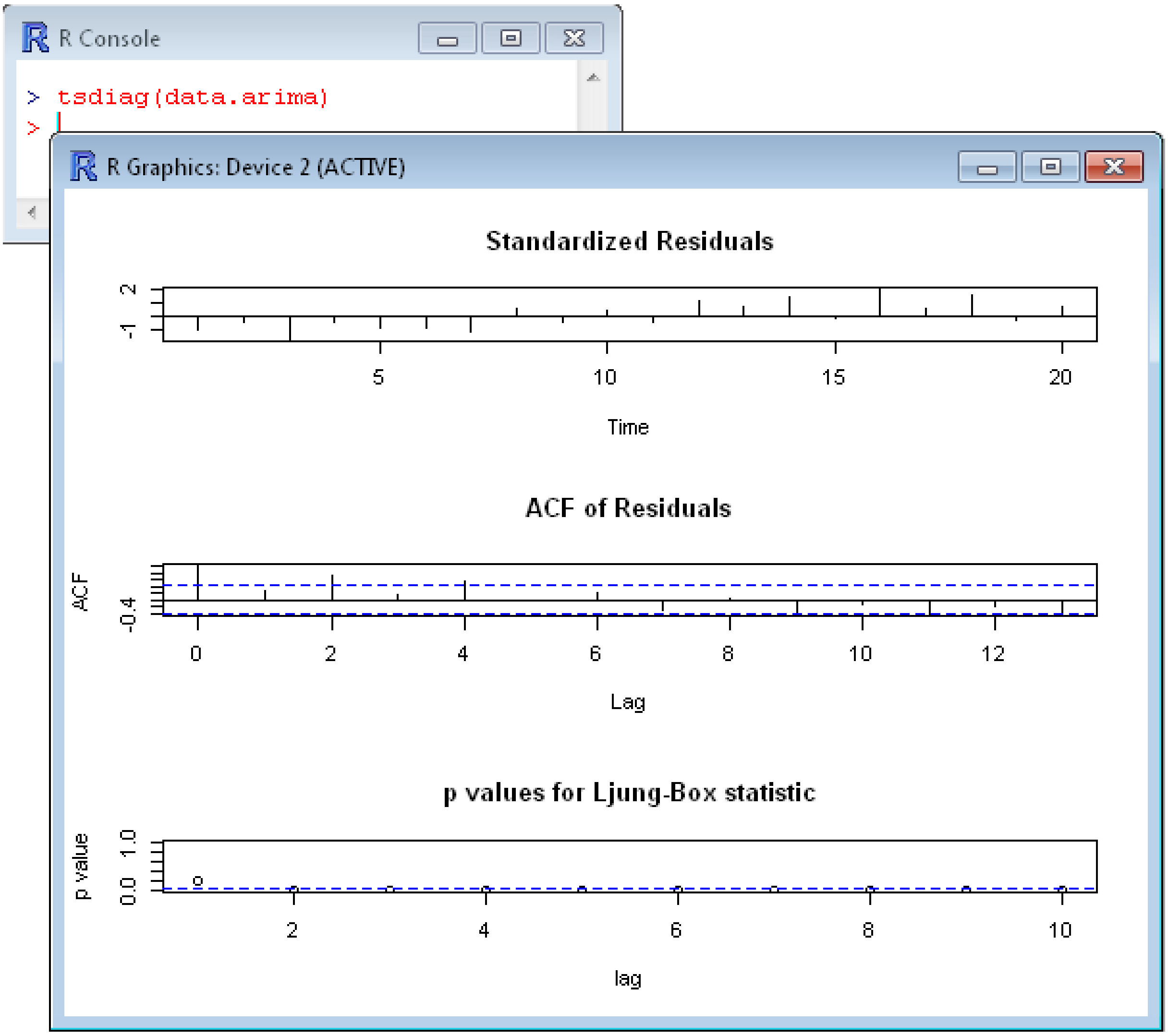
et le graphique associé (donc rien de surprenant dans les résultats obtenus):



et une petite analyse des résidus:



avec la command native **ts.diag( )** nous pouvons avoir quelques informations équivalents sous forme graphique et d'autres qui sont complémentaires:

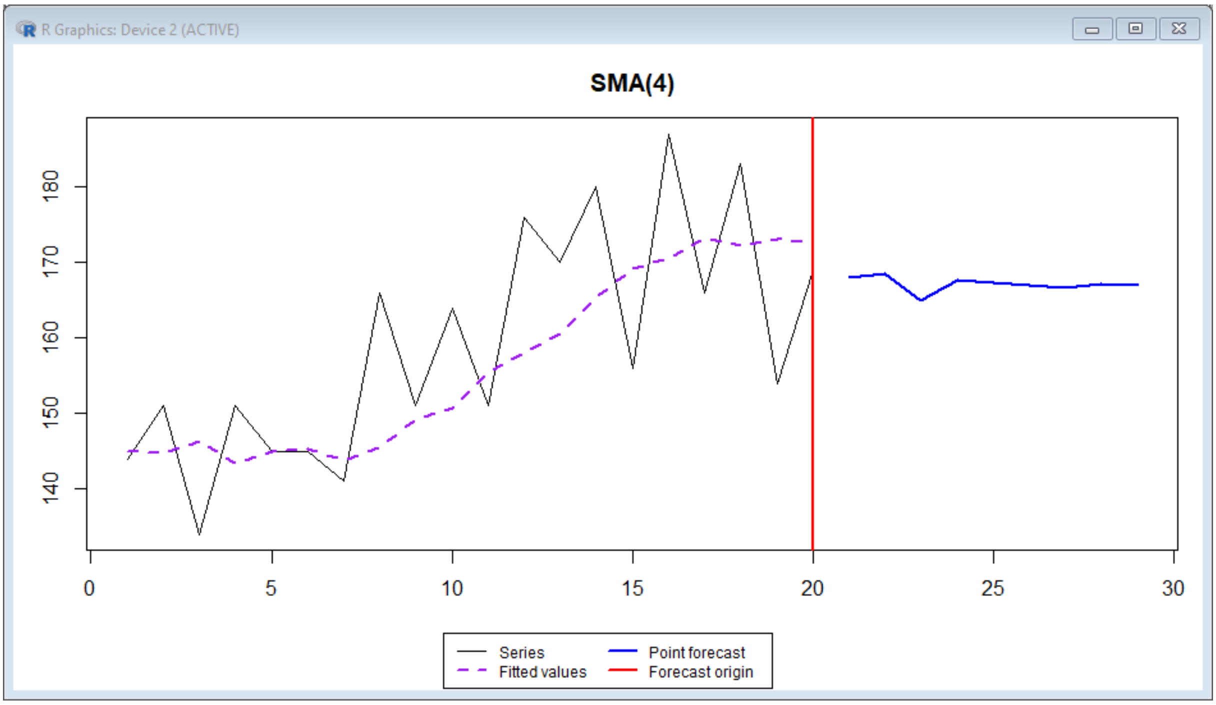


Sinon il y a aussi le package **smooth** et sa fonction **sma( )**:

```
> library("smooth")
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,183,154,169)
> (MA<-sma(data,h=9,silent=FALSE))
Time elapsed: 0.02 seconds
Model estimated: SMA(4)
Initial values were produced using backcasting.

Loss function type: MSE; Loss function value: 124.7062
Error standard deviation: 11.1672
Sample size: 20
Number of estimated parameters: 2
Number of degrees of freedom: 18
Information criteria:
     AIC     AICc      BIC     BICc
157.2768 157.9826 159.2682 160.3255
> cbind(MA$fitted,MA$forecast)
Time Series:
Start = 1
End = 29
Frequency = 1
   MA$fitted MA$forecast
 1    145.00          NA
 2    144.75          NA
 3    146.25          NA
 4    143.50          NA
 5    145.00          NA
 6    145.25          NA
 7    143.75          NA
 8    145.50          NA
 9    149.25          NA
10    150.75          NA
11    155.50          NA
12    158.00          NA
13    160.50          NA
14    165.25          NA
15    169.25          NA
16    170.50          NA
17    173.25          NA
18    172.25          NA
19    173.00          NA
20    172.50          NA
21        NA    168.0000
22        NA    168.5000
23        NA    164.8750
24        NA    167.5938
25        NA    167.2422
26        NA    167.0527
27        NA    166.6909
28        NA    167.1449
29        NA    167.0327
```

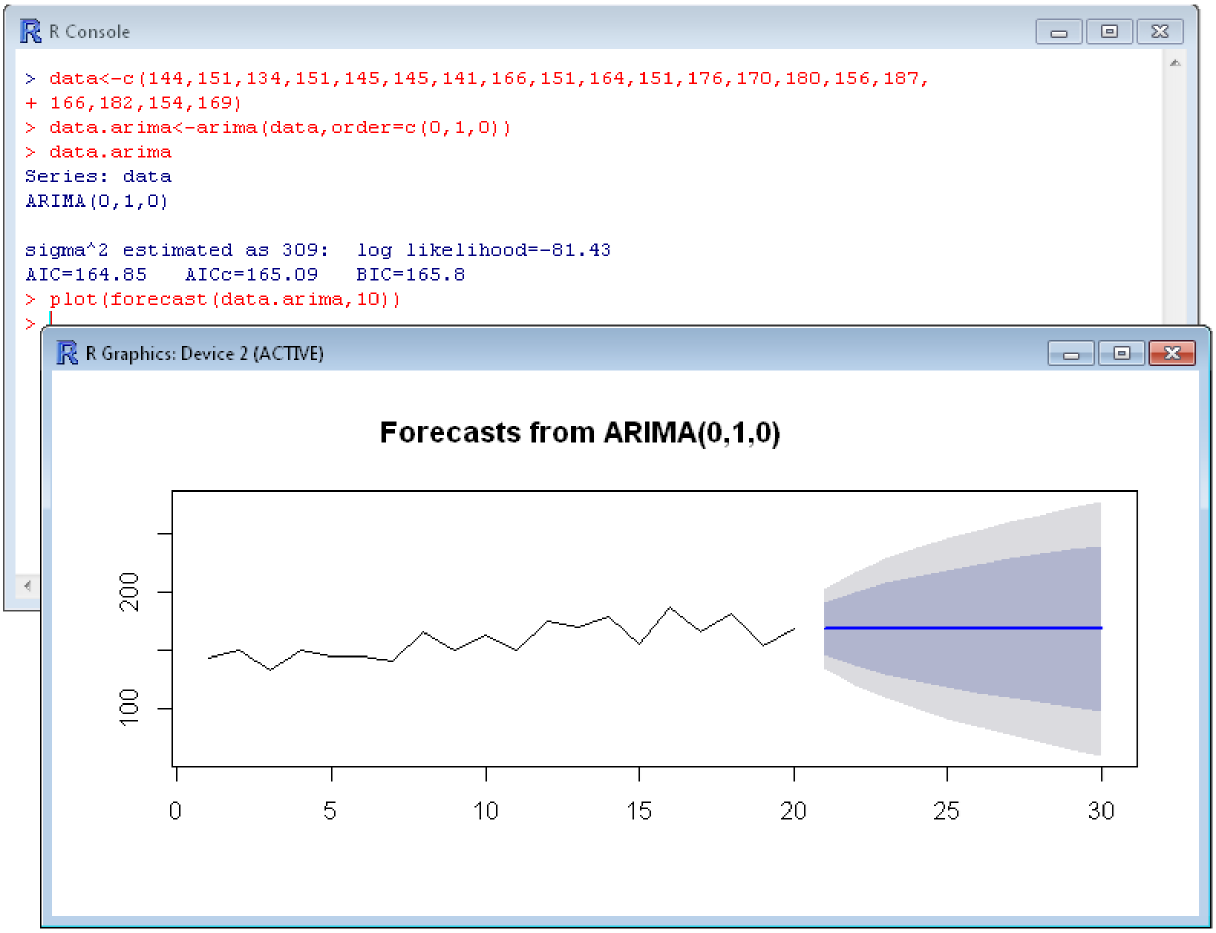
Et le graphique correspondant:



On voit que les résultats sont (sans trop de surprises) significativement différents de ceux obtenus précédemment...

Ajuster un modèle ARIMA(0,1,0)

Ensuite, dans la complexité croissante, voyons le modèle ARIMA(0,1,0) naïf:



et une petite analyse des résidus:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,1,0))
> data.arima$residuals
Time Series:
Start = 1
End = 20
Frequency = 1
 [1]   0.1439999   7.0000000 -17.0000000  17.0000000  -6.0000000
 [6]   0.0000000  -4.0000000  25.0000000 -15.0000000  13.0000000
[11] -13.0000000  25.0000000  -6.0000000  10.0000000 -24.0000000
[16]  31.0000000 -21.0000000  16.0000000 -28.0000000  15.0000000
> sum(data.arima$residuals^2)
[1] 5871.021
>
```

et toujours avec **tsdiag( )** :

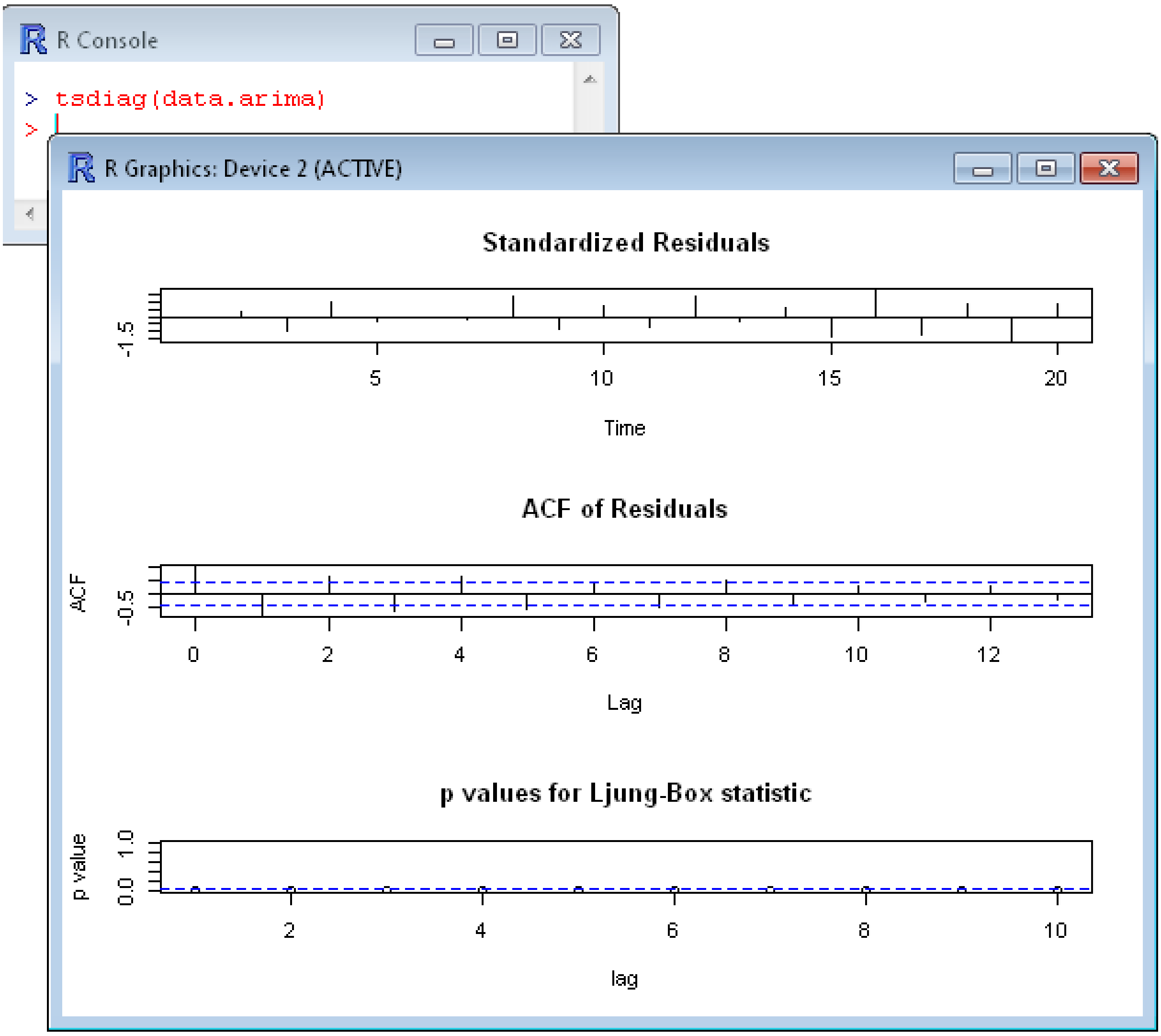
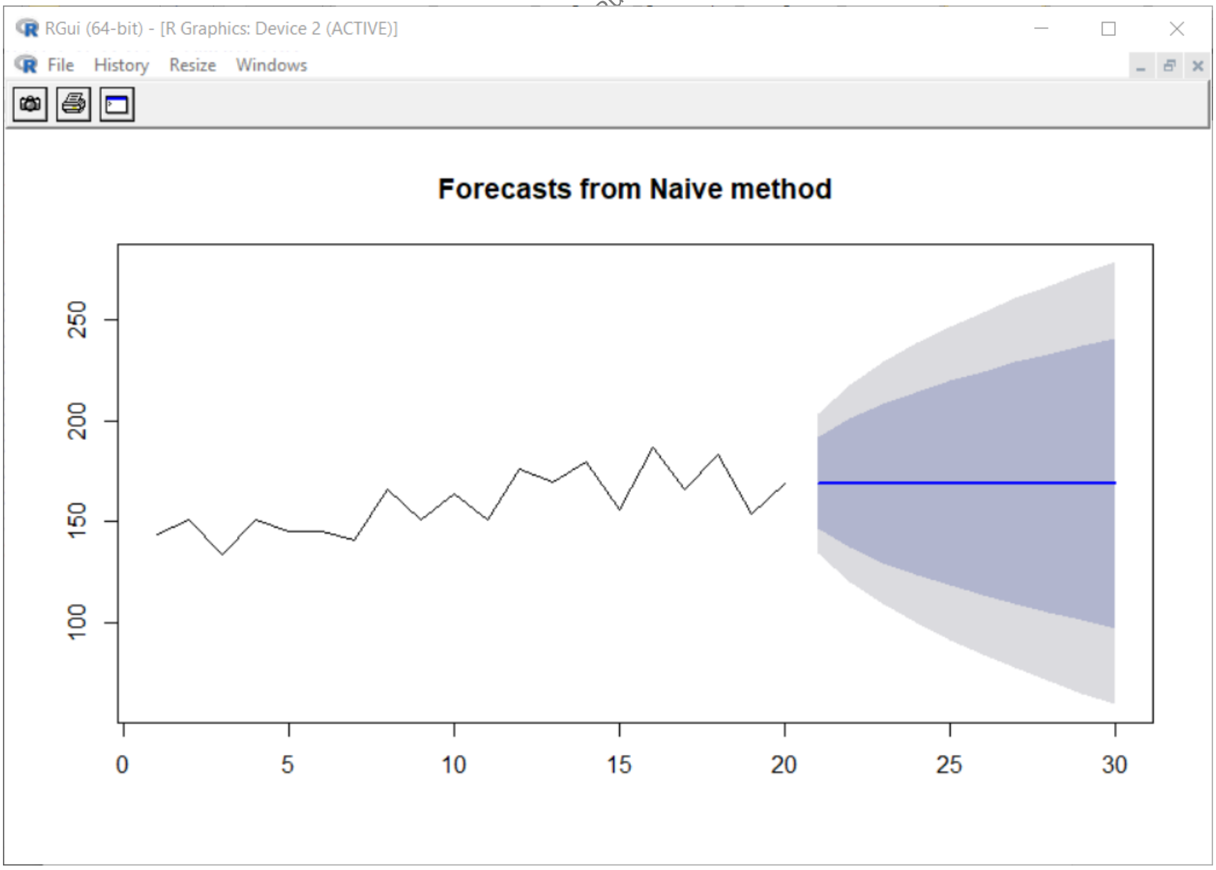


Figure 312 Graph des résidus standardisé, ACF des résidus et p-valeurs de la statistique de Ljung-Box

On peut obtenir le même résultat avec la fonction **naive( )** du package **forecast**:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,183,154,169)
> library("forecast")
> naive(data,h=10)
   Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
21            169 146.30033 191.6997 134.28385 203.7162
22            169 136.89781 201.1022 119.90394 218.0961
23            169 129.68301 208.3170 108.86986 229.1301
24            169 123.60065 214.3993  99.56769 238.4323
25            169 118.24199 219.7580  91.37232 246.6277
26            169 113.39738 224.6026  83.96314 254.0369
27            169 108.94231 229.0577  77.14969 260.8503
28            169 104.79563 233.2044  70.80789 267.1921
29            169 100.90098 237.0990  64.85154 273.1485
30            169  97.21733 240.7827  59.21788 278.7821
> plot(naive(data,h=10))
> 
```

Ce qui donne:



Ajuster un modèle ARIMA(0,1,1)/Lissage exponentiel simple

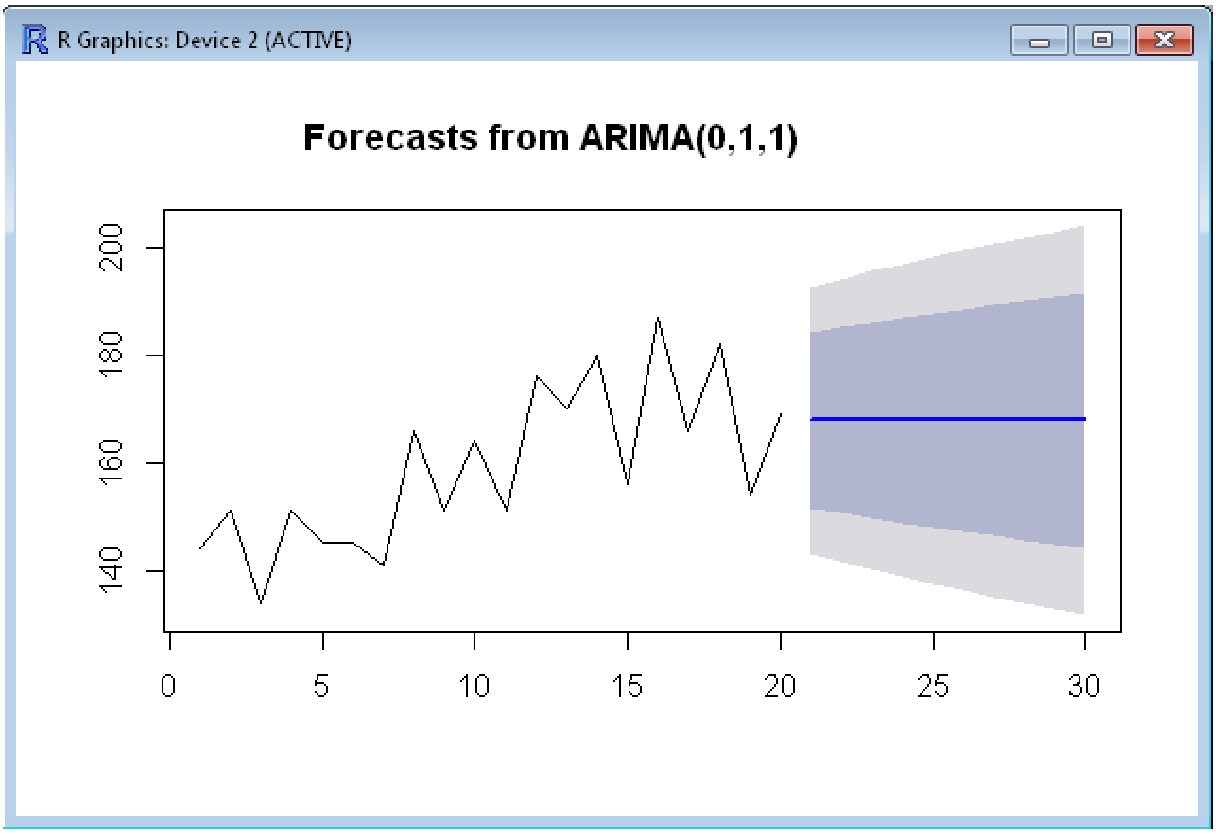
Toujours en complexité croissante, voyons maintenant le modèle à lissage exponentiel simple ARIMA(0,1,1):

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,1,1))
> data.arima
Series: data
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.6457
s.e.   0.1307

sigma^2 estimated as 160:  log likelihood=-75.45
AIC=154.89   AICc=155.64   BIC=156.78
> plot(forecast(data.arima,10))
>
```

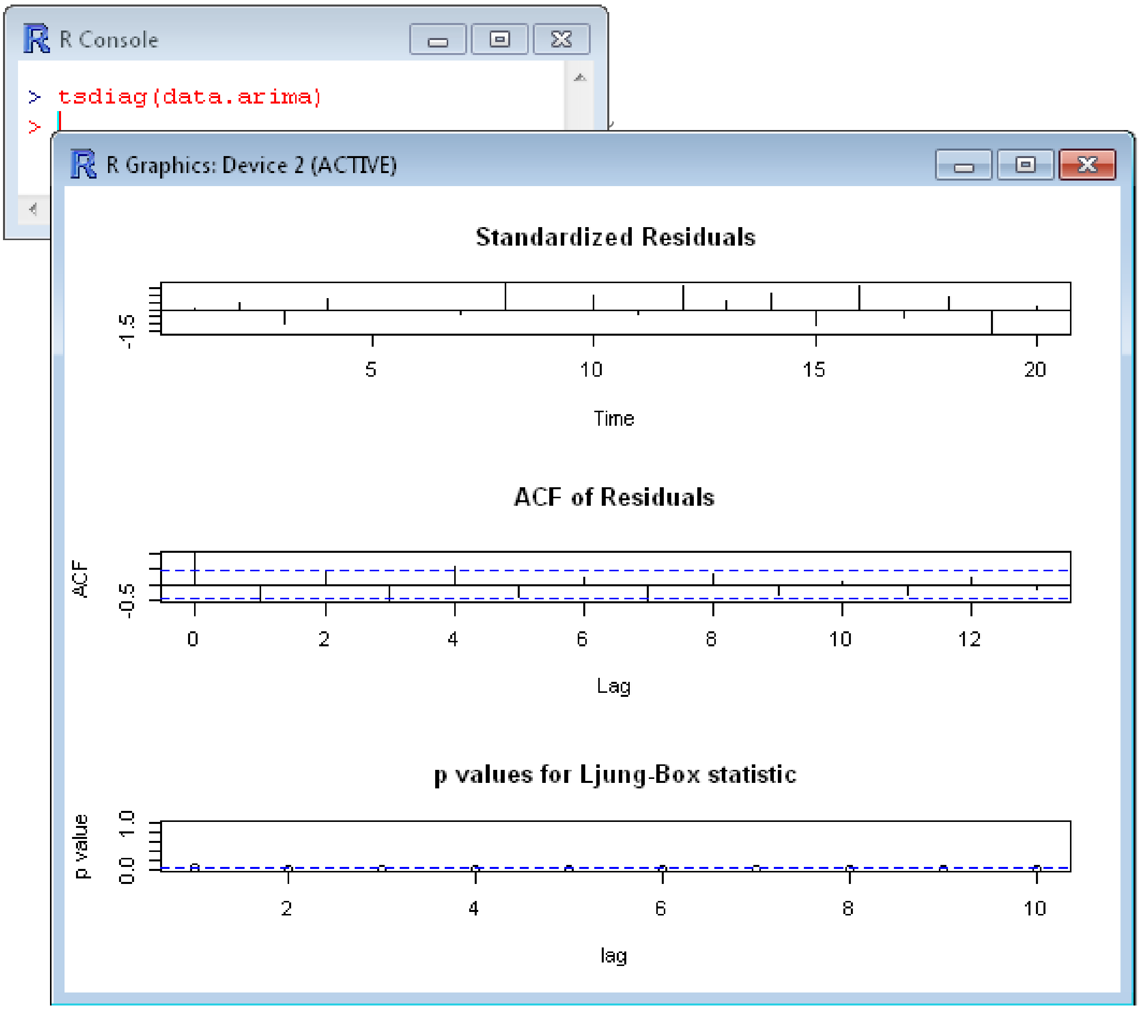
Nous retrouvons donc bien la valeur de alpha vue plus haut (au signe près). Le graphique associé donne:



et une petite analyse des résidus:

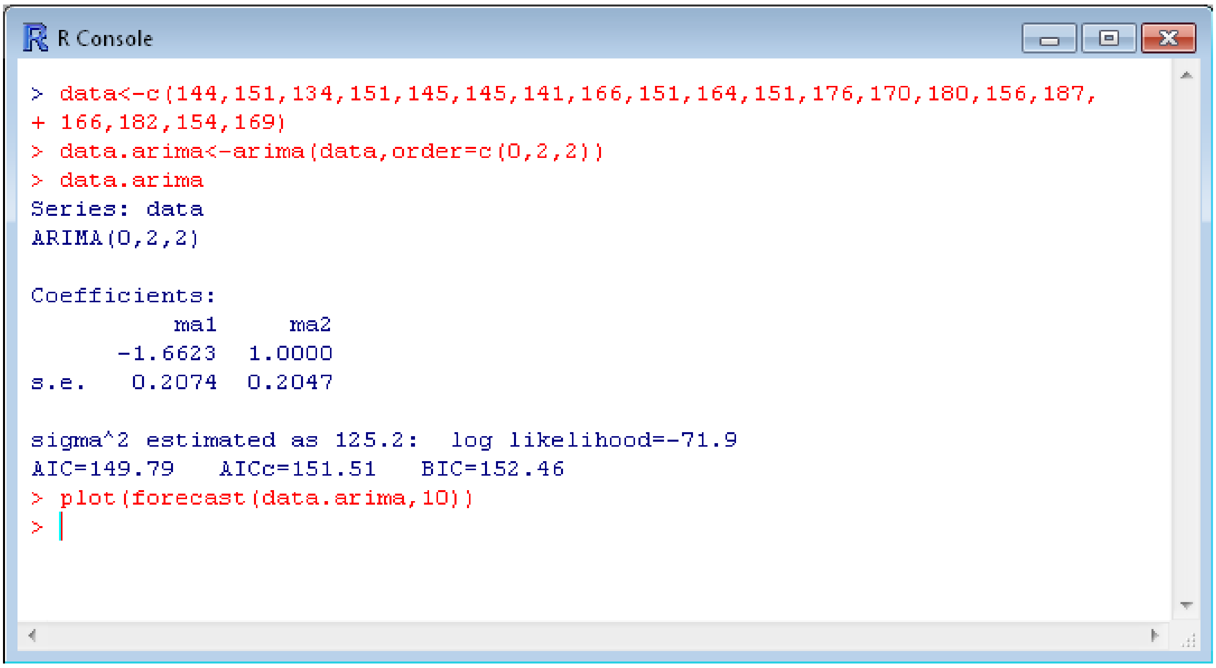
```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,1,1))
> data.arima$residuals
Time Series:
Start = 1
End = 20
Frequency = 1
 [1]   0.1439999   5.8806607 -13.0335553   8.8575980  -0.4028543
 [6]  -0.2568451  -4.1588370  22.3044129  -0.6066620  12.6069708
[11]  -4.8601168  21.8614529   8.1160206  15.2405765 -14.1588814
[16]  21.8573394  -6.8863043  11.5533815 -20.5397604   1.7370842
> sum(data.arima$residuals^2)
[1] 3040.955
>
```

et toujours avec **tsdiag( )** :



Ajuster un modèle ARIMA(0,2,2)/Lissage exponentiel double de Holt

Toujours en complexité croissante, voyons maintenant le modèle à lissage exponentiel double de Holt ARIMA(0,2,2):

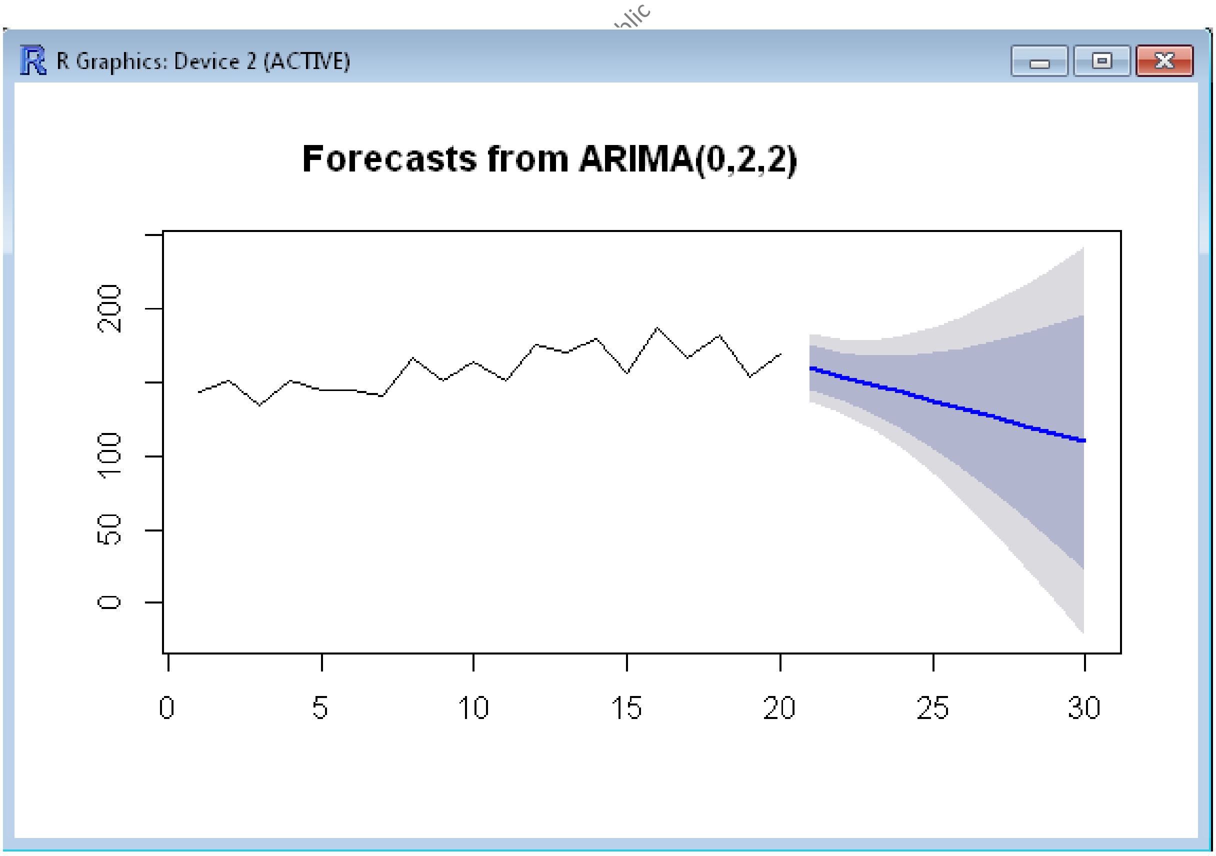


```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,2,2))
> data.arima
Series: data
ARIMA(0,2,2)

Coefficients:
          ma1     ma2
      -1.6623  1.0000
s.e.   0.2074  0.2047

sigma^2 estimated as 125.2:  log likelihood=-71.9
AIC=149.79   AICc=151.51   BIC=152.46
> plot(forecast(data.arima,10))
>
```

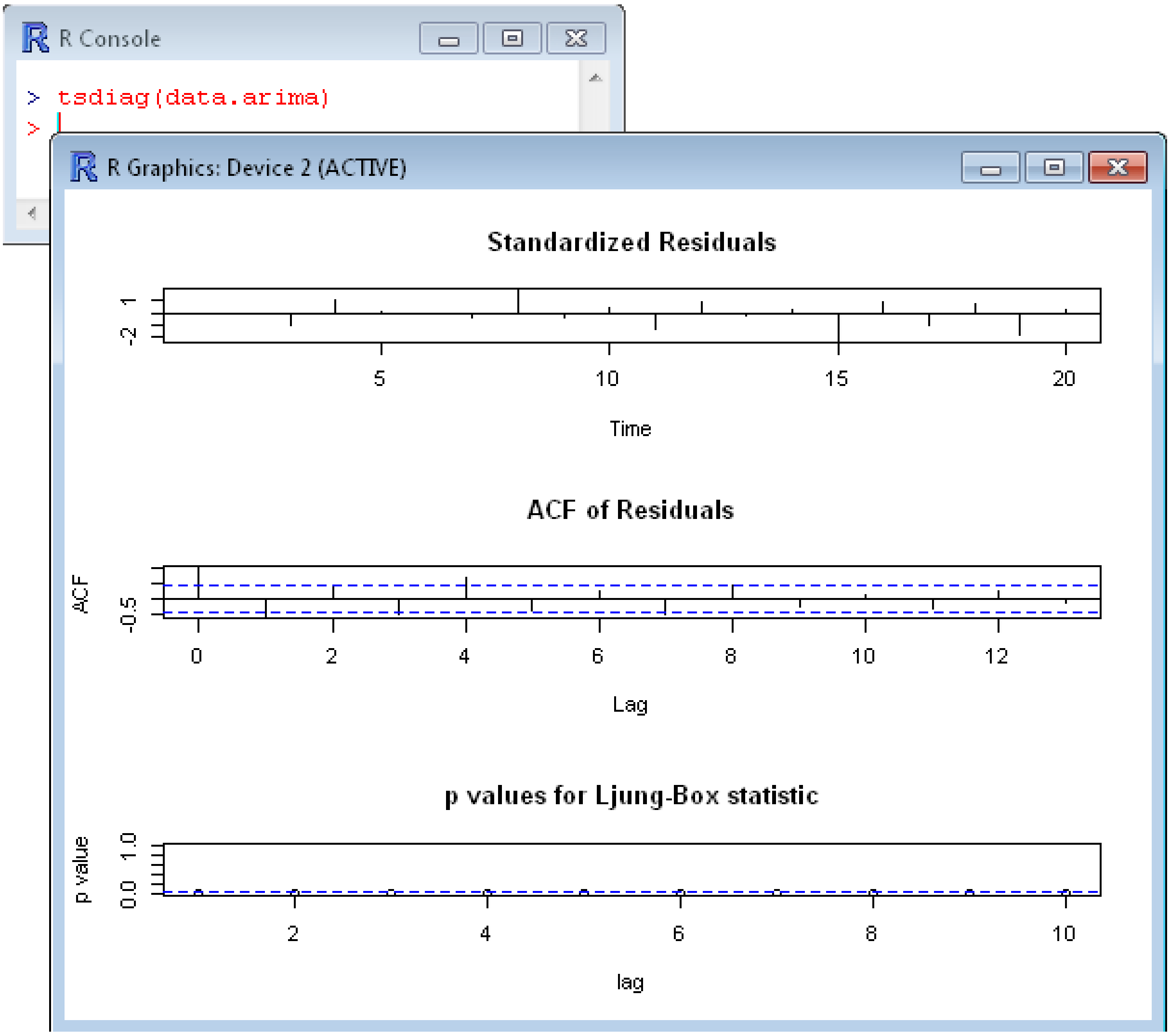
et le graphique associé:



et une petite analyse des résidus:

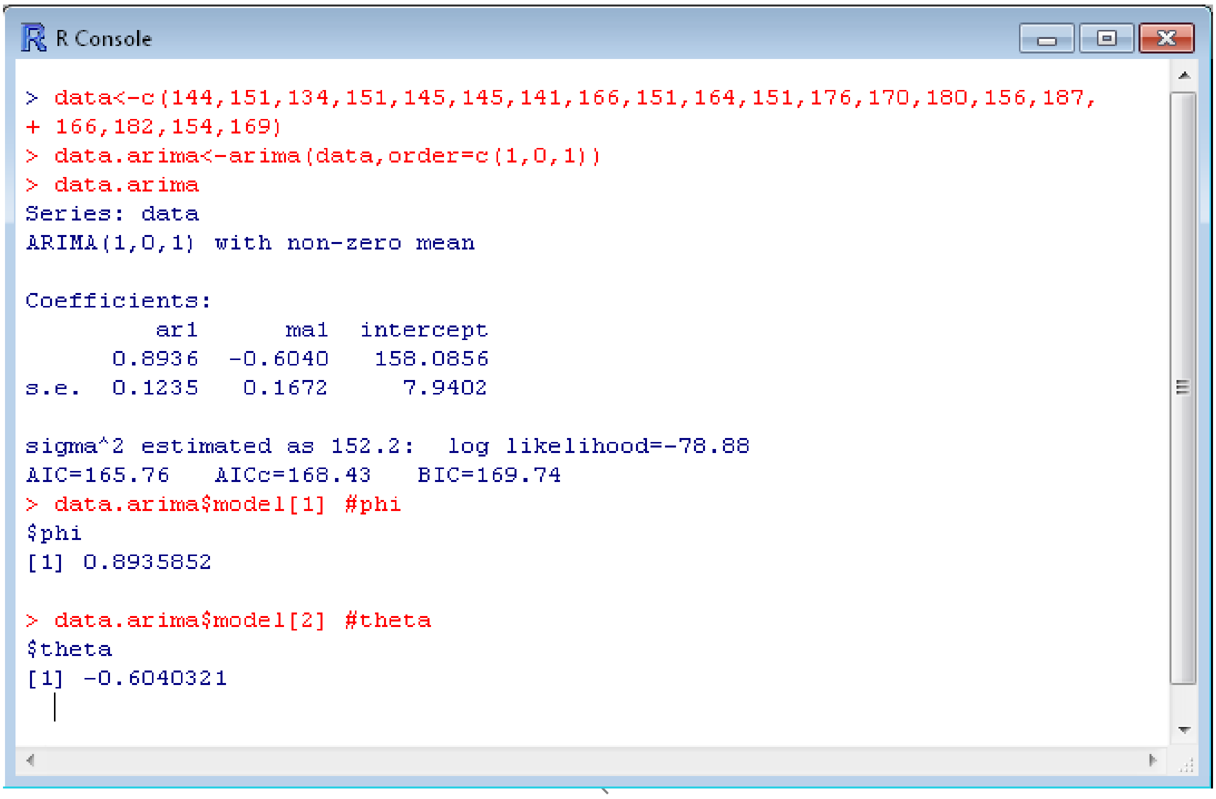
```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(0,2,2))
> data.arima$residuals
Time Series:
Start = 1
End = 20
Frequency = 1
 [1]   0.06439873  -0.17754243 -10.99725004  11.03580010   0.44512808
 [6]  -0.25061634  -4.15153158  19.56826255  -3.67587189   4.23639438
[11] -14.25773470   9.58203302  -1.94611391   3.65407235 -24.28070432
[16]  10.53392446 -11.21852087   8.01544738 -19.04796663   3.60439995
> sum(data.arima$residuals^2)
[1] 2253.31
>
```

et avec la commande **tsdiag( )** :



Ajuster un modèle ARMA(1,1)/ARIMA(1,0,1)

Toujours en complexité croissante, voyons maintenant le modèle ARMA(1,1) dont l'équivalent est ARIMA(1,0,1):



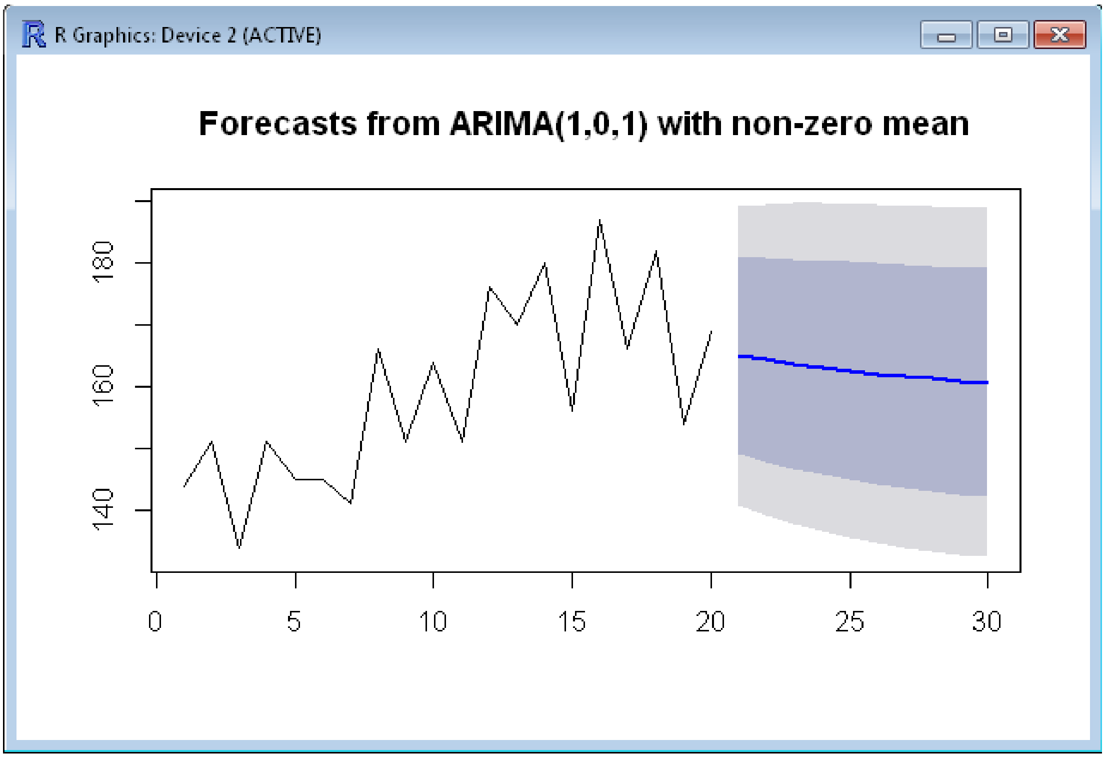
```
R Console
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(1,0,1))
> data.arima
Series: data
ARIMA(1,0,1) with non-zero mean

Coefficients:
         ar1      ma1  intercept
      0.8936  -0.6040   158.0856
s.e.  0.1235   0.1672     7.9402

sigma^2 estimated as 152.2:  log likelihood=-78.88
AIC=165.76   AICc=168.43   BIC=169.74
> data.arima$model[1] #phi
$phi
[1] 0.8935852

> data.arima$model[2] #theta
$theta
[1] -0.6040321
```

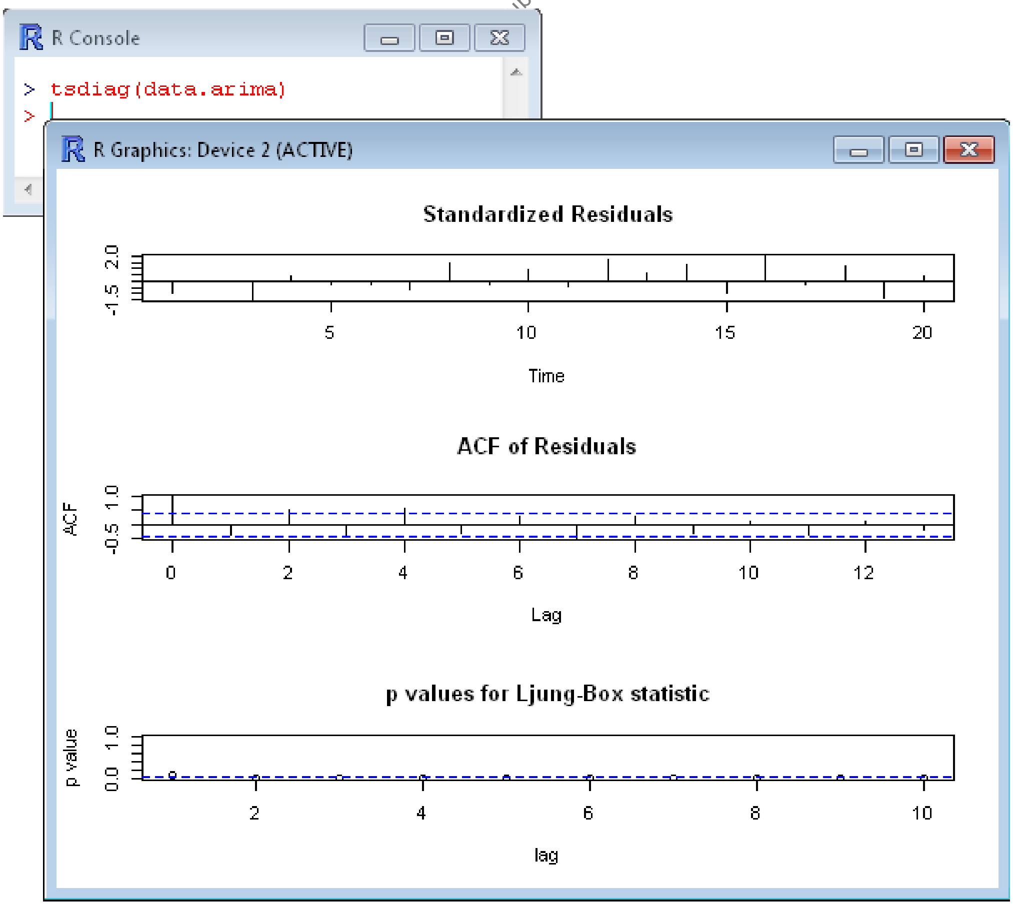
et le graphique associé:



et une petite analyse des résidus:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(1,0,1))
> data.arima$residuals
Time Series:
Start = 1
End = 20
Frequency = 1
 [1] -11.8367151  -0.4819892 -17.7204109   3.8932791  -4.4069710  -4.0452031
 [7]  -7.8316169  18.4507028  -3.0140566  10.4253268  -6.0734427  20.5774175
[13]   8.3357525  16.3029272 -11.8204894  23.6381108  -3.6448987  14.6405780
[19] -16.6117707   4.5311937
> sum(data.arima$residuals^2)
[1] 3043.016
> 
```

et toujours avec la commande **ts.diag( )** :



Recherche du meilleur modèle ARIMA

Nous allons ici utiliser le fichier de données habituel suivant:

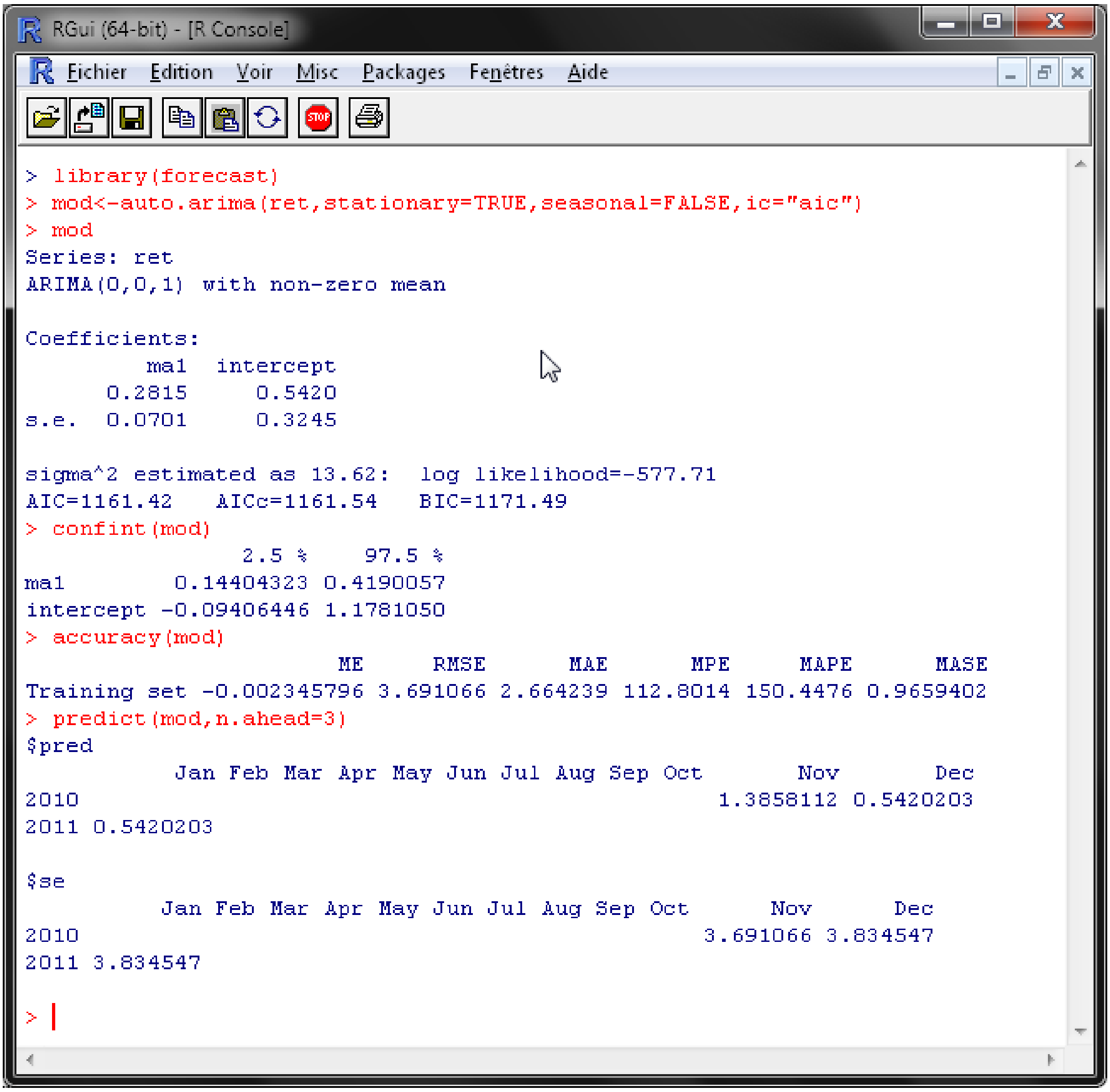
```
YahooSPY.csv - Bloc-notes
Fichier  Edition  Format  Affichage  ?
"Date";"Open";"High";"Low";"Close";"Volume";"Adj.Close"
"44";2010-10-29;118.28;118.72;118.07;118.49;144305500;110.71
"45";2010-10-28;119.06;119.11;117.83;118.4;168576000;110.62
"46";2010-10-27;117.89;118.51;117.26;118.38;190024000;110.61
"47";2010-10-26;118.1;118.84;117.87;118.72;158982900;110.92
"48";2010-10-25;119.14;119.76;118.61;118.7;151145700;110.9
"49";2010-10-22;118.31;118.53;118;118.35;108212400;110.58
"50";2010-10-21;118.4;119.09;117.21;118.13;221585500;110.37
"51";2010-10-20;116.94;118.44;116.87;117.87;200051800;110.13
"52";2010-10-19;117.19;117.85;116.02;116.73;280604700;109.06
"53";2010-10-18;117.74;118.67;117.31;118.28;141204800;110.51
"54";2010-10-15;118.28;118.35;116.76;117.7;243705000;109.97
"55";2010-10-14;117.81;118.01;116.72;117.46;217764300;109.75
"56";2010-10-13;117.66;118.55;117.38;117.92;194347200;110.18
"57";2010-10-12;116.27;117.35;115.65;117.01;182210000;109.33
"58";2010-10-11;116.72;116.97;116.25;116.65;103098300;108.99
"59";2010-10-08;116.05;116.86;115.61;116.54;177760100;108.89
```

Avec le package **zoo** et la commande **aggregate** par mois (as.yearmon), nous regroupons ces données par mois:

```
RGui (64-bit) - [R Console]
Fichier  Edition  Voir  Misc  Packages  Fenêtres  Aide
> tsa<-read.zoo("C:/YahooSPY.csv",sep=";",header=T,format="%Y-%m-%d")
> tsa<-aggregate(tsa,FUN=mean,as.yearmon)
> frequency(tsa) #nous vérifions la fréquence
[1] 12
> head(tsa)
               Open     High      Low    Close   Volume Adj.Close
févr. 1993 44.30737 44.45000 44.05000 44.29158 285136.8  30.01842
mars 1993  45.08304 45.27130 44.93739 45.12000 131269.6  30.67174
avr. 1993  44.44762 44.56810 44.21667 44.35762 128438.1  30.29238
mai 1993   44.62700 44.79550 44.46650 44.66300  90400.0  30.50050
juin 1993  45.01955 45.12045 44.86591 44.99182 156272.7  30.80227
juil. 1993 44.80857 44.92857 44.64476 44.80048 291314.3  30.78429
> #nous calculons les rendements
> ret<-diff(tsa$Close)/lag(tsa$Close,k=-1)*100
> head(ret)
 mars 1993  avr. 1993   mai 1993  juin 1993 juil. 1993  août 1993
 1.8703805 -1.6896741  0.6884521  0.7362205 -0.4252817  1.7278148
>
```

Bon une fois ceci fait, nous chargeons le package **forecast** et utilisons la commande **auto.arima( )** qui va chercher le meilleur modèle ARIMA possible (la recherche peut suivant

le nombre de données prendre plusieurs dizaines de secondes) ainsi que les commandes **confint( )** , **accuracy( )** et **predict( )** qui vont nous donner quelques informations diverses complémentaires:



```
> library(forecast)
> mod<-auto.arima(ret,stationary=TRUE,seasonal=FALSE,ic="aic")
> mod
Series: ret
ARIMA(0,0,1) with non-zero mean

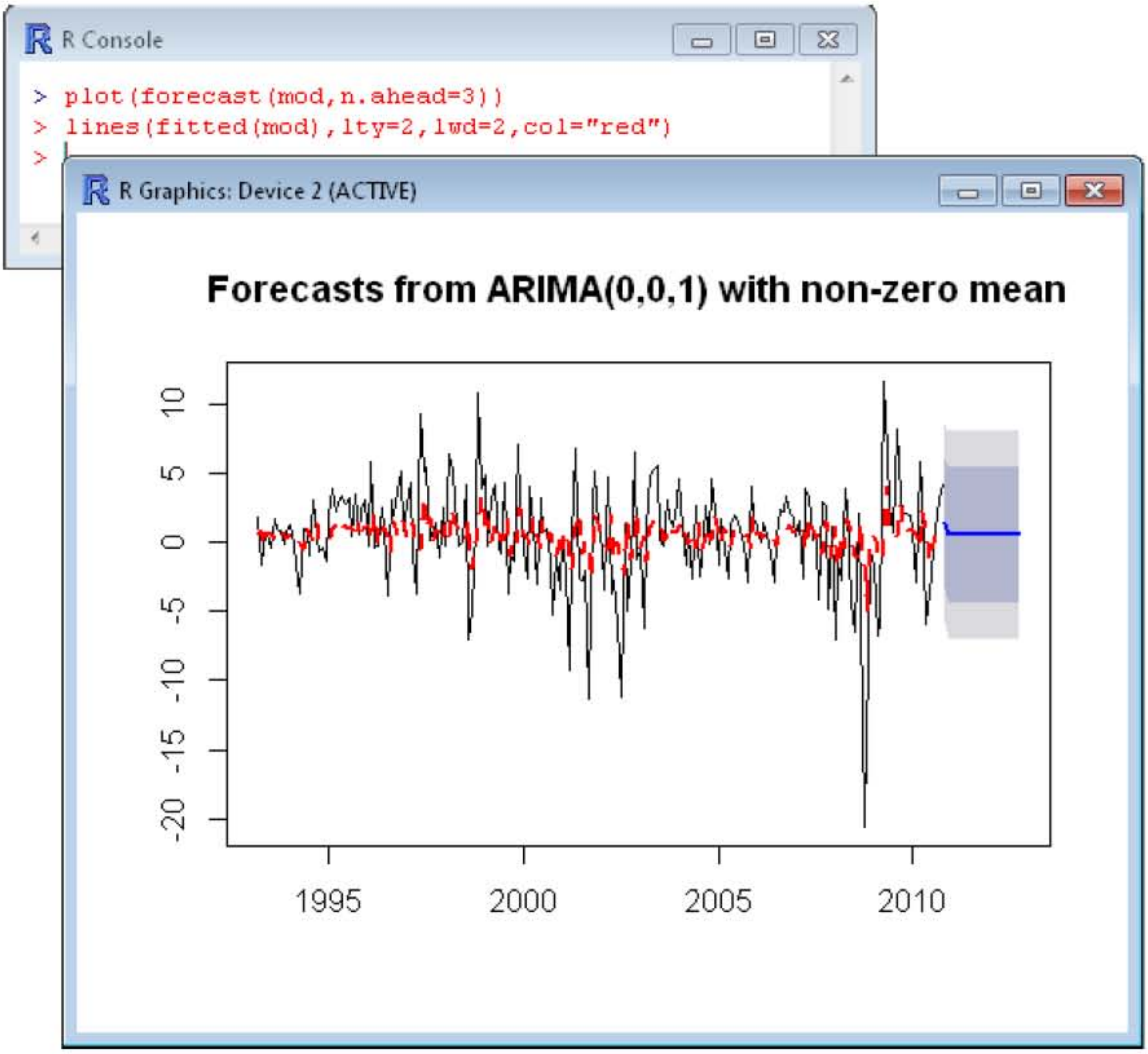
Coefficients:
         ma1  intercept
      0.2815     0.5420
s.e.  0.0701     0.3245

sigma^2 estimated as 13.62:  log likelihood=-577.71
AIC=1161.42   AICc=1161.54   BIC=1171.49
> confint(mod)
                2.5 %    97.5 %
ma1          0.14404323 0.4190057
intercept -0.09406446 1.1781050
> accuracy(mod)
                      ME     RMSE      MAE      MPE     MAPE      MASE
Training set -0.002345796 3.691066 2.664239 112.8014 150.4476 0.9659402
> predict(mod,n.ahead=3)
$pred
           Jan Feb Mar Apr May Jun Jul Aug Sep Oct       Nov       Dec
2010                                               1.3858112 0.5420203
2011 0.5420203

$se
          Jan Feb Mar Apr May Jun Jul Aug Sep Oct      Nov      Dec
2010                                              3.691066 3.834547
2011 3.834547

> 
```

Ensuite faisons un plot de ce que nous avons vu dans le cours théorique (sachant qu'il y a beaucoup de choses que nous n'avons pas étudié non plus et que nous ne présenterons donc pas ici):


R Console
> plot(forecast(mod,n.ahead=3))
> lines(fitted(mod),lty=2,lwd=2,col="red")
>
R Graphics: Device 2 (ACTIVE)
Forecasts from ARIMA(0,0,1) with non-zero mean
10
5
0
-5
-10
-15
-20
1995
2000
2005
2010

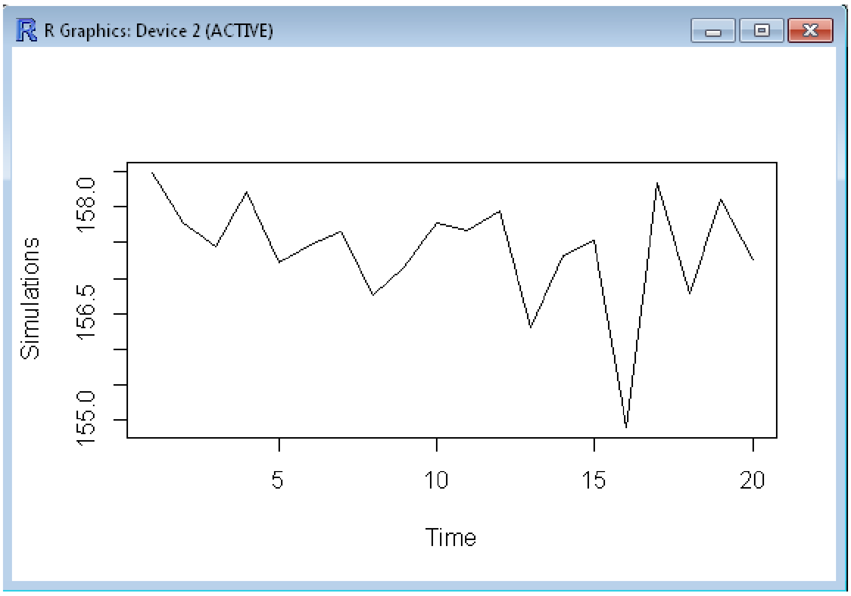
Simulations de modèles ARIMA

Nous allons ici reproduire les modèles obtenus plus haut avec leurs coefficients/paramètres respectifs. Évidemment, et comme nous l'avons vu dans le cours théorique, chaque fois que vous relancerez une simulation pour un même modèle ARIMA avec des paramètres identiques, le résultat sera différent.

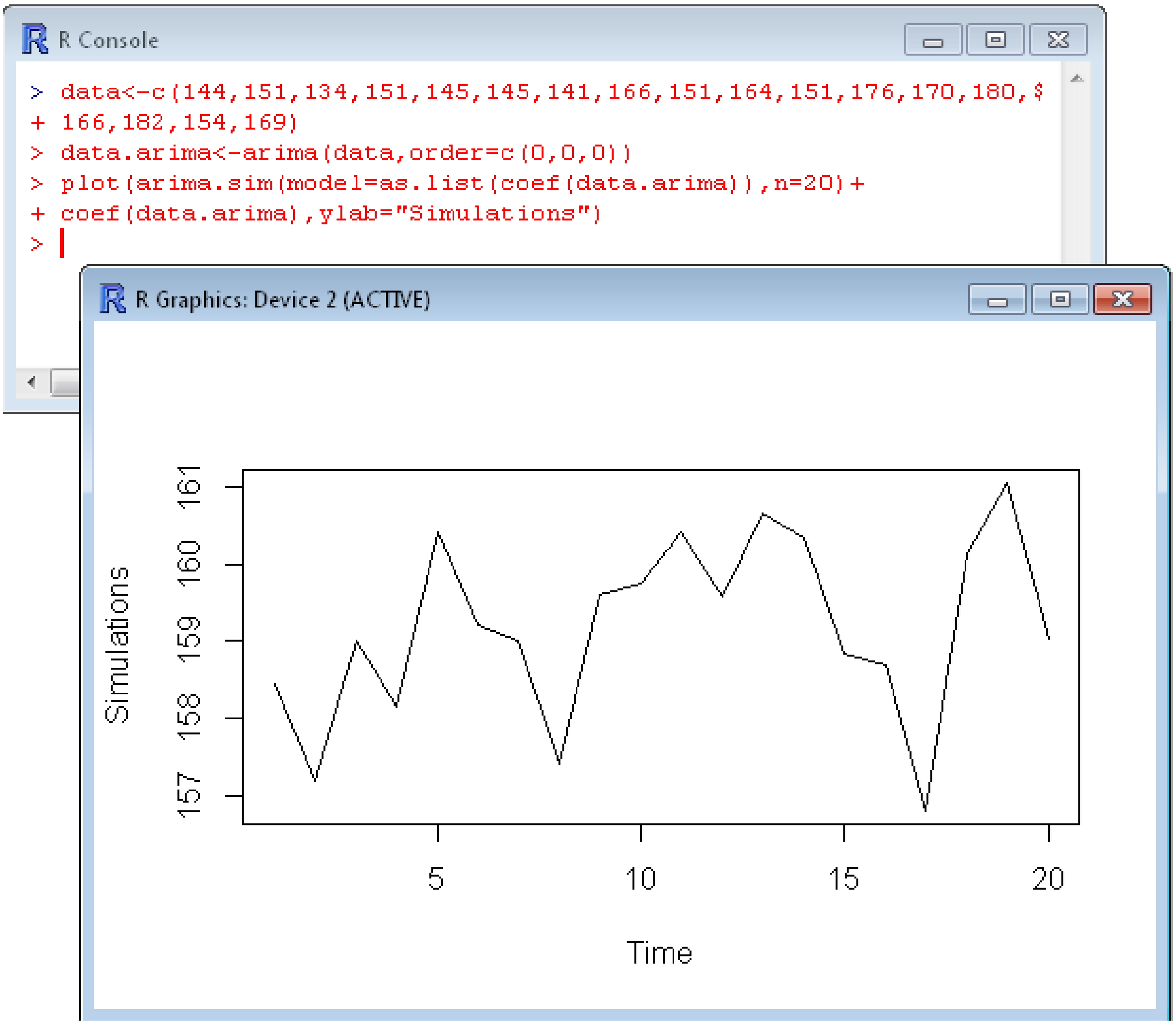
Nous commençons avec le modèle AR(2) que nous avons obtenu plus haut). Comme la commande native **arima.sim( )** ignore l'interception, nous devons l'ajouter systématiquement:

```
> data<-c(144,151,134,151,145,145,141,166,151,164,151,176,170,180,156,187,
+ 166,182,154,169)
> data.arima<-arima(data,order=c(2,0,0))
> arima.sim(model=as.list(coef(data.arima)),n=20)
Time Series:
Start = 1
End = 20
Frequency = 1
 [1] -1.0017888  0.7948052  0.4641716  1.3756805  1.1423079 -1.2082263  1.8963943
 [8]  0.6935560 -0.5743589 -1.0843954 -2.1165998 -1.1970653 -0.7388205  1.4042732
[15] -0.0209877 -2.4011538  0.4855412  0.1948019  0.9000237 -0.5412893
> plot(arima.sim(model=as.list(coef(data.arima)),n=20)+
+ coef(data.arima)[3])
>
```

et le graphique associé (qui donc au même titre que les valeurs sera différent à chaque réexécution des commandes ci-dessus):



Pour le modèle ARIMA(0,0,0), la syntaxe sera:

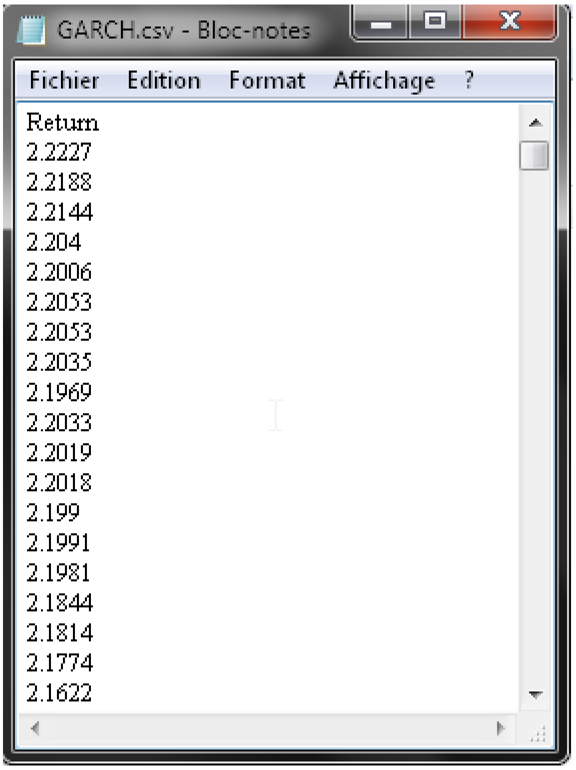


et ainsi de suite... avec en fonction des modèles de subtilités pour rapatrier le constante donc n'oubliez pas d'utiliser la commande **str(data.arima)** pour voir le contenu da la liste.

Ajuster un modèle GARCH(1,1)

Le but ici va être de vérifier si nous obtenons le même résultat que dans le cours théorique et surtout de déterminer (si possible) la méthode utilisée par R pour déterminer les paramètres du modèle GARCH(1,1).

Nous partons donc des mêmes données que dans le cours théorique de 1267 lignes:



Utilisons d'abord la fonction **garch( )** du package **tseries** et comparons le résultat à ce qui nous est connu:

```
> library("tseries")
> mydata<-read.csv("c:/tmp/GARCH.csv")
> fit<-garch(mydata,order=c(1,1),trace=FALSE)
> summary(fit)

Call:
garch(x = mydata, order = c(1, 1), trace = FALSE)

Model:
GARCH(1,1)

Residuals:
   Min      1Q Median     3Q    Max
0.9472 0.9704 0.9786 0.9853 1.0052

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 3.001e-01   3.811e+01    0.008    0.994
a1 9.555e-01   1.048e+02    0.009    0.993
b1 8.583e-15   1.092e+02    0.000    1.000

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 47.406, df = 2, p-value = 5.081e-11


        Box-Ljung test

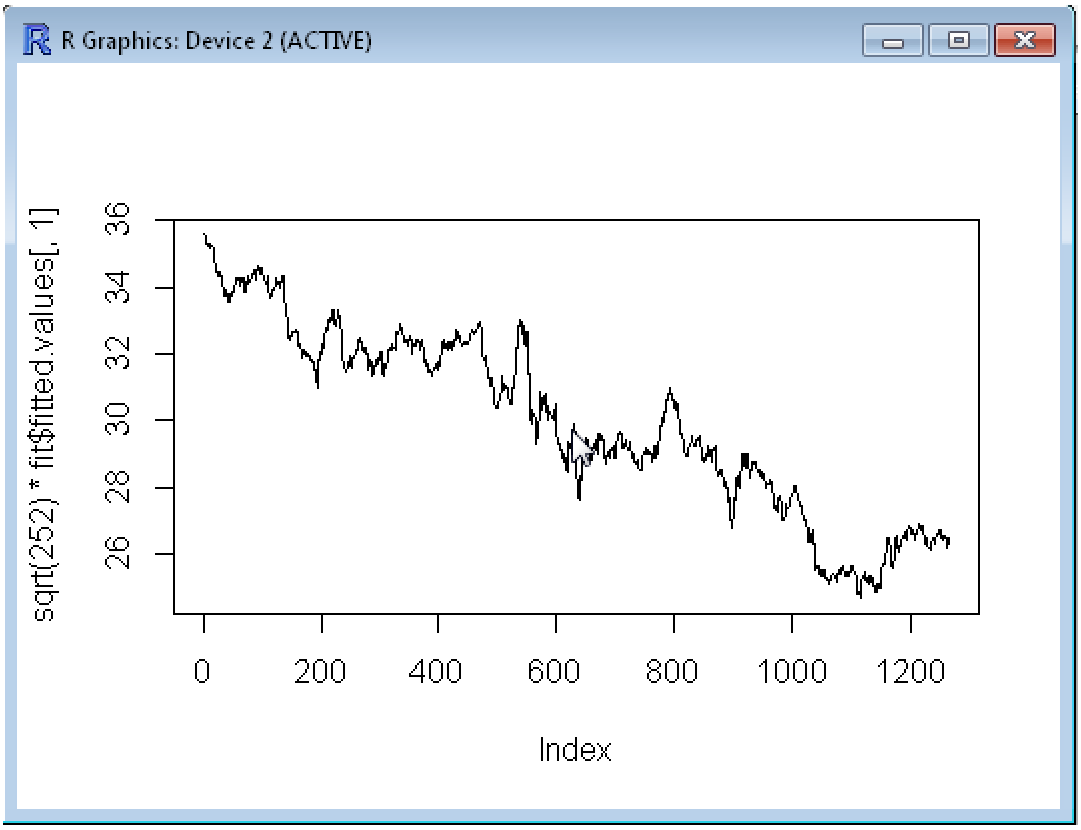
data:  Squared.Residuals
X-squared = 842.03, df = 1, p-value < 2.2e-16

> coef(fit)
          a0           a1           b1
3.001021e-01 9.555199e-01 8.582746e-15
> plot(sqrt(252)*fit$fitted.values[,1],type="l")
>
```

Les intervalles de confiances des paramètres:

```
> confint(fit)
        2.5 %    97.5 %
a0  -74.38524  74.98545
a1 -204.41109 206.32213
b1 -214.04178 214.04178
>
```

et le graphique associé:



Attention à ne pas oublier de contrôler avant les calculs que les résidus satisfont les conditions d'application du modèle GARCH (voir le cours théorique)!

Alors avec cette première méthode que ce soit au niveau des paramètres ou du graphique cela est très différent de toutes les méthodes théoriques que nous avons déterminées. Donc nous laisserons de côté cette méthode à défaut de perdre du temps à décortiquer l'algorithme R utilisé.

La deuxième méthode consiste à utiliser la fonction **garchFit( )** du package **fGarch**:

```
> library(fGarch)
> fit<-garchFit(~garch(1, 1), data = mydata)

Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(1, 1)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               1 1
 Max GARCH Order:           1
 Maximum Order:             1
 Conditional Dist:          norm
 h.start:                   2
 llh.start:                 1
 Length of Series:          1267
 Recursion Init:            mci
 Series Scale:              0.1854356

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
```

```
                     U          V   params includes
    mu     -99.60141986  99.60142 9.960142     TRUE
    omega    0.00000100 100.00000 0.100000     TRUE
    alpha1   0.00000001   1.00000 0.100000     TRUE
    gamma1  -0.99999999   1.00000 0.100000    FALSE
    beta1    0.00000001   1.00000 0.800000     TRUE
    delta    0.00000000   2.00000 2.000000    FALSE
    skew     0.10000000  10.00000 1.000000    FALSE
    shape    1.00000000  10.00000 4.000000    FALSE
 Index List of Parameters to be Optimized:
    mu  omega alpha1  beta1
     1      2      3      5
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     1592.9842:  9.96014 0.100000 0.100000 0.800000
  1:     1525.1318:  10.0290 0.0612052 0.104741 0.781950
  2:     1311.4984:  9.82965 1.00000e-06 0.236028 0.740576
  3:     1302.6332:  9.80341 0.00553903 0.236171 0.740654
  4:     1225.7267:  9.72065 0.00142212 0.243210 0.736777
  5:     1214.0427:  9.61690 1.00000e-06 0.243323 0.736375
  6:     1208.8214:  9.61736 0.00106142 0.243329 0.736378
  7:     1199.7247:  9.65521 0.000921870 0.263952 0.729836
  8:     1197.8632:  9.65472 0.000474024 0.263952 0.729821
  9:     1197.7460:  9.65366 0.000591098 0.264306 0.729591
 10:     1197.5499:  9.65317 0.000493939 0.264675 0.729356
```

...

```
 24:      1152.6593:  9.65750 0.00275105 0.998451 1.00000e-08
 25:      1152.6329:  9.65864 0.00270435 0.991199 1.00000e-08
 26:      1152.6319:  9.65855 0.00270867 0.990816 1.00000e-08
 27:      1152.6296:  9.65828 0.00273906 0.988861 1.00000e-08
 28:      1152.6295:  9.65818 0.00273563 0.988826 1.00000e-08
 29:      1152.6295:  9.65822 0.00273622 0.988876 1.00000e-08
 30:      1152.6295:  9.65822 0.00273622 0.988875 1.00000e-08

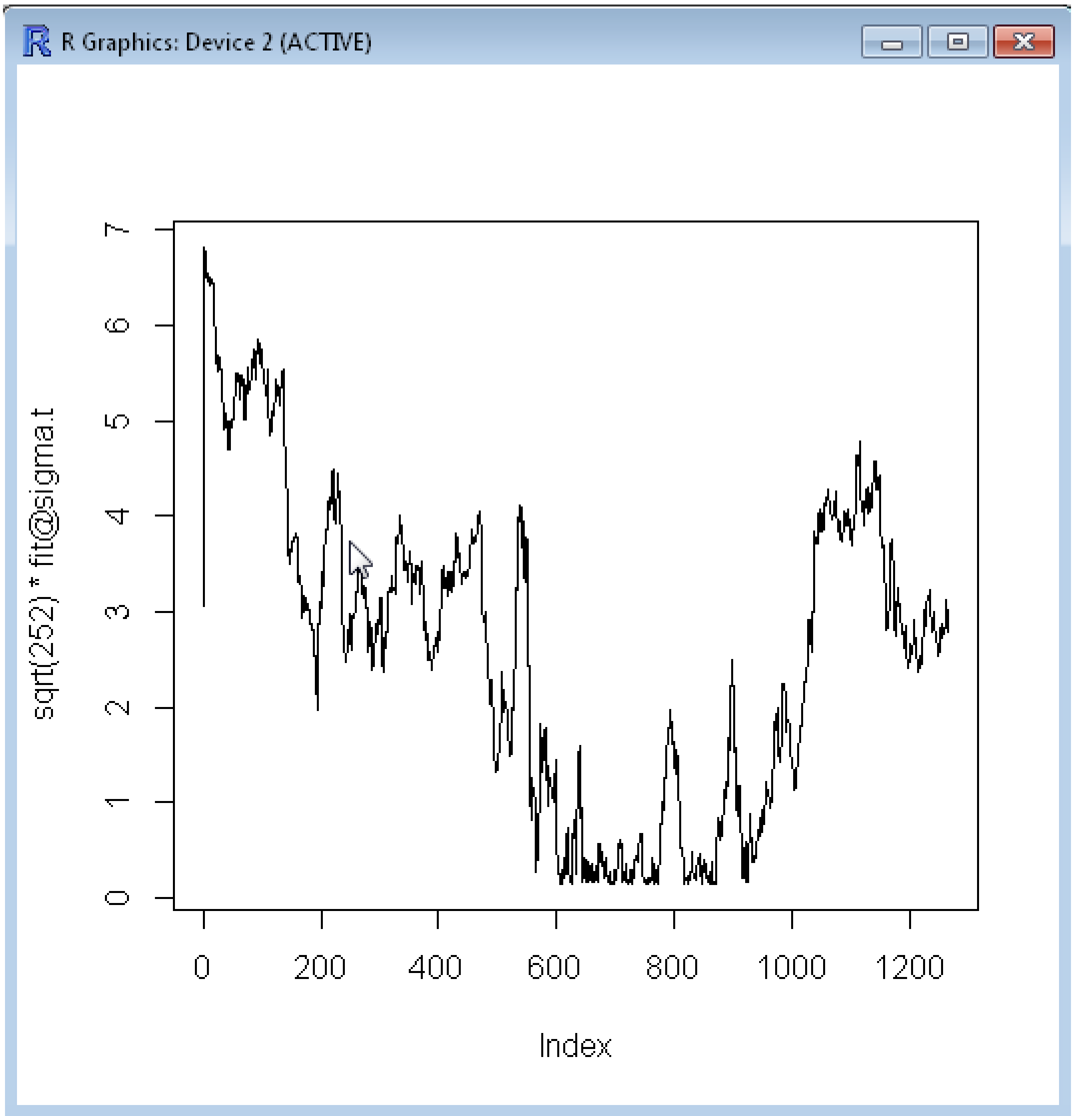
Final Estimate of the Negative LLH:
 LLH:  -982.3256    norm LLH:  -0.7753161
          mu        omega       alpha1        beta1
1.790978e+00 9.408872e-05 9.888748e-01 1.000000e-08

R-optimhess Difference Approximated Hessian Matrix:
                  mu         omega       alpha1         beta1
mu     -2.888344e+05   -11132706.2     419.9592    -2730.8111
omega  -1.113271e+07 -7312863526.1 -252050.0002  -927298.2752
alpha1  4.199592e+02     -252050.0     -569.0740     -605.1578
beta1  -2.730811e+03     -927298.3     -605.1578     -977.7221
attr(,"time")
Time difference of 0.04000211 secs

--- END OF TRACE ---

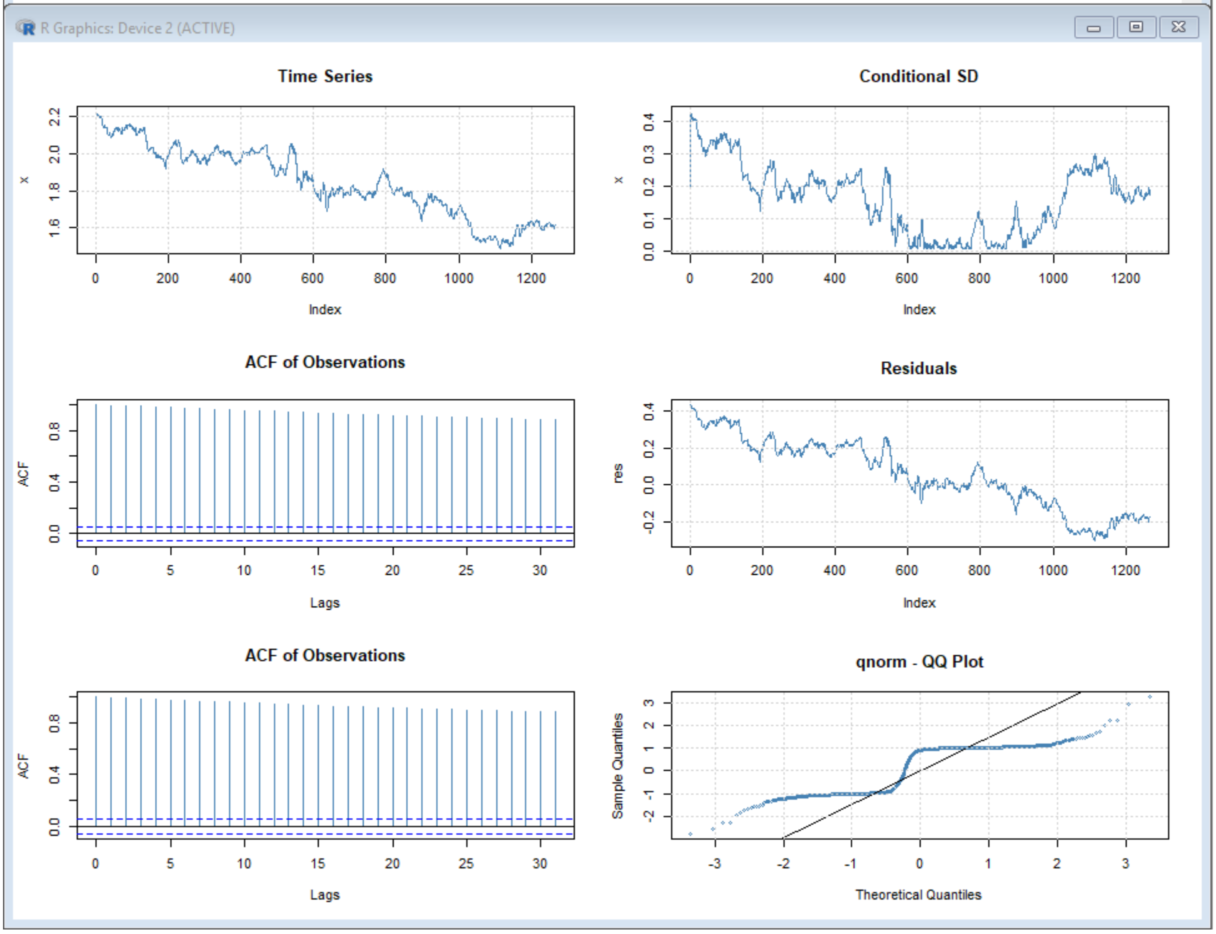

Time to Estimate Parameters:
 Time difference of 0.1860101 secs
> coef(fit)
          mu        omega       alpha1        beta1
1.790978e+00 9.408872e-05 9.888748e-01 1.000000e-08
> plot(sqrt(252) * fit@sigma.t, type="l")
> 
```

Avec le graphique associé:



On peut aussi obtenir le graphique ci-dessus de la variance conditionnelle, ainsi que d'autres, de la manière suivante:

```
> par(mfrow=c(3,2))
> plot(fit, which = 1) #série temporelle
> plot(fit, which = 2) #variance conditionelle
> plot(fit, which = 4) #acf des observation
> plot(fit, which = 7) #résidus
> plot(fit, which = 4) #acf des résidus
> plot(fit, which = 13) #qq-plot ou résidus standardisés
>
```



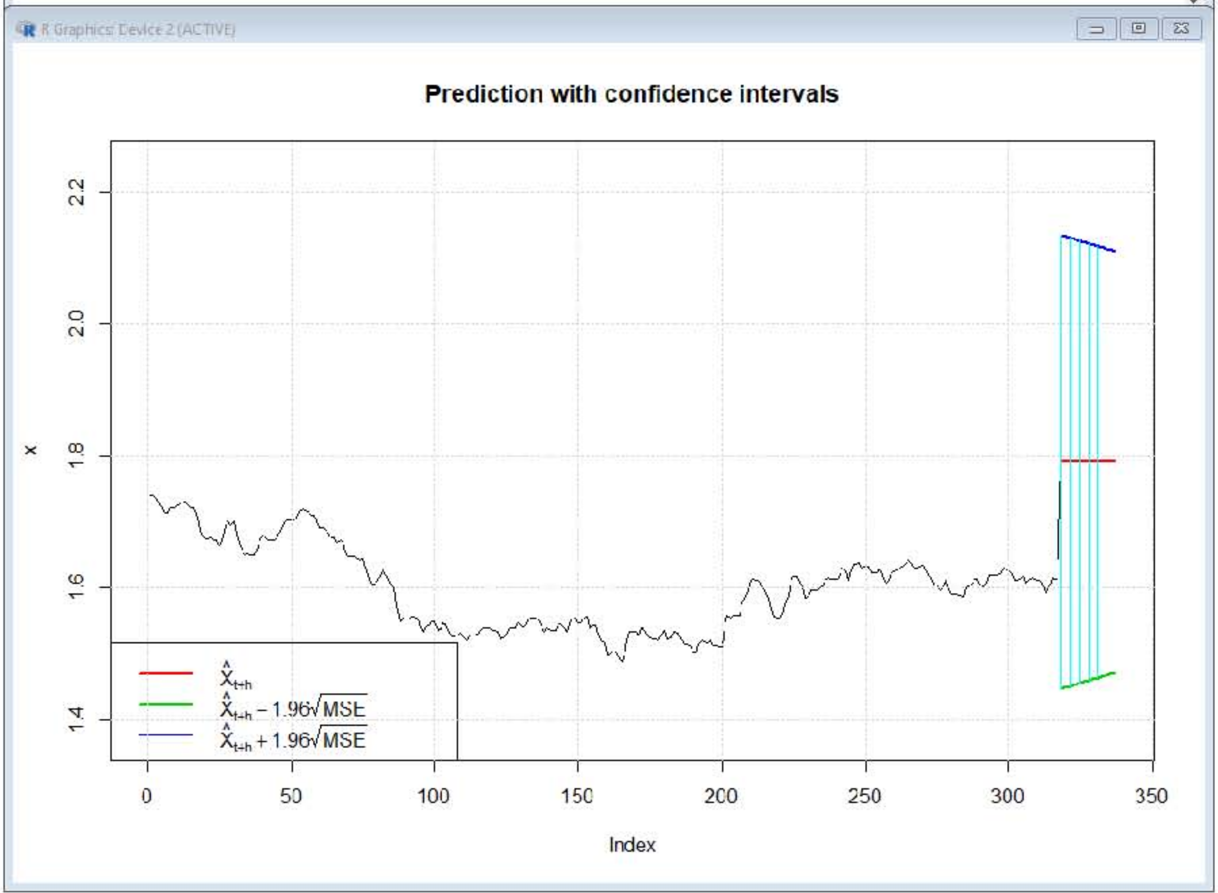
Alors que dire du résultat... D'abord la manière que R (disons plutôt les packages correspondants…) détermine la moyenne empirique nécessiterai de décortiquer le script car ce n'est a priori pas la moyenne arithmétique, ni géométrique ni la médiane...

Sinon pour les trois autres paramètres nous sommes à peu près bon et nous retrouvons les mêmes résultats que dans le cours théorique que si nous considérons le ε_t par rapport à la moyenne empirique (alors que MATLAB le fait par rendement successifs).

En ce qui concerne le graphique, là aussi c'est très différent de calculs faits à la main dans le cours théorique et aussi très différent de ce que nous avons obtenons avec MATLAB.

Nous pouvons sinon (sujet non étudié dans le cours théorique), faire aussi de la "prédiction" mais le problème… c'est que nous ne sommes pas arrivés à déterminer le graphique qui est fait car il ne correspond ni au graphique de la série temporelle, ni à celui de la variance conditionnelle??????:

```
> predict(fit, n.ahead = 20, plot=TRUE)
  meanForecast meanError standardDeviation lowerInterval upperInterval
1     1.790978 0.1758605         0.1758605      1.446298      2.135658
2     1.790978 0.1751483         0.1751483      1.447693      2.134262
3     1.790978 0.1744412         0.1744412      1.449079      2.132876
4     1.790978 0.1737391         0.1737391      1.450455      2.131500
5     1.790978 0.1730421         0.1730421      1.451822      2.130134
6     1.790978 0.1723500         0.1723500      1.453178      2.128778
```



Exemple 384.: Test de causalité de Granger WP

R 3.6.3

Le test de causalité de Granger est utilisé pour déterminer si une série chronologique est utile ou non pour en prévoir une autre.

Ce test utilise les hypothèses nulles et alternatives suivantes :

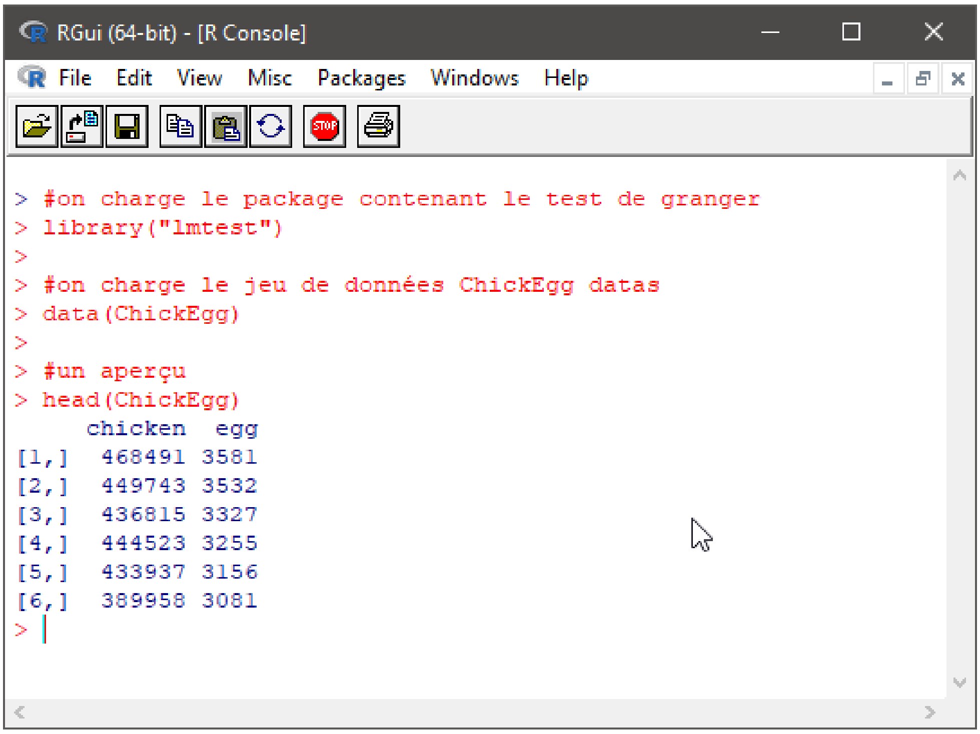
- Hypothèse nulle (H0) : la série chronologique *x* ne Granger-cause pas la série chronologique *y*

- Hypothèse alternative (HA) : la série chronologique *x* Granger-cause la série chronologique *y*

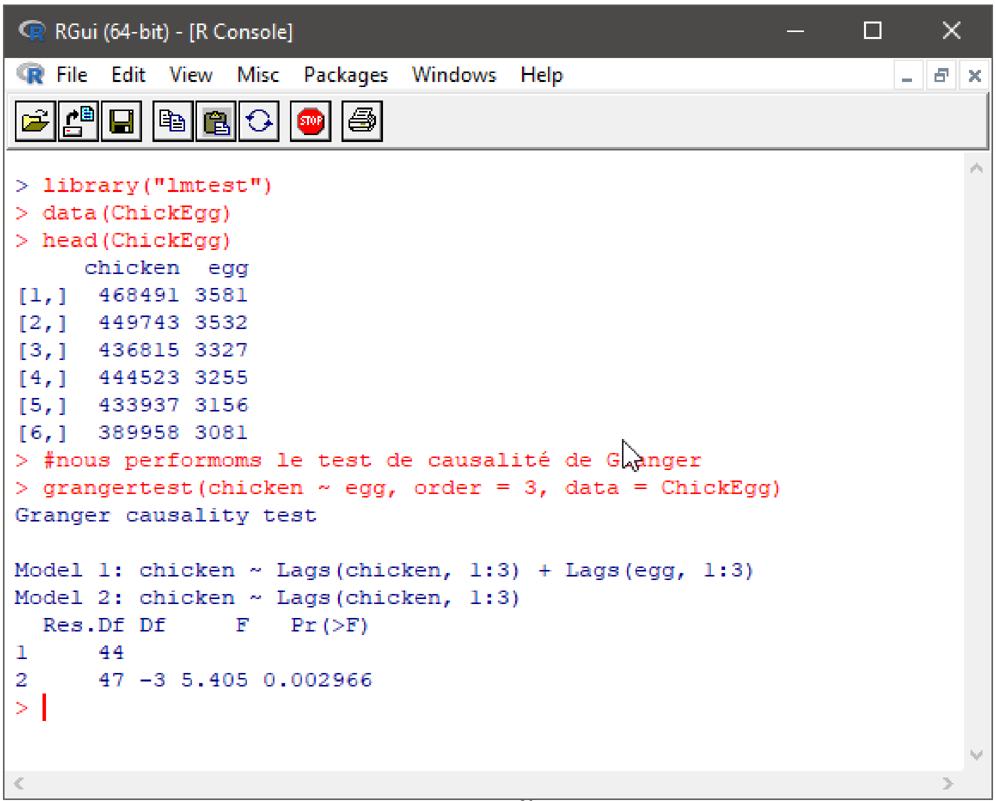
Le terme « causes de Granger » signifie que connaître la valeur de la série chronologique *x* à un certain décalage est utile pour prédire la valeur de la série chronologique *y* à une période ultérieure.

Ce test produit une statistique de test *F* avec une valeur p correspondante. Si la *p*-valeur est inférieure à un certain niveau de signification (c'est-à-dire $\alpha = 0.05$), alors nous pouvons rejeter l'hypothèse nulle et conclure que nous avons suffisamment de preuves pour dire que la série chronologique *x* Granger-cause la série chronologique *y*.

Pour cet exemple, nous utiliserons l'ensemble de données *ChickEgg* qui est préchargé dans le package **lmtest**. Cet ensemble de données contient des valeurs pour le nombre d'œufs fabriqués ainsi que le nombre de poulets aux États-Unis de 1930 à 1983 :



Ensuite, nous utilisons la fonction **grangertest( )** pour effectuer un test de causalité de Granger pour voir si le nombre d'œufs fabriqués est prédictif du nombre futur de poulets. Nous allons exécuter le test en utilisant trois décalages :



Voici comment interpréter la sortie :

- Modèle 1 : Ce modèle tente de prédire le nombre de poulets en utilisant le nombre de poulets au cours des trois années précédentes et le nombre d'œufs au cours des trois années précédentes comme variables prédictives.

- Modèle 2 : Ce modèle tente de prédire le nombre de poulets en utilisant uniquement le nombre de poulets au cours des trois années précédentes comme variables prédictives.

- F : Il s'agit de la statistique du test F. Il s'avère que c'est 5.405.

- Pr(>F) : Il s'agit de la valeur de p qui correspond à la statistique de test F. Il s'avère que c'est .002966.

Puisque la *p*-valeur est inférieure à 0.05, nous pouvons rejeter l'hypothèse nulle du test et conclure que connaître le nombre d'œufs est utile pour prédire le nombre futur de poulets.

Bien que nous ayons rejeté l'hypothèse nulle du test, il est en fait possible qu'un cas de causalité inverse se produise. C'est-à-dire qu'il est possible que le nombre de poules modifie le nombre d'œufs.

Pour exclure cette possibilité, nous devons effectuer le test de Granger-Causalité à l'envers, en utilisant les poulets comme variable prédictive et les œufs comme variable de réponse :

```
> library("lmtest")
> data(ChickEgg)
> head(ChickEgg)
     chicken  egg
[1,]  468491 3581
[2,]  449743 3532
[3,]  436815 3327
[4,]  444523 3255
[5,]  433937 3156
[6,]  389958 3081
> #nous performoms le test de causalité de Granger
> grangertest(chicken ~ egg, order = 3, data = ChickEgg)
Granger causality test

Model 1: chicken ~ Lags(chicken, 1:3) + Lags(egg, 1:3)
Model 2: chicken ~ Lags(chicken, 1:3)
  Res.Df Df      F   Pr(>F)
1     44
2     47 -3 5.405 0.002966
> #et dans le sens inverse
> grangertest(egg ~ chicken, order = 3, data = ChickEgg)
Granger causality test

Model 1: egg ~ Lags(egg, 1:3) + Lags(chicken, 1:3)
Model 2: egg ~ Lags(egg, 1:3)
  Res.Df Df      F Pr(>F)
1     44
2     47 -3 0.5916 0.6238
> 
```

La *p*-valeur du test est de 0.6238. Comme ce n'est pas inférieur à 0.05, nous ne pouvons pas rejeter l'hypothèse nulle. Autrement dit, le nombre de poulets n'est pas prédictif du nombre futur d'œufs.

Ainsi, nous pouvons conclure que connaître le nombre d'œufs est utile pour prédire le nombre futur de poulets.

Exemple 385.: Modèle TBATS

R 3.6.3

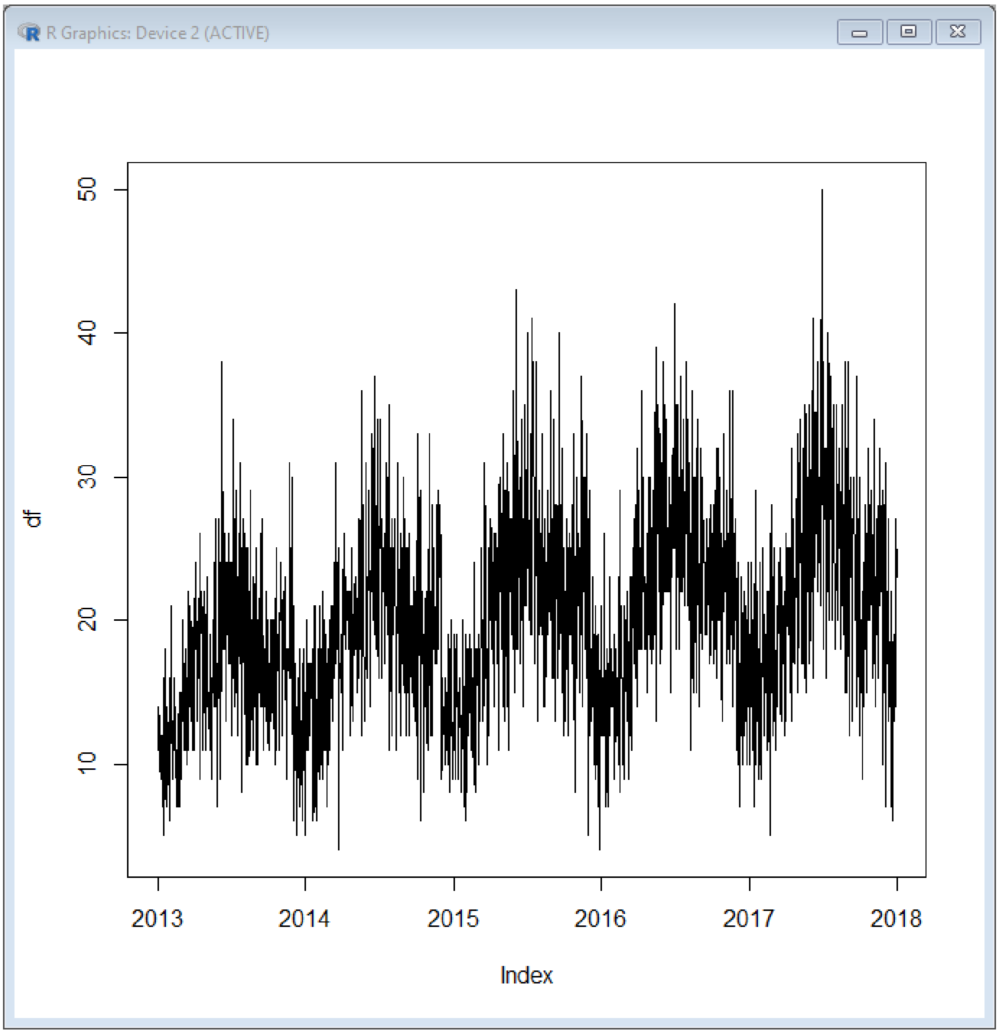
Comme nous l'avons mentionné dans le cours théorique, c'est une méthode empirique. Pour rappel, l'abréviation signifie:

- Trigonometric terms for seasonality
- Box-Cox transformations for heterogeneity
- ARMA errors for short-term dynamics
- Trend (possibly damped)
- Seasonal (including multiple and non-integer periods)

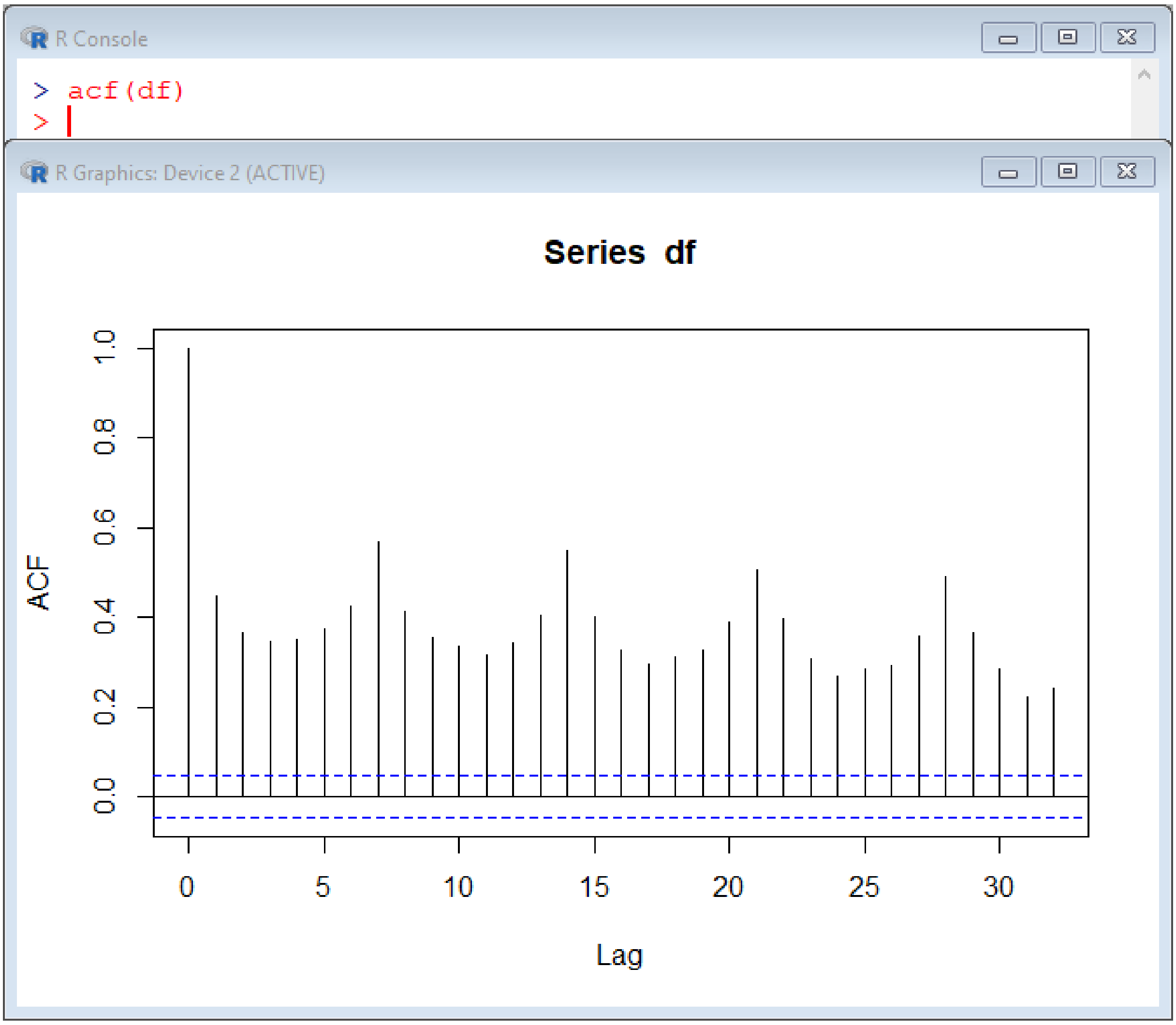
D'abord chargeons et découvrons le jeu de données avec lequel nous allons travailler:

```
> library("forecast")
> library("zoo")
> library("lubridate")
> mydata<-read.csv("c:/tmp/sales.csv",header=T,sep=",",stringsAsFactors=FALSE)
> #on filtre un peu pour prendre seulement un article d'un magasin donné
> df <- mydata[which(mydata$store==1 & mydata$item ==1),c(1,4)]
> str(df)
'data.frame':   1826 obs. of  2 variables:
 $ date : chr  "2013-01-01" "2013-01-02" "2013-01-03" "2013-01-04" ...
 $ sales: int  13 11 14 13 10 12 10 9 12 9 ...
> head(df)
        date sales
1 2013-01-01    13
2 2013-01-02    11
3 2013-01-03    14
4 2013-01-04    13
5 2013-01-05    10
6 2013-01-06    12
> tail(df)
           date sales
1821 2017-12-26    16
1822 2017-12-27    14
1823 2017-12-28    19
1824 2017-12-29    15
1825 2017-12-30    27
1826 2017-12-31    23
>
> #nombre d'années dans le jeu initial utile pour tard
> (Annees<-lubridate::time_length(difftime(as.Date(max(df$date)), as.Date(min(df$date))), "years"))
[1] 5
>
> df<- read.zoo(df, format = "%Y-%m-%d")
> plot(df)
>
```

Avec le graphique correspondant:



On fait une analyse de l'ACF:



On lance maintenant l'ajustement du modèle:

```
> #Les ventes ont toutes l'air d'avoir une périodicité hebdomadaire et annuelle
> (fit<-tbats(df,seasonal.periods = c(7,365)))
TBATS(0.239, {0,0}, -, {<7,3>, <365,6>})

Call: tbats(y = df, seasonal.periods = c(7, 365))

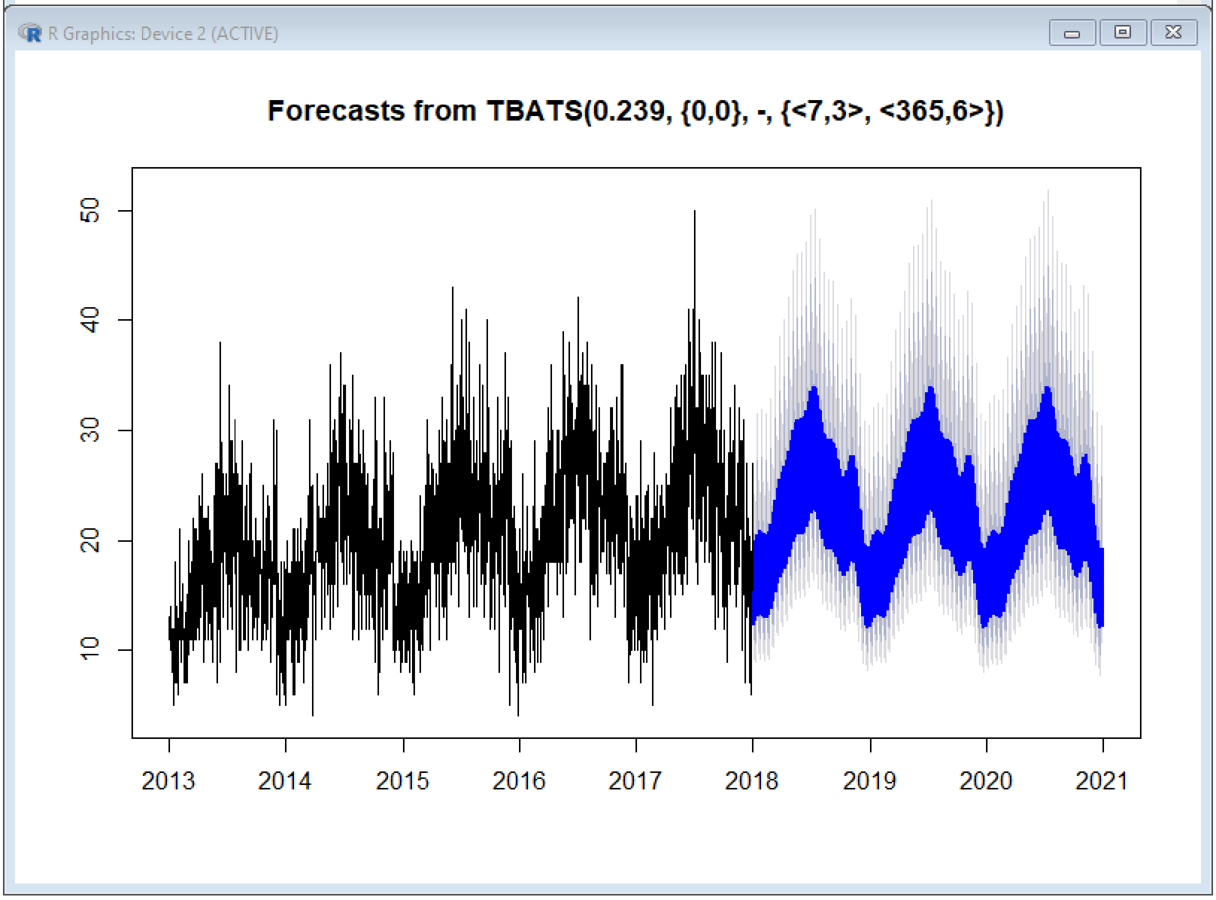
Parameters
  Lambda: 0.238803
  Alpha: 0.01558787
  Gamma-1 Values: 0.0003275112 -0.001321891
  Gamma-2 Values: 0.002166953 -0.0001732746

Seed States:
              [,1]
 [1,]  3.878958012
 [2,] -0.277719696
 [3,] -0.027443156
 [4,]  0.132211894
 [5,] -0.055119018
 [6,]  0.191895672
 [7,]  0.089115666
 [8,] -0.497609479
 [9,] -0.037368335
[10,] -0.102748946
[11,] -0.020661916
[12,] -0.050842363
[13,]  0.002238892
[14,] -0.082358064
[15,] -0.103466007
[16,] -0.071637563
[17,]  0.022690156
[18,]  0.060276708
[19,]  0.066820947
attr(,"lambda")
[1] 0.2388028

Sigma: 0.4760076
AIC: 19203.96
>
```

Et faisons maintenant une visulation graphique d'une projection sur 3 ans:

```
R Console
> horizon_fcast<-3*365 #en jours
> fcast<-forecast(fit,level=c(80,95),h=horizon_fcast)
> plot(fcast,xaxt = "n")
> #on corrige le problème des axes (il y a probablement plus élégant...)
> maxdate<-max(round(as.numeric(rownames(as.data.frame(fcast)))))
>
> axis(1,at=seq(maxdate-Annees-horizon_fcast/365,maxdate),
+ labels=seq(year(min(mydata$date)),year(min(mydata$date))+Annees+horizon_f$
> 
```



Voyons comment obtenir les valeurs directement plutôt que sous forme graphique:

```
> #exemple projection sur 10 jours
> #il faut évidemment corriger les dates
> forecast(fit,level=c(80,95),h=10)
           Point Forecast        Lo 80    Hi 80     Lo 95    Hi 95
15711.0027       12.29998   8.674693 16.97762  7.121063 19.94486
15711.0055       14.28397  10.203109 19.50266  8.440206 22.79367
15711.0082       15.26365  10.962773 20.74260  9.098443 24.18894
15711.0110       15.20027  10.913430 20.66270  9.055589 24.09920
15711.0137       16.77788  12.143041 22.65055 10.124296 26.33134
15711.0164       19.19659  14.041898 25.67930 11.781800 29.72221
15711.0192       19.98664  14.665193 26.66443 12.327469 30.82292
15711.0219       12.72066   8.996786 17.51599  7.398073 20.55394
15711.0247       14.76264  10.573156 20.11023  8.760306 23.47836
15711.0274       15.76888  11.354943 21.38166  9.438510 24.90793
>
```

Exemple 386.: Modèle Prophet

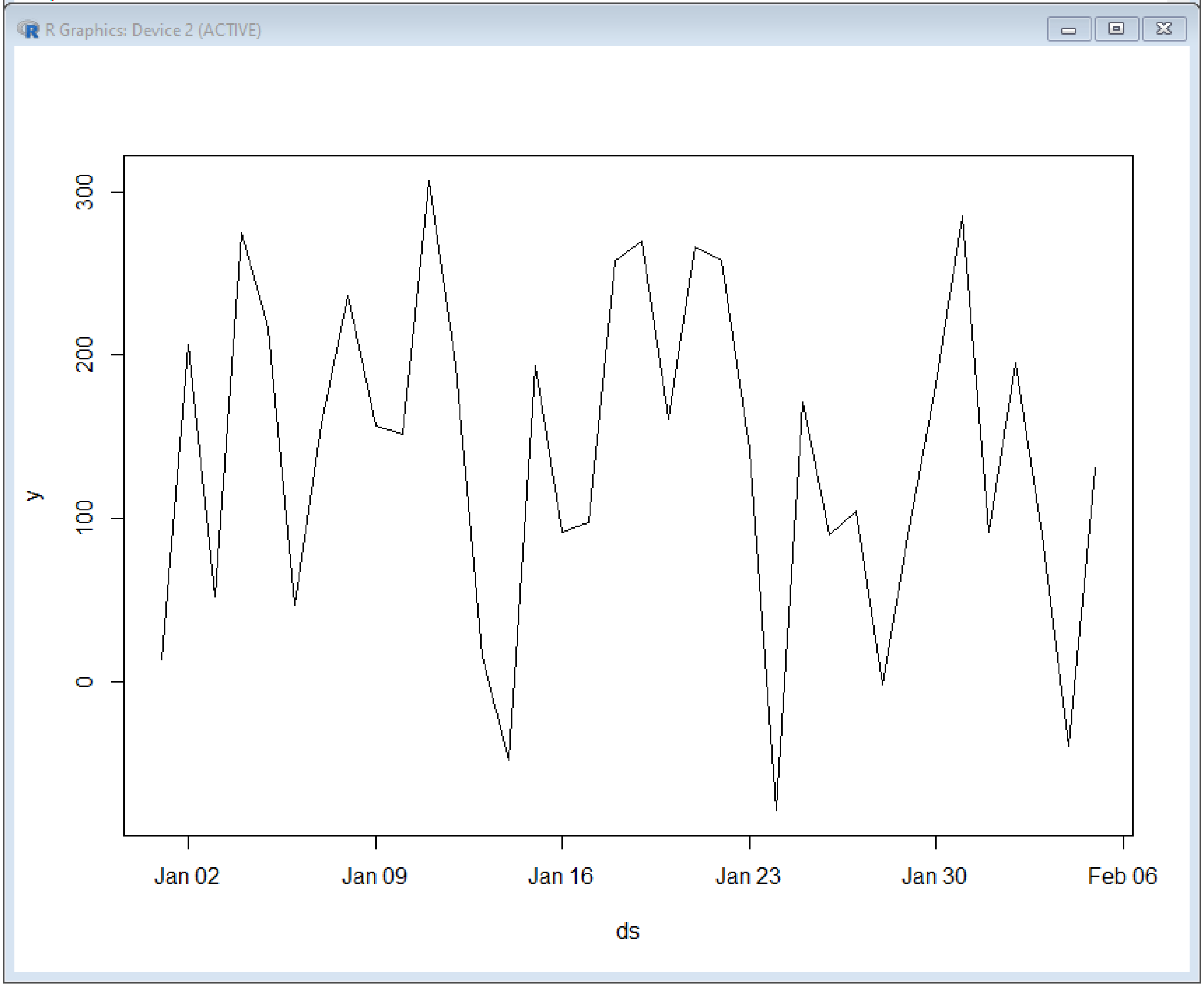
R 3.6.3

Modèle empirique mis en place par l'équipe de Facebook:

https://facebook.github.io/prophet/

Voyons comment le mettre en place. D'abord construisons un jeu de données aléatoire relativement petit:

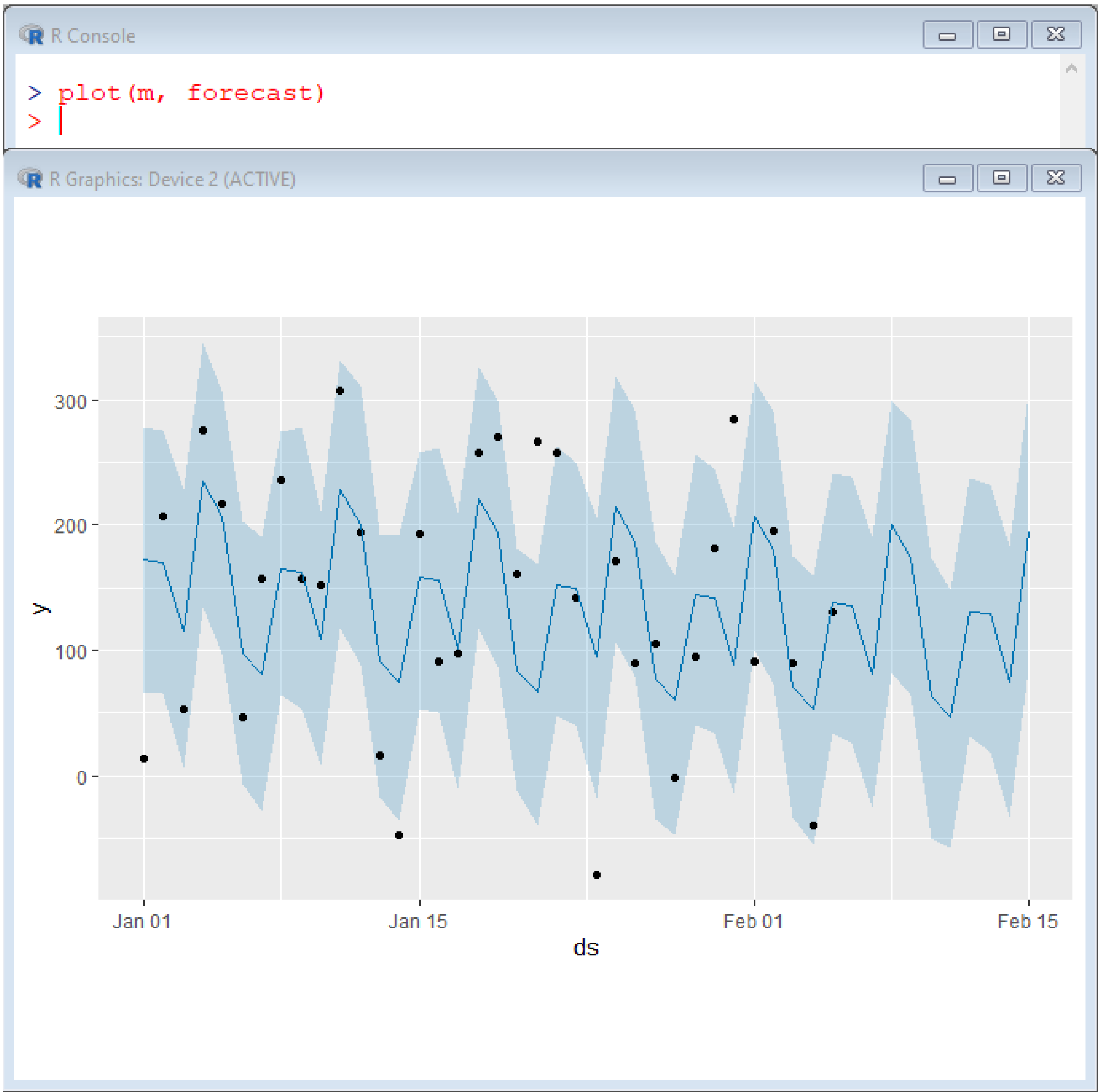
```
> library("prophet") #Bayesian based curve fitting method
> set.seed(1)
> alea<-round(rnorm(36,10,20),2)
> y<-alea+4*(alea+20)+50*sin(alea*2*pi*1/12)+20*sin(alea*2*pi*1/64)
> ds<- seq(from = as.Date("2017-01-01"), to = as.Date("2017-02-05"), by = 'day')
> data.ts<-data.frame(ds,y)
> plot(data.ts,type="l")
>
```



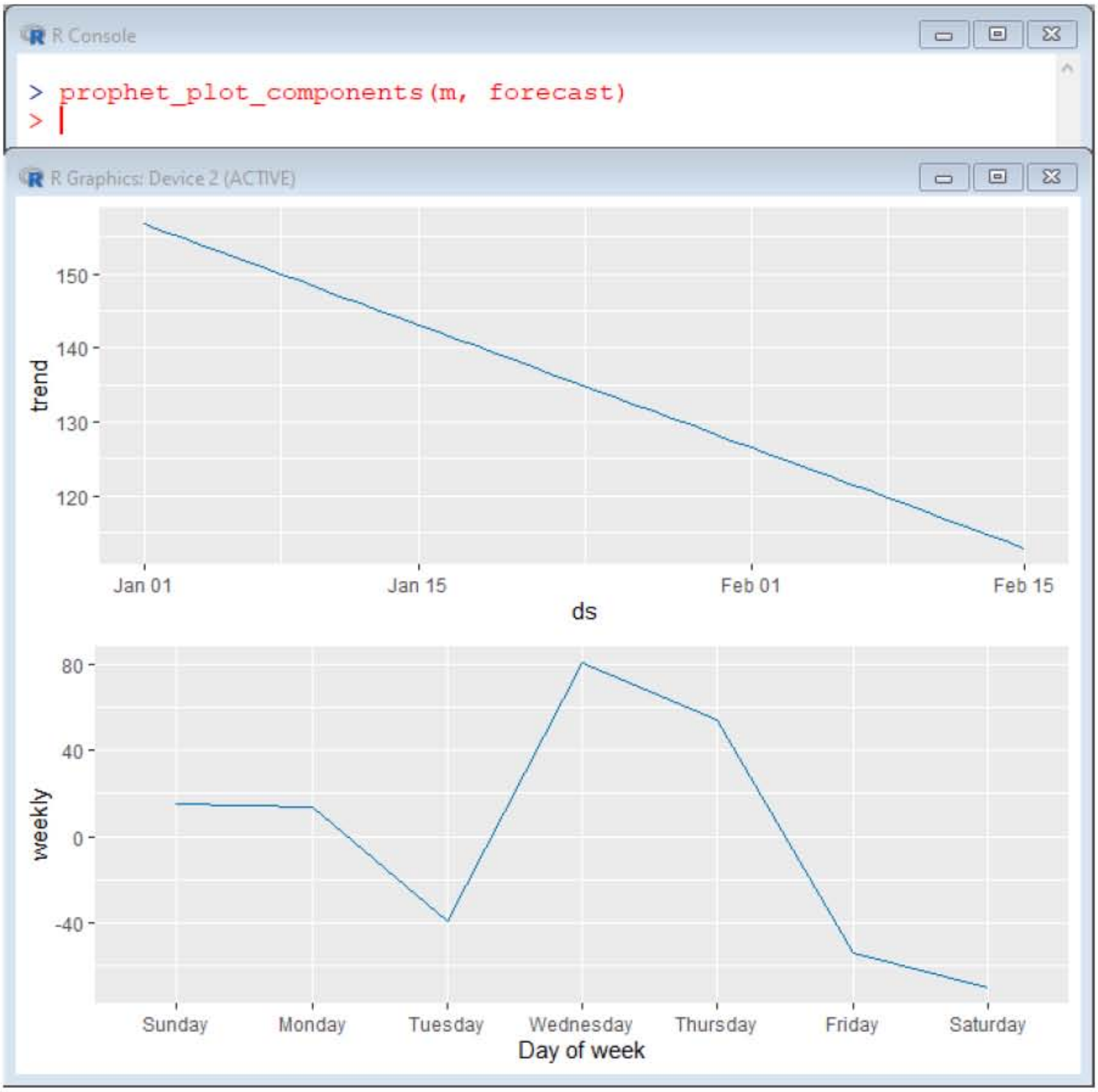
Maintenant utilisons les fonctions **prophet( )**, **make_future_dataframe( )** et **predict( )** du package **prophet** pour obtenir les valeurs futures:

```
> m <- prophet(data.ts)
Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.
Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
> future <- make_future_dataframe(m, periods = 10)
> forecast <- predict(m, future)
> tail(forecast,10)[,c(1:2)]
           ds    trend
37 2017-02-06 121.7123
38 2017-02-07 120.7382
39 2017-02-08 119.7641
40 2017-02-09 118.7900
41 2017-02-10 117.8158
42 2017-02-11 116.8417
43 2017-02-12 115.8676
44 2017-02-13 114.8935
45 2017-02-14 113.9193
46 2017-02-15 112.9452
>
```

Voyons maintenant les graphiques que nous pouvons obtenir:



Et traçons les composantes du modèle Prophet à l'aide de la fonction **prophet_plot_components( )**:



Exemple 387.: Détection des valeurs aberrantes

R 3.6.2

L'exemple illustré ci-dessous vient de:

https://towardsdatascience.com/time-series-anomaly-detection-with-anomalize-library-67472003c003

Nous conseillons aussi la lecture de la vignette du package correspondant:

https://cran.r-project.org/web/packages/anomalize/vignettes/anomalize_methods.html

Voyons l'exemple sans parler de la théorie sous-jacent car il s'agit de techniques totalement empiriques et formellement peu élégantes.

Generalized Extreme Studentized Deviate (GESTD)

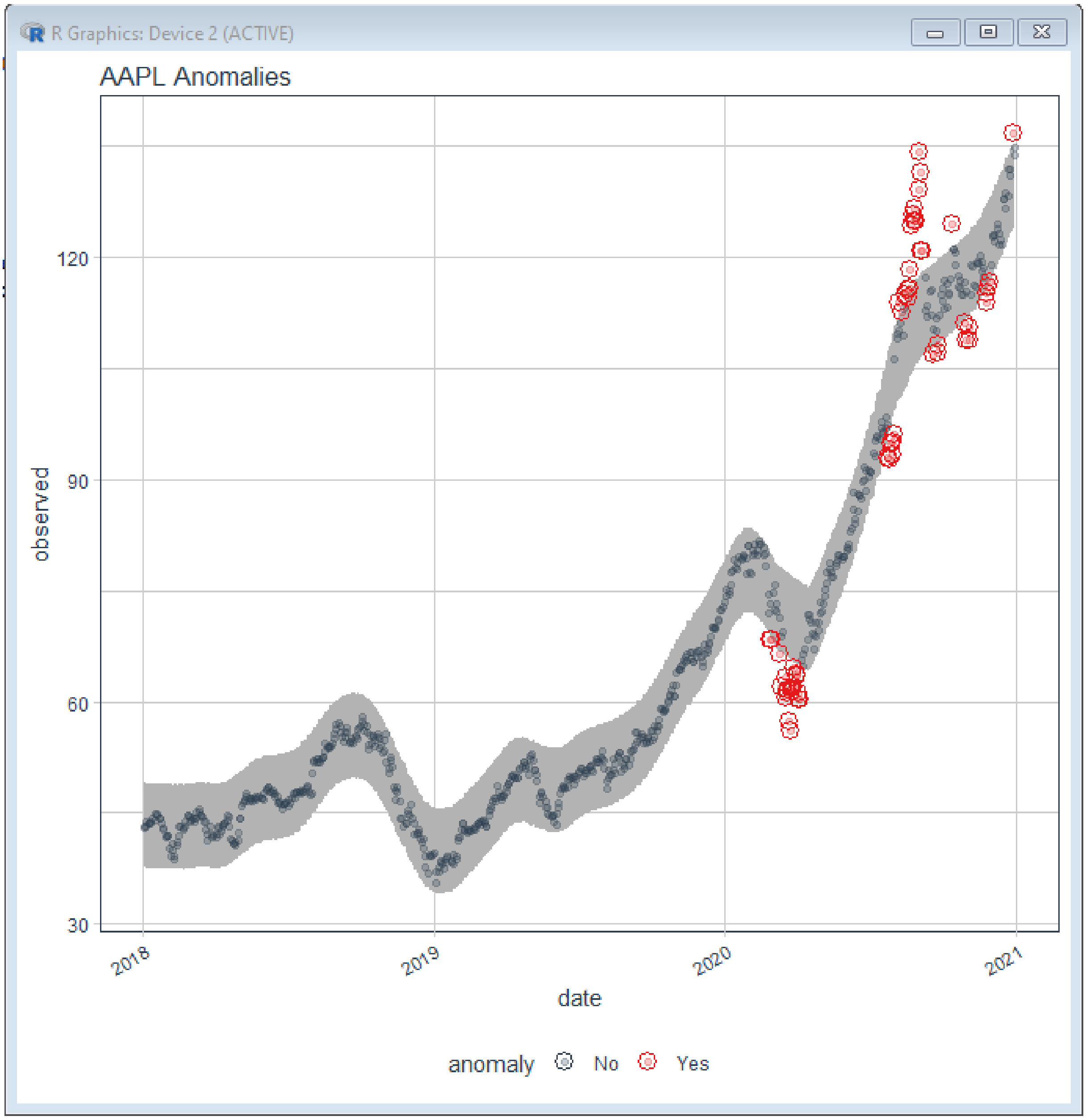
Commençons avec la méthode basée sur le Generalized Extreme Studentized Deviate test[40] (GESD) :

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

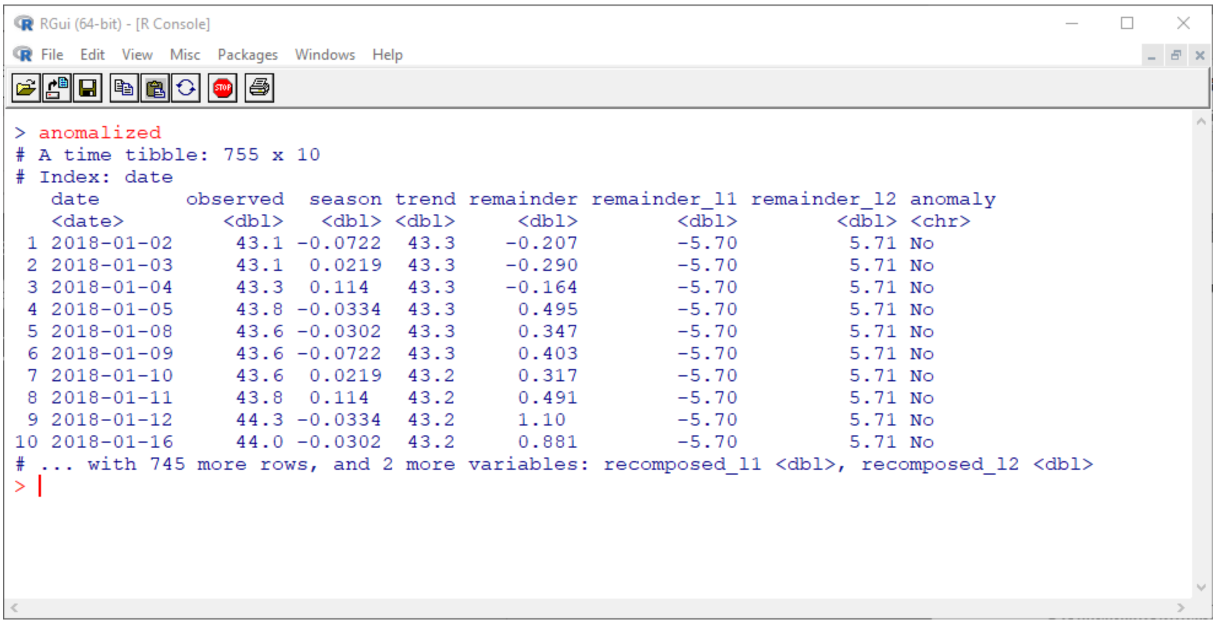
> library("anomalize")
> library("tidyverse")
> library("tibbletime")
> library("tidyquant")
>
> #obtention de quelques données
> data <- tq_get('AAPL',from = "2018-01-01",to = "2020-12-31",
+               get = "stock.prices")
> #un petit coup d'oeil
> head(data)
# A tibble: 6 x 8
  symbol date        open  high   low close    volume adjusted
  <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
1 AAPL   2018-01-02  42.5  43.1  42.3  43.1 102223600     41.4
2 AAPL   2018-01-03  43.1  43.6  43.0  43.1 118071600     41.4
3 AAPL   2018-01-04  43.1  43.4  43.0  43.3  89738400     41.6
4 AAPL   2018-01-05  43.4  43.8  43.3  43.8  94640000     42.1
5 AAPL   2018-01-08  43.6  43.9  43.5  43.6  82271200     41.9
6 AAPL   2018-01-09  43.6  43.8  43.4  43.6  86336000     41.9
>
> #détection des anomalies
> anomalized <- data %>%
+   time_decompose(close,method = "stl") %>% #il y a la méthode Twitter aussi...
+   anomalize(remainder,method="gesd",alpha=0.05,max_anoms=0.2) %>%
+   time_recompose()
Converting from tbl_df to tbl_time.
Auto-index message: index = date
frequency = 5 days
trend = 63 days
>
> #et le graph
> anomalized %>%
+   plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.25)+
+    labs(title = "AAPL Anomalies")
> |
```

[40] Notez que l'ESD généralisée est essentiellement le test de Grubbs appliqué séquentiellement.

Ce qui donne le graphique suivant:

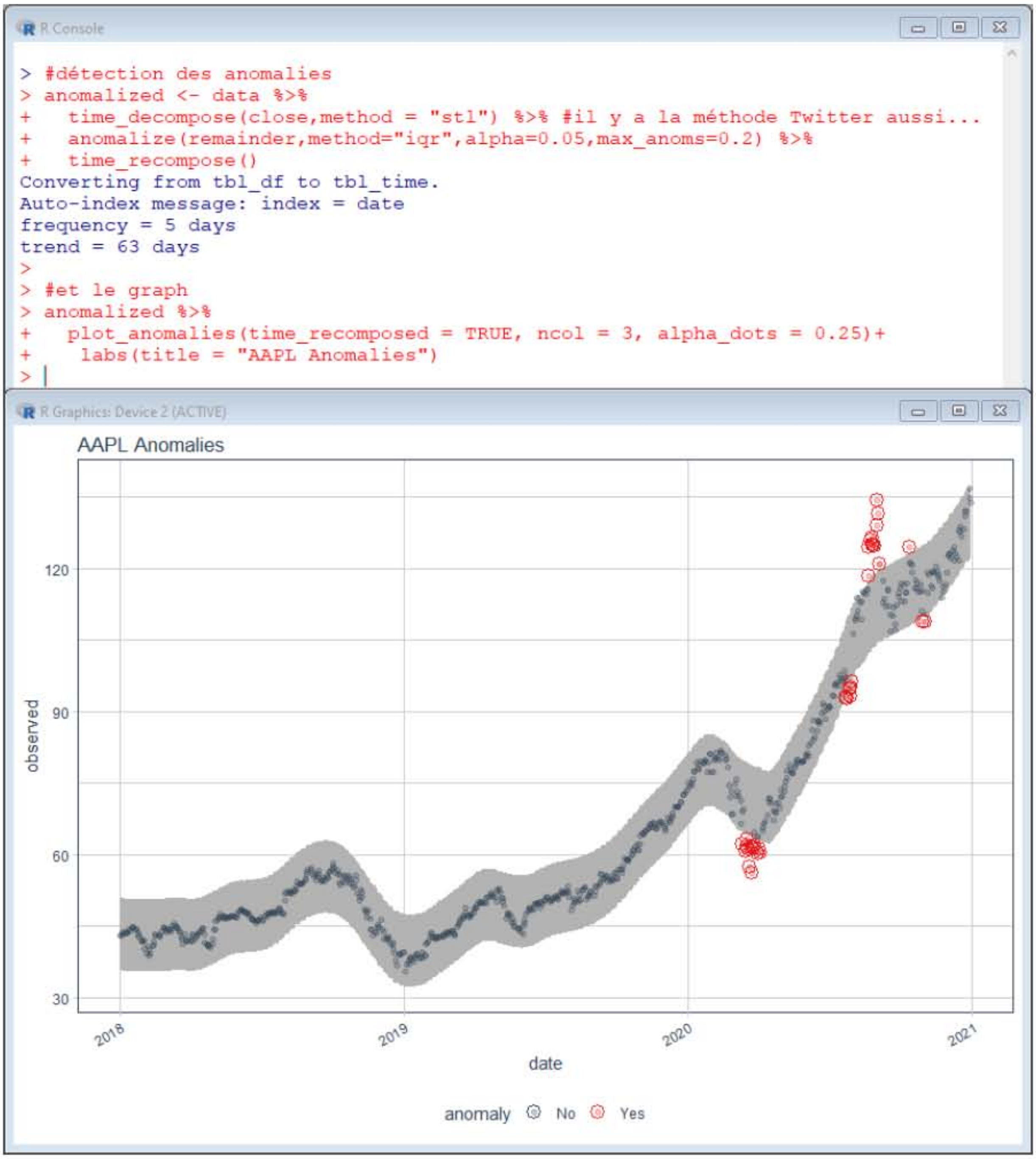


On voit que le jeu de donnéss anomalized contient un colonne *anomaly* qui nous permettra de filter les anomalies au besoin:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> anomalized
A time tibble: 755 x 10
Index: date
date observed season trend remainder remainder_l1 remainder_l2 anomaly
<date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 2018-01-02 43.1 -0.0722 43.3 -0.207 -5.70 5.71 No
2 2018-01-03 43.1 0.0219 43.3 -0.290 -5.70 5.71 No
3 2018-01-04 43.3 0.114 43.3 -0.164 -5.70 5.71 No
4 2018-01-05 43.8 -0.0334 43.3 0.495 -5.70 5.71 No
5 2018-01-08 43.6 -0.0302 43.3 0.347 -5.70 5.71 No
6 2018-01-09 43.6 -0.0722 43.3 0.403 -5.70 5.71 No
7 2018-01-10 43.6 0.0219 43.2 0.317 -5.70 5.71 No
8 2018-01-11 43.8 0.114 43.2 0.491 -5.70 5.71 No
9 2018-01-12 44.3 -0.0334 43.2 1.10 -5.70 5.71 No
10 2018-01-16 44.0 -0.0302 43.2 0.881 -5.70 5.71 No
... with 745 more rows, and 2 more variables: recomposed_l1 <dbl>, recomposed_l2 <dbl>
>

InterQuartile Range (IQR)

Et maintenant comparons juste le graphique avec la méthode IQR:



Exemple 388.: Transformée de Fourier Rapide (FFT)

R 3.0.2

La transformée de Fourier Rapide est un outil très important dans l'étude des séries chronologiques pour authentifier s'il y a des harmoniques périodiques dans une série de mesures et la fréquence des ces dernières.

Je n'ai malheureusement pas trouvé de fonction ou de package proposant une FFT avec exactement le même résultat que celui fourni par Matlab (qui reste souvent le logiciel étalon pour ce qui est relatif au traitement du signal).

Donc à défaut voici une série de commandes (qui peuvent bien évidemment être mis sous la forme d'une fonction simple) permettant d'obtenir exactement le même résultat que Matlab et faisant appel à la fonction native **fft( )** de R (merci à un internaute anonyme!):

C:\Users\Isoz Vincent\Documents\Professionel\Cours\R\Exercices\Finance\FFT.R - Editeur R

```
#paramètres de base de l'algorithme
   T    <- 5
   dt   <- 0.002 #secondes diminuer cette valeur si les fréquences sont élevées!
   n    <- T/dt
   F    <- 1/dt # domaine des fréquences analysées -F/2 -> F/2
   df   <- 1/T
   t    <- seq(0,T,by=dt)
   freq <- 5 #Hz

#function ou vecteur d'entrée à analyser
   y     <- cos(2*pi*60*t) + sin(2*pi*220*t)

#vecteur de fréquences
   f <- 1:length(t)/T

#Transformée de Fourier
   Y     <- fft(y)
   mag   <- sqrt(Re(Y)^2+Im(Y)^2)*2/n #Amplitude

#plot des différentes graphs
   par(mfrow=c(2, 1))
   plot(t,y,type="l",xlim=c(0,T))
   grid(NULL,NULL, col = "lightgray", lty = "dotted",lwd = 1)

   plot(f[1:length(f)/2],mag[1:length(f)/2],type="l",xlab="Frequency,Hz",ylab="Amplitude")
   grid(NULL,NULL, col = "lightgray", lty = "dotted",lwd = 1)
```

Ce qui donne graphiquement en sortie:

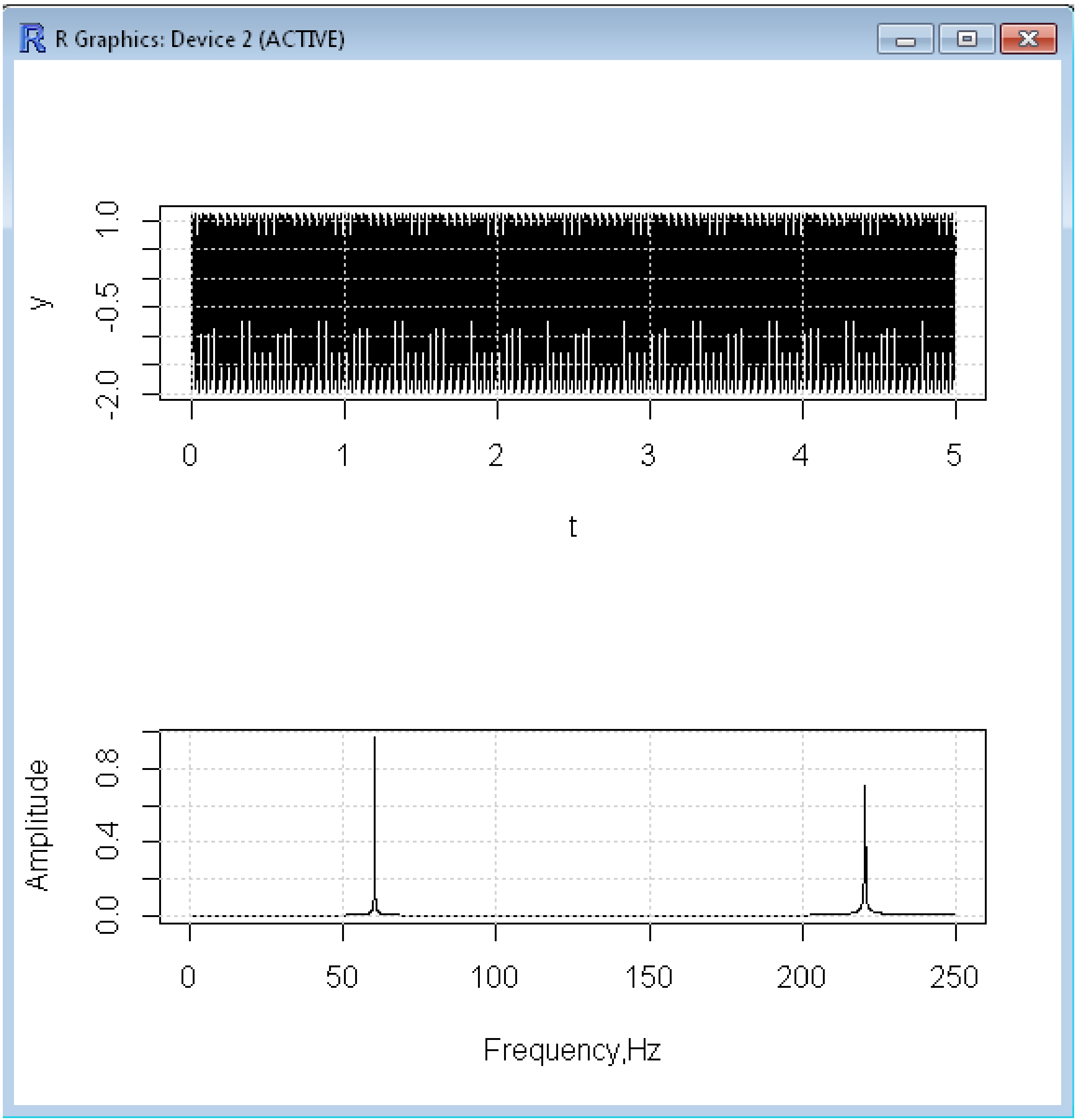


Figure 313 Graph Transformée de Fourier

et voilà :-)

ToDo: Rapatrier les valeurs des pics comme dans MATLAB.

Certaines personnes préfèrent cependant les périodogrammes (dont je ne suis pas un fan):

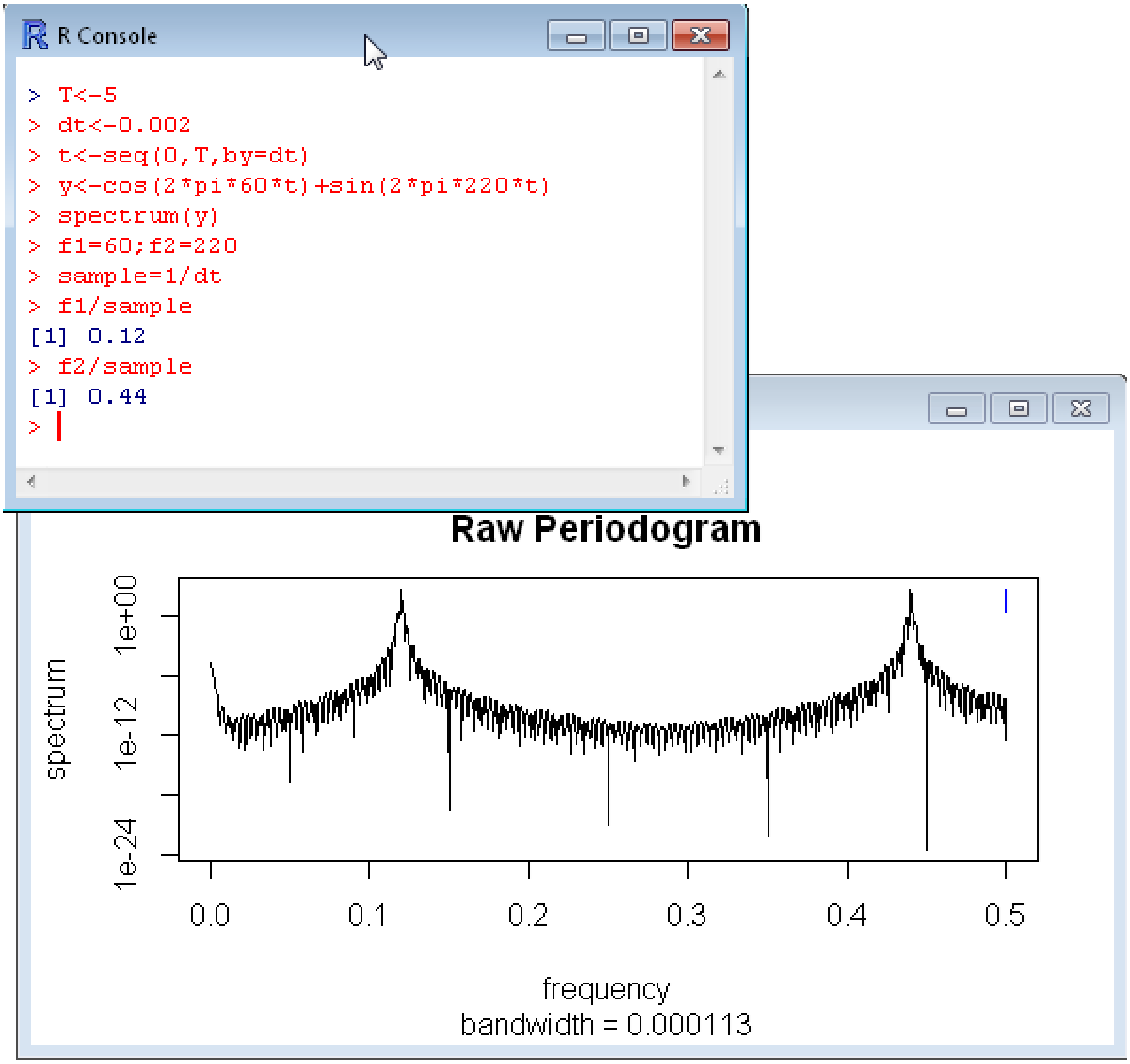


Figure 314 Graph périodogramme transformée de Fourier

Exemple 389.: Transformée de Fourier Inverse Rapide (IFFT)

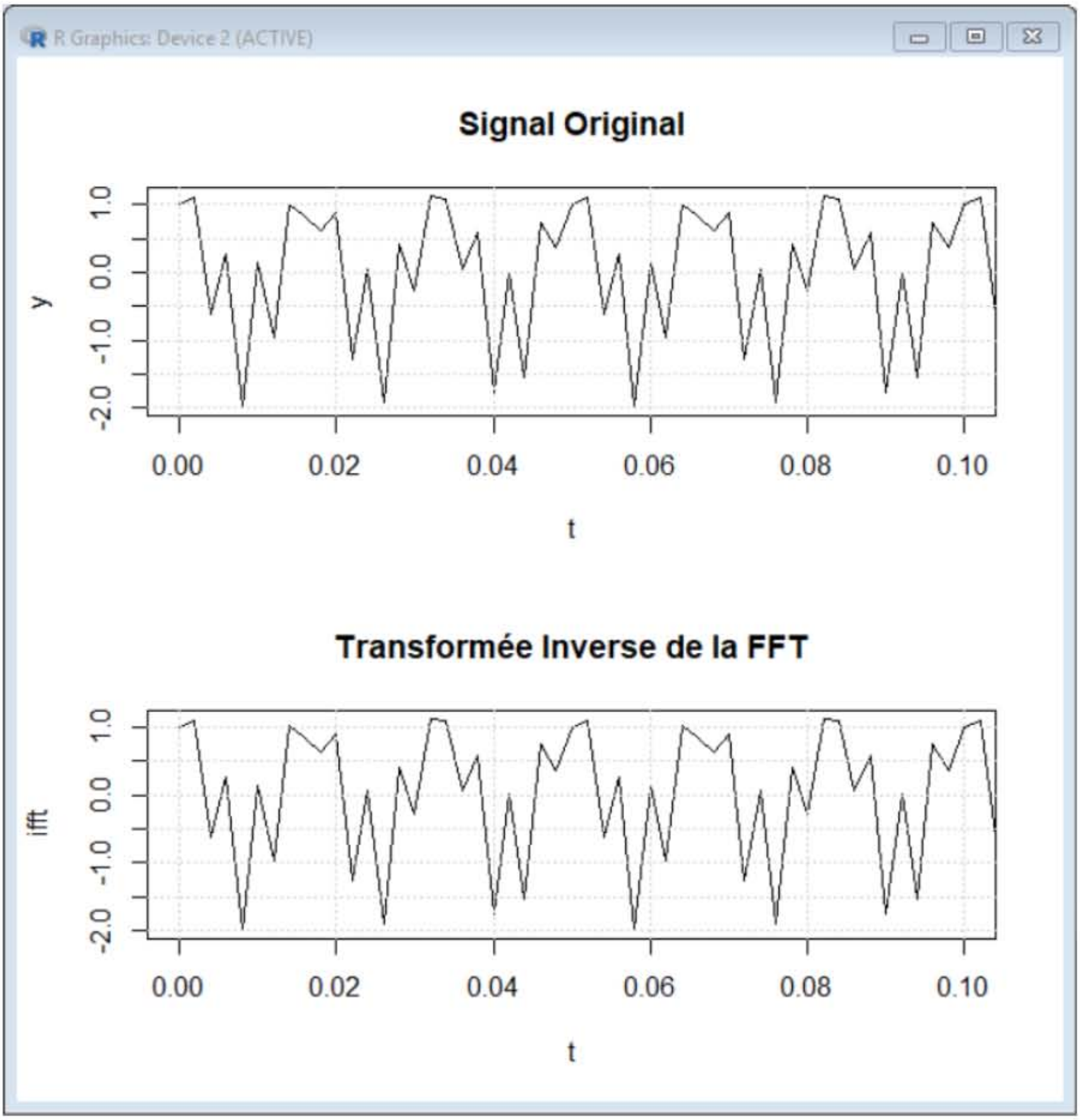
R 3.4.2

Nous allons vérifier ici comment nous pouvons faire la transformée de Fourier inverse du signal dont nous avons fait la transformée de Fourier afin de vérifier si nous retombons bien sur nos pattes!

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #Paramètres de base de l'algorithme
>    T    <- 5
>    dt   <- 0.002 #secondes diminuer cette valeur si les fréquences sont élevées!
>    n    <- T/dt
>    F    <- 1/dt # domaine des fréquences analysées -F/2 -> F/2
>    df   <- 1/T
>    t    <- seq(0,T,by=dt)
>    freq <- 5 #Hz
>
> #Function ou vecteur d'entrée à analyser
>    y     <- cos(2*pi*60*t) + sin(2*pi*220*t)
>
> #Vecteur de fréquences
>    f <- 1:length(t)/T
>
> #Transformée de Fourier
>    Y     <- fft(y)
> #Transformée de Fourier inverse
>    ifft <- fft(fft(y), inverse = TRUE)/length(t)
>
>
> #plot des différentes graphs
>    par(mfrow=c(2, 1))
>    plot(t,y,type="l",xlim=c(0,0.1),main="Signal Original")
>    grid(NULL,NULL, col = "lightgray", lty = "dotted",lwd = 1)
>
>    plot(t,ifft ,type="l",xlim=c(0,0.1),main="Transformée Inverse de la FFT")
Warning message:
In xy.coords(x, y, xlabel, ylabel, log) :
  imaginary parts discarded in coercion
>    grid(NULL,NULL, col = "lightgray", lty = "dotted",lwd = 1)
> 
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Signal Original
y
t
Transformée Inverse de la FFT
ifft
t

Exemple 390.: Analyses de données de panel

R 3.1.1

Les données de Panel sont simplement un groupe des données longitudinales mesurées donc pendant un certain temps. L'exemple le plus typique en entreprise étant la mesure des ventes pendant plusieurs années d'articles (produits différents).

Voyons comment analyser ce type de données avec R. Nous chargeons d'abord le package **pscl** qui contient un jeu de données bien adapté à ce type d'exemple (% de participation aux votations présidentielles américaines depuis 1932 par État et par éléction):

```
> library(pscl)
> print(xyplot(demVote ~ year | state,
+ data=presidentialElections,
+ scales=list(y=list(cex=.35),x=list(cex=.35)),
+ strip=strip.custom(par.strip.text=list(cex=.5)),
+ xlab="Year",
+ ylab="Democratic Vote (%)",
+ type="l")
+ )
> 
```

Cela nous donne:

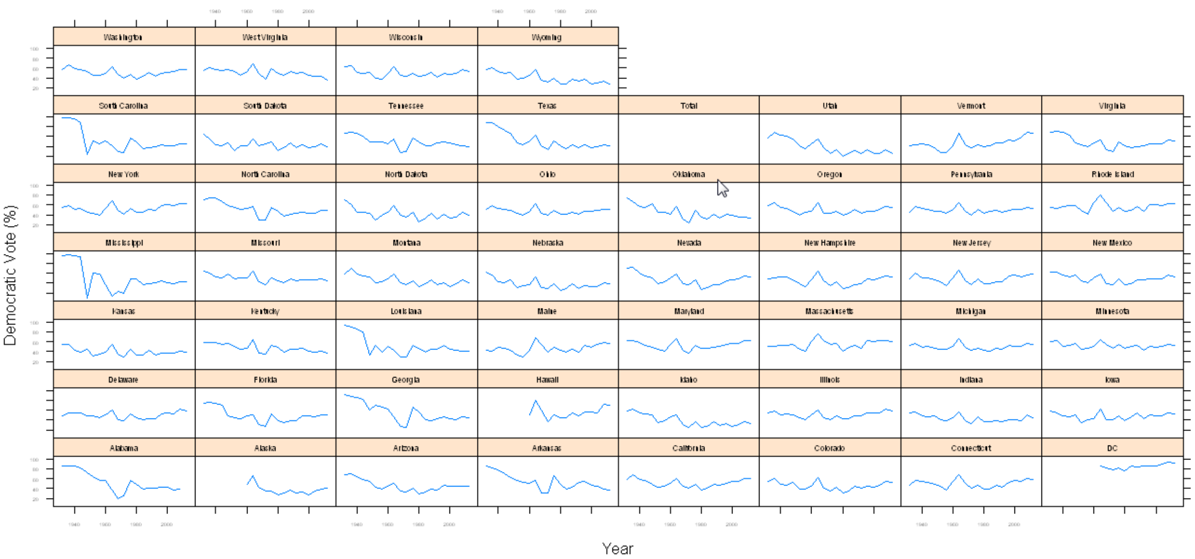
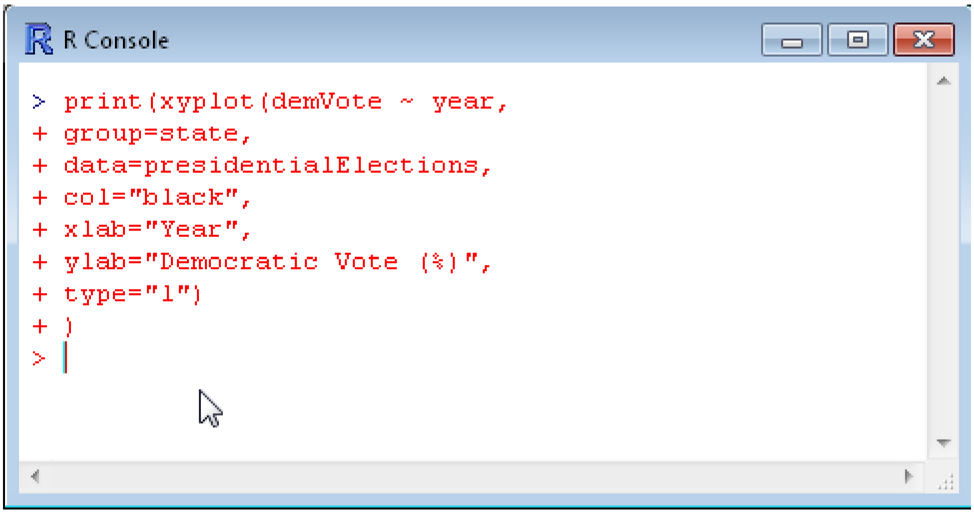
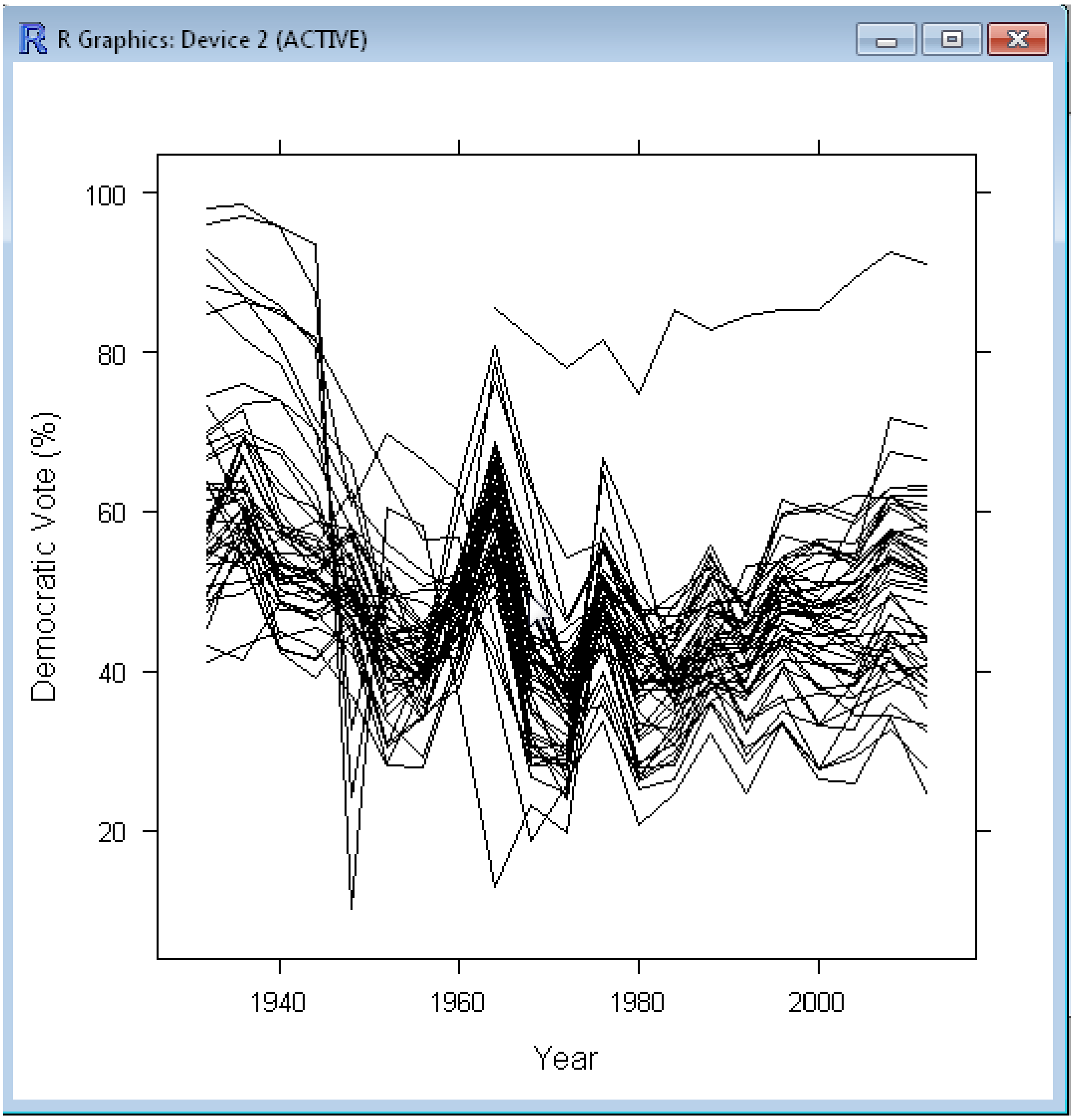


Figure 315 Graph de données de panel

Ceci état pour poser le contexte. Ce qui nous intéresse est de comparer le tout. Donc déjà nous pouvons superposer toutes les données:



Ce qui donne:



Nous savons que rien ne nous interdit de faire une ANOVA par État en passant par une régression car nous avons étudié cela en cours.

Nous avons alors

```
> w <- lm(demVote ~ as.factor(state),subset=year>1968,presidentialElections)
> anova(w)
Analysis of Variance Table

Response: demVote
                  Df Sum Sq Mean Sq F value    Pr(>F)
as.factor(state)  51  38394  752.82  16.734 < 2.2e-16 ***
Residuals        509  22898   44.99
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

La valeur de Fisher est significative ce qui indique que la régression par États peut être considérée comme ayant tous ses coefficients significativement non nuls et donc que ce type d'analyse à un intérêt certain.

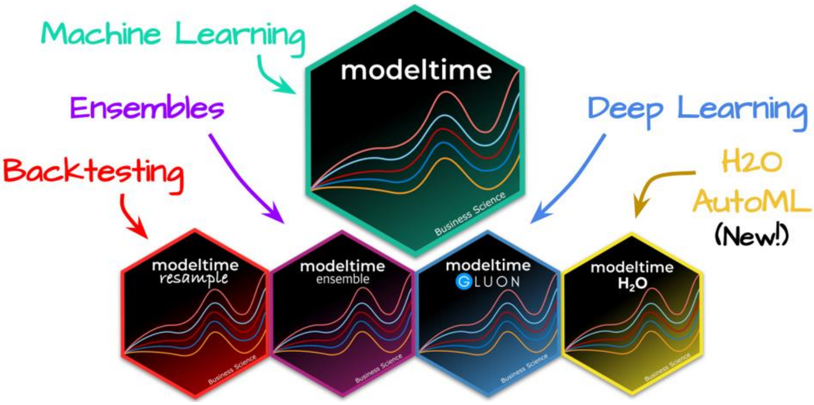
Exemple 391.: Prévisions avec l'apprentissage machine et le package ModelTime

R 4.0.5

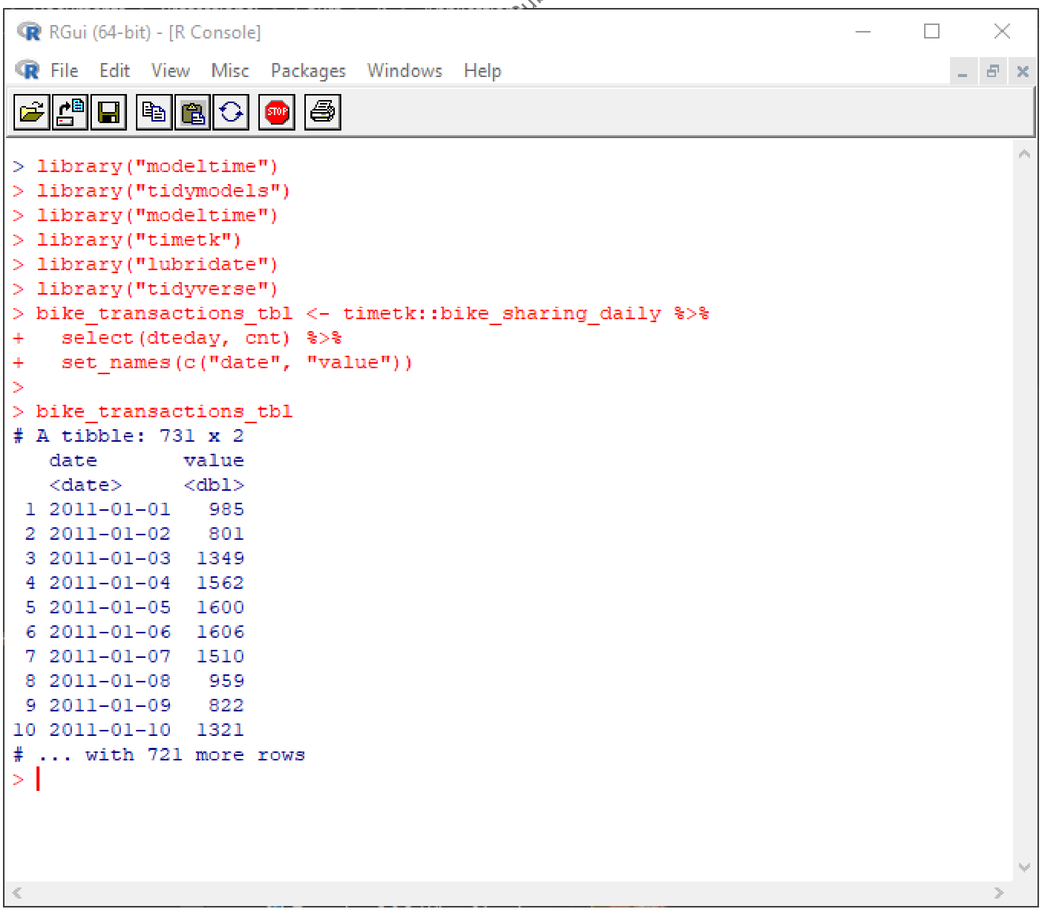
Ce qui suit est un copie traduite de l'excellent article disponible ici (de Matt Dancho):

https://www.business-science.io/code-tools/2020/06/29/introducing-modeltime.html

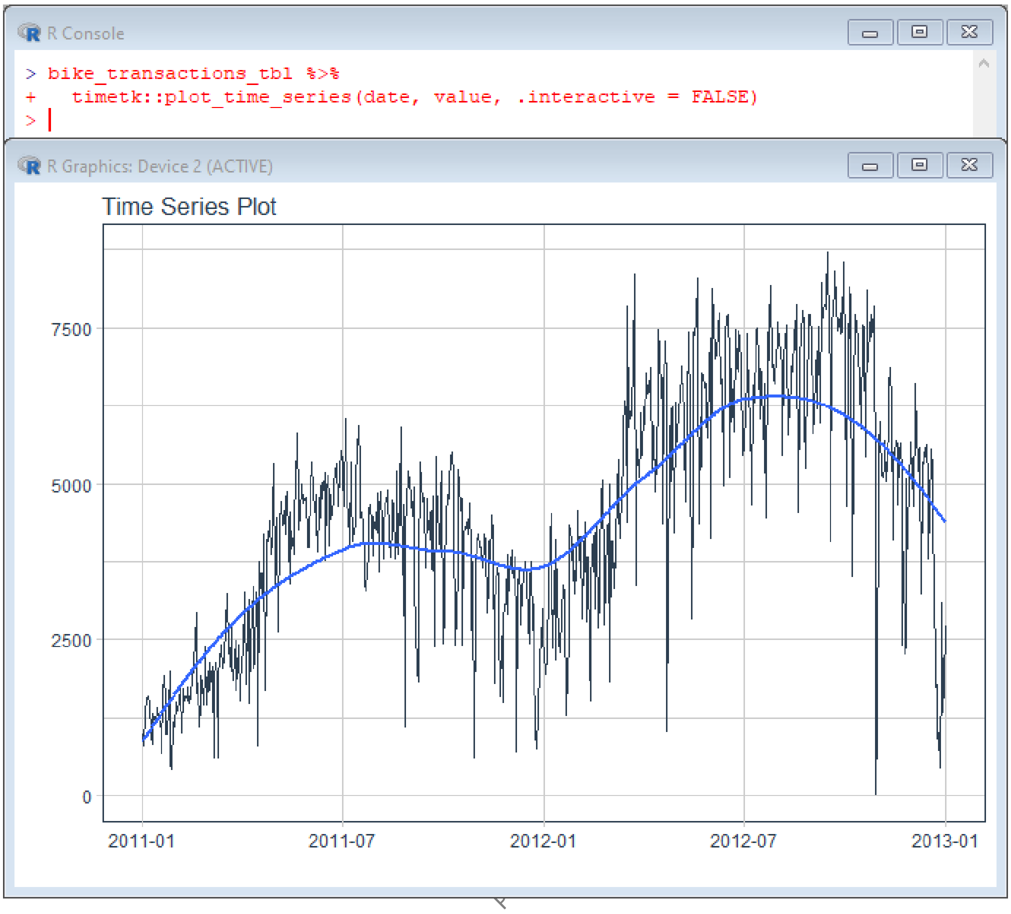
L'idée est d'introduite l'écosystème ModelTime qu'on peut résumer par l'image suivante:



Nous chargeons les packages et le jeu de données que nous allons utiliser tout au long de cet exemple:

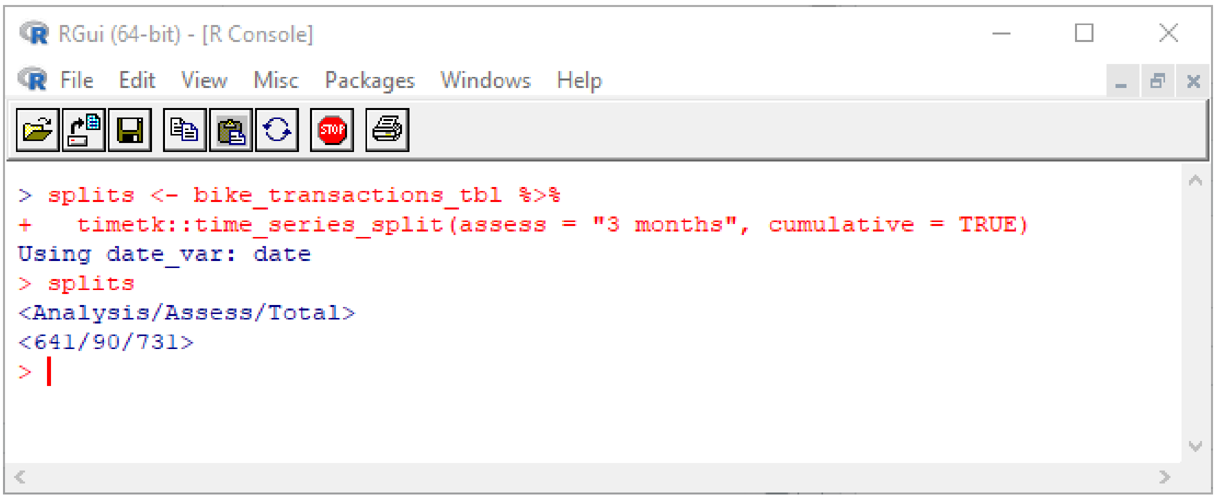


Voyons à quoi cela ressemble:



Ensuite, nous utilisons **time_series_split( )** pour créer un ensemble d'apprentissage/test:

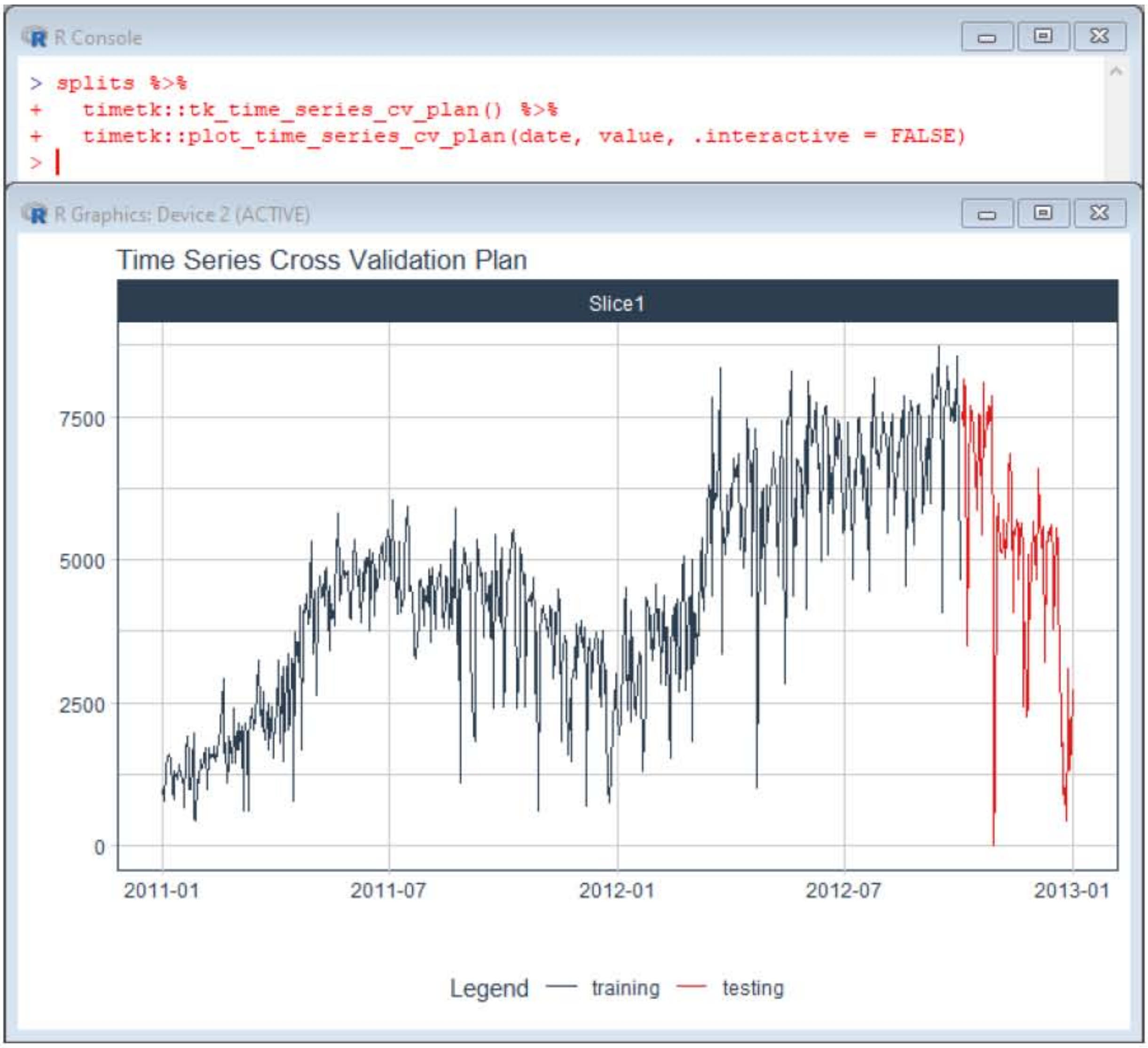
- Le réglage `assess = "3 mois"` indique à la fonction d'utiliser les 3 derniers mois de données comme ensemble de test.
- Le réglage `cumulative = TRUE` indique à l'échantillonnage d'utiliser toutes les données antérieures comme ensemble d'apprentissage.



Ensuite, nous visualisons la répartition apprentissage/test:

- **tk_time_series_cv_plan( )** : convertit l'objet fractionné en un data.frame

- **plot_time_series_cv_plan( )** : trace les données d'échantillonnage de la série chronologique à l'aide des colonnes « date » et « valeur ».



26. Finance Quantitative

La finance quantitative nécessite l'utilisation de nombreuses sources de données financières. Actuellement, l'accès gratuit à ces dernières se réduit comme peau de chagrin… (Yahoo!, Google, Quandl, etc.). Évidemment dans un contexte professionnel, on aura tendance à utiliser principalement Bloomberg pour extraire des données de qualité et sans trop de limites contraignantes. Par ailleurs il existe déjà un package R pour se connecter au terminal Bloomberg via son API et dont les détails sont disponibles en suivant le lien suivant:

https://cran.r-project.org/web/packages/Rblpapi/index.html

Pour ceux souhaitant aussi une liste relativement exhaustive des packages les plus importants en finance pour R, nous recommandons la lecture complète de la page suivante:

https://cran.r-project.org/web/views/Finance.html

Ceci ayant été dit… commençons!

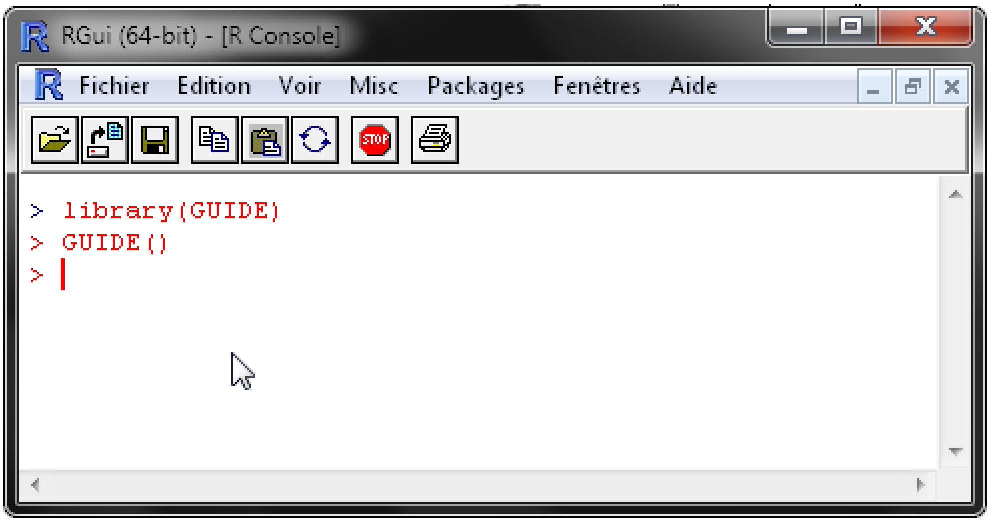
Exemples 392.: Finance "scolaire"

R 3.0.2

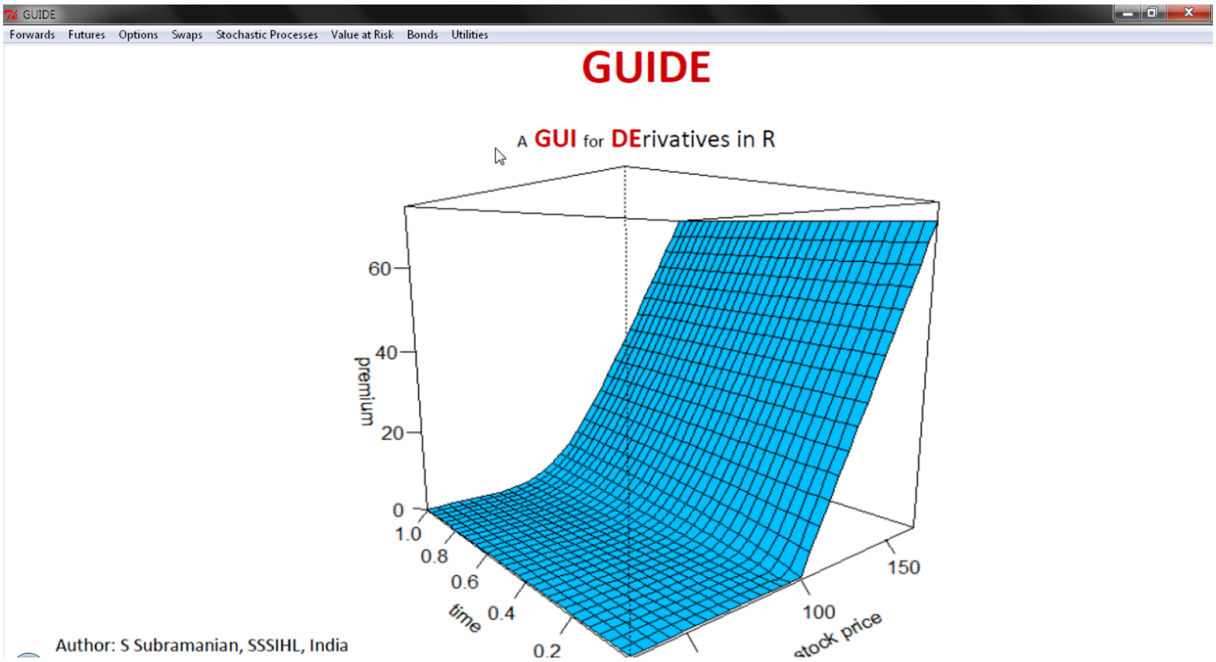
Réviser les bases de la finance avec GUIDE

Nous allons voir ici quelques démos du package **GUIDE** qui intéressera particulièrement les enseignants pour dispenser des formations de finance niveau L1 à M2. Il s'agit vraiment d'un magnifique outil pour accélérer l'apprentissage de nombreux concepts typiquement étudiés lors des CFA, FRM, CAIA et autres du même genre!

Pour cela, après avoir installé ce package, nous écrivons:

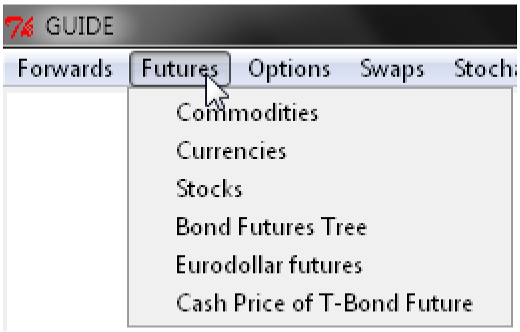


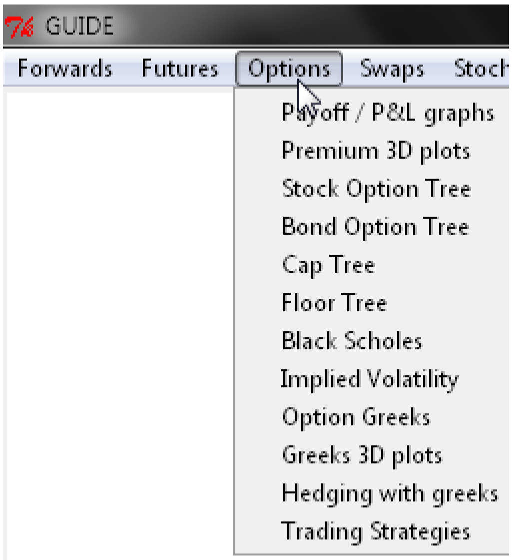
Ce qui donne en plein écran:

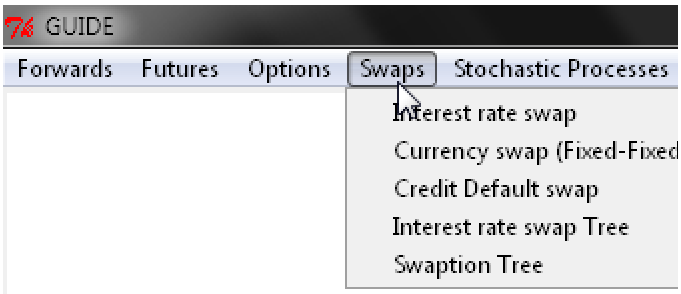


Les outils disponibles sont de gauche à droite:

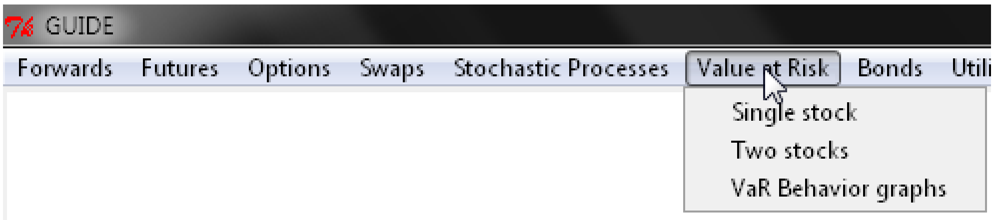


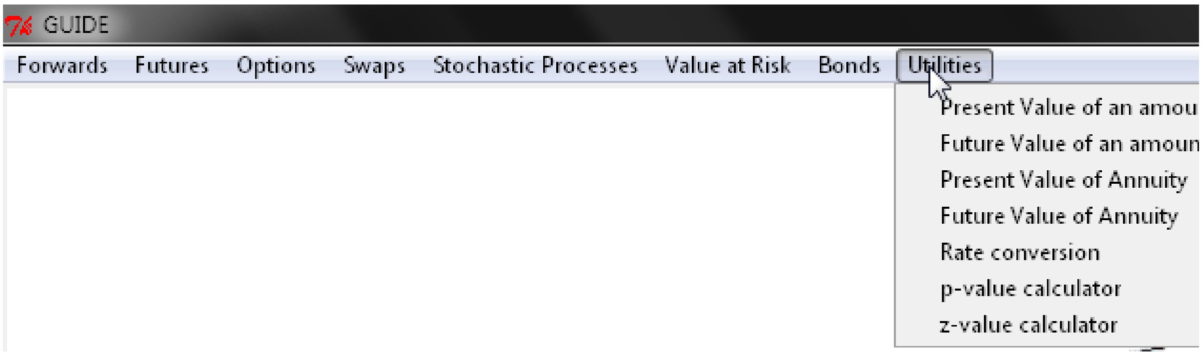










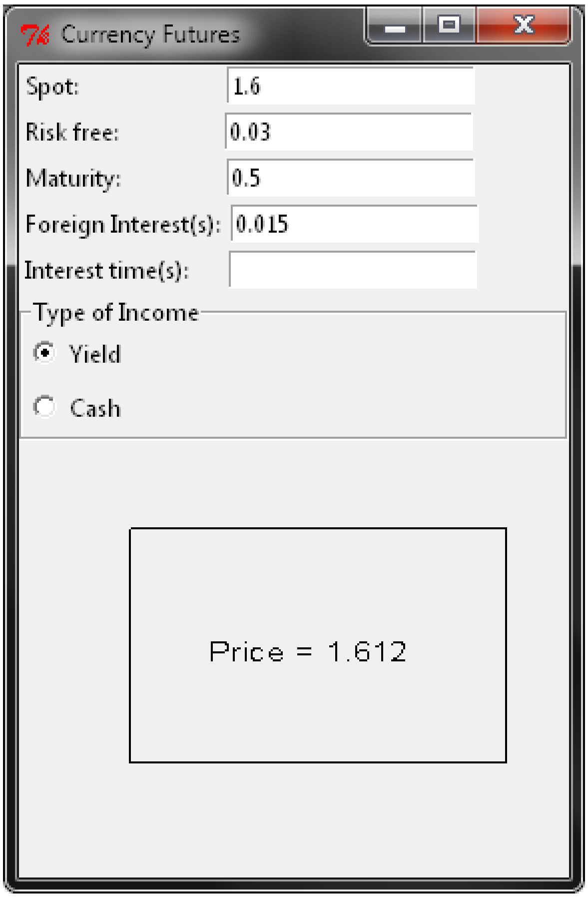


Nous détaillerons chacune des options avec un cas concret ultérieurement pour comparer les résultats au cours théorique. Cela est important car le package **GUIDE** contient le minimum qu'est censé connaître tout bon manager ou employé au bénéfice d'un MBA/CFA/CAIA/FRM ou autre certification du même genre!

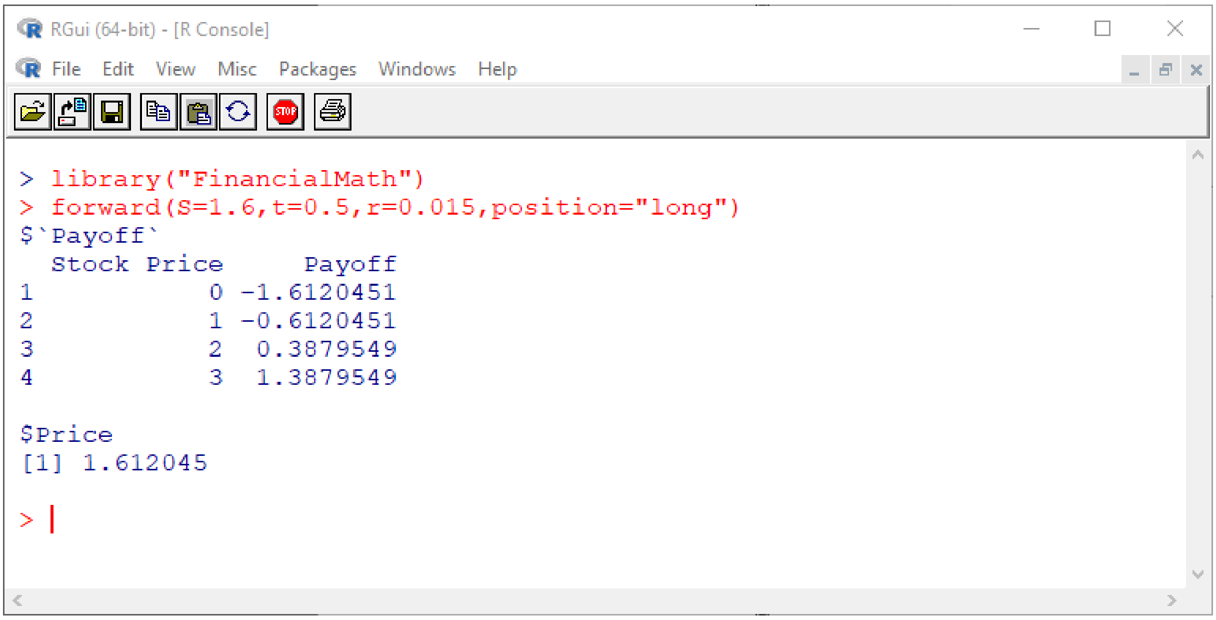
Attention!!! Lorsque vous changez les paramètres dans une boîte de dialogue, il faut valider par la touche ENTER pour que les calculs prennent effet avec les nouveaux paramètres.

Pricing naïf d'un Future/Forward

Nous allons pouvoir vérifier le bon fonctionnement de ce package avec par exemple un petit calcul simple pour lequel nous ne trouvons aucune fonction dans MATLAB. Il s'agit du calcul du prix future sur taux de change que nous avons calculé dans le cours théorique:



Ou sans interface, en utilisant la fonction **forward( )** du package **FinancialMath**:




```
> library("FinancialMath")
> forward(S=1.6,t=0.5,r=0.015,position="long")
$`Payoff`
  Stock Price      Payoff
1           0 -1.6120451
2           1 -0.6120451
3           2  0.3879549
4           3  1.3879549

$Price
[1] 1.612045

>
```

Nous retrouvons donc bien la même valeur que dans le cours théorique, quelle que soit la méthode!

Rendements discrets, rendements logarithmiques leur moyenne et écart-type

Nous allons reproduire ici les exemples du cours théoriques calculés à la main et faits aussi avec Microsoft Excel.

Nous avons alors en comparant:

- La moyenne arithmétique et l'écart-type des rendements
- La moyenne géométrique avec l'écart-type géométrique des rendements décimaux
- La moyenne arithmétique avec l'écart-type des rendements logarithmiques

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #quelques rendements
> R<-c(2,1.5,3,5,6,0)
> #on les normalise
> R_norm<-R/100+1
>
> #d'abord la moyenne arithmétique
> mean(R)
[1] 2.916667
> sd(R)
[1] 2.245366
>
> #la moyenne géometrique
> library("EnvStats")
> (geoMean(R_norm)-1)*100
[1] 2.896302
> (geoSD(R_norm)-1)*100
[1] 2.202035
>
> #enfin pour terminer avec les rendements logarithmiques
> (mean(log(R/100+1,base=exp(1))))*100
[1] 2.855152
> (sd(log(R/100+1,base=exp(1))))*100
[1] 2.17814
>
```

Ce qui est conforme aux calculs faits à la main et avec Microsoft Excel.

Le lecteur aura probablement noté l'utilisation du package **EnvStats** et ses fonctions **geoMean( )** et **geoSD( )**.

Plus proche matrice de corrélation réelle (semi-positive définie)

Dans le cours théorique nous avons démontré qu'une matrice de variances-covariances (et donc de corrélation) devait être positive définie et nous avons parlé du fait que des praticiens dans la finance (et pas seulement!) construisent des matrices de corrélations empiriques (quasi-corrélations) et passent ces dernières dans des algorithmes (il en existe plusieurs types) pour les mettre en conformité avec les propriétés mathématiques ad hoc.

Nous allons dans l'exemple qui va suivre comparer au résultat obtenu avec le solveur de Microsoft Excel et la méthode basée sur la norme de Frobenius mais en utilisant la fonction **nearPD( )** (pour "nearest positive definite") du package **Matrix** de R.

D'abord vérifions que la matrice estimée soit effectivement pas positive semi-définie:

```
> EstimatedCM<-matrix(c(1,0.8,-0.4,-0.2,
+      0.8,1,-0.3,0.5,
+      -0.4,-0.3,1,0.5,
+      -0.2,0.5,0.5,1),nrow=4,ncol=4)
> #on affiche la matrice pour le principe
> EstimatedCM
     [,1] [,2] [,3] [,4]
[1,]  1.0  0.8 -0.4 -0.2
[2,]  0.8  1.0 -0.3  0.5
[3,] -0.4 -0.3  1.0  0.5
[4,] -0.2  0.5  0.5  1.0
> #pour le fun on check les valeurs propres
> eigen(ECMatrix_Corr)$values
[1]  2.03852755  1.57260336  0.47346368 -0.08459459
> 
```

Donc oui… Voyons comment corriger le tir:

```
> Matrix::nearPD(EstimatedCM)
$mat
4 x 4 Matrix of class "dpoMatrix"
           [,1]       [,2]       [,3]       [,4]
[1,]  1.0209364  0.7720221 -0.4098328 -0.1787085
[2,]  0.7720221  1.0373877 -0.2868602  0.4715476
[3,] -0.4098328 -0.2868602  1.0046179  0.4900005
[4,] -0.1787085  0.4715476  0.4900005  1.0216526

$eigenvalues
[1] 2.038528e+00 1.572603e+00 4.734637e-01 2.038528e-08

$corr
[1] FALSE

$normF
[1] 0.08459461

$iterations
[1] 2

$rel.tol
[1] 0

$converged
[1] TRUE

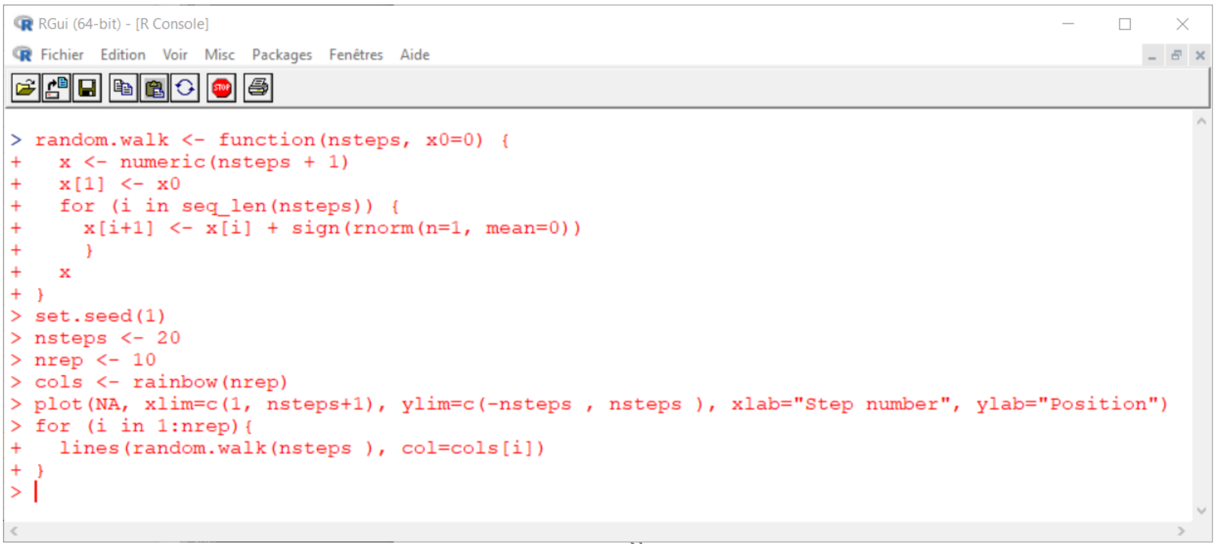
attr(,"class")
[1] "nearPD"
> 
```

Comme nous nous y attendions, le choix de l'algorithme influe quand même significativement sur les valeurs de la plus proche matrice de corrélation. Mais au moins c'est une vraie matrice positive définie maintenant!

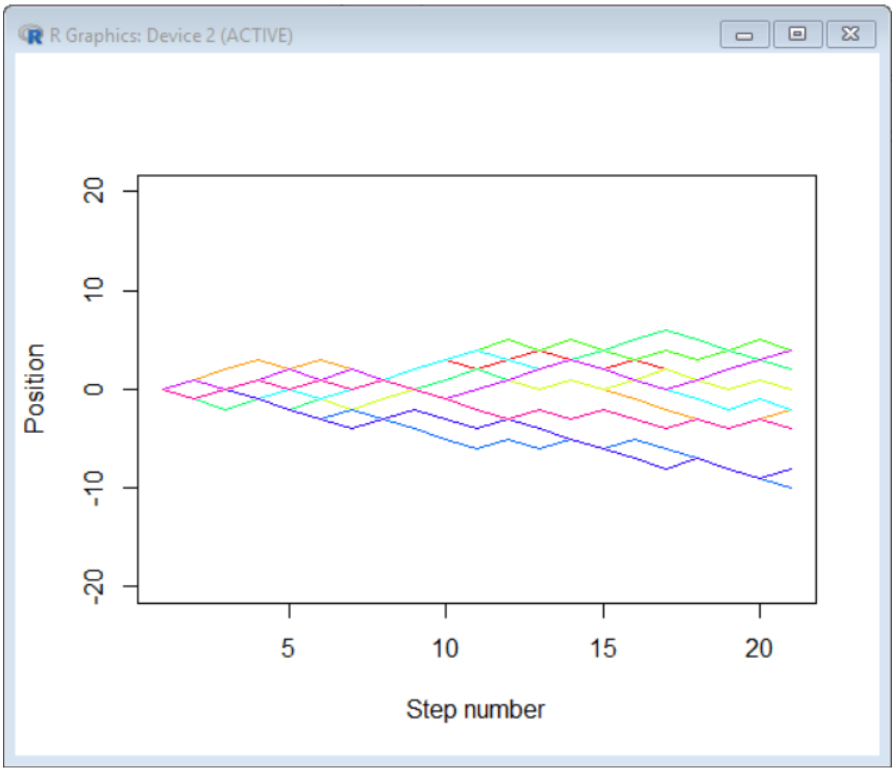
Marche aléatoire

Nous avons vu dans le cours théorique que aussi bien en physique théorique (mécanique statistique) qu'en finance (dont une proportion importante découle de la mécanique statistique…) la marche aléatoire est utile pour comprendre la construction intellectuelle de la valorisation d'instruments financier.

Voyons donc comment créer une marche aléatoire simple avec le graph fameux correspondant (notez que comprendre ce genre de script vous sera utile plus tard pour apprendre la programmation dans R):



Ce qui donne graphiquement:



Reste à faire des arbres de type CRR avec drift, Jarrow-Rudd, Leisen-Reimer et Tian comme nous le verrons à la page 2538.

Exemple 393.: Comptes de produits et charges

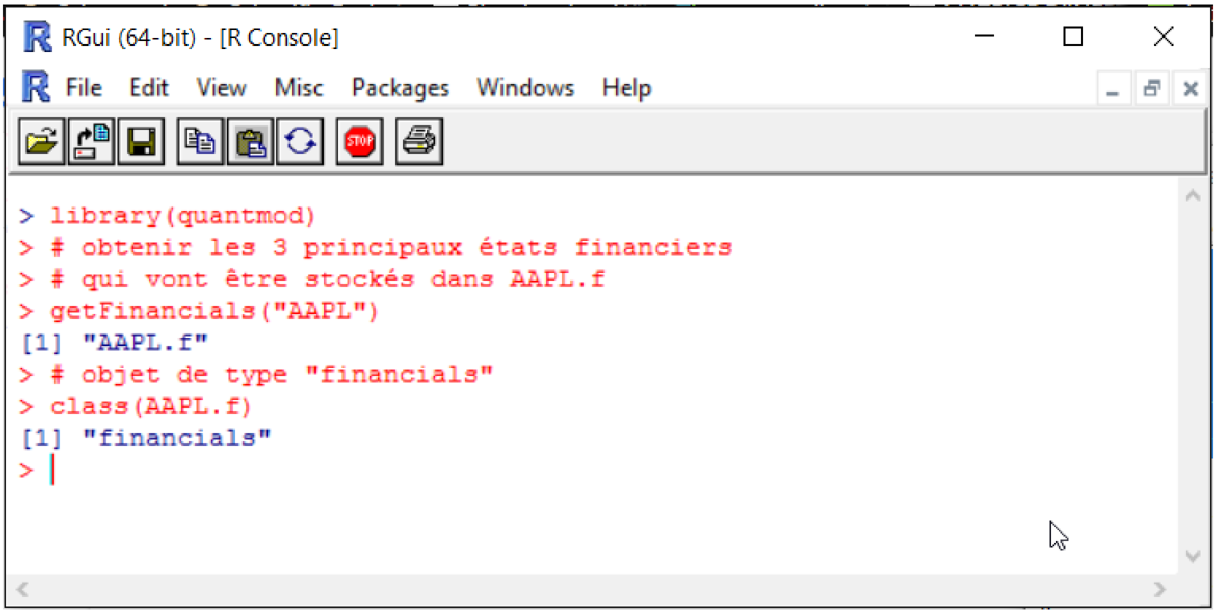
R 3.2.3

En finance avant d'investir des certaines obligations ou actions il peut être intéressant d'avoir une idée de leur état financier via une analyse de leurs comptes de produits et charges (CPC) conformément aux standards du International Financial Reporting Standards (IFRS).

Le package **quantmod** permet d'obtenir ces informations rapidement via les bases de données Google et les fonctions principales **getFinancial** et **viewFinancials**.

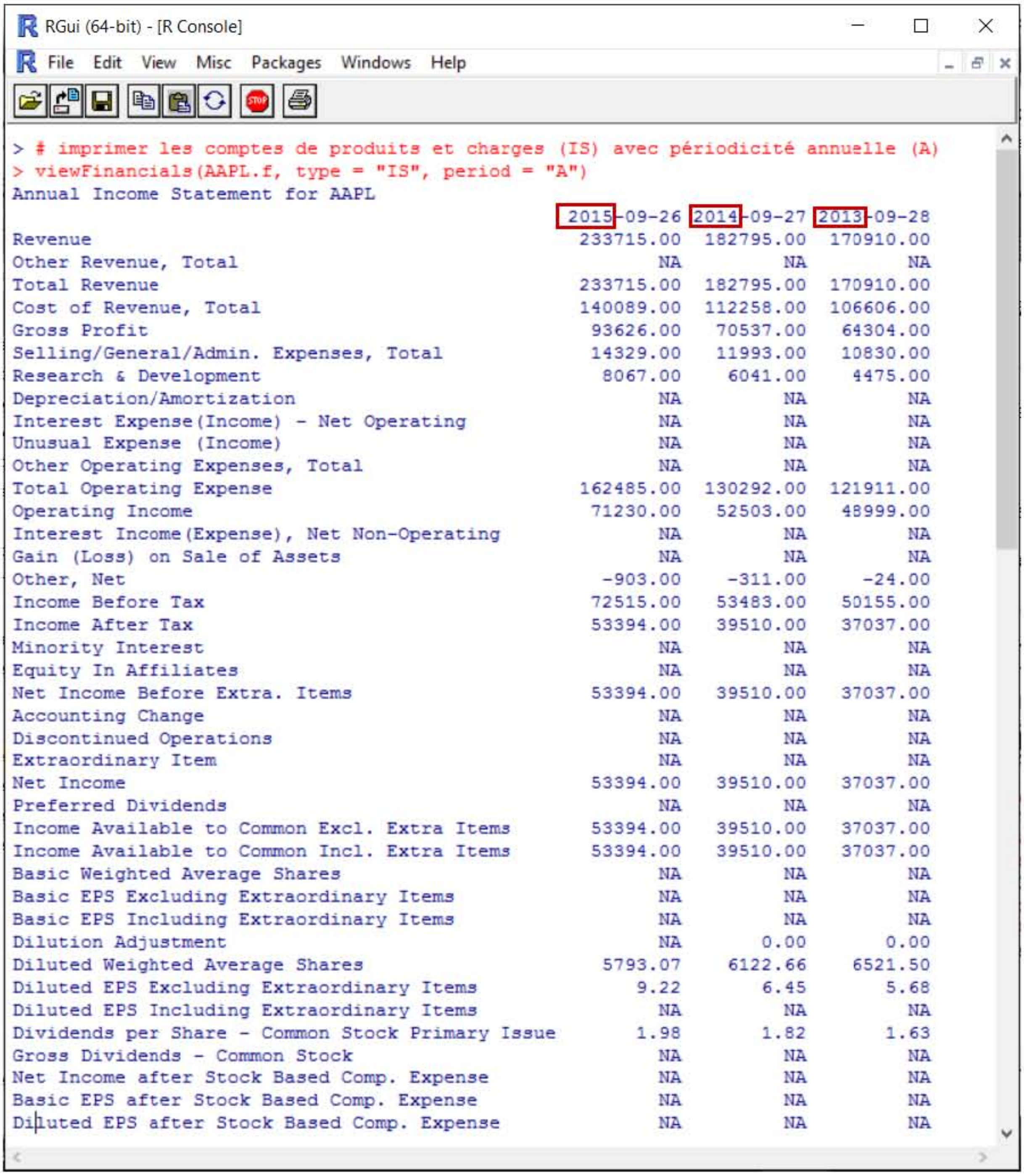
Voyons un exemple avec Apple et les données qui sont publiques… (faut pas compter au 21ème siècle y avoir les grandes entreprises suisses car elles savent à peine utiliser correctement un système informatique…).

D'abord on charge les données:



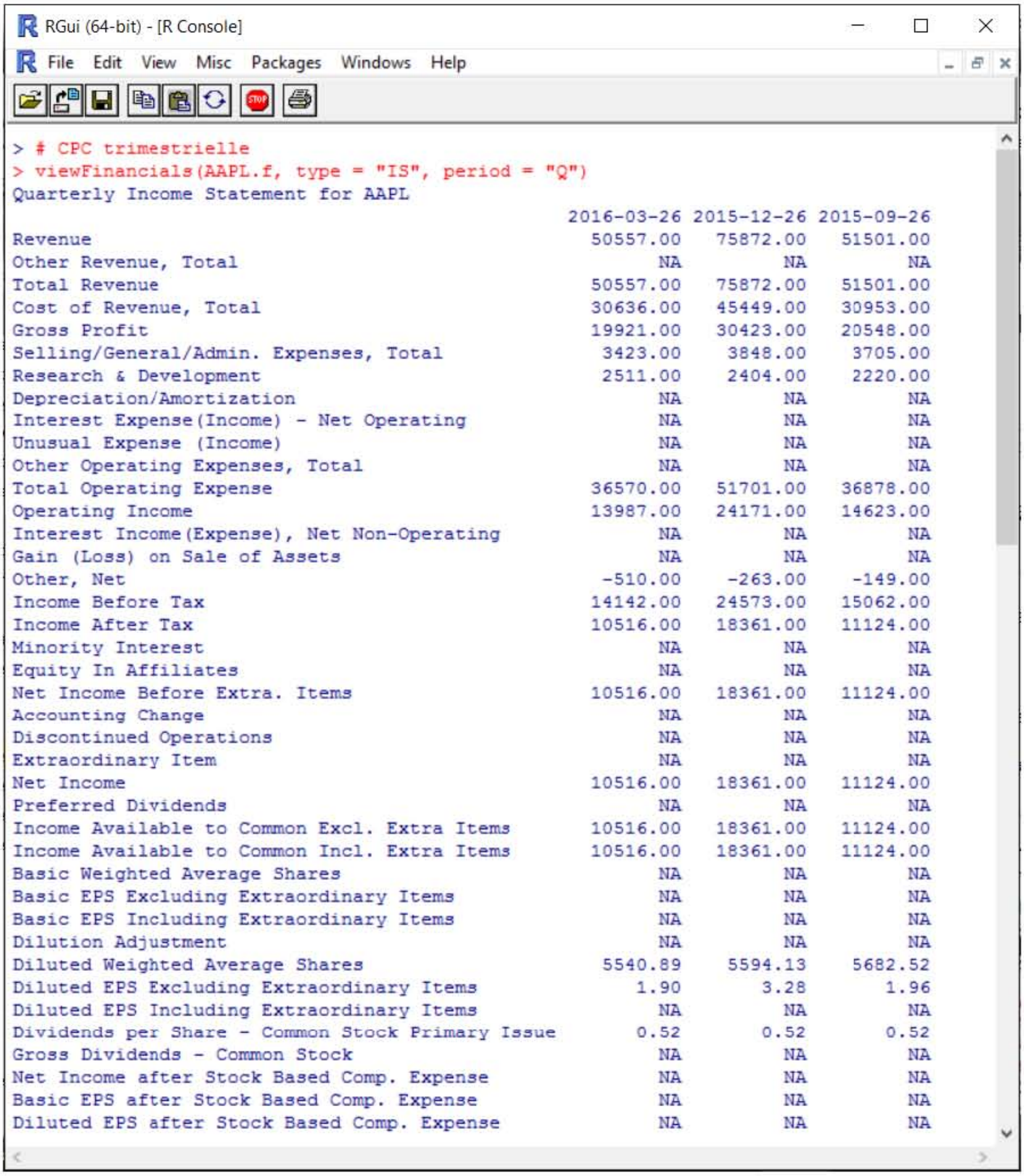
Attention!!!! Depuis Mars 2018, Google a désactivé le service qu'utilisait le package *quantmod* ci-dessus!!!!!!!! Mieux vaut passer par la package *finreportr* comme nous allons le voir plus bas!

Ce qui donne annuellement:



```
> # imprimer les comptes de produits et charges (IS) avec périodicité annuelle (A)
> viewFinancials(AAPL.f, type = "IS", period = "A")
Annual Income Statement for AAPL
                                              2015-09-26 2014-09-27 2013-09-28
Revenue                                        233715.00  182795.00  170910.00
Other Revenue, Total                                  NA         NA         NA
Total Revenue                                  233715.00  182795.00  170910.00
Cost of Revenue, Total                         140089.00  112258.00  106606.00
Gross Profit                                    93626.00   70537.00   64304.00
Selling/General/Admin. Expenses, Total          14329.00   11993.00   10830.00
Research & Development                           8067.00    6041.00    4475.00
Depreciation/Amortization                             NA         NA         NA
Interest Expense(Income) - Net Operating              NA         NA         NA
Unusual Expense (Income)                              NA         NA         NA
Other Operating Expenses, Total                       NA         NA         NA
Total Operating Expense                        162485.00  130292.00  121911.00
Operating Income                                71230.00   52503.00   48999.00
Interest Income(Expense), Net Non-Operating           NA         NA         NA
Gain (Loss) on Sale of Assets                         NA         NA         NA
Other, Net                                       -903.00    -311.00     -24.00
Income Before Tax                               72515.00   53483.00   50155.00
Income After Tax                                53394.00   39510.00   37037.00
Minority Interest                                     NA         NA         NA
Equity In Affiliates                                  NA         NA         NA
Net Income Before Extra. Items                  53394.00   39510.00   37037.00
Accounting Change                                     NA         NA         NA
Discontinued Operations                               NA         NA         NA
Extraordinary Item                                    NA         NA         NA
Net Income                                      53394.00   39510.00   37037.00
Preferred Dividends                                   NA         NA         NA
Income Available to Common Excl. Extra Items    53394.00   39510.00   37037.00
Income Available to Common Incl. Extra Items    53394.00   39510.00   37037.00
Basic Weighted Average Shares                         NA         NA         NA
Basic EPS Excluding Extraordinary Items               NA         NA         NA
Basic EPS Including Extraordinary Items               NA         NA         NA
Dilution Adjustment                                   NA       0.00       0.00
Diluted Weighted Average Shares                  5793.07    6122.66    6521.50
Diluted EPS Excluding Extraordinary Items           9.22       6.45       5.68
Diluted EPS Including Extraordinary Items             NA         NA         NA
Dividends per Share - Common Stock Primary Issue    1.98       1.82       1.63
Gross Dividends - Common Stock                        NA         NA         NA
Net Income after Stock Based Comp. Expense            NA         NA         NA
Basic EPS after Stock Based Comp. Expense             NA         NA         NA
Diluted EPS after Stock Based Comp. Expense           NA         NA         NA
```

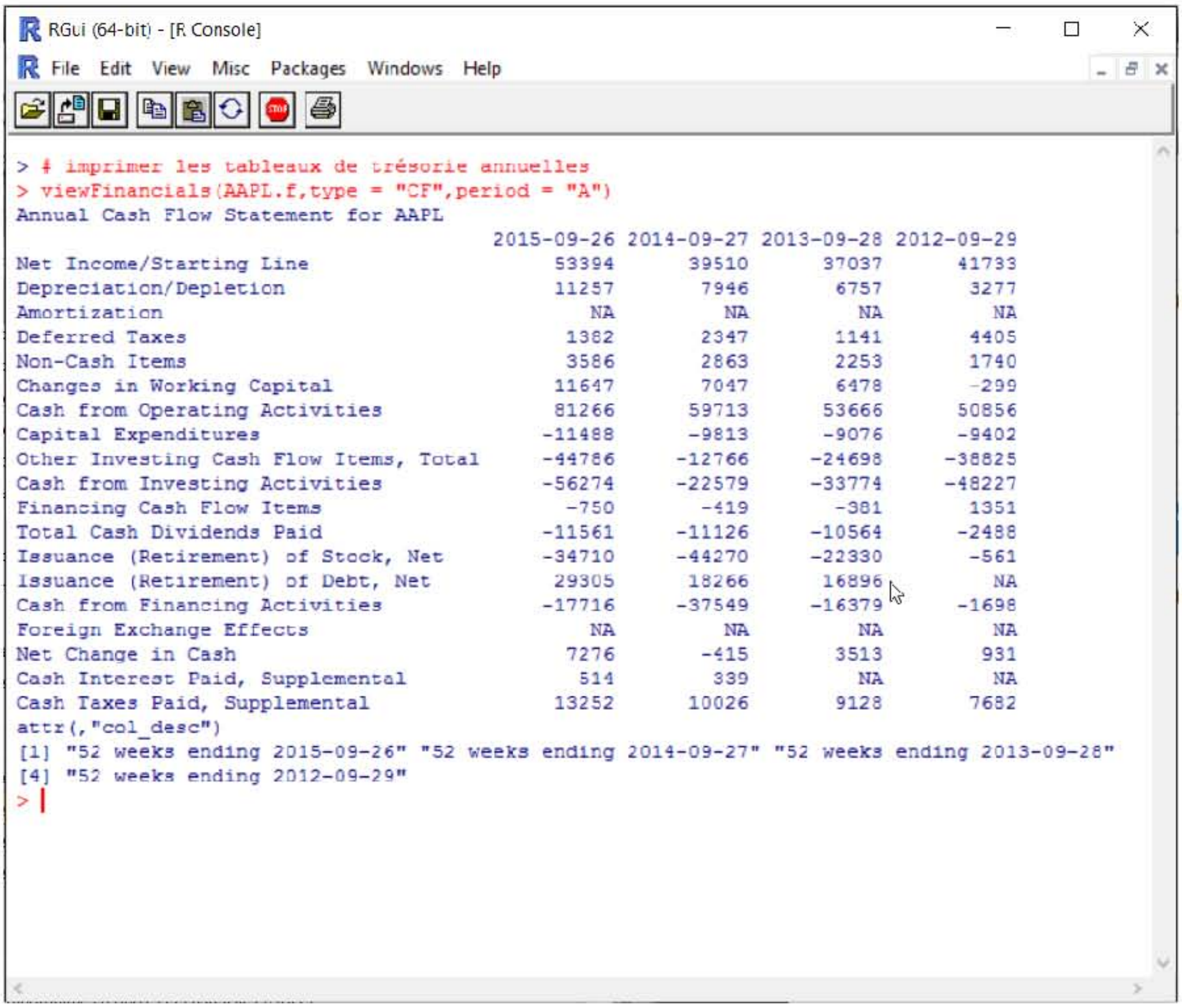
Et en trimestriel:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> # CPC trimestrielle
> viewFinancials(AAPL.f, type = "IS", period = "Q")
Quarterly Income Statement for AAPL
                                                2016-03-26 2015-12-26 2015-09-26
Revenue                                           50557.00   75872.00   51501.00
Other Revenue, Total                                    NA         NA         NA
Total Revenue                                     50557.00   75872.00   51501.00
Cost of Revenue, Total                            30636.00   45449.00   30953.00
Gross Profit                                      19921.00   30423.00   20548.00
Selling/General/Admin. Expenses, Total             3423.00    3848.00    3705.00
Research & Development                             2511.00    2404.00    2220.00
Depreciation/Amortization                               NA         NA         NA
Interest Expense(Income) - Net Operating                NA         NA         NA
Unusual Expense (Income)                                NA         NA         NA
Other Operating Expenses, Total                         NA         NA         NA
Total Operating Expense                           36570.00   51701.00   36878.00
Operating Income                                  13987.00   24171.00   14623.00
Interest Income(Expense), Net Non-Operating             NA         NA         NA
Gain (Loss) on Sale of Assets                           NA         NA         NA
Other, Net                                         -510.00    -263.00    -149.00
Income Before Tax                                 14142.00   24573.00   15062.00
Income After Tax                                  10516.00   18361.00   11124.00
Minority Interest                                       NA         NA         NA
Equity In Affiliates                                    NA         NA         NA
Net Income Before Extra. Items                    10516.00   18361.00   11124.00
Accounting Change                                       NA         NA         NA
Discontinued Operations                                 NA         NA         NA
Extraordinary Item                                      NA         NA         NA
Net Income                                        10516.00   18361.00   11124.00
Preferred Dividends                                     NA         NA         NA
Income Available to Common Excl. Extra Items      10516.00   18361.00   11124.00
Income Available to Common Incl. Extra Items      10516.00   18361.00   11124.00
Basic Weighted Average Shares                           NA         NA         NA
Basic EPS Excluding Extraordinary Items                 NA         NA         NA
Basic EPS Including Extraordinary Items                 NA         NA         NA
Dilution Adjustment                                     NA         NA         NA
Diluted Weighted Average Shares                    5540.89    5594.13    5682.52
Diluted EPS Excluding Extraordinary Items             1.90       3.28       1.96
Diluted EPS Including Extraordinary Items               NA         NA         NA
Dividends per Share - Common Stock Primary Issue      0.52       0.52       0.52
Gross Dividends - Common Stock                          NA         NA         NA
Net Income after Stock Based Comp. Expense              NA         NA         NA
Basic EPS after Stock Based Comp. Expense               NA         NA         NA
Diluted EPS after Stock Based Comp. Expense             NA         NA         NA
```

Et les tableaux de trésorerie annuelles:



```
> # imprimer les tableaux de trésorie annuelles
> viewFinancials(AAPL.f,type = "CF",period = "A")
Annual Cash Flow Statement for AAPL
                                     2015-09-26 2014-09-27 2013-09-28 2012-09-29
Net Income/Starting Line                  53394      39510      37037      41733
Depreciation/Depletion                    11257       7946       6757       3277
Amortization                                 NA         NA         NA         NA
Deferred Taxes                             1382       2347       1141       4405
Non-Cash Items                             3586       2863       2253       1740
Changes in Working Capital                11647       7047       6478       -299
Cash from Operating Activities            81266      59713      53666      50856
Capital Expenditures                     -11488      -9813      -9076      -9402
Other Investing Cash Flow Items, Total   -44786     -12766     -24698     -38825
Cash from Investing Activities           -56274     -22579     -33774     -48227
Financing Cash Flow Items                  -750       -419       -381       1351
Total Cash Dividends Paid                -11561     -11126     -10564      -2488
Issuance (Retirement) of Stock, Net      -34710     -44270     -22330       -561
Issuance (Retirement) of Debt, Net        29305      18266      16896         NA
Cash from Financing Activities           -17716     -37549     -16379      -1698
Foreign Exchange Effects                     NA         NA         NA         NA
Net Change in Cash                         7276       -415       3513        931
Cash Interest Paid, Supplemental            514        339         NA         NA
Cash Taxes Paid, Supplemental             13252      10026       9128       7682
attr(,"col_desc")
[1] "52 weeks ending 2015-09-26" "52 weeks ending 2014-09-27" "52 weeks ending 2013-09-28"
[4] "52 weeks ending 2012-09-29"
>
```

Et les bilans annuels:

```
> # imprimer les bilans annuels
> viewFinancials(AAPL.f, type = "BS", period = "A")
Annual Balance Sheet for AAPL
                                              2015-09-26 2014-09-27 2013-09-28 2012-09-29
Cash & Equivalents                               9731.00    3612.00    5554.00    7637.00
Short Term Investments                          20481.00   11233.00   26287.00   18383.00
Cash and Short Term Investments                 41601.00   25077.00   40546.00   29129.00
Accounts Receivable - Trade, Net                16849.00   17460.00   13102.00   10930.00
Receivables - Other                                   NA         NA         NA         NA
Total Receivables, Net                          30343.00   27219.00   20641.00   18692.00
Total Inventory                                  2349.00    2111.00    1764.00     791.00
Prepaid Expenses                                      NA         NA         NA         NA
Other Current Assets, Total                     15085.00   14124.00   10335.00    9041.00
Total Current Assets                            89378.00   68531.00   73286.00   57653.00
Property/Plant/Equipment, Total - Gross         49257.00   39015.00   28519.00   21887.00
Accumulated Depreciation, Total                -26786.00  -18391.00  -11922.00   -6435.00
Goodwill, Net                                    5116.00    4616.00    1577.00    1135.00
Intangibles, Net                                 3893.00    4142.00    4179.00    4224.00
Long Term Investments                          164065.00  130162.00  106215.00   92122.00
Other Long Term Assets, Total                    5556.00    3764.00    5146.00    5478.00
Total Assets                                   290479.00  231839.00  207000.00  176064.00
Accounts Payable                                35490.00   30196.00   22367.00   21175.00
Accrued Expenses                                24169.00    7689.00    4782.00    3283.00
Notes Payable/Short Term Debt                    8499.00    6308.00       0.00       0.00
Current Port. of LT Debt/Capital Leases          2500.00         NA         NA         NA
Other Current liabilities, Total                 9952.00   19255.00   16509.00   14084.00
Total Current Liabilities                       80610.00   63448.00   43658.00   38542.00
Long Term Debt                                  53463.00   28987.00   16960.00         NA
Capital Lease Obligations                             NA         NA         NA         NA
Total Long Term Debt                            53463.00   28987.00   16960.00       0.00
Total Debt                                      64462.00   35295.00   16960.00       0.00
Deferred Income Tax                             24062.00   20259.00   16489.00   13847.00
Minority Interest                                     NA         NA         NA         NA
Other Liabilities, Total                        12989.00    7598.00    6344.00    5465.00
Total Liabilities                              171124.00  120292.00   83451.00   57854.00
Redeemable Preferred Stock, Total                     NA         NA         NA         NA
Preferred Stock - Non Redeemable, Net                 NA         NA         NA         NA
Common Stock, Total                             27416.00   23313.00   19764.00   16422.00
Additional Paid-In Capital                            NA         NA         NA         NA
Retained Earnings (Accumulated Deficit)         92284.00   87152.00  104256.00  101289.00
Treasury Stock - Common                               NA         NA         NA         NA
Other Equity, Total                              -464.00     -40.00    -191.00     731.00
Total Equity                                   119355.00  111547.00  123549.00  118210.00
Total Liabilities & Shareholders&#39; Equity  290479.00  231839.00  207000.00  176064.00
```

Maintenant voyons avec le package finreportr et d'abord ses deux fonctions et **CompanyInfo( )** et **GetBalanceSheet( )**:

```
> library("finreportr")
> CompanyInfo("AAPL")
    company        CIK  SIC state state.inc FY.end      street.address        city.state
1 APPLE INC 0000320193 3571    CA        CA   0930 ONE APPLE PARK WAY CUPERTINO CA 95014
> GetBalanceSheet("AAPL",2016)
                                        Metric Units       Amount startDate    endDate
1            Cash and Cash Equivalents, at Carrying Value   usd  14259000000      <NA> 2013-09-28
2            Cash and Cash Equivalents, at Carrying Value   usd  13844000000      <NA> 2014-09-27
3            Cash and Cash Equivalents, at Carrying Value   usd  21120000000      <NA> 2015-09-26
4            Cash and Cash Equivalents, at Carrying Value   usd  20484000000      <NA> 2016-09-24
5                  Available-for-sale Securities, Current   usd  20481000000      <NA> 2015-09-26
6                  Available-for-sale Securities, Current   usd  46671000000      <NA> 2016-09-24
7                       Accounts Receivable, Net, Current   usd  16849000000      <NA> 2015-09-26
8                       Accounts Receivable, Net, Current   usd  15754000000      <NA> 2016-09-24
9                                          Inventory, Net   usd   2349000000      <NA> 2015-09-26
10                                         Inventory, Net   usd   2132000000      <NA> 2016-09-24
11                        Nontrade Receivables, Current   usd  13494000000      <NA> 2015-09-26
12                        Nontrade Receivables, Current   usd  13545000000      <NA> 2016-09-24
13                                Other Assets, Current   usd  15085000000      <NA> 2015-09-26
14                                Other Assets, Current   usd   8283000000      <NA> 2016-09-24
15                                      Assets, Current   usd  89378000000      <NA> 2015-09-26
16                                      Assets, Current   usd 106869000000      <NA> 2016-09-24
17            Available-for-sale Securities, Noncurrent   usd 164065000000      <NA> 2015-09-26
18            Available-for-sale Securities, Noncurrent   usd 170430000000      <NA> 2016-09-24
19                   Property, Plant and Equipment, Net   usd  22471000000      <NA> 2015-09-26
20                   Property, Plant and Equipment, Net   usd  27010000000      <NA> 2016-09-24
21                                             Goodwill   usd   5116000000      <NA> 2015-09-26
22                                             Goodwill   usd   5414000000      <NA> 2016-09-24
23             Intangible Assets, Net (Excluding Goodwill)   usd   3893000000      <NA> 2015-09-26
24             Intangible Assets, Net (Excluding Goodwill)   usd   3206000000      <NA> 2016-09-24
25                             Other Assets, Noncurrent   usd   5422000000      <NA> 2015-09-26
26                             Other Assets, Noncurrent   usd   8757000000      <NA> 2016-09-24
27                                               Assets   usd 290345000000      <NA> 2015-09-26
28                                               Assets   usd 321686000000      <NA> 2016-09-24
29                           Accounts Payable, Current   usd  35490000000      <NA> 2015-09-26
30                           Accounts Payable, Current   usd  37294000000      <NA> 2016-09-24
31                        Accrued Liabilities, Current   usd  25181000000      <NA> 2015-09-26
32                        Accrued Liabilities, Current   usd  22027000000      <NA> 2016-09-24
33                           Deferred Revenue, Current   usd   8940000000      <NA> 2015-09-26
34                           Deferred Revenue, Current   usd   8080000000      <NA> 2016-09-24
35                                    Commercial Paper   usd   8499000000      <NA> 2015-09-26
36                                    Commercial Paper   usd   8500000000      <NA> 2015-09-26
37                                    Commercial Paper   usd   8100000000      <NA> 2016-09-24
38                                    Commercial Paper   usd   8105000000      <NA> 2016-09-24
39                  Long-term Debt, Current Maturities   usd   2500000000      <NA> 2015-09-26
40                  Long-term Debt, Current Maturities   usd   3500000000      <NA> 2016-09-24
41                             Liabilities, Current   usd  80610000000      <NA> 2015-09-26
42                             Liabilities, Current   usd  79006000000      <NA> 2016-09-24
43                        Deferred Revenue, Noncurrent   usd   3624000000      <NA> 2015-09-26
```

Et la fonction **GetCashFlow( )**:


```
> GetCashFlow("AAPL",2016)
                                                              Metric Units      Amount  startDate    endDate
1                       Cash and Cash Equivalents, at Carrying Value   usd 14259000000       <NA> 2013-09-28
2                       Cash and Cash Equivalents, at Carrying Value   usd 13844000000       <NA> 2014-09-27
3                       Cash and Cash Equivalents, at Carrying Value   usd 21120000000       <NA> 2015-09-26
4                       Cash and Cash Equivalents, at Carrying Value   usd 20484000000       <NA> 2016-09-24
5                          Net Income (Loss) Attributable to Parent   usd 39510000000 2013-09-29 2014-09-27
6                          Net Income (Loss) Attributable to Parent   usd 18024000000 2014-09-28 2014-12-27
7                          Net Income (Loss) Attributable to Parent   usd 13569000000 2014-12-28 2015-03-28
8                          Net Income (Loss) Attributable to Parent   usd 10677000000 2015-03-29 2015-06-27
9                          Net Income (Loss) Attributable to Parent   usd 11124000000 2015-06-28 2015-09-26
10                         Net Income (Loss) Attributable to Parent   usd 53394000000 2014-09-28 2015-09-26
11                         Net Income (Loss) Attributable to Parent   usd 18361000000 2015-09-27 2015-12-26
12                         Net Income (Loss) Attributable to Parent   usd 10516000000 2015-12-27 2016-03-26
13                         Net Income (Loss) Attributable to Parent   usd  7796000000 2016-03-27 2016-06-25
14                         Net Income (Loss) Attributable to Parent   usd  9014000000 2016-06-26 2016-09-24
15                         Net Income (Loss) Attributable to Parent   usd 45687000000 2015-09-27 2016-09-24
16                      Depreciation, Amortization and Accretion, Net   usd  7946000000 2013-09-29 2014-09-27
17                      Depreciation, Amortization and Accretion, Net   usd 11257000000 2014-09-28 2015-09-26
18                      Depreciation, Amortization and Accretion, Net   usd 10505000000 2015-09-27 2016-09-24
19                                      Share-based Compensation   usd  2863000000 2013-09-29 2014-09-27
20                                      Share-based Compensation   usd  3586000000 2014-09-28 2015-09-26
21                                      Share-based Compensation   usd  4210000000 2015-09-27 2016-09-24
22                          Deferred Income Tax Expense (Benefit)   usd  2347000000 2013-09-29 2014-09-27
23                          Deferred Income Tax Expense (Benefit)   usd  1382000000 2014-09-28 2015-09-26
24                          Deferred Income Tax Expense (Benefit)   usd  4938000000 2015-09-27 2016-09-24
25                      Increase (Decrease) in Accounts Receivable   usd  4232000000 2013-09-29 2014-09-27
26                      Increase (Decrease) in Accounts Receivable   usd  -611000000 2014-09-28 2015-09-26
27                      Increase (Decrease) in Accounts Receivable   usd -1095000000 2015-09-27 2016-09-24
28                            Increase (Decrease) in Inventories   usd    76000000 2013-09-29 2014-09-27
29                            Increase (Decrease) in Inventories   usd   238000000 2014-09-28 2015-09-26
30                            Increase (Decrease) in Inventories   usd  -217000000 2015-09-27 2016-09-24
31                       Increase (Decrease) in Other Receivables   usd  2220000000 2013-09-29 2014-09-27
32                       Increase (Decrease) in Other Receivables   usd  3735000000 2014-09-28 2015-09-26
33                       Increase (Decrease) in Other Receivables   usd    51000000 2015-09-27 2016-09-24
34                   Increase (Decrease) in Other Operating Assets   usd  -167000000 2013-09-29 2014-09-27
35                   Increase (Decrease) in Other Operating Assets   usd   179000000 2014-09-28 2015-09-26
36                   Increase (Decrease) in Other Operating Assets   usd -1090000000 2015-09-27 2016-09-24
37                        Increase (Decrease) in Accounts Payable   usd  5938000000 2013-09-29 2014-09-27
38                        Increase (Decrease) in Accounts Payable   usd  5400000000 2014-09-28 2015-09-26
39                        Increase (Decrease) in Accounts Payable   usd  1791000000 2015-09-27 2016-09-24
40                        Increase (Decrease) in Deferred Revenue   usd  1460000000 2013-09-29 2014-09-27
41                        Increase (Decrease) in Deferred Revenue   usd  1042000000 2014-09-28 2015-09-26
42                        Increase (Decrease) in Deferred Revenue   usd -1554000000 2015-09-27 2016-09-24
43              Increase (Decrease) in Other Operating Liabilities   usd  6010000000 2013-09-29 2014-09-27
44              Increase (Decrease) in Other Operating Liabilities   usd  8746000000 2014-09-28 2015-09-26
45              Increase (Decrease) in Other Operating Liabilities   usd -2104000000 2015-09-27 2016-09-24
46  Net Cash Provided by (Used in) Operating Activities, Continuing Operations   usd 59713000000 2013-09-29 2014-09-27
47  Net Cash Provided by (Used in) Operating Activities, Continuing Operations   usd 81266000000 2014-09-28 2015-09-26
```

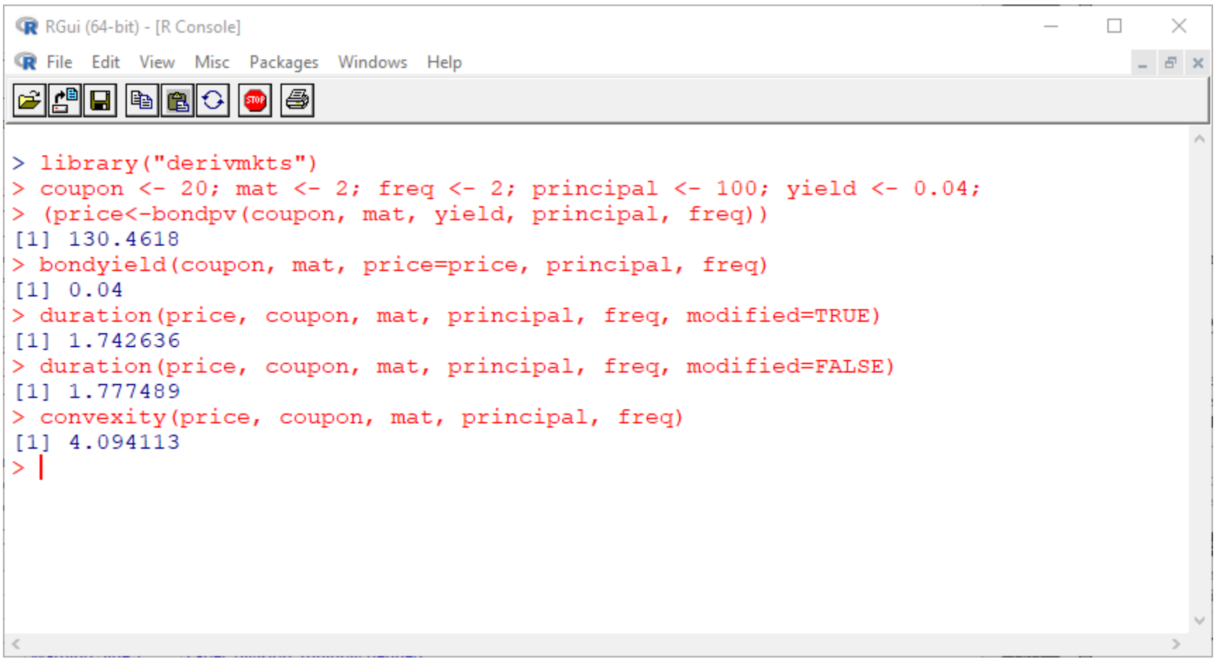
Exemple 394.: Valeur présente d'une obligation et rendement

R 3.0.2

Nous allons reprendre ici l'exemple du cours que nous avons fait après la démonstration mathématique détaillée. Rappelons juste que l'exemple numérique est celui repris de la page Wikipédia:

https://en.wikipedia.org/wiki/Bond_duration

D'abord nous allons calculer la valeur présente (present value) d' une obligation de 2 ans d'une valeur nominale de 100 dollars, un coupon semestriel de 20% et un rendement de 4% composé semestriellement. Le PV total sera à l'aide de la fonction **bondpv( )** du package **derivmkts**:



Ce qui est parfaitement conforme aux calculs "à la main" (excepté pour la convexité que nous avons la flemme de calculer)!

Exemple 395.: Liquidités et illiquidités

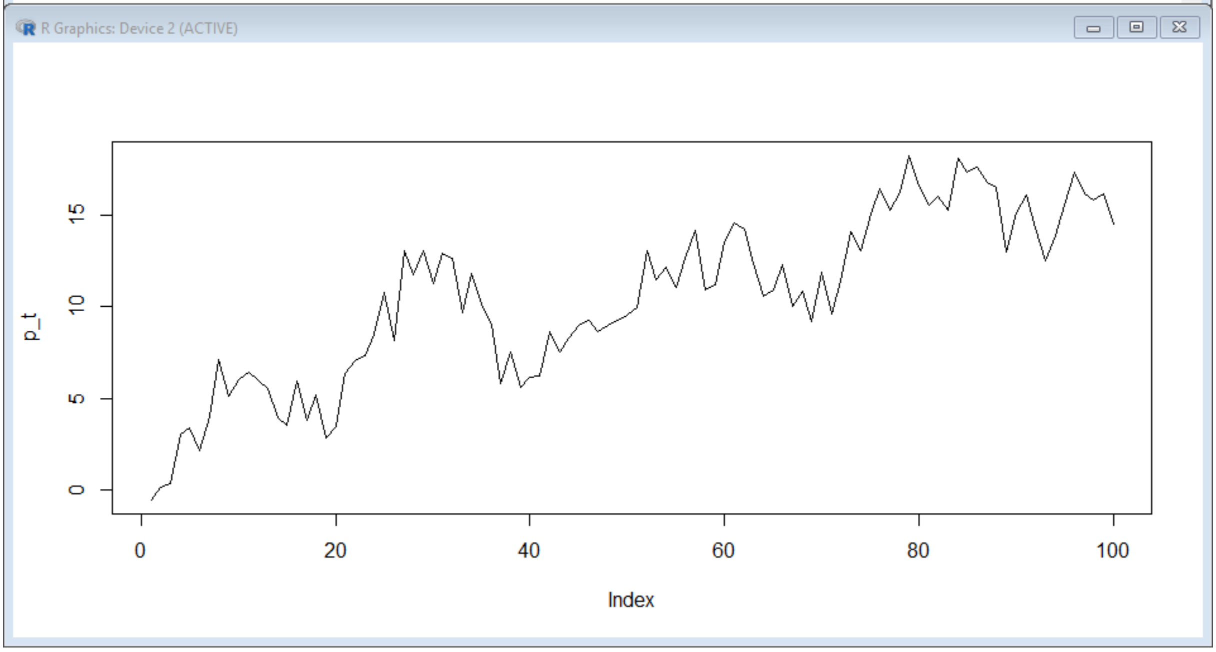
R 3.5.3

Illiquidité de Rolle (1984)

Nous avons démontré dans le cours théorique l'origine de relation du Bid et Ask spread de Roll (selon son article de 1984 qui est très détaillé ce qui est plutôt rare…) qui permet donc d'estimer le prix statistique de transaction d'un instrument (puisque c'est justement la différence statistique entre le bid et le ask). Malheureusement, à notre connaissance, aucun package à ce jour n'intègre le calcul correspondant. Voyons comment alors écrire le calcul depuis zéro car ce n'est pas bien long et c'est aussi un excellent exemple d'application de la fonction **lag( )**:

R Console

```
> library("zoo")
> c = 1
> time = 1:100
> epsilon = rnorm(time)
> prices = cumsum(epsilon)
> m_t = zoo(prices)
> a_t = m_t + c
> b_t = m_t - c
>
> q_t = sign(rnorm(time))
> p_t = m_t + (c * q_t)
> plot(p_t)
>
> returns <- na.omit((m_t/lag(m_t))-1)
> cost <- sqrt(2 * cov(returns [1:length(returns )-1], lag(returns , na.pad=FALSE)))
> cost
[1] 0.08847571
>
```



Ratio de Biais (bias ratio)

Le but ici va être simplement de reproduire cet indicateur empirique tel que calculé "à la main dans le cours théorique" avec Microsoft Excel.

Pour cela nous partons du même jeu de données (Fairfield Sentry pour rappel est connu pour avoir une grande part de sens rendements grâce aux fonds de Maddoff…) mis dans un fichier CSV dont nous avons éléminé les valeurs manquantes:

Table 1: **Fairfield Sentry monthly returns from Dec. 1990 to Oct. 2008.**

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2008	0.63	0.06	0.18	0.93	0.81	-0.06	0.72	0.71	0.50	-0.06	na	na
2007	0.29	-0.11	1.64	0.98	0.81	0.34	0.17	0.31	0.97	0.46	1.04	0.23
2006	0.70	0.20	1.31	0.94	0.70	0.51	1.06	0.77	0.68	0.42	0.86	0.86
2005	0.51	0.37	0.85	0.14	0.63	0.46	0.13	0.16	0.89	1.61	0.75	0.54
2004	0.88	0.44	-0.01	0.37	0.59	1.21	0.02	1.26	0.46	0.03	0.79	0.24
2003	-0.35	-0.05	1.85	0.03	0.9	0.93	1.37	0.16	0.86	1.26	-0.14	0.25
2002	-0.04	0.53	0.39	1.09	2.05	0.19	3.29	-0.13	0.06	0.66	0.09	0.00
2001	2.14	0.08	1.07	1.26	0.26	0.17	0.38	0.94	0.66	1.22	1.14	0.12
2000	2.14	0.13	1.77	0.27	1.30	0.73	0.58	1.26	0.18	0.86	0.62	0.36
1999	1.99	0.11	2.22	0.29	1.45	1.70	0.36	0.87	0.66	1.05	1.54	0.32
1998	0.85	1.23	1.68	0.36	1.69	1.22	0.76	0.21	0.98	1.86	0.78	0.26
1997	2.38	0.67	0.80	1.10	0.57	1.28	0.68	0.28	2.32	0.49	1.49	0.36
1996	1.42	0.66	1.16	0.57	1.34	0.15	1.86	0.20	1.16	1.03	1.51	0.41
1995	0.85	0.69	0.78	1.62	1.65	0.43	1.02	-0.24	1.63	1.53	0.44	1.03
1994	2.11	-0.44	1.45	1.75	0.44	0.23	1.71	0.35	0.75	1.81	-0.64	0.60
1993	-0.09	1.86	1.79	-0.01	1.65	0.79	0.02	1.71	0.28	1.71	0.19	0.39
1992	0.42	2.72	0.94	2.79	-0.27	1.22	-0.09	0.85	0.33	1.33	1.35	1.36
1991	3.01	1.40	0.52	1.32	1.82	0.30	1.98	1.00	0.73	2.75	0.01	1.56
1990												2.77

Nous utilisons pour cela le package **YRmisc** et sa fonction pt.bias( ):

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> library("YRmisc")
> library("moments")
>
> yields<-read.csv("c:/tmp/Bias_ratio_Yields.csv",sep=",",header=T)
> summary(yields)
      Year        Attribute              Value
 Min.   :1990   Length:215         Min.   :-0.6400
 1st Qu.:1995   Class :character   1st Qu.: 0.2950
 Median :1999   Mode  :character   Median : 0.7300
 Mean   :1999                      Mean   : 0.8421
 3rd Qu.:2004                      3rd Qu.: 1.2700
 Max.   :2008                      Max.   : 3.2900
> head(yields)
  Year Attribute Value
1 2008       Jan  0.63
2 2008       Feb  0.06
3 2008       Mar  0.18
4 2008       Apr  0.93
5 2008       May  0.81
6 2008       Jun -0.06
> tail(yields)
    Year Attribute Value
210 1991       Aug  1.00
211 1991       Sep  0.73
212 1991       Oct  2.75
213 1991       Nov  0.01
214 1991       Dec  1.56
215 1990       Dec  2.77
> r<-yields$Value
> pt.bias(r)
[1] 0.6555556
> #jour pour comparer avec le skewness
> moments::skewness(r)
[1] 0.7811937
>
```

Ce qui **ne correspond pas du tout au calcul fait avec Microsoft Excel qui est de 5.18** !!! Notons par ailleurs que le Skewness (sans surprises!) nous amène à la même conclusion!

Donc pourquoi cette différence importante entre le 5.18 obtenu avec Microsoft Excel et ce package R? La raison est simple! Malheureusement le code de ce package utilise:

$$\text{Bias Ratio} = \text{BR} = \frac{\#\left(r_i \mid r_i \in [\hat{\mu}, \hat{\mu} + \hat{\sigma}]\right)}{1 + \#\left(r_i \mid r_i \in [\hat{\mu} - \hat{\sigma}, \hat{\mu})\right)}$$

Au lieu de:

$$\text{Bias Ratio} = \text{BR} = \frac{\#\left(r_i \mid r_i \in [0, +\hat{\sigma}]\right)}{1 + \#\left(r_i \mid r_i \in [-\hat{\sigma}, 0)\right)}$$

Exemple 396.: Mouvement Brownien vs Mouvement de Lévy

R 3.0.2

Dans le cours théorique de finance quantitative nous avons assez longuement étudié le mouvement brownien. Nous avons cependant aussi parlé des processus de Lévy pour les distributions à queues épaisses. Voyons-en un exemple en commençant d'abord avec le processus gaussien bivarié à la main:

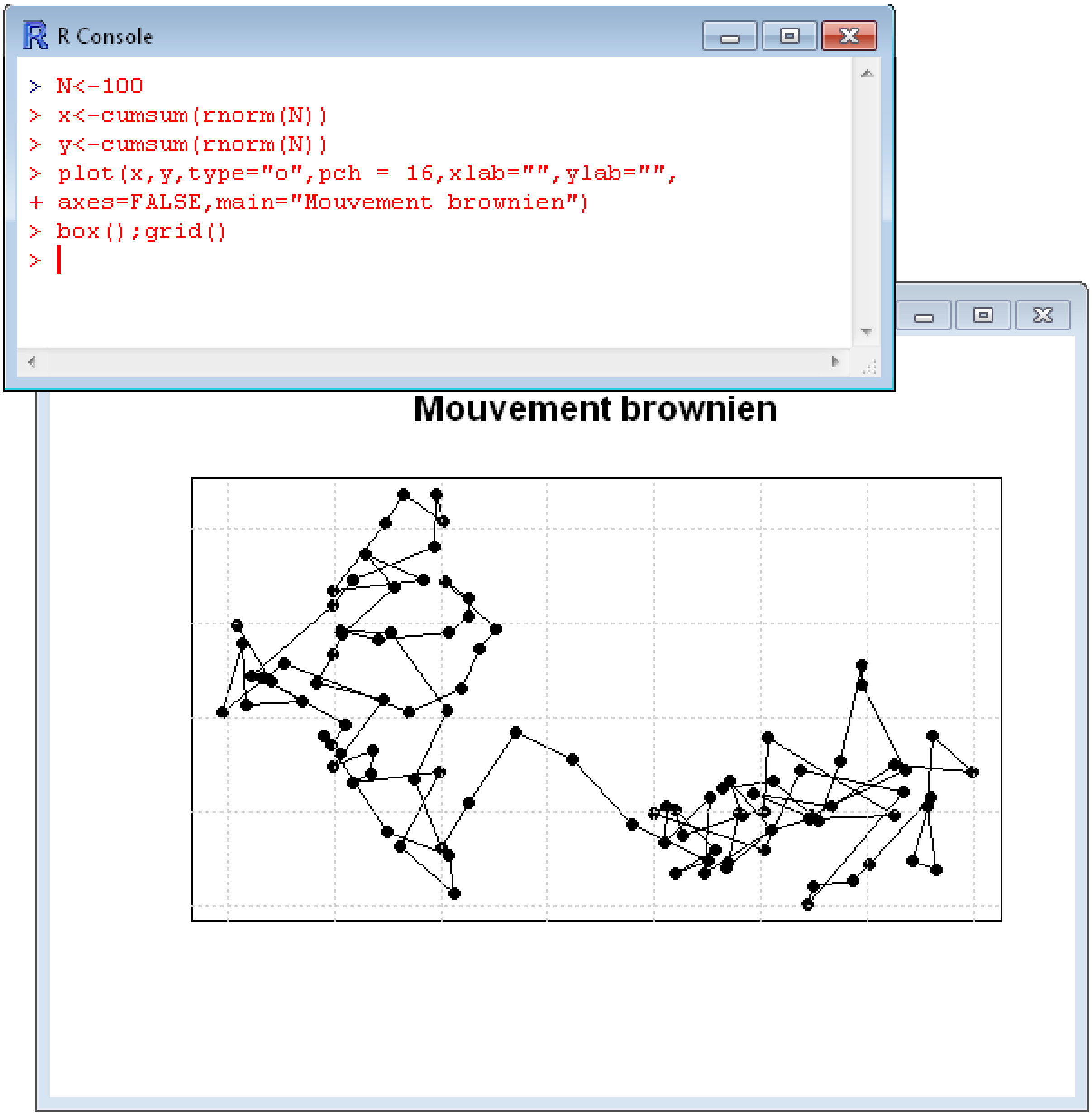
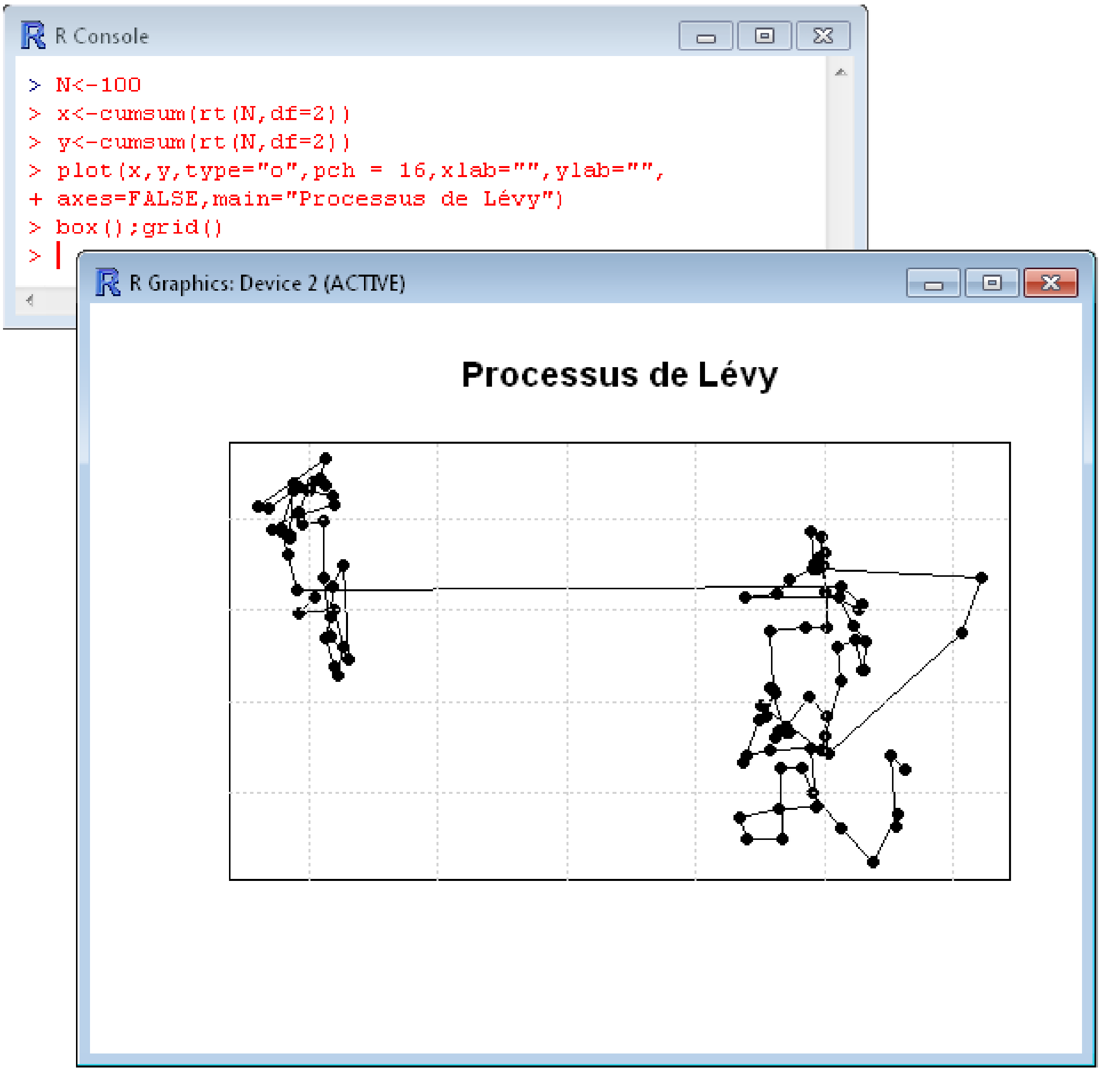


Figure 316 Graph Mouvement Brownien

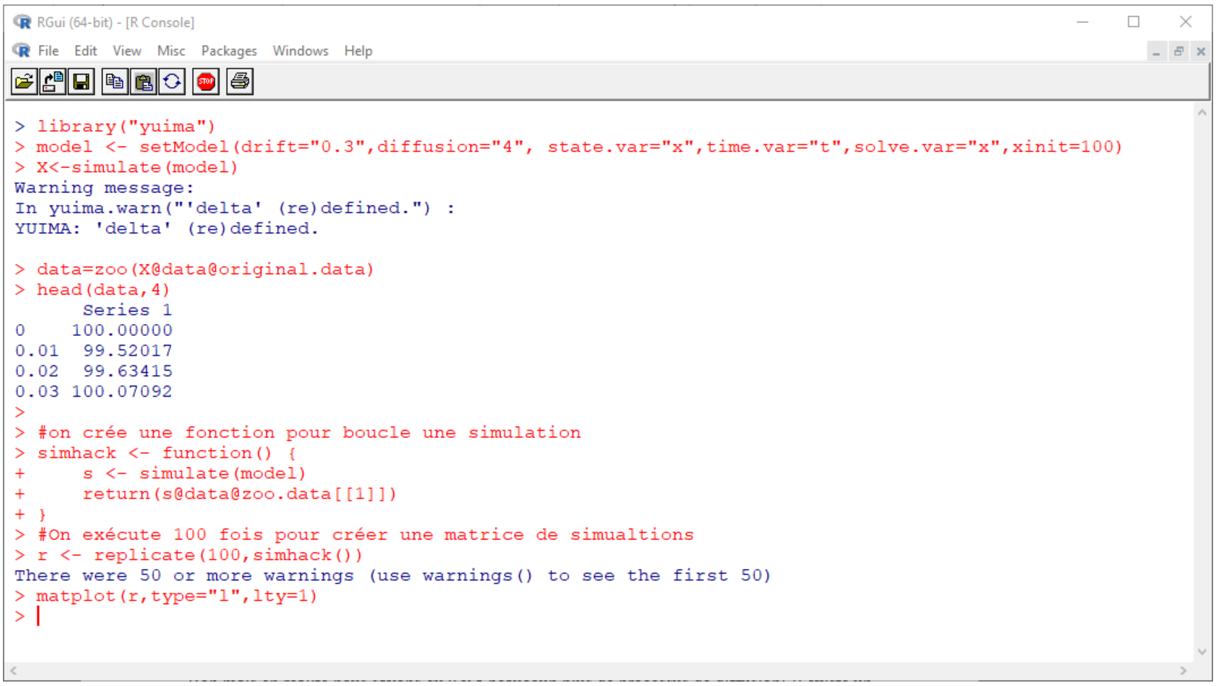
et ensuite avec le cas classique du processus de Lévy (aussi bidimensionnel) basé sur une distribution de Student (remarquez le fait que parfois vous obtiendrez un grand saut):



Bon mais en réalité nous savons qu'il y a beaucoup plus de processus de diffusion! Il existe un package très complet, nommé **yuima**, pour lequel il existe un livre dédié de relativement bonne qualité et qui aussi sa propre app Shiny (très utile pour comprendre certains points du fonctionnement du package):

https://yuimaproject.com/

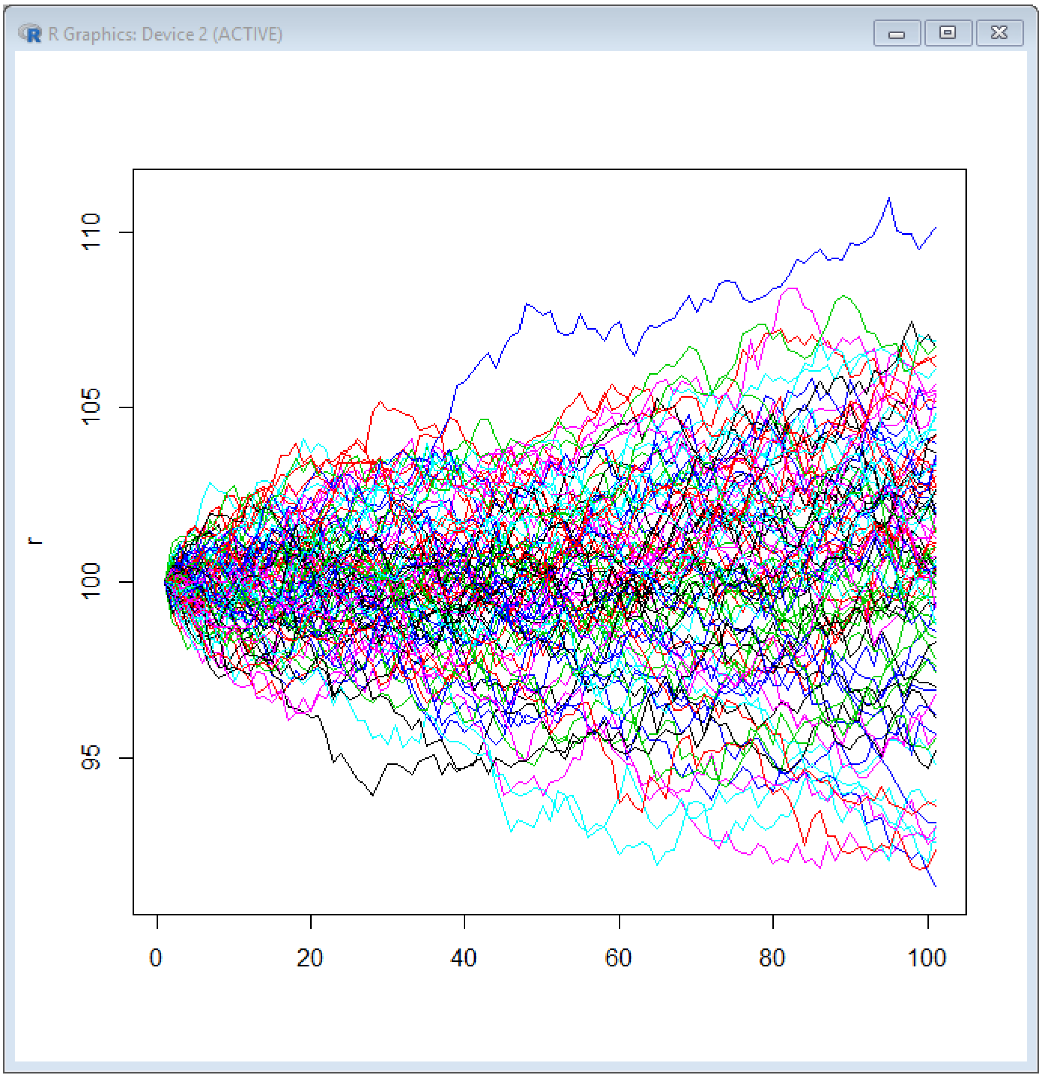
Voyons d'abord comment simuler une mouvement brownien avec ce package:



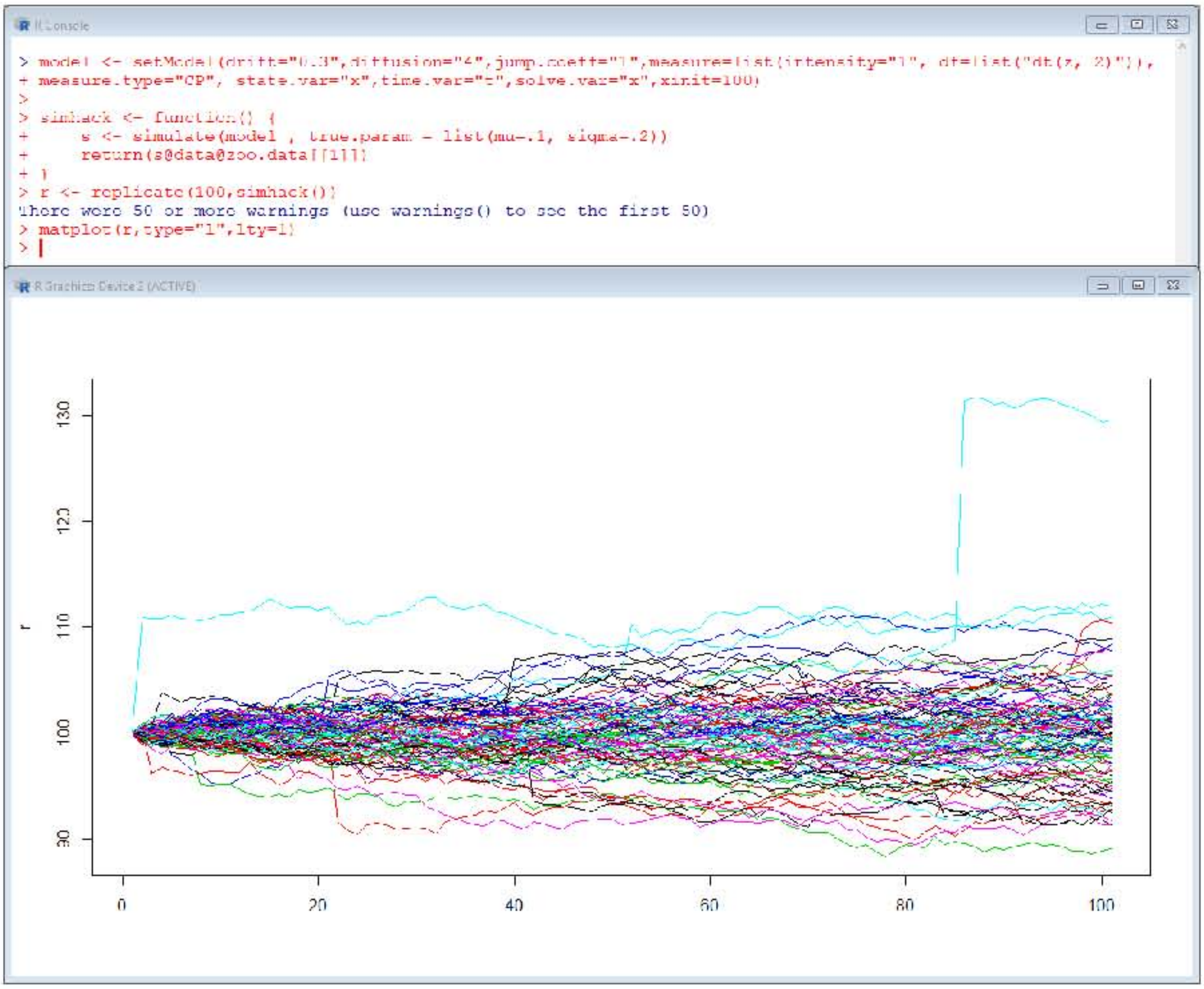
```
> library("yuima")
> model <- setModel(drift="0.3",diffusion="4", state.var="x",time.var="t",solve.var="x",xinit=100)
> X<-simulate(model)
Warning message:
In yuima.warn("'delta' (re)defined.") :
YUIMA: 'delta' (re)defined.

> data=zoo(X@data@original.data)
> head(data,4)
      Series 1
0    100.00000
0.01  99.52017
0.02  99.63415
0.03 100.07092
>
> #on crée une fonction pour boucle une simulation
> simhack <- function() {
+     s <- simulate(model)
+     return(s@data@zoo.data[[1]])
+ }
> #On exécute 100 fois pour créer une matrice de simualtions
> r <- replicate(100,simhack())
There were 50 or more warnings (use warnings() to see the first 50)
> matplot(r,type="l",lty=1)
> |
```

Ce qui donnera:



Et voyons maintenant comment simuler un processus de Lévy à sauts de Poisson de type Student à deux degrés de liberté:

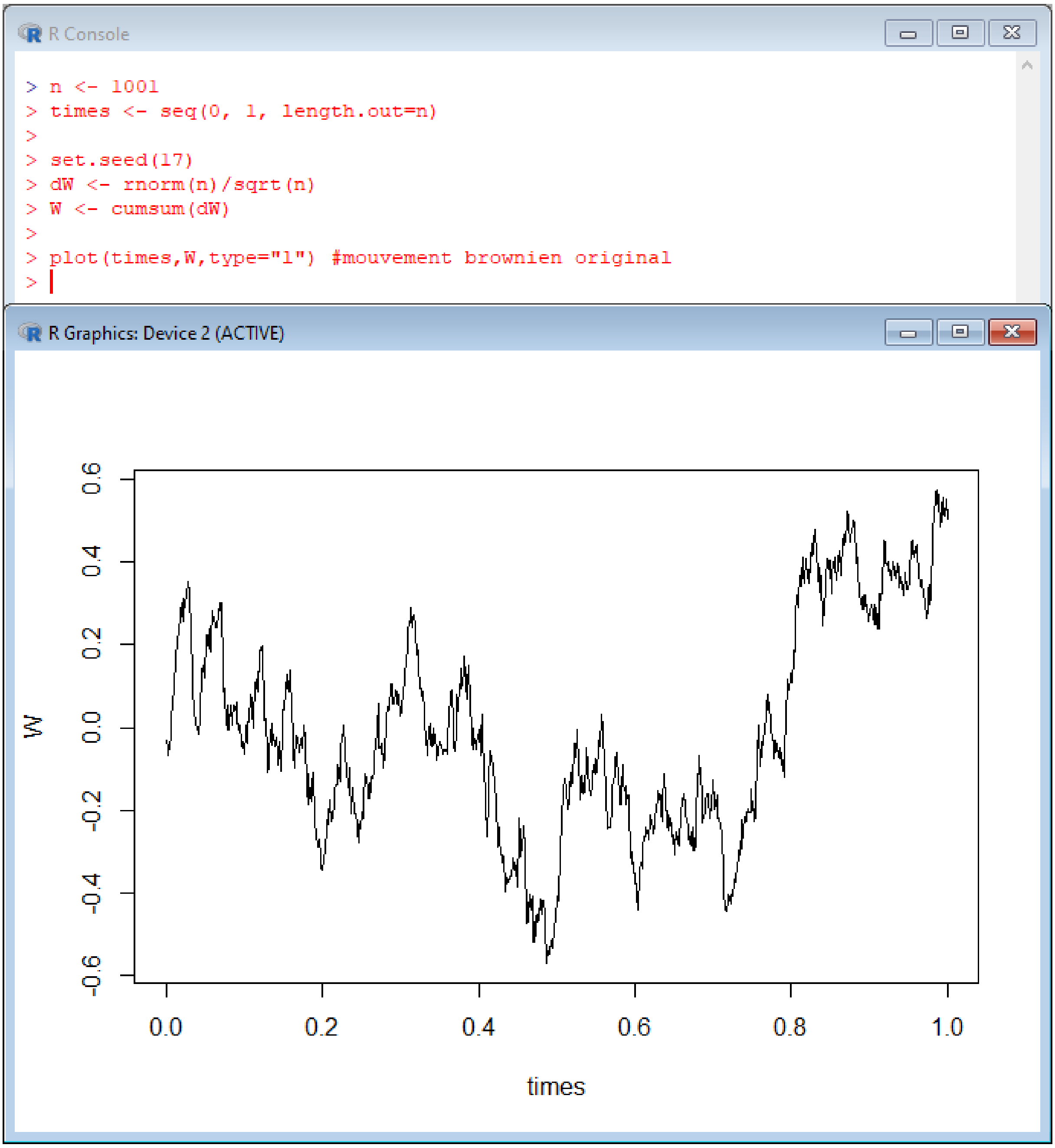


Exemple 397.: Pont Brownien et Excursion Brownienne

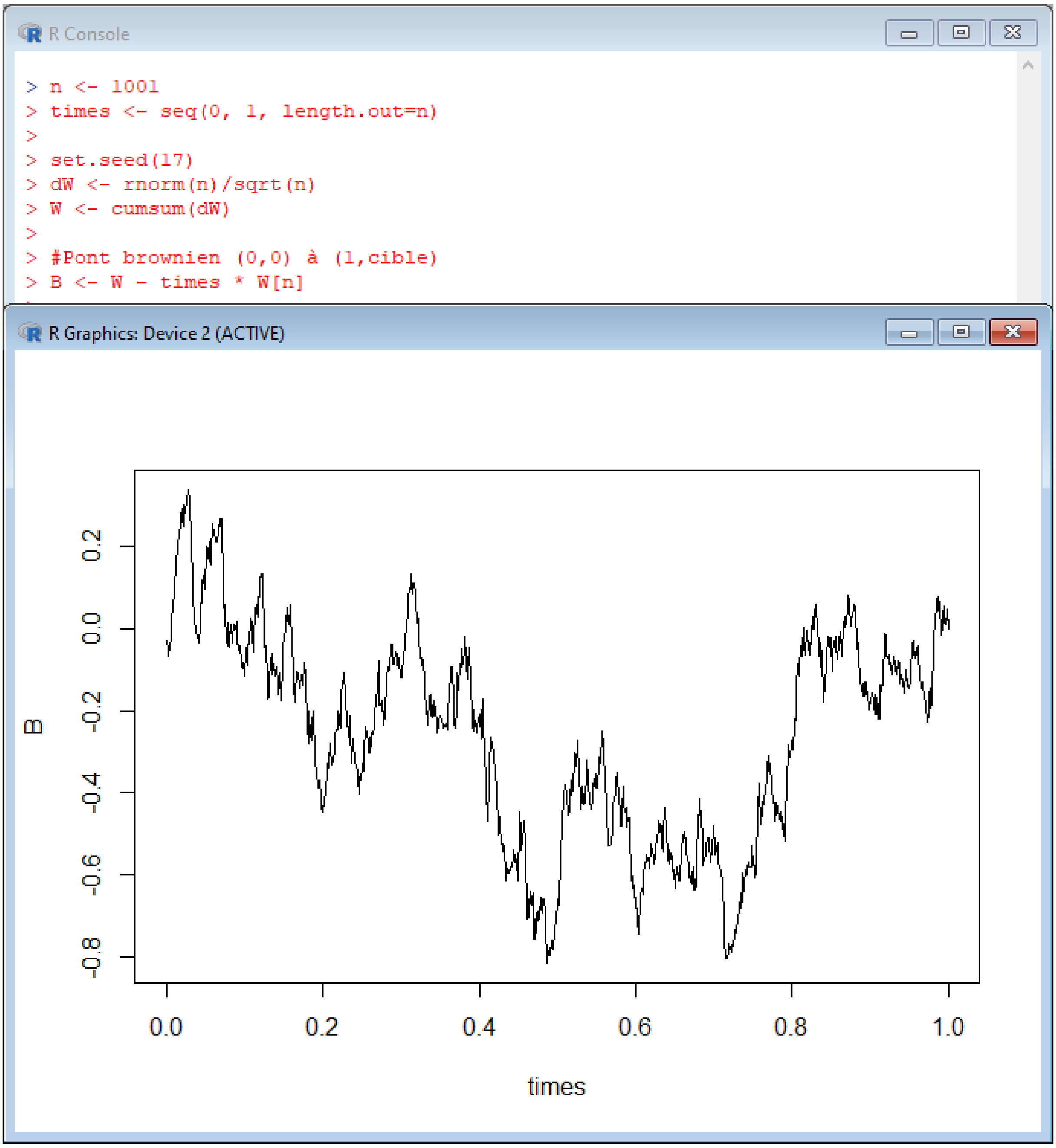
R 4.1.2

Voyons comment obtenir ces deux concepts vus dans le cours théorique.

Pour cela, rappelons d'abord le mouvement brownian standard:



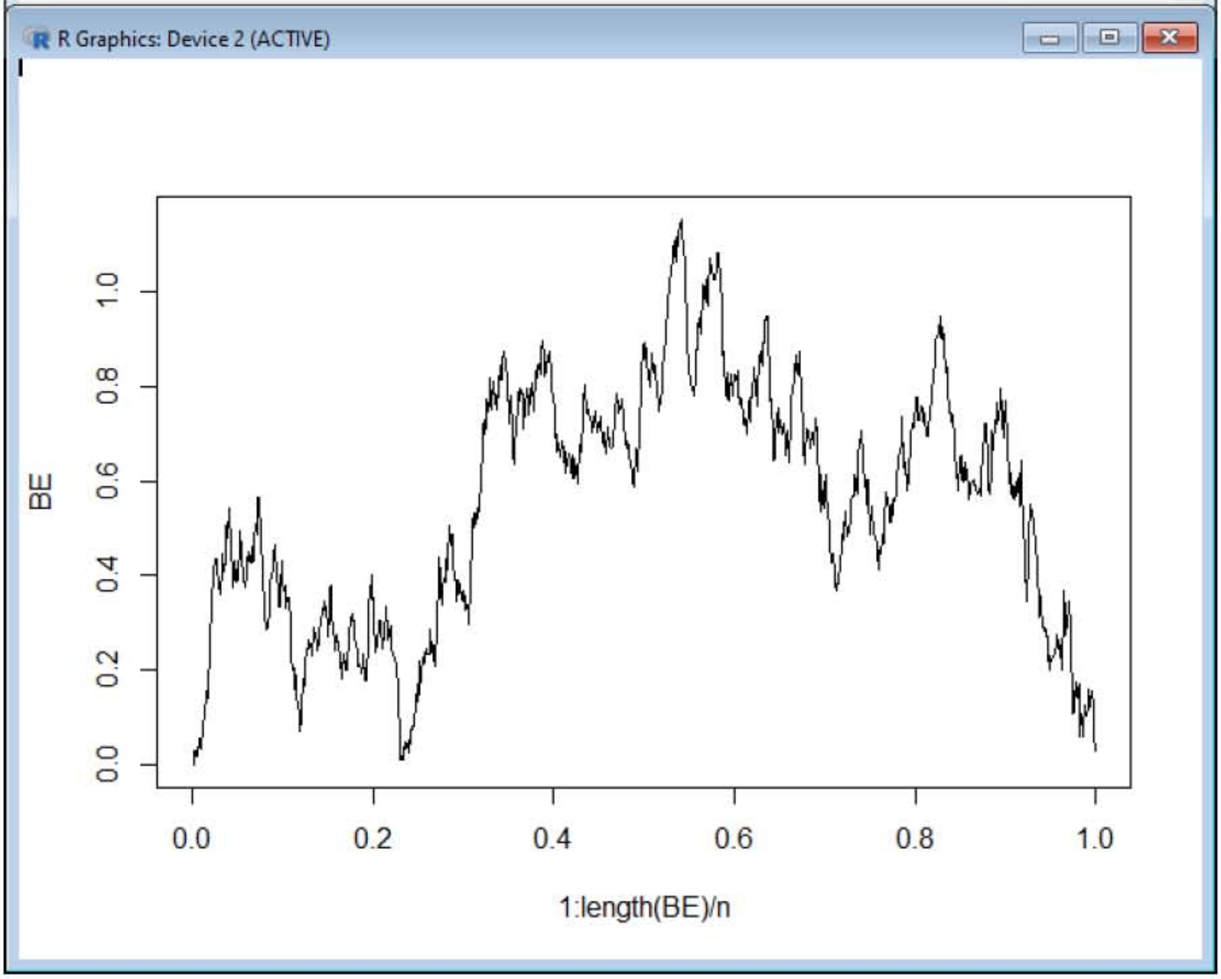
Et voici comment obtenir le pont brownien:



Et enfin l'excursion Brownienne[41]:

[41] Selon la méthode décrite ici: https://projecteuclid.org/download/pdf_1/euclid.aop/1176995155

```
> n <- 1001
> times <- seq(0, 1, length.out=n)
>
> set.seed(17)
> dW <- rnorm(n)/sqrt(n)
> W <- cumsum(dW)
>
> #Pont brownien (0,0) à (1,cible)
> B <- W - times * W[n]
>
> # construction de Vervaat de l'excursion brownienne
> Bmin <- min(B)
> tmin <- which(B == Bmin)
> newtimes <- (times[tmin] + times) %% 1
> J<-floor(newtimes * n)
> BE <- B[J] - Bmin
> plot(1:length(BE)/n,BE,type="l")
>
```



Exemple 398.: Mouvement Brownien Géométrique et test d'adéquation

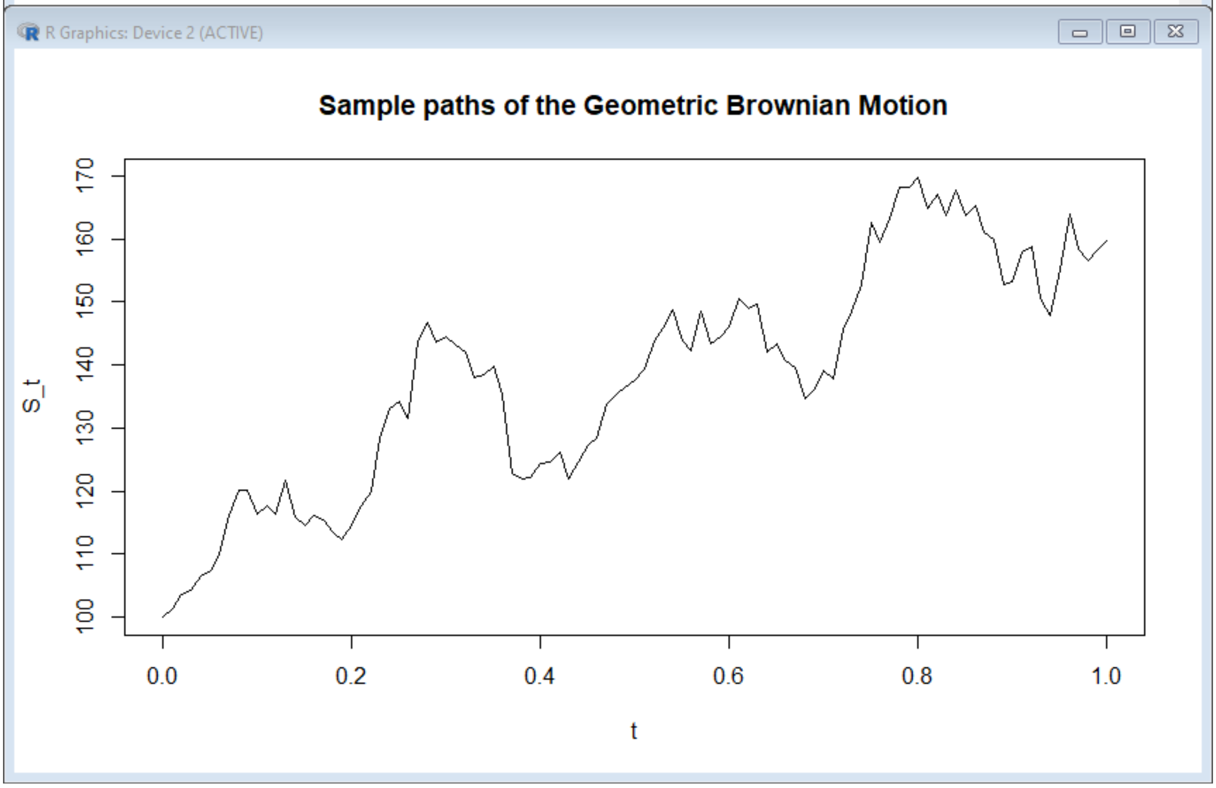
R 3.6.2

Nous savons quelle est l'import de mouvement brownien géométrique dans le modèle de Black and Scholes, Black-Merton et aussi KMV.

Quand on traite avec de tels processus de diffusion, un minimum est de vérifier si on a bien affaire à ces derniers (et non pas à autre chose...). Donc voyons comment rapidement en simuler et tester leur adéquation (le rendement logarithmique devant suivre une loi normale).

D'abord voyons comment simuler un mouvement brownien standard ou géométrique avec le package **fExpressCertificates** et sa fonction **GBM( )** (ou respectivement sa fonction **BM( )**):

```
> library("fExpressCertificates")
> T<-1
> mc.steps<-100
> dt<-T/mc.steps
> t<-seq(0, T, by=dt)
> #il y aussi la fonction BM...
> S_t<-GBM(S0=100, mu=0.05, sigma=0.3, T=T, N=mc.steps)
> plot(t,S_t,type="l",main="Sample paths of the Geometric Brownian Motion")
>
```



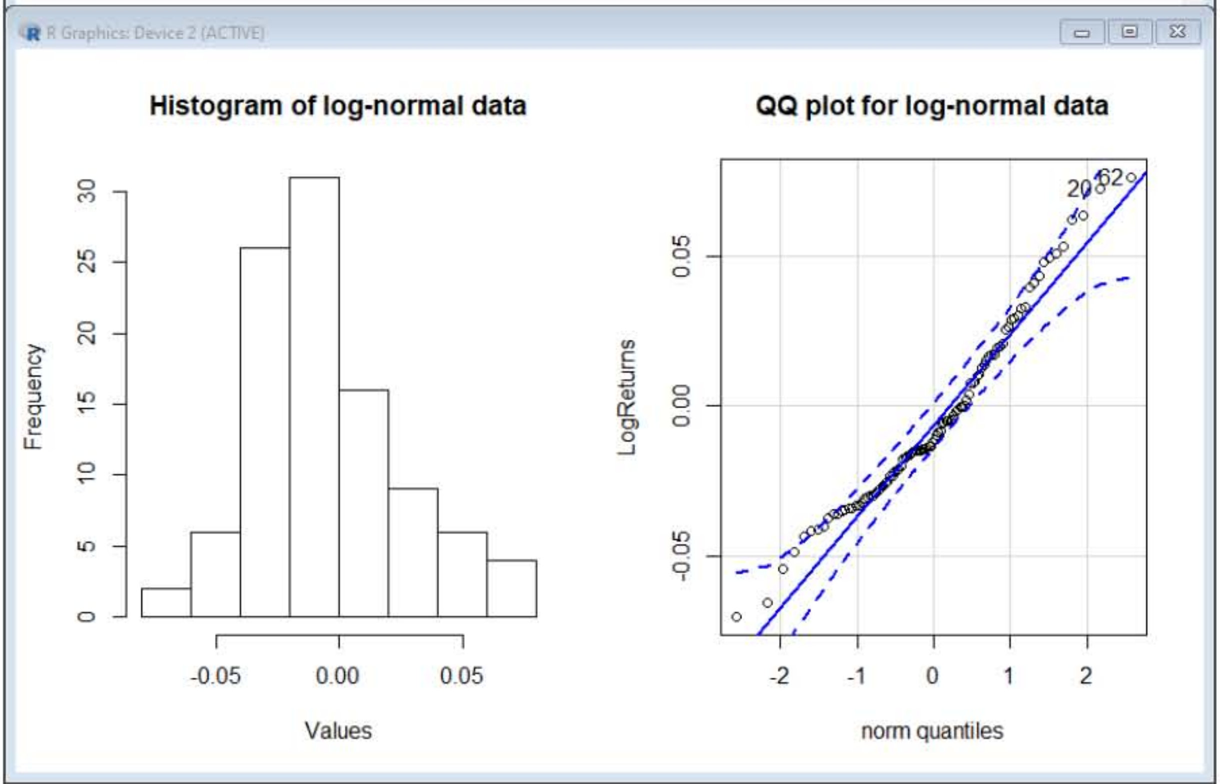
Maintenant regardons s'il passe le test de Normalité de Jaque-Berra avec la fonction **jarque.bera.test( )** du package **tseries** et quelques graphiques avec la fonction **hist( )** du package base et la fonction **qqPlot( )** du package **car**:

```
> library("tseries")
> jarque.bera.test(LogReturns)

        Jarque Bera Test

data:  LogReturns
X-squared = 5.9288, df = 2, p-value = 0.05159

>
> library("car")
> par(mfrow = c(1, 2))
> hist(LogReturns, main="Histogram of log-normal data", xlab="Values")
> qqPlot(LogReturns, main="QQ plot for log-normal data")
[1] 62 20
>
```

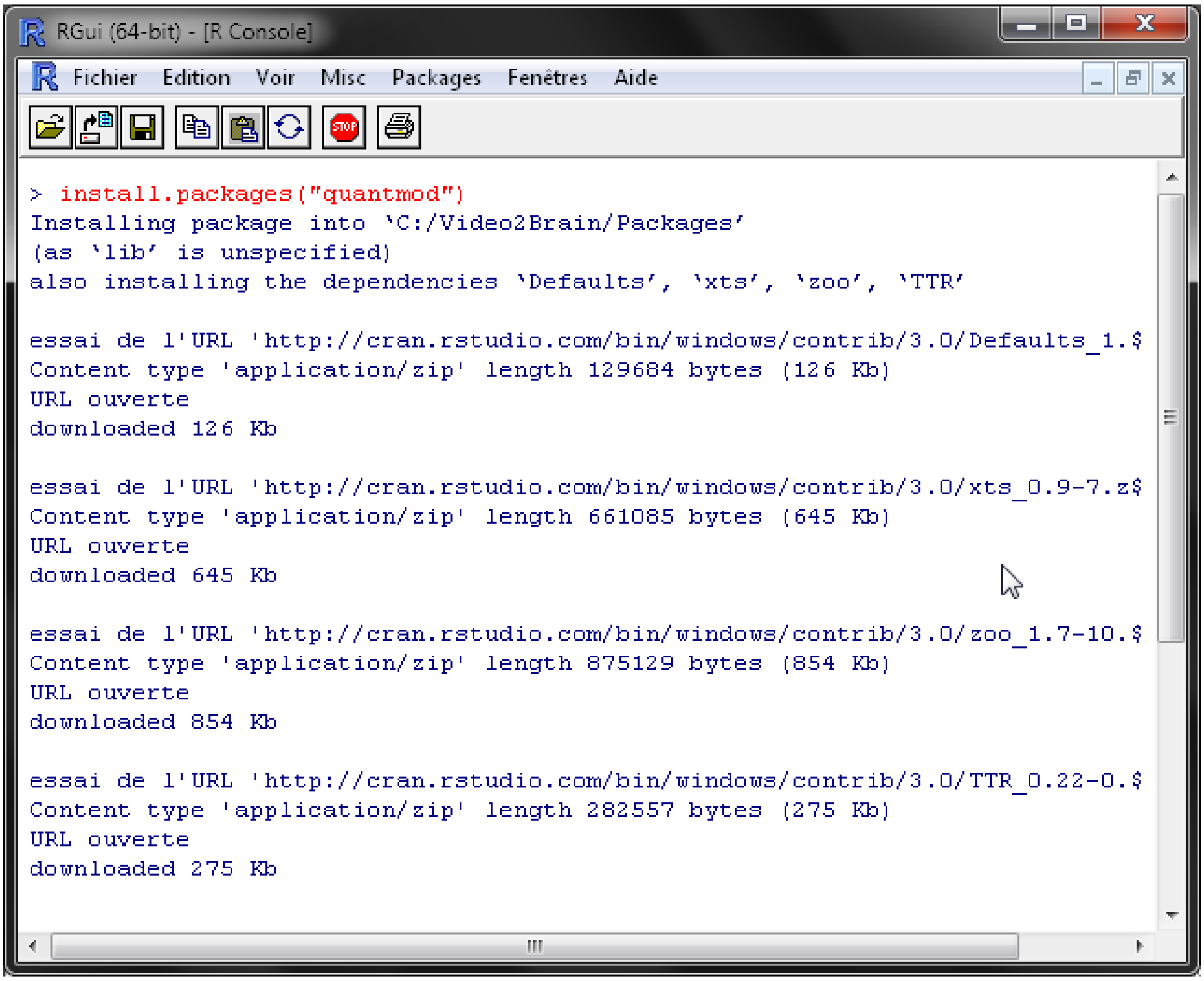


Exemples 399.: Import de données avec les packages quantmod, xts et Quandl

R 3.0.2

Accéder aux cours (trades) des titres est normalement tâche facile vu les nombreux packages dont dispose R à cet effet. Toutefois ces packages communiquent avec des API de plateformes quoique officielles YahooFinance, GoogleFinance, Quandl, Morningstar… mais qui changent souvent les spécifications d'où le dysfonctionnement récurrent qui touche très souvent ce type de package. Donc ne soyez pas étonné si une des exemples ci-dessous marche avec des messages d'avertissements, marche partiellement ou ne marche pas du tout…

Nous allons d'abord installer le package **quantmod**:

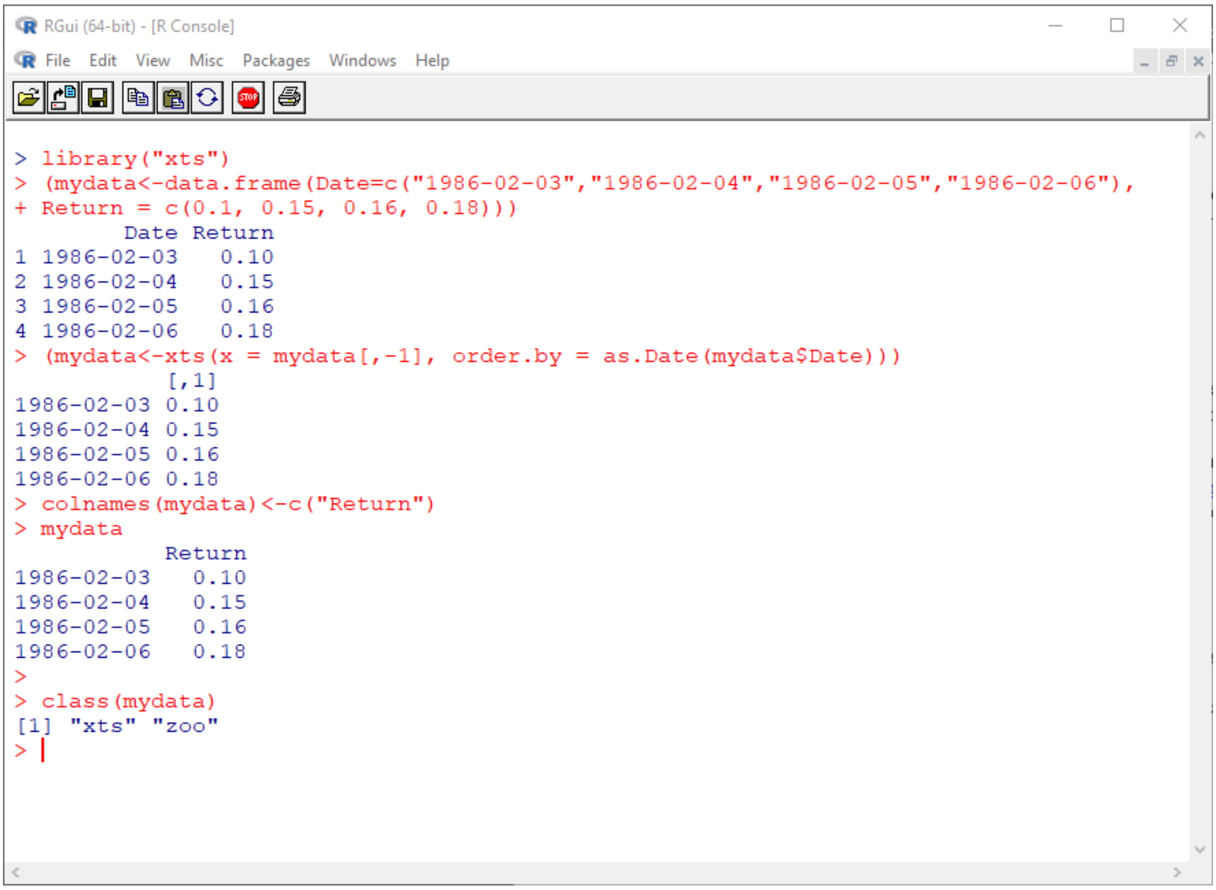


et nous faisons quelques tests dans l'ordre de difficulté croissant au niveau du bagage mathématique nécessaire que nous avons étudié dans le cours théorique d'économie quantitative:

Récupérer des données brutes

Rappels sur les objets XTS et ZOO

La grande majorité des packages en finance utilisent des objets zoo, xts, ou ts. Il est peut-être bon avant de continuer de faire un rappel de comment construire un objet zoo ou xts à partir d'un data frame:



Voilà ceci est fait! Maintenant pour d'autres notions importantes, le lecteur pourra se référer à la CheatSheet ci-dessous et particulièrement les zones en rouge (!):

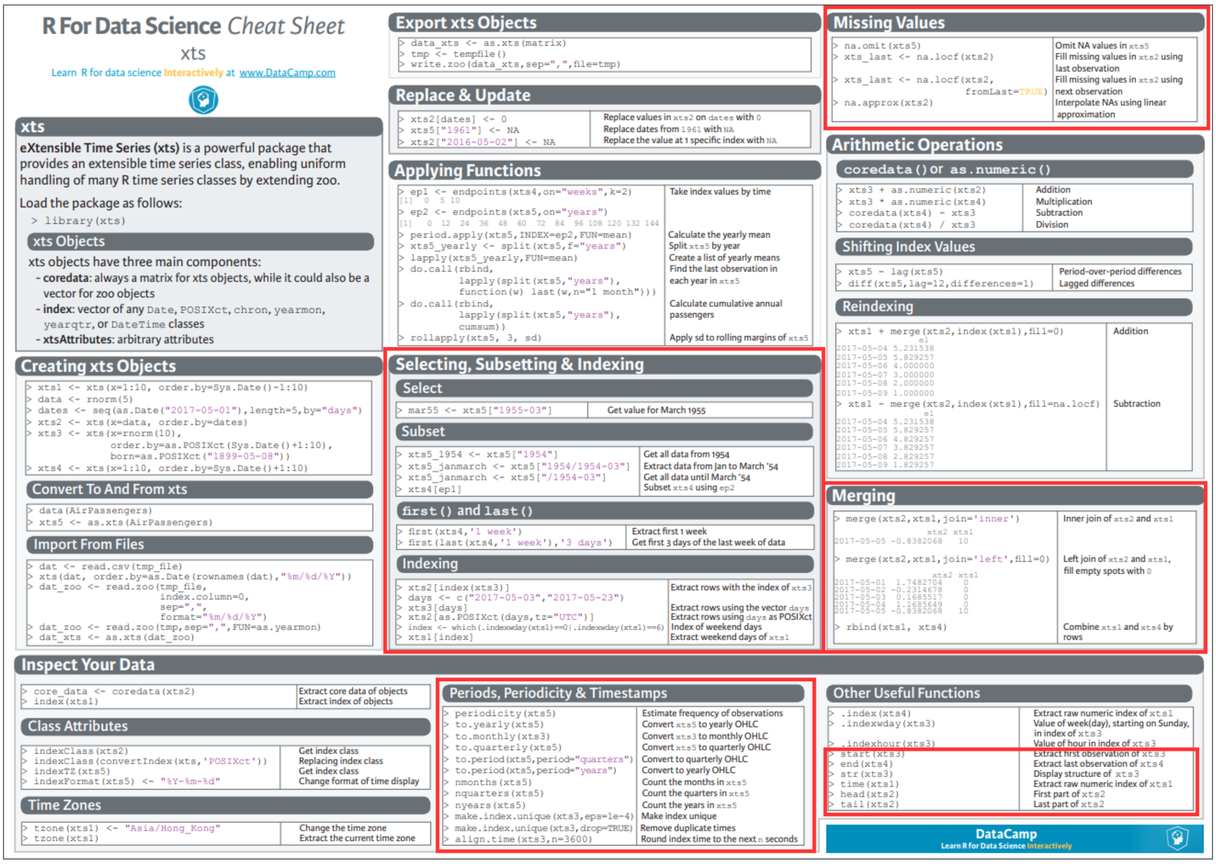


Figure 317 Cheatsheet XTS

Sans packages, non automatisé

Maintenant, nous allons d'abord commencer par la finance qualitative (simpler is better...). C'est-à-dire l'affichage des valeurs d'indices ou d'actifs via de simples graphiques.

Nous utiliserons ici exclusivement le système de données de Yahoo (si vous souhaitez utiliser le service Bloomberg voyez du côté de http://docenti.luiss.it/fasano/bloomr/). Si vous cherchez à récupérer les données pour un indice ou une entreprise particulière dont vous ne connaissez pas le code, vous pouvez utiliser la page suivante:

http://finance.yahoo.com/lookup

Pour commencer, nous allons et importer des données de Yahoo à l'ancienne (sans packages!) et utiliser le package **ggplot2** pour les plotter et utiliser la commande **geom_ribbon( )** pour faire une bande de fluctuation des cours par rapport à la moyenne arithmétique (le SPY est un fond qui a pour but de répliquer le S&P 500):

R Console

```
> library(ggplot2)
> #on choisit l'indice du stock
> stock_index<-"SPY"
> #on définit le début
> start_year<-2008;start_month<-1;start_day<-15
> #on définit la fin
> finish_year<-2010;finish_month<-12;finish_day<-31
> #fréquence de mesure
> frequency="w"  #d, w ou  m
>
> URL<-paste("http://ichart.finance.yahoo.com/table.csv?s=",stock_index,"&a=",
+ start_month-1,"&b=",start_day,"&c=",start_year,
+ "&d=",finish_month-1,"&e=",finish_day,"&f=",finish_year,
+ "&g=",frequency,sep="")
>
> mydata <- read.csv(URL)
> mydata$Date <- as.Date(mydata$Date, "%Y-%m-%d")
> #voir à quoi cela ressemble
> mydata[1:5,]
        Date   Open   High    Low  Close    Volume Adj.Close
1 2010-12-27 125.13 126.20 125.04 125.75  67860800    118.11
2 2010-12-20 124.64 125.82 123.98 125.60  90745700    117.97
3 2010-12-13 125.05 125.23 123.75 124.30 153599100    116.75
4 2010-12-06 122.63 124.60 122.41 124.48 137785500    116.30
5 2010-11-29 118.50 123.03 117.74 122.89 204222500    114.82
> ggplot(mydata,aes(x=Date,y=(High+Low)/2))+
+ geom_ribbon(aes(ymin=Low,ymax=High),alpha=0.2)+
+ geom_line()
>
```

Attention!!!! Depuis Mai 2017, Yahoo a rendu les choses compliquées (avec des cookies sous formes de sessions)… et le code suivant ne fonctionne plus!!!!!!! Mieux vaut passer par la package *quantmod* (dans une certaine mesure…) ou *Quandl* comme nous allons le voir plus bas!

Ce qui donne:

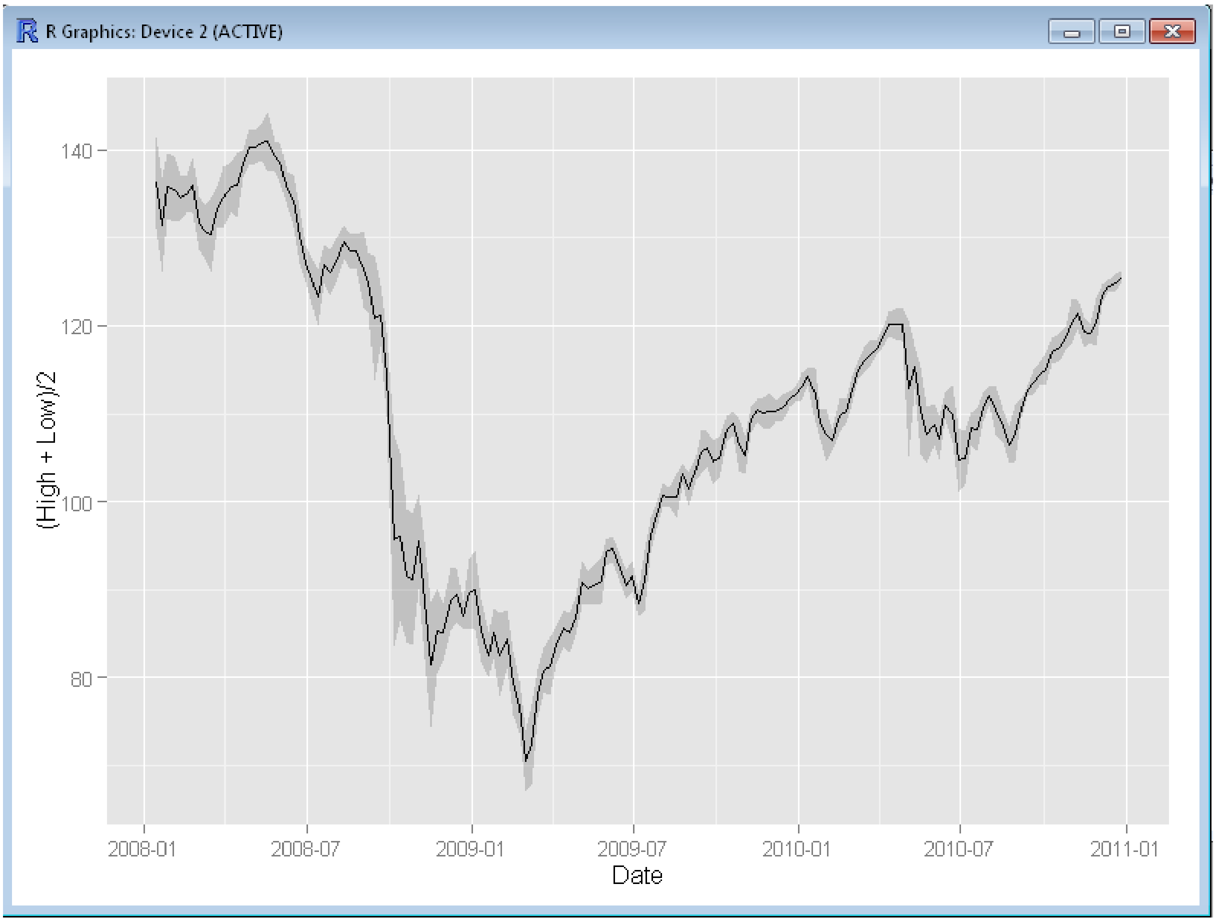
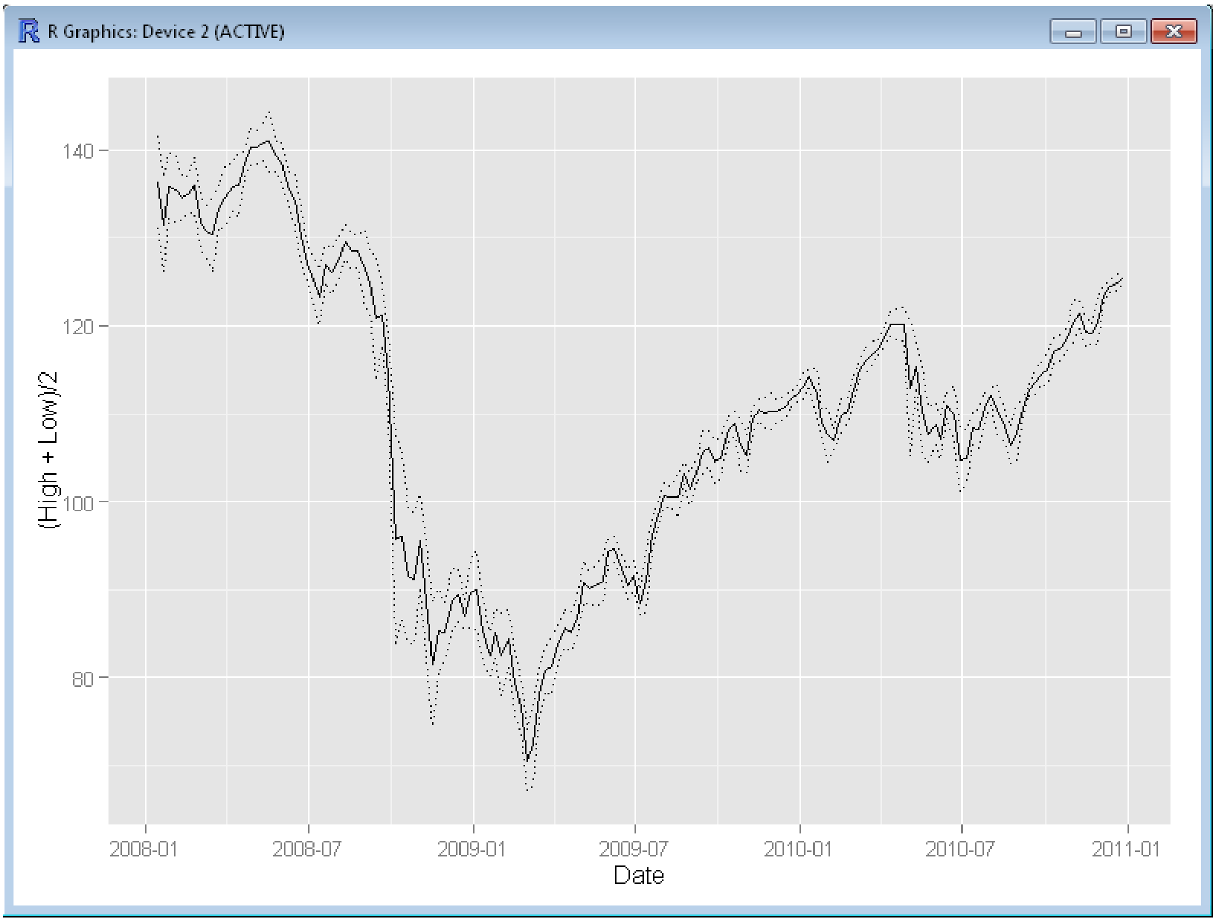


Figure 318 Graph d'une série financier avec intervalle de fluctuation (ggplot)

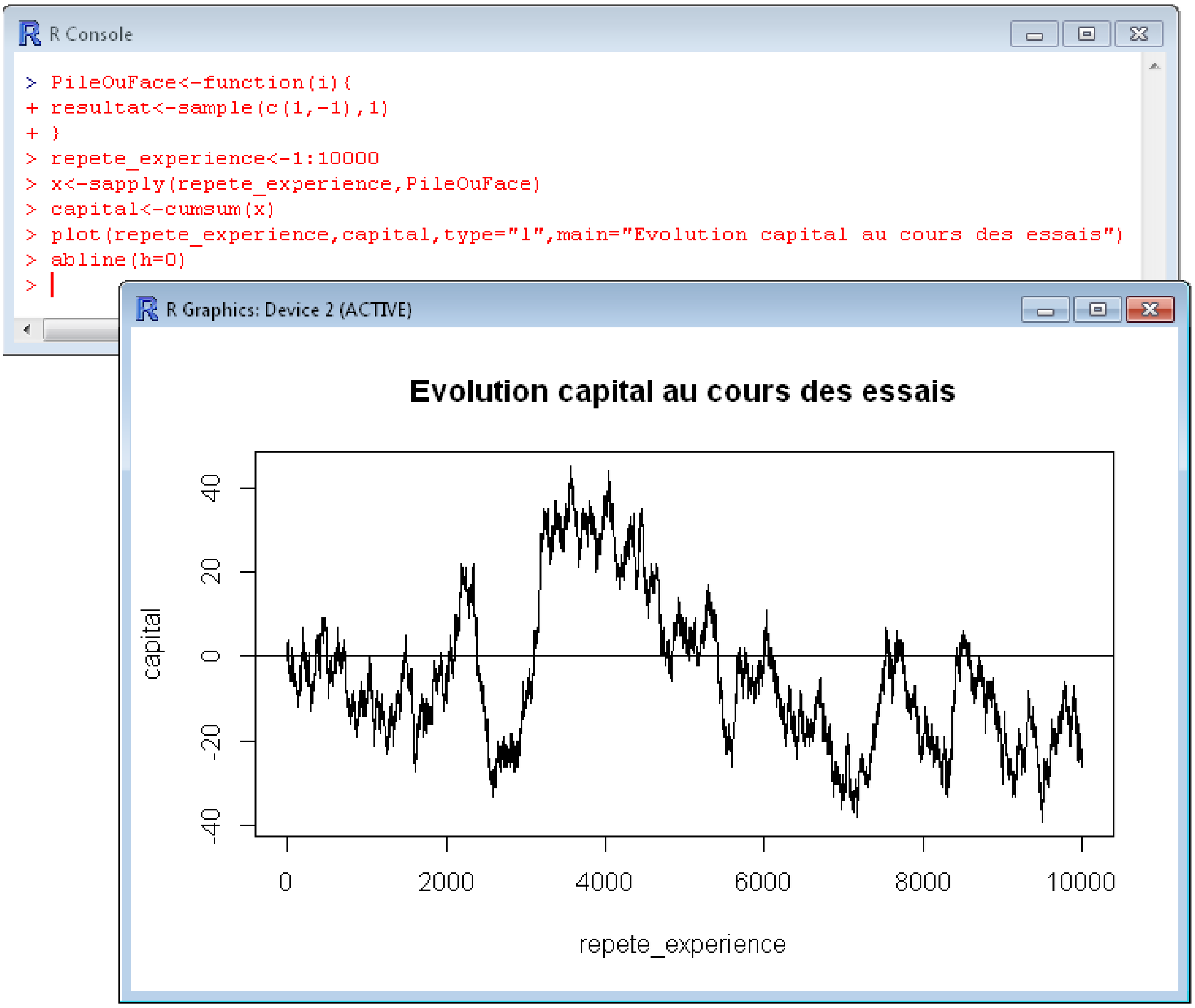
ou avec un autre style visuel en utilisant le script R suivant:

```
> library(ggplot2)
> #on choisit l'indice du stock
> stock_index<-"SPY"
> #on définit le début
> start_year<-2008;start_month<-1;start_day<-15
> #on définit la fin
> finish_year<-2010;finish_month<-12;finish_day<-31
> #fréquence de mesure
> frequency="w"  #d, w ou  m
>
> URL<-paste("http://ichart.finance.yahoo.com/table.csv?s=",stock_index,"&a=",
+ start_month-1,"&b=",start_day,"&c=",start_year,
+ "&d=",finish_month-1,"&e=",finish_day,"&f=",finish_year,
+ "&g=",frequency,sep="")
>
> mydata <- read.csv(URL)
> mydata$Date <- as.Date(mydata$Date, "%Y-%m-%d")
>
> mydata$Date <- as.Date(mydata$Date, "%Y-%m-%d")
> mydata<-mydata[1:100,]
> ggplot(mydata,aes(x=Date,y=(High+Low)/2))+
+ geom_line(aes(y=High),linetype="dotted")+
+ geom_line(aes(y=Low),linetype="dotted")+
+ geom_line()
>
```

Ce qui donne:

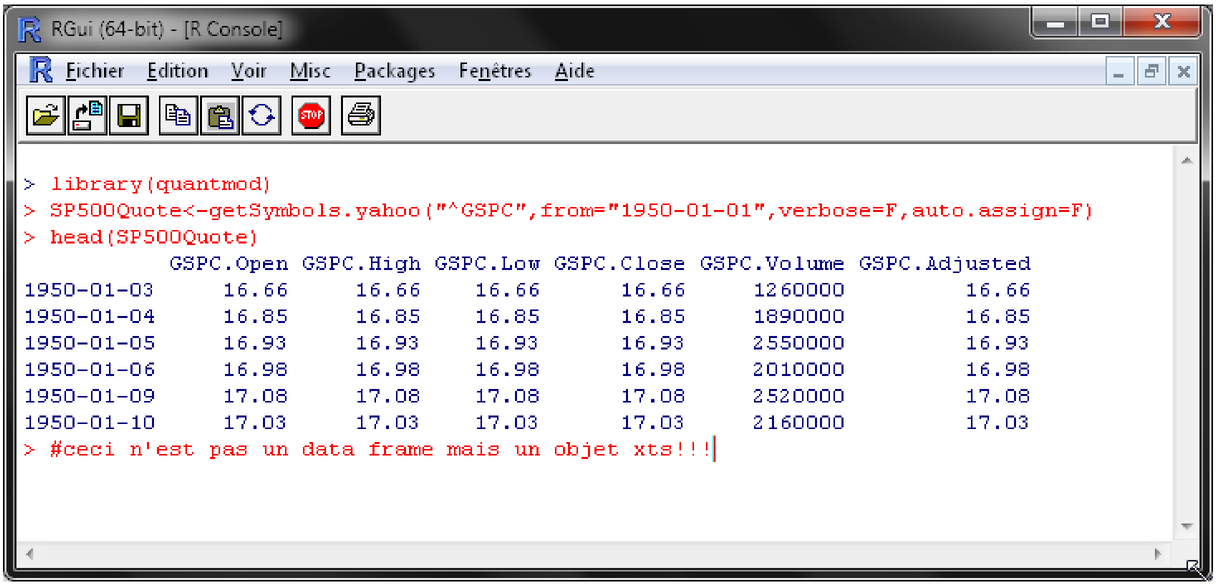


...et puis comme la finance c'est un peu jouer au casino (pile ou face) sur les marchés financier un bon petit jeu est souvent ludique (et au passage une application ludique de la commande **sapply( )**):


R Console
> PileOuFace<-function(i){
+ resultat<-sample(c(1,-1),1)
+ }
> repete_experience<-1:10000
> x<-sapply(repete_experience,PileOuFace)
> capital<-cumsum(x)
> plot(repete_experience,capital,type="l",main="Evolution capital au cours des essais")
> abline(h=0)
>
R Graphics: Device 2 (ACTIVE)
Evolution capital au cours des essais
capital
-40
-20
0
20
40
0
2000
4000
6000
8000
10000
repete_experience

Avec package, non automatisé différé ou actuel (indices/titres/options/dividendes)

Récupérons des données brutes de l'indice S&P 500 (abrégé ^GSPC, INX ou $SPX) avec la commande **getSymbols.yahoo( )** du package **quantmod**:

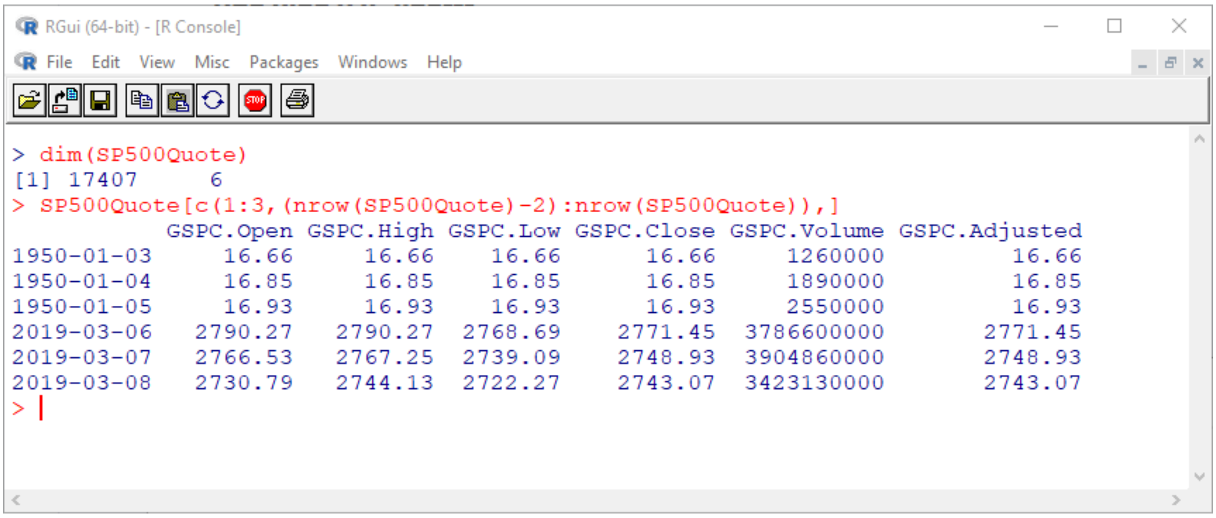


```
> library(quantmod)
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> head(SP500Quote)
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1950-01-03     16.66     16.66    16.66      16.66     1260000         16.66
1950-01-04     16.85     16.85    16.85      16.85     1890000         16.85
1950-01-05     16.93     16.93    16.93      16.93     2550000         16.93
1950-01-06     16.98     16.98    16.98      16.98     2010000         16.98
1950-01-09     17.08     17.08    17.08      17.08     2520000         17.08
1950-01-10     17.03     17.03    17.03      17.03     2160000         17.03
> #ceci n'est pas un data frame mais un objet xts!!!
```

C'est simple et limpide! Mais il y a de fortes chances que quantmod ne fonctionne plus un jour ou l'autre vu qu'il utilise Yahoo! en arrière-plan et que cette dernière entreprise n'est pas fan de ce genre de pratiques…

Rappel: Pour extraire la première colonne (colonnes des dates) d'un objet XTS ou ZOO, il faudra utiliser la fonction **index( )**, ainsi cela donnera dans le cas présent **index(***SPQ500Quote***)**.

Souvent le résultat est assez gigantesque. Il est parfois intéressant d'observer la différence entre les quelques premières et dernières lignes. Pour rappel, cela se fait avec:

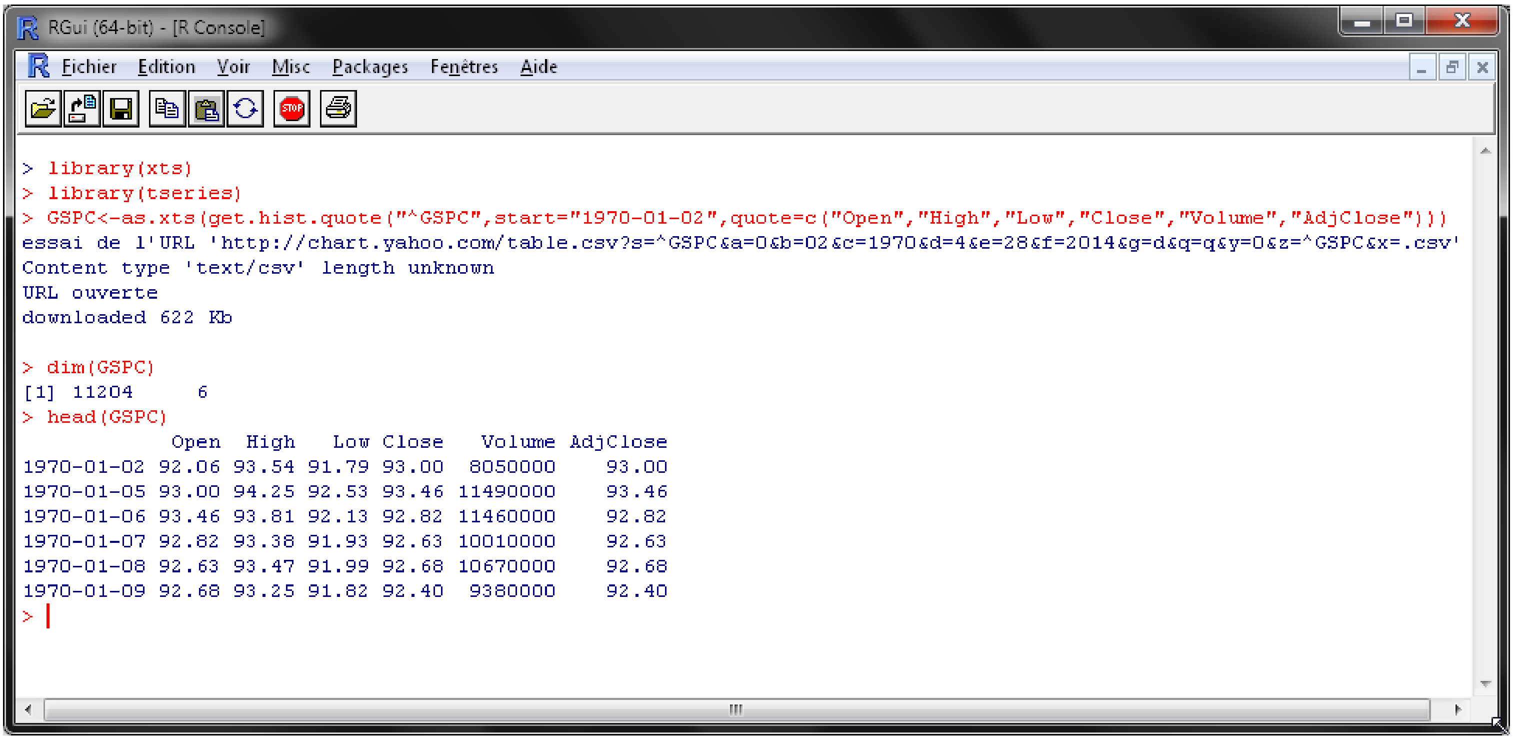


```
> dim(SP500Quote)
[1] 17407     6
> SP500Quote[c(1:3,(nrow(SP500Quote)-2):nrow(SP500Quote)),]
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1950-01-03     16.66     16.66    16.66      16.66     1260000         16.66
1950-01-04     16.85     16.85    16.85      16.85     1890000         16.85
1950-01-05     16.93     16.93    16.93      16.93     2550000         16.93
2019-03-06   2790.27   2790.27  2768.69    2771.45  3786600000       2771.45
2019-03-07   2766.53   2767.25  2739.09    2748.93  3904860000       2748.93
2019-03-08   2730.79   2744.13  2722.27    2743.07  3423130000       2743.07
>
```

Notons qu'on peut obtenir la liste des stocks disponibles avec certaines métadonnées de la manière suivante:

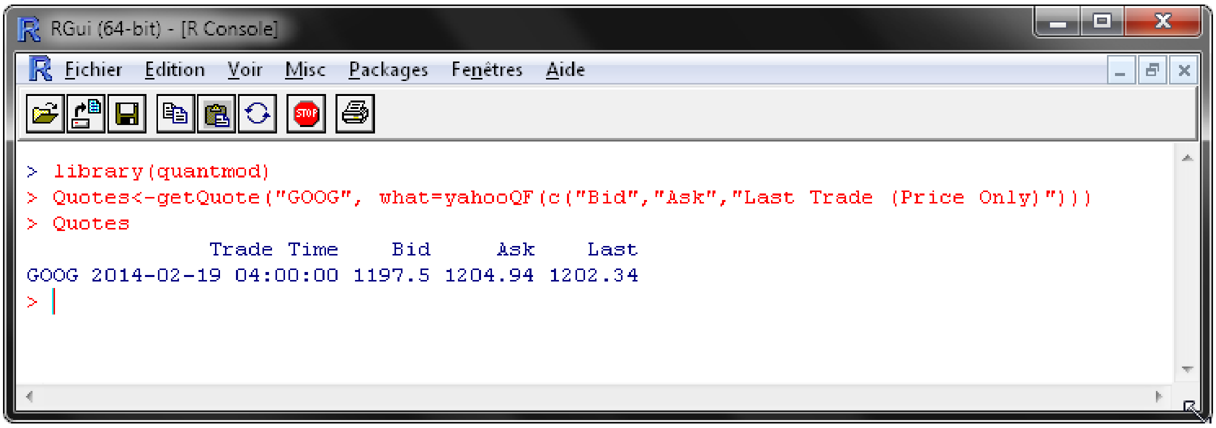


Nous pouvons obtenir l'équivalent avec la commande **as.xts( )** du package **xts** associée à la commande **get.hist.quote( )** du package **tseries**:

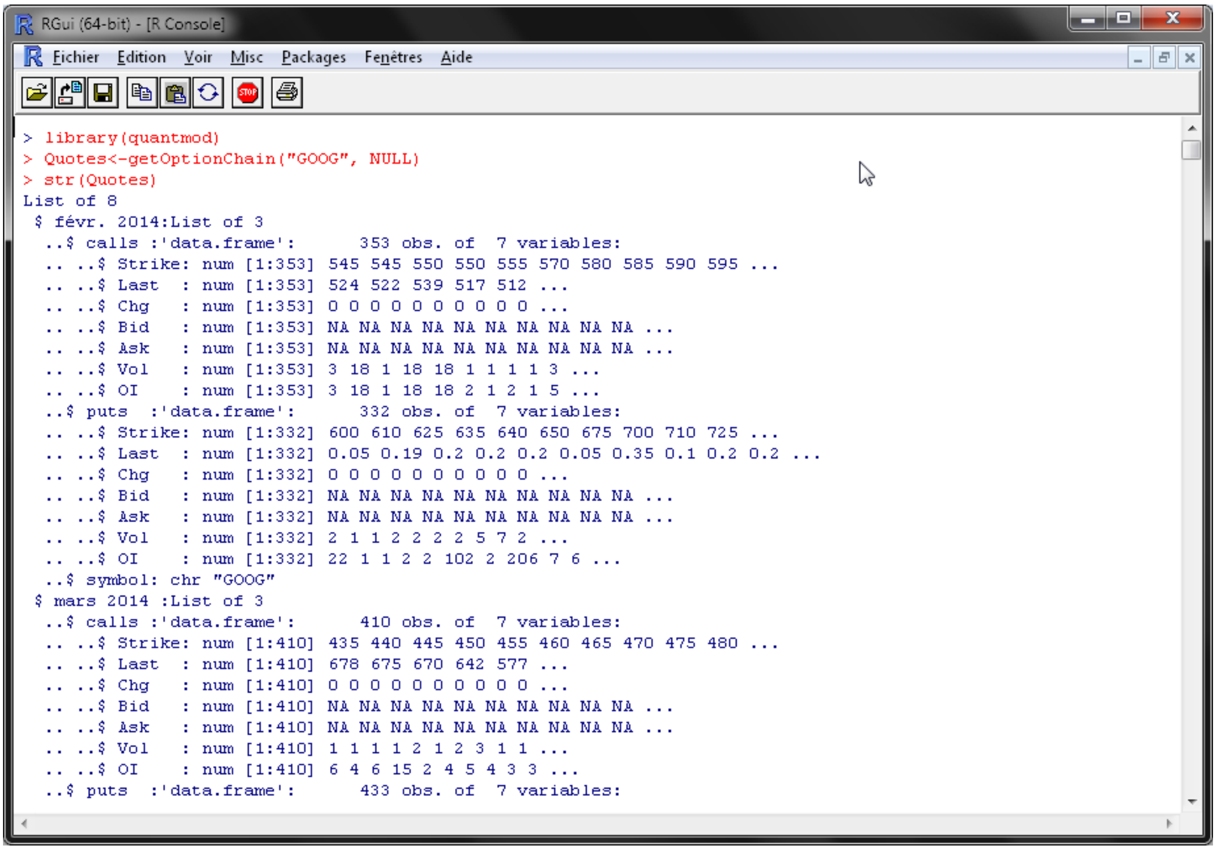


Si nous voulons le dernier cours du jour[42] d'un titre (ci-dessous le titre de Google) avec le prix de l'offre et de la demande nous utiliserons la commande **getQuote( )** :

[42] Cette fonction ne peut en renvoyer qu'un seul et le dernier! Il n'existe pas à jour de paramètres pour en rappatrier d'autres. Normalement, Yahoo met à jour cette valeur chaque minute!



et avec la commande **getOptionChain( )** nous avons les informations concernant les options (dérivés) du sous-jacent choisi (du moins quand certaines des informations ne sont pas manquantes...):



On peut donc extraire un sous élément facilement:

```
> head(Quotes[[1]]$calls,10)
                    Strike   Last      Chg   Bid   Ask Vol OI
GOOG190322C00820000  820.0 367.36 48.38998 363.0 368.0   2  2
GOOG190322C00900000  900.0 296.80  0.00000 283.0 288.0   1 17
GOOG190322C00920000  920.0 201.23  0.00000 263.0 268.0   0 19
GOOG190322C00930000  930.0 171.00  0.00000 253.0 258.0   1  1
GOOG190322C00940000  940.0 182.20  0.00000 243.0 248.0   0 19
GOOG190322C01000000 1000.0 196.81  0.00000 183.0 188.0  20 22
GOOG190322C01030000 1030.0  99.45  0.00000 153.0 158.0   0  5
GOOG190322C01035000 1035.0 151.14 56.04000 148.0 153.0   1  5
GOOG190322C01037500 1037.5  70.40  0.00000 145.5 150.5   1  1
GOOG190322C01045000 1045.0  94.50  0.00000 138.0 143.0   1  1
> 
```

La premier élément correspondant au "ticker" que nous décortiquerons plus bas!

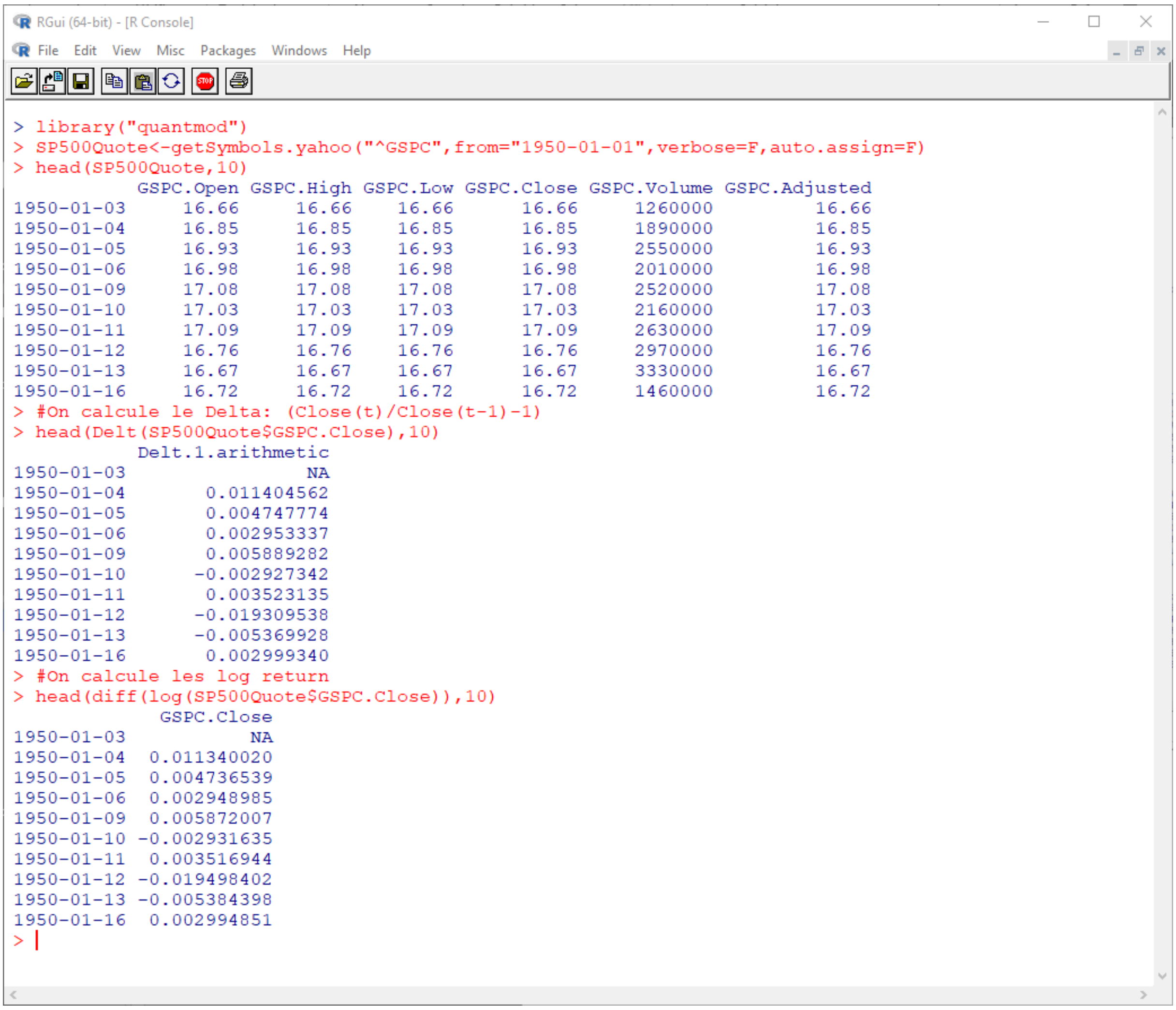
Dont la source est en réalité ici:

http://finance.yahoo.com/q/op?s=GOOG&ql=1

Revenant au package **quantmod**, nous pouvons transformer simplement la période de fenêtre d'analyse avec les fonctions **to.weekly( )**, **to.monthly( )**, **to.quarterly( )**, **to.yearly( )** et nous pouvons faire encore plus (se référer à la documentation du package!):

```
> library("quantmod")
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> head(SP500Quote)
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1950-01-03     16.66     16.66    16.66      16.66     1260000         16.66
1950-01-04     16.85     16.85    16.85      16.85     1890000         16.85
1950-01-05     16.93     16.93    16.93      16.93     2550000         16.93
1950-01-06     16.98     16.98    16.98      16.98     2010000         16.98
1950-01-09     17.08     17.08    17.08      17.08     2520000         17.08
1950-01-10     17.03     17.03    17.03      17.03     2160000         17.03
> SP500QuoteWeekly<-to.weekly(SP500Quote)
> head(SP500QuoteWeekly)
           SP500Quote.Open SP500Quote.High SP500Quote.Low SP500Quote.Close SP500Quote.Volume SP500Quote.Adjusted
1950-01-06           16.66           16.98          16.66            16.98           7710000               16.98
1950-01-13           17.08           17.09          16.67            16.67          13610000               16.67
1950-01-20           16.72           16.90          16.72            16.90           7430000               16.90
1950-01-27           16.92           16.92          16.73            16.82           6690000               16.82
1950-02-03           17.02           17.29          17.02            17.29           9390000               17.29
1950-02-10           17.32           17.32          17.21            17.24           7920000               17.24
> SP500QuoteMontly<-to.monthly(SP500Quote)
> head(SP500QuoteMontly)
         SP500Quote.Open SP500Quote.High SP500Quote.Low SP500Quote.Close SP500Quote.Volume SP500Quote.Adjusted
Jan 1950           16.66           17.09          16.66            17.05          38770000               17.05
Feb 1950           17.05           17.32          16.99            17.22          30200000               17.22
Mar 1950           17.24           17.56          17.07            17.29          37450000               17.29
Apr 1950           17.53           18.05          17.53            17.96          42630000               17.96
May 1950           18.22           18.78          18.11            18.78          42230000               18.78
Jun 1950           18.77           19.40          17.44            17.69          45660000               17.69
> SP500QuoteQuarterly<-to.quarterly(SP500Quote)
> head(SP500QuoteQuarterly)
        SP500Quote.Open SP500Quote.High SP500Quote.Low SP500Quote.Close SP500Quote.Volume SP500Quote.Adjusted
1950 Q1           16.66           17.56          16.66            17.29         106420000               17.29
1950 Q2           17.53           19.40          17.44            17.69         130520000               17.69
1950 Q3           17.64           19.45          16.68            19.45         122060000               19.45
1950 Q4           19.69           20.43          19.00            20.43         141300000               20.43
1951 Q1           20.77           22.20          20.69            21.48         135110000               21.48
1951 Q2           21.32           22.81          20.96            20.96         104140000               20.96
> SP500QuoteYearly<-to.yearly(SP500Quote)
> head(SP500QuoteYearly)
           SP500Quote.Open SP500Quote.High SP500Quote.Low SP500Quote.Close SP500Quote.Volume SP500Quote.Adjusted
1950-12-29           16.66           20.43          16.66            20.43         500300000               20.43
1951-12-31           20.77           23.85          20.69            23.77         427680000               23.77
1952-12-31           23.80           26.59          23.09            26.57         327040000               26.57
1953-12-31           26.54           26.66          22.71            24.81         354700000               24.81
1954-12-31           24.95           35.98          24.80            35.98         571980000               35.98
1955-12-30           36.75           46.41          34.58            45.48         648450000               45.48
>
```

On peut s'amuser à appliquer les fonction **Delt( )**, **diff( )** et **log( )** vu au début du cours pour calculer les ratios et rendements:

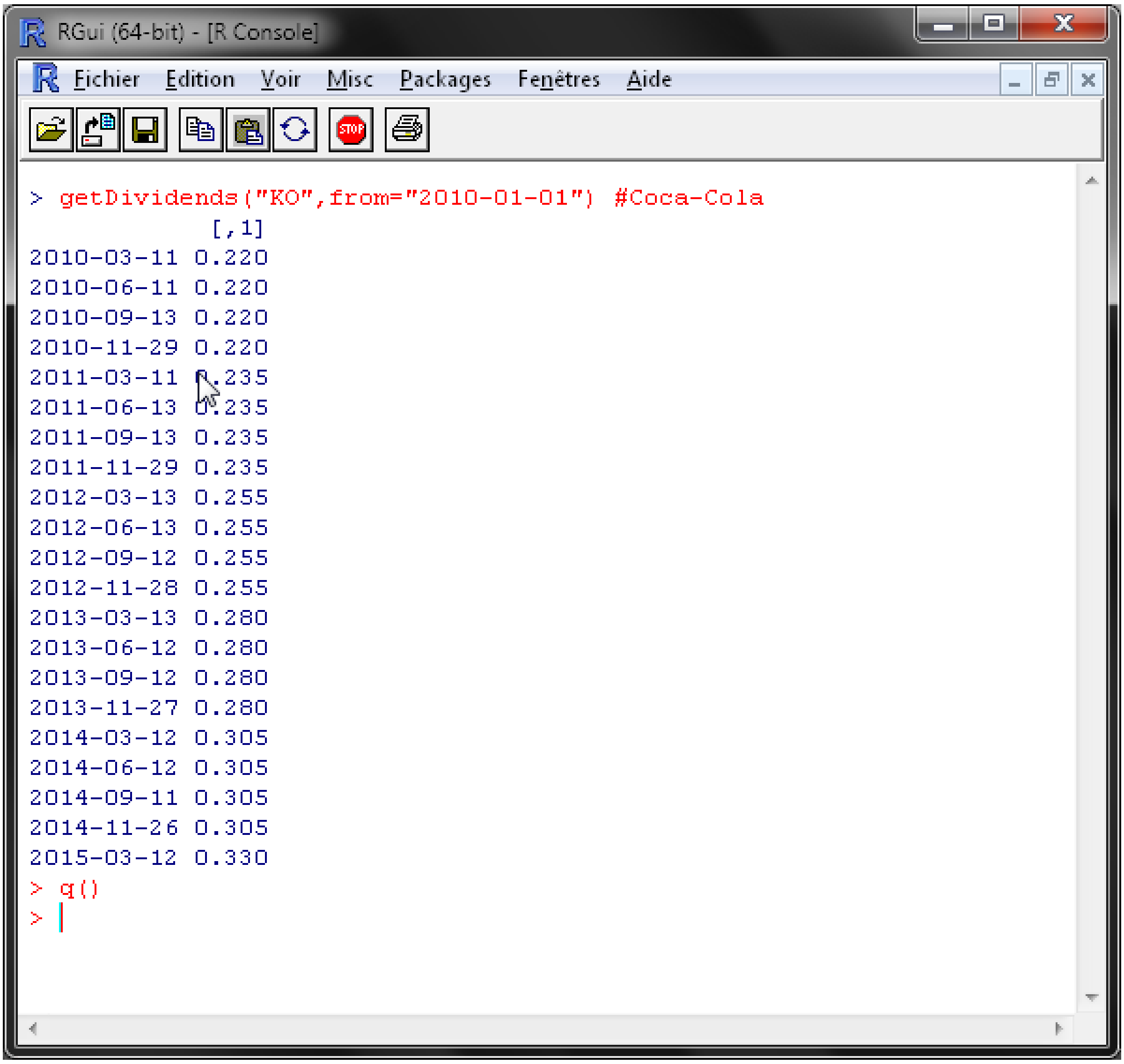


```
> library("quantmod")
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> head(SP500Quote,10)
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1950-01-03     16.66     16.66    16.66      16.66     1260000         16.66
1950-01-04     16.85     16.85    16.85      16.85     1890000         16.85
1950-01-05     16.93     16.93    16.93      16.93     2550000         16.93
1950-01-06     16.98     16.98    16.98      16.98     2010000         16.98
1950-01-09     17.08     17.08    17.08      17.08     2520000         17.08
1950-01-10     17.03     17.03    17.03      17.03     2160000         17.03
1950-01-11     17.09     17.09    17.09      17.09     2630000         17.09
1950-01-12     16.76     16.76    16.76      16.76     2970000         16.76
1950-01-13     16.67     16.67    16.67      16.67     3330000         16.67
1950-01-16     16.72     16.72    16.72      16.72     1460000         16.72
> #On calcule le Delta: (Close(t)/Close(t-1)-1)
> head(Delt(SP500Quote$GSPC.Close),10)
           Delt.1.arithmetic
1950-01-03                NA
1950-01-04       0.011404562
1950-01-05       0.004747774
1950-01-06       0.002953337
1950-01-09       0.005889282
1950-01-10      -0.002927342
1950-01-11       0.003523135
1950-01-12      -0.019309538
1950-01-13      -0.005369928
1950-01-16       0.002999340
> #On calcule les log return
> head(diff(log(SP500Quote$GSPC.Close)),10)
             GSPC.Close
1950-01-03           NA
1950-01-04  0.011340020
1950-01-05  0.004736539
1950-01-06  0.002948985
1950-01-09  0.005872007
1950-01-10 -0.002931635
1950-01-11  0.003516944
1950-01-12 -0.019498402
1950-01-13 -0.005384398
1950-01-16  0.002994851
>
```

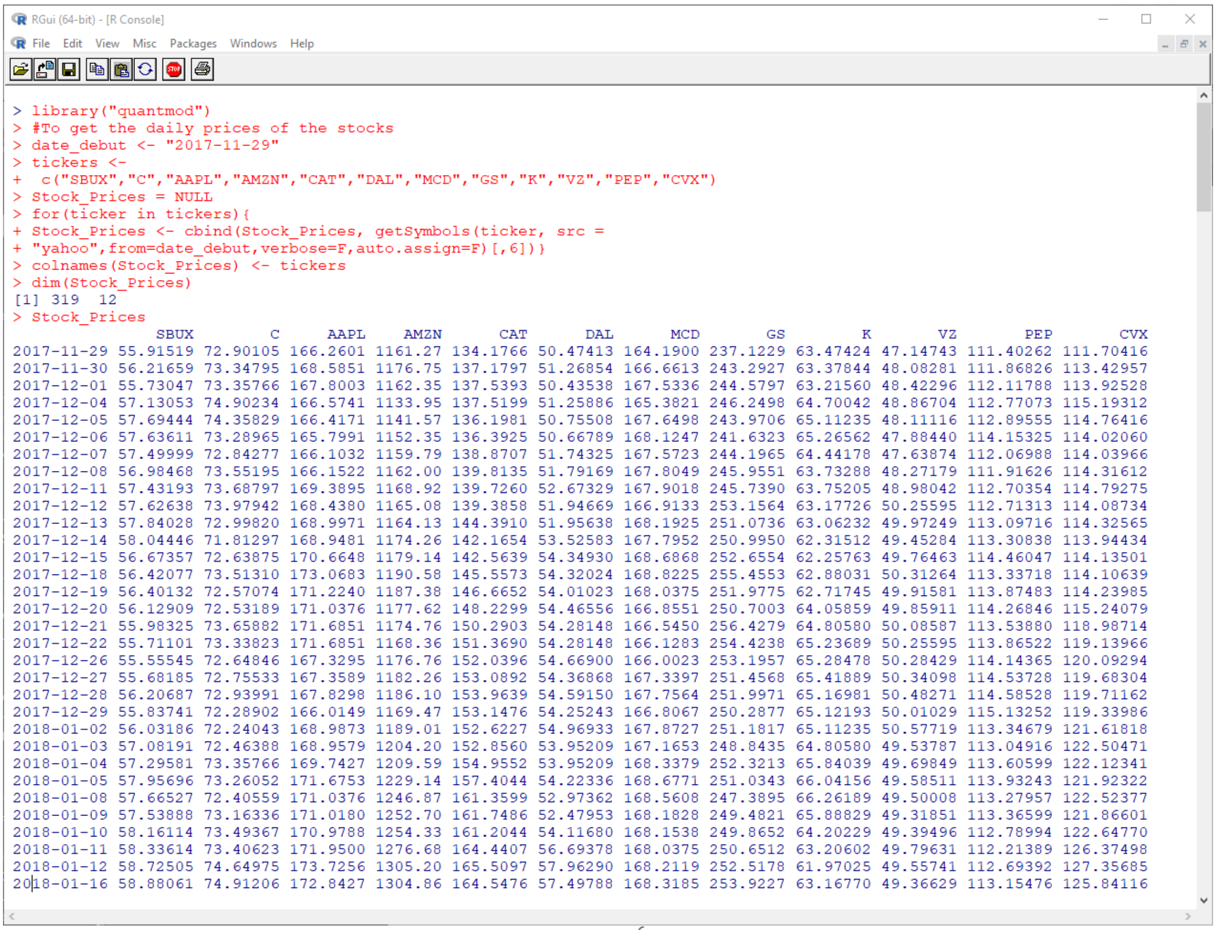
Ou on peut calculer les rendements aussi avec la fonction **monthlyReturn( )** (la même fonction existe pour le jour, l'année et autre…):

```
> head(yearlyReturn(SP500Quote[,4]),10)
           yearly.returns
1950-12-29     0.22629052
1951-12-31     0.16348507
1952-12-31     0.11779554
1953-12-31    -0.06624016
1954-12-31     0.45022174
1955-12-30     0.26403558
1956-12-31     0.02616530
1957-12-31    -0.14313255
1958-12-31     0.38059505
1959-12-31     0.08476725
> head(monthlyReturn(SP500Quote[,4]),10)
           monthly.returns
1950-01-31     0.023409304
1950-02-28     0.009970675
1950-03-31     0.004065157
1950-04-28     0.038750605
1950-05-31     0.045657129
1950-06-30    -0.058040465
1950-07-31     0.008479310
1950-08-31     0.032511211
1950-09-29     0.055917535
1950-10-31     0.004113110
> head(dailyReturn(SP500Quote[,4]),10)
           daily.returns
1950-01-03   0.000000000
1950-01-04   0.011404562
1950-01-05   0.004747774
1950-01-06   0.002953337
1950-01-09   0.005889282
1950-01-10  -0.002927342
1950-01-11   0.003523135
1950-01-12  -0.019309538
1950-01-13  -0.005369928
1950-01-16   0.002999340
>
```

Nous pouvons aussi obtenir les dividendes avec la fonction **getDividends( )** :

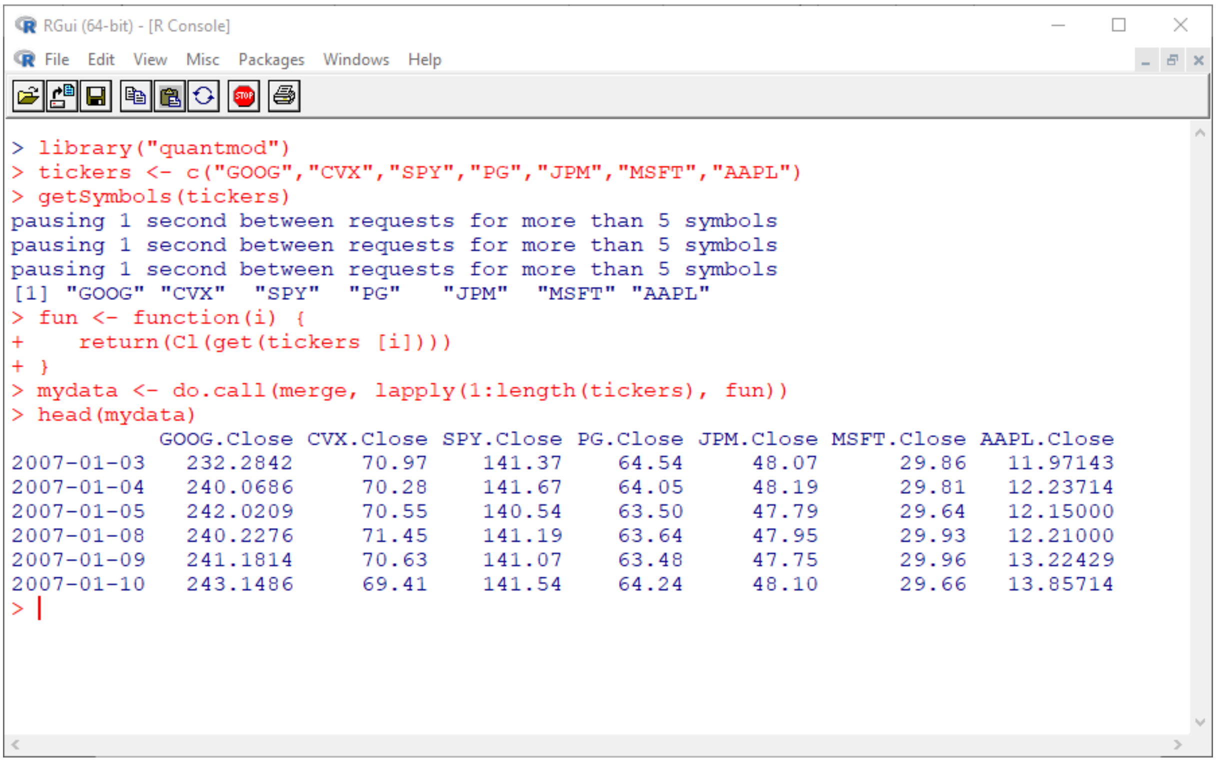


Pour extraire un ensemble d'indices, nous pouvons utiliser une simple boucle *for*:



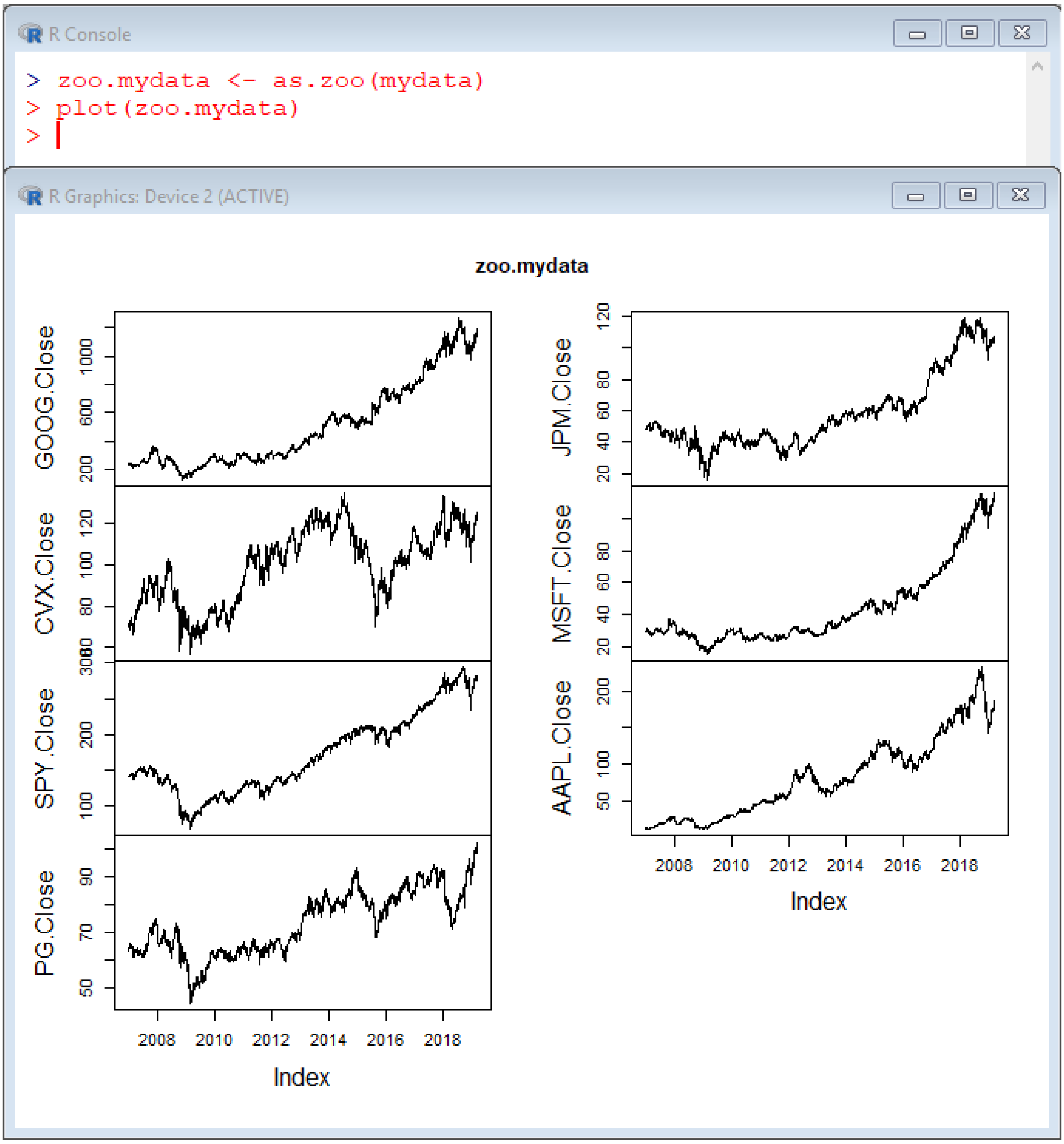
```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("quantmod")
> #To get the daily prices of the stocks
> date_debut <- "2017-11-29"
> tickers <-
+  c("SBUX","C","AAPL","AMZN","CAT","DAL","MCD","GS","K","VZ","PEP","CVX")
> Stock_Prices = NULL
> for(ticker in tickers){
+ Stock_Prices <- cbind(Stock_Prices, getSymbols(ticker, src =
+ "yahoo",from=date_debut,verbose=F,auto.assign=F)[,6])}
> colnames(Stock_Prices) <- tickers
> dim(Stock_Prices)
[1] 319  12
> Stock_Prices
               SBUX        C     AAPL    AMZN      CAT      DAL      MCD       GS        K       VZ       PEP       CVX
2017-11-29 55.91519 72.90105 166.2601 1161.27 134.1766 50.47413 164.1900 237.1229 63.47424 47.14743 111.40262 111.70416
2017-11-30 56.21659 73.34795 168.5851 1176.75 137.1797 51.26854 166.6613 243.2927 63.37844 48.08281 111.86826 113.42957
2017-12-01 55.73047 73.35766 167.8003 1162.35 137.5393 50.43538 167.5336 244.5797 63.21560 48.42296 112.11788 113.92528
2017-12-04 57.13053 74.90234 166.5741 1133.95 137.5199 51.25886 165.3821 246.2498 64.70042 48.86704 112.77073 115.19312
2017-12-05 57.69444 74.35829 166.4171 1141.57 136.1981 50.75508 167.6498 243.9706 65.11235 48.11116 112.89555 114.76416
2017-12-06 57.63611 73.28965 165.7991 1152.35 136.3925 50.66789 168.1247 241.6323 65.26562 47.88440 114.15325 114.02060
2017-12-07 57.49999 72.84277 166.1032 1159.79 138.8707 51.74325 167.5723 244.1965 64.44178 47.63874 112.06988 114.03966
2017-12-08 56.98468 73.55195 166.1522 1162.00 139.8135 51.79169 167.8049 245.9551 63.73288 48.27179 111.91626 114.31612
2017-12-11 57.43193 73.68797 169.3895 1168.92 139.7260 52.67329 167.9018 245.7390 63.75205 48.98042 112.70354 114.79275
2017-12-12 57.62638 73.97942 168.4380 1165.08 139.3858 51.94669 166.9133 253.1564 63.17726 50.25595 112.71313 114.08734
2017-12-13 57.84028 72.99820 168.9971 1164.13 144.3910 51.95638 168.1925 251.0736 63.06232 49.97249 113.09716 114.32565
2017-12-14 58.04446 71.81297 168.9481 1174.26 142.1654 53.52583 167.7952 250.9950 62.31512 49.45284 113.30838 113.94434
2017-12-15 56.67357 72.63875 170.6648 1179.14 142.5639 54.34930 168.6868 252.6554 62.25763 49.76463 114.46047 114.13501
2017-12-18 56.42077 73.51310 173.0683 1190.58 145.5573 54.32024 168.8225 255.4553 62.88031 50.31264 113.33718 114.10639
2017-12-19 56.40132 72.57074 171.2240 1187.38 146.6652 54.01023 168.0375 251.9775 62.71745 49.91581 113.87483 114.23985
2017-12-20 56.12909 72.53189 171.0376 1177.62 148.2299 54.46556 166.8551 250.7003 64.05859 49.85911 114.26846 115.24079
2017-12-21 55.98325 73.65882 171.6851 1174.76 150.2903 54.28148 166.5450 256.4279 64.80580 50.08587 113.53880 118.98714
2017-12-22 55.71101 73.33823 171.6851 1168.36 151.3690 54.28148 166.1283 254.4238 65.23689 50.25595 113.86522 119.13966
2017-12-26 55.55545 72.64846 167.3295 1176.76 152.0396 54.66900 166.0023 253.1957 65.28478 50.28429 114.14365 120.09294
2017-12-27 55.68185 72.75533 167.3589 1182.26 153.0892 54.36868 167.3397 251.4568 65.41889 50.34098 114.53728 119.68304
2017-12-28 56.20687 72.93991 167.8298 1186.10 153.9639 54.59150 167.7564 251.9971 65.16981 50.48271 114.58528 119.71162
2017-12-29 55.83741 72.28902 166.0149 1169.47 153.1476 54.25243 166.8067 250.2877 65.12193 50.01029 115.13252 119.33986
2018-01-02 56.03186 72.24043 168.9873 1189.01 152.6227 54.96933 167.8727 251.1817 65.11235 50.57719 113.34679 121.61818
2018-01-03 57.08191 72.46388 168.9579 1204.20 152.8560 53.95209 167.1653 248.8435 64.80580 49.53787 113.04916 122.50471
2018-01-04 57.29581 73.35766 169.7427 1209.59 154.9552 53.95209 168.3379 252.3213 65.84039 49.69849 113.60599 122.12341
2018-01-05 57.95696 73.26052 171.6753 1229.14 157.4044 54.22336 168.6771 251.0343 66.04156 49.58511 113.93243 121.92322
2018-01-08 57.66527 72.40559 171.0376 1246.87 161.3599 52.97362 168.5608 247.3895 66.26189 49.50008 113.27957 122.52377
2018-01-09 57.53888 73.16336 171.0180 1252.70 161.7486 52.47953 168.1828 249.4821 65.88829 49.31851 113.36599 121.86601
2018-01-10 58.16114 73.49367 170.9788 1254.33 161.2044 54.11680 168.1538 249.8652 64.20229 49.39496 112.78994 122.64770
2018-01-11 58.33614 73.40623 171.9500 1276.68 164.4407 56.69378 168.0375 250.6512 63.20602 49.79631 112.21389 126.37498
2018-01-12 58.72505 74.64975 173.7256 1305.20 165.5097 57.96290 168.2119 252.5178 61.97025 49.55741 112.69392 127.35685
2018-01-16 58.88061 74.91206 172.8427 1304.86 164.5476 57.49788 168.3185 253.9227 63.16770 49.36629 113.15476 125.84116
```

Nous pouvons vectoriser aussi la procédure:

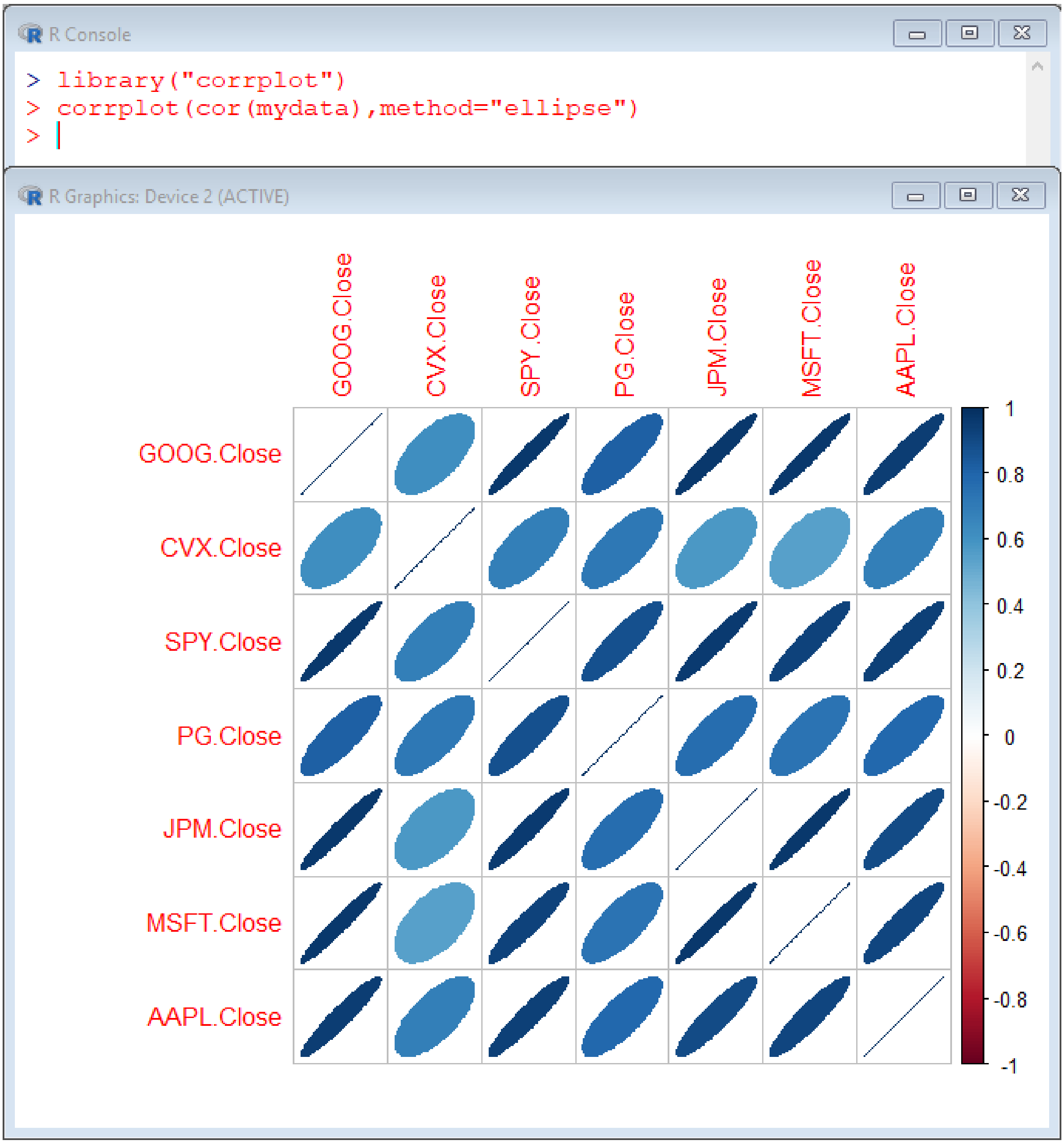


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("quantmod")
> tickers <- c("GOOG","CVX","SPY","PG","JPM","MSFT","AAPL")
> getSymbols(tickers)
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
[1] "GOOG" "CVX"  "SPY"  "PG"   "JPM"  "MSFT" "AAPL"
> fun <- function(i) {
+     return(Cl(get(tickers [i])))
+ }
> mydata <- do.call(merge, lapply(1:length(tickers), fun))
> head(mydata)
           GOOG.Close CVX.Close SPY.Close PG.Close JPM.Close MSFT.Close AAPL.Close
2007-01-03   232.2842     70.97    141.37    64.54     48.07      29.86   11.97143
2007-01-04   240.0686     70.28    141.67    64.05     48.19      29.81   12.23714
2007-01-05   242.0209     70.55    140.54    63.50     47.79      29.64   12.15000
2007-01-08   240.2276     71.45    141.19    63.64     47.95      29.93   12.21000
2007-01-09   241.1814     70.63    141.07    63.48     47.75      29.96   13.22429
2007-01-10   243.1486     69.41    141.54    64.24     48.10      29.66   13.85714
> |
```

Et bien évidemment, rien ne nous empêche de faire un graphique:



Ou une matrice de corrélation:



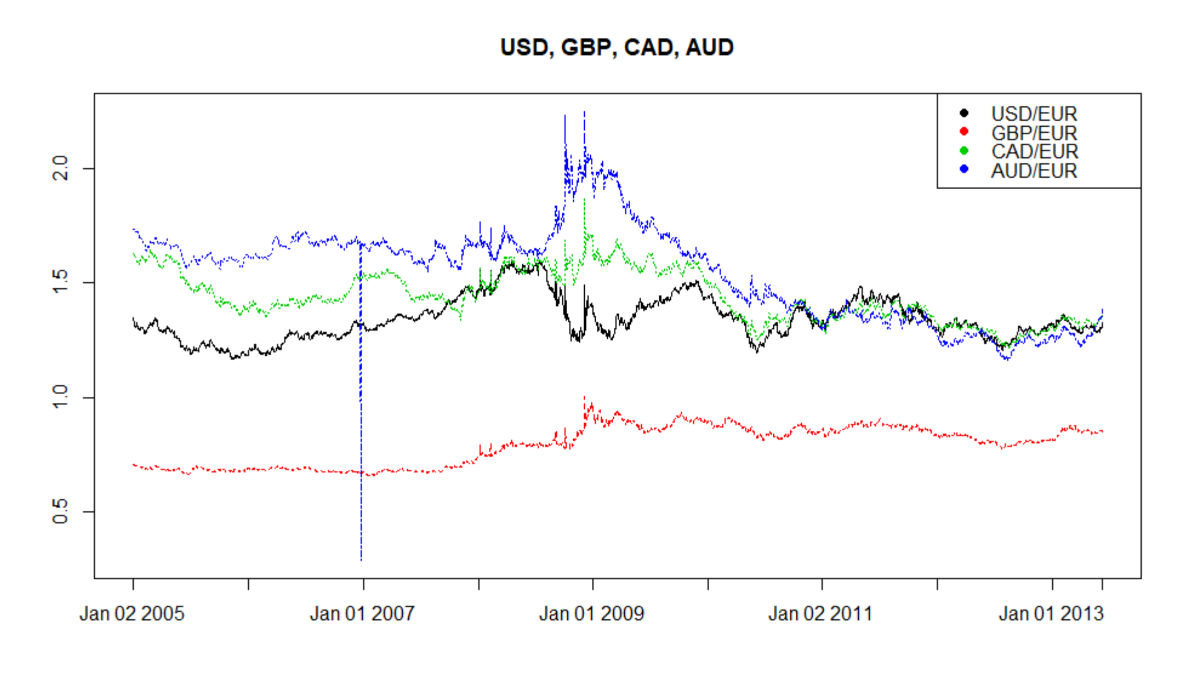
Signalons que nous pouvons aussi obtenir les cours du FOREX par rapport à une monnaie donnée à l'aide du package **quantmod** comme ci-dessous:

```
RGui (64-bit) - [R Console]
> library("quantmod")
> #Marche seulement pour les 180 derniers jours!
> last(getSymbols("GBP/USD",src="oanda", from="2019-01-10",to="2019-03-14",verbose=FALSE,auto.assign=FALSE),10)
            GBP.USD
2019-03-03 1.320812
2019-03-04 1.320898
2019-03-05 1.315972
2019-03-06 1.315070
2019-03-07 1.313595
2019-03-08 1.305868
2019-03-09 1.301565
2019-03-10 1.301193
2019-03-11 1.305700
2019-03-12 1.314172
> 
```

Ou avec le package **Quandl** dans une version un peu plus complexe..:

```
R Console
> library("Quandl")
> library("xts")
> currencies <- c( "USD", "CHF", "GBP", "JPY", "RUB", "CAD", "AUD")
> currencies <- paste("CURRFX/EUR", currencies, sep = "")
> currency_ts <- lapply(as.list(currencies), Quandl, start_date="2005-01-01"
+ ,end_date="2013-06-07", type="xts")
> #on prend que les USD, GBP, CAD et AUD
> Q <- cbind(
+ currency_ts[[1]]$Rate,currency_ts[[3]]$Rate,currency_ts[[6]]$Rate,currency_ts[[7]]$Rate)
>
> matplot(Q, type = "l", xlab = "", ylab = "", main = "USD, GBP, CAD, AUD",xaxt = 'n')
> ticks = axTicksByTime(currency_ts[[1]])
> axis(1, at = ticks, labels = names(ticks))
> legend("topright", legend = c("USD/EUR", "GBP/EUR", "CAD/EUR", "AUD/EUR"),
+ col = 1:4, pch = 19)
> 
```

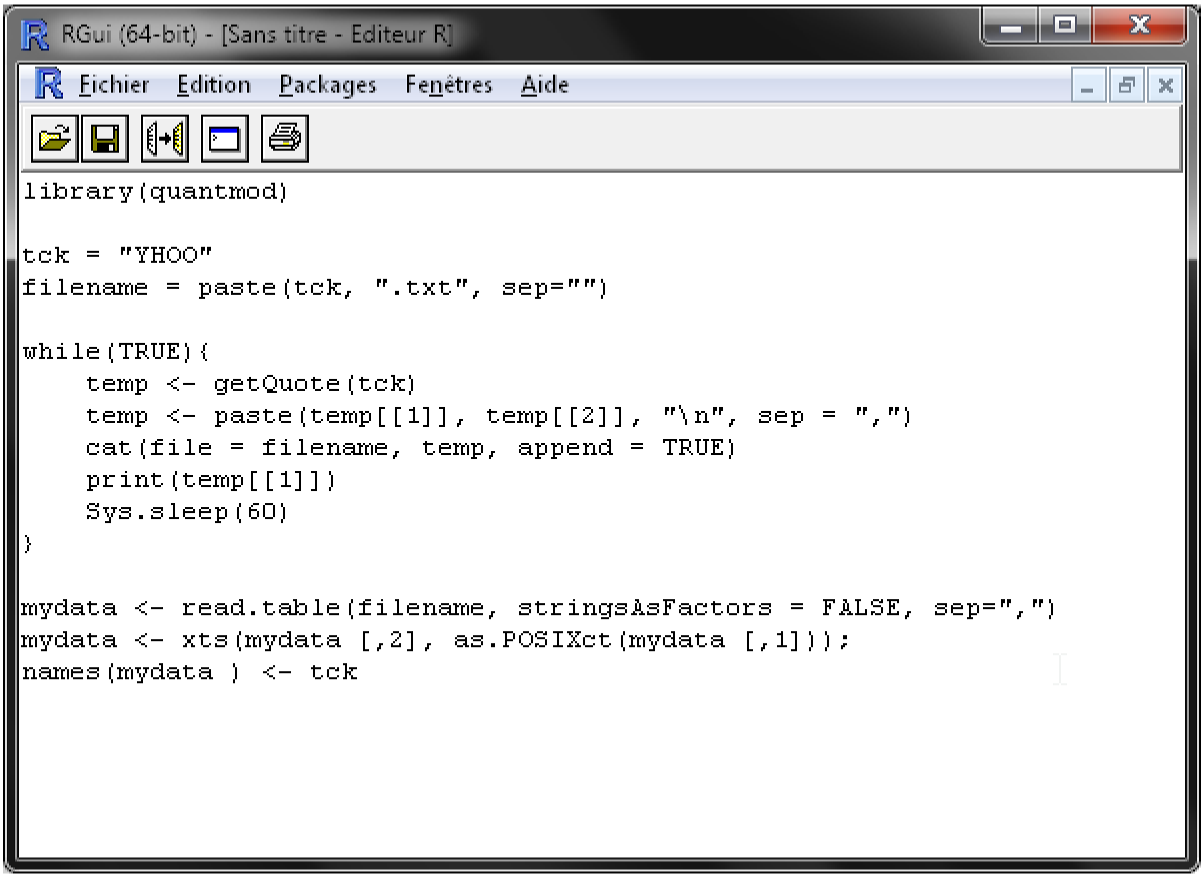
R Graphics: Device 2 (ACTIVE)



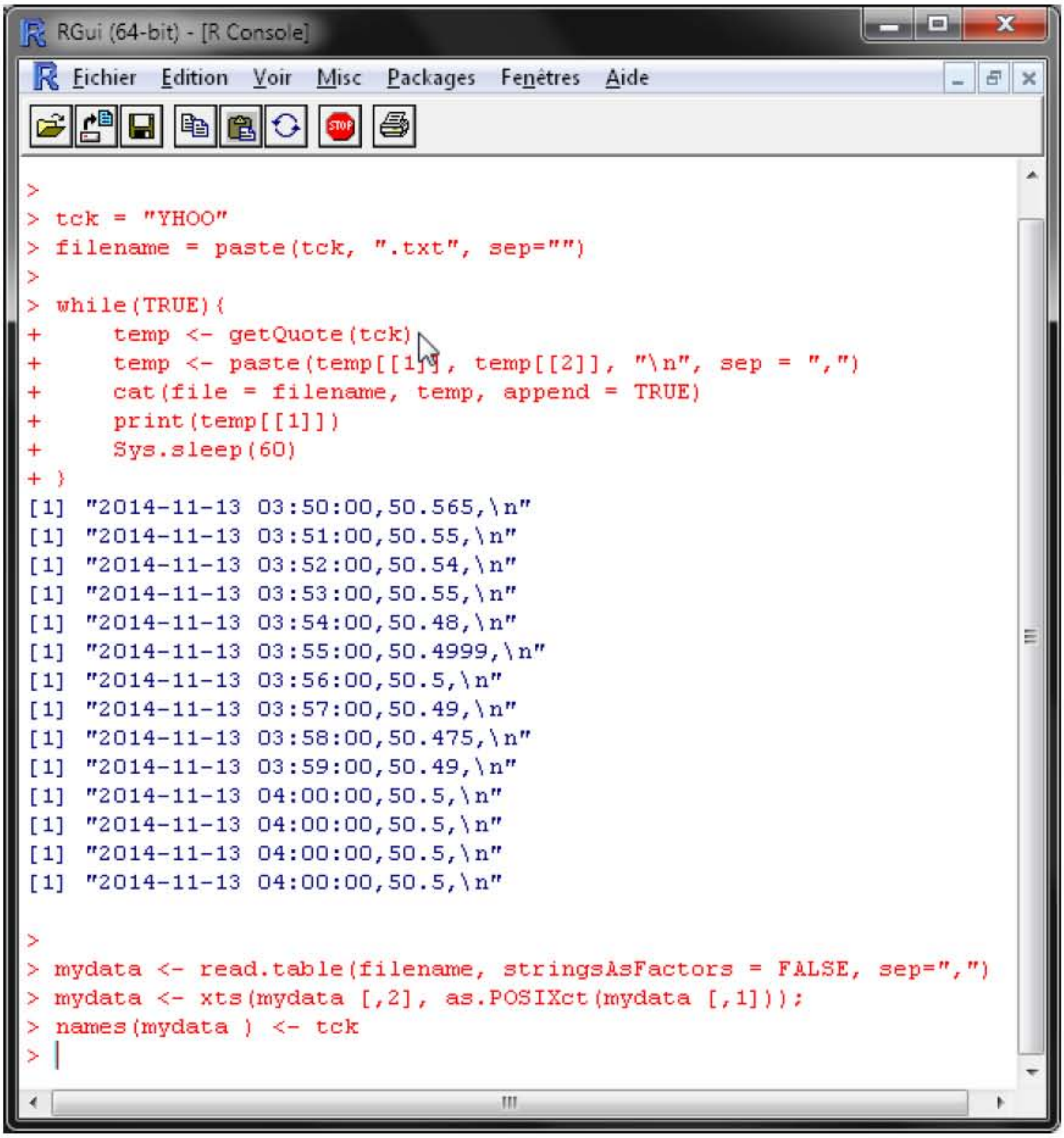
Avec package, automatisé en temps réel actualisé (indices/titres/options)

Voici un script qui tourne (tant qu'il n'y a pas d'erreurs) indéfiniment à stocker dans un fichier *.txt du même nom que l'action, le prix de la demande d'une action à l'aide à nouveau du package **quantmod** et de la fonction **getQuote( )** que nous avons déjà vus plus haut.

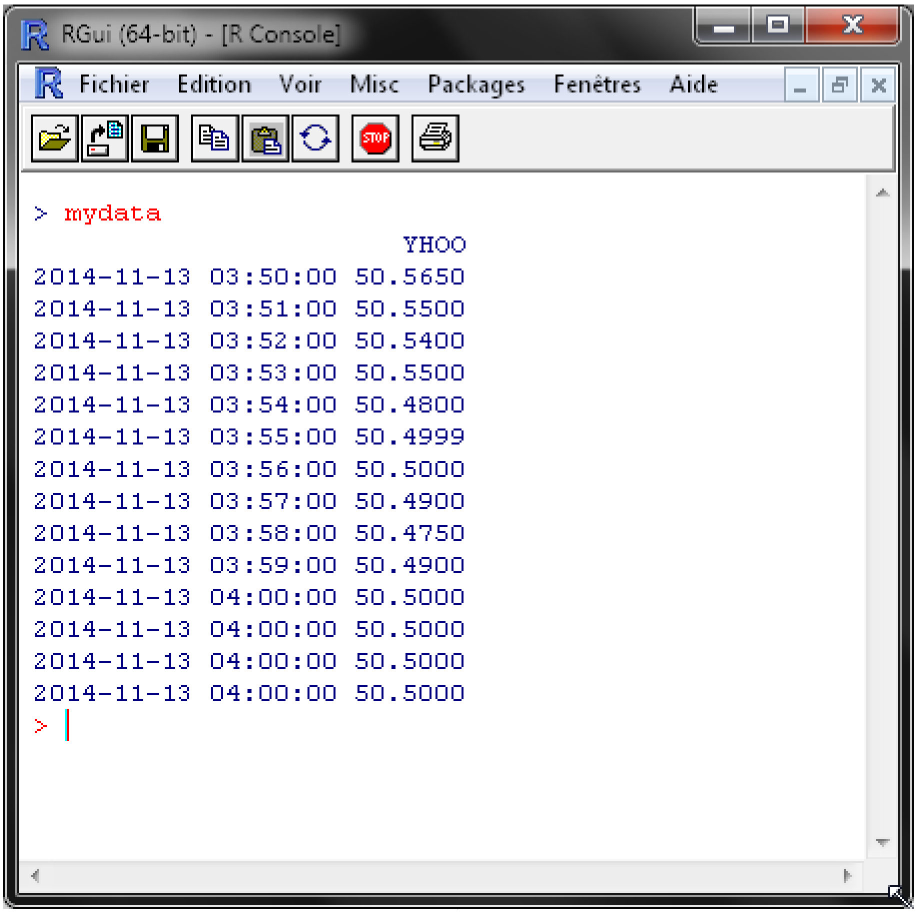
Empiriquement j'ai mis un timer de "1 minute" soit "60 secondes" qui affiche régulièrement l'avancement dans la fenêtre de commande:



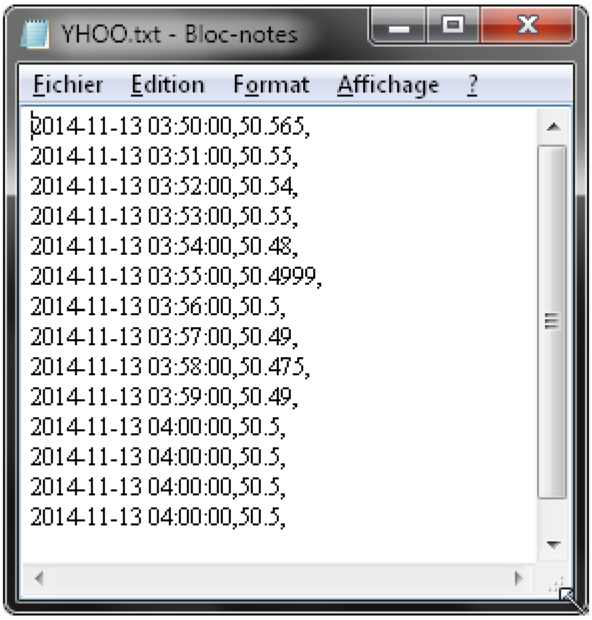
Ce qui donne en quelques minutes:



Soit:



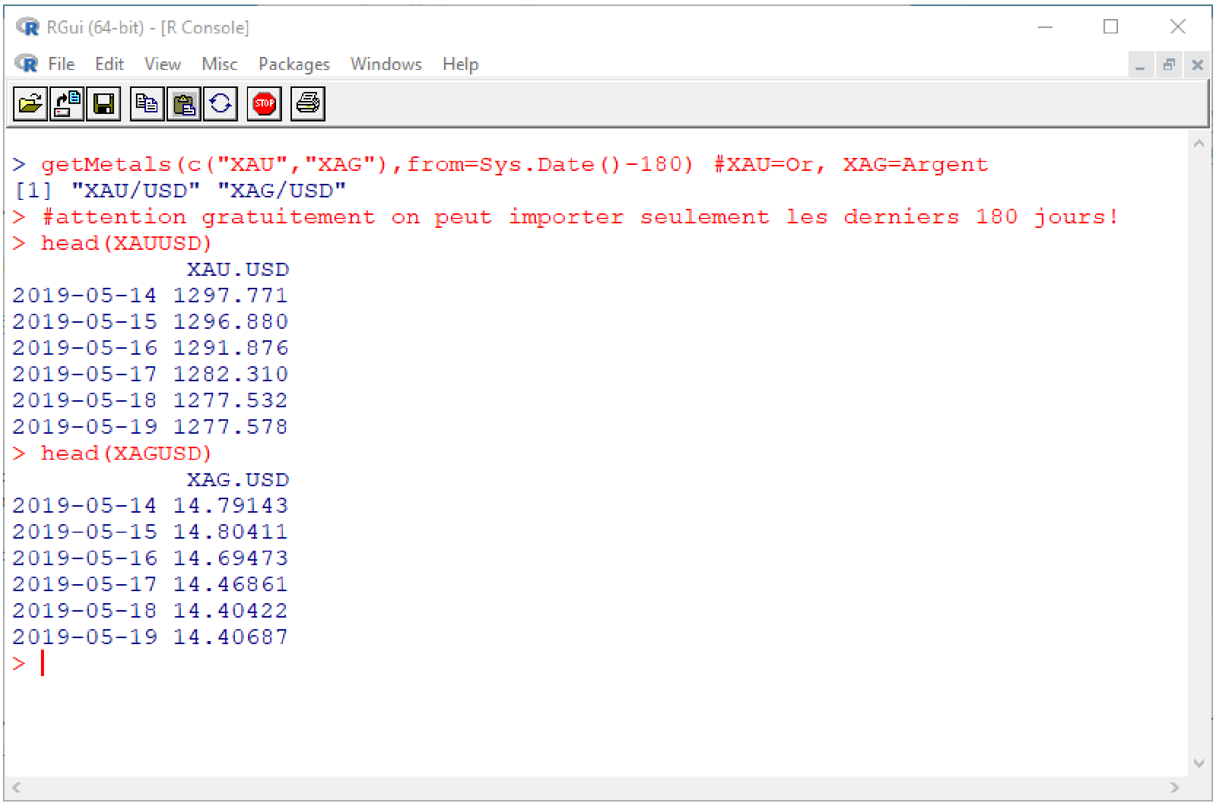
et le fichier texte correspondant:



Avec package, non automatisé différé ou actuel (matières premières)

Le suivi du cours des matières premières est évidemment tout aussi important que les indices/titres ou options.

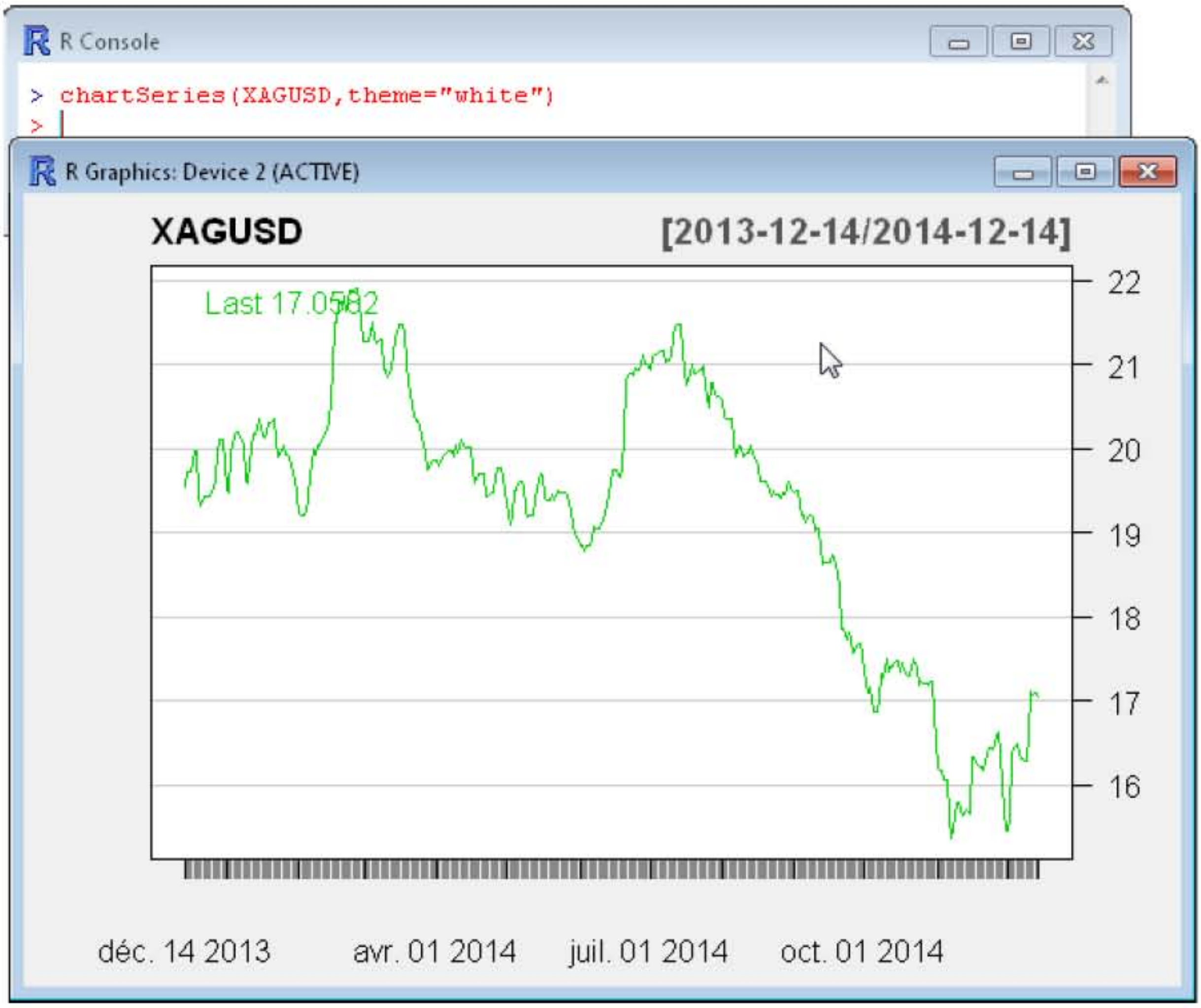
Nous utilisons pour cela la commande **getMetals( )** :



Et sous forme graphique:


R Console
> chartSeries(XAUUSD,theme="white")
R Graphics: Device 2 (ACTIVE)
XAUUSD
[2013-12-14/2014-12-14]
Last 1222.79
1350
1300
1250
1200
1150
déc. 14 2013
avr. 01 2014
juil. 01 2014
oct. 01 2014

et:


R Console
> chartSeries(XAGUSD,theme="white")
R Graphics: Device 2 (ACTIVE)
XAGUSD
[2013-12-14/2014-12-14]
Last 17.0582
22
21
20
19
18
17
16
déc. 14 2013
avr. 01 2014
juil. 01 2014
oct. 01 2014

Volatilité Historique

Nous avons mentionné dans le cours théorique qu'il existe 6 types de volatilité historique (nous avons démontré mathématiquement l'origine uniquement de la première) qui sont pour rappel:

- Volatilité historique Cloture-Cloture:

$$\sigma_{cl} = \sqrt{\frac{N}{n-2}\sum_{i=1}^{n-1}(r_i - \bar{r})^2}$$

où:

$$r_i = \log\left(\frac{C_i}{C_{i-1}}\right)$$

- Volatilité historique de Garman et Klass:

$$\sigma = \sqrt{\frac{N}{n}\sum\left[\frac{1}{2}\left(\log\frac{H_i}{L_i}\right)^2 - (2\log 2 - 1)\left(\log\frac{C_i}{O_i}\right)^2\right]}$$

- Volatilité historique de Parkinson:

$$\sigma = \sqrt{\frac{N}{4n \times \log 2}\sum_{i=1}^{n}\left(\log\frac{H_i}{L_i}\right)^2}$$

- Volatilité historique de Rogers et Satchell:

$$\sigma = \sqrt{\frac{N}{n}\sum\left[\log\frac{H_i}{C_i} \times \log\frac{H_i}{O_i} + \log\frac{L_i}{C_i} \times \log\frac{L_i}{O_i}\right]}$$

- Volatilité historique de Garman-Klass-Yang-Zhang:

$$\sigma = \sqrt{\frac{N}{n}\sum\left[\left(\log\frac{O_i}{C_{i-1}}\right)^2 + \frac{1}{2}\left(\log\frac{H_i}{L_i}\right)^2 - (2 \times \log 2 - 1)\left(\log\frac{C_i}{O_i}\right)^2\right]}$$

- Volatilité historique de Yang de Zhang:

$$\sigma^2 = \sigma_o^2 + k\sigma_c^2 + (1-k)\sigma_{rs}^2$$

$$\sigma_o^2 = \frac{N}{n-1}\sum\left(\log\frac{O_i}{C_{i-1}} - \mu_o\right)^2$$

$$\mu_o = \frac{1}{n}\sum\log\frac{O_i}{C_{i-1}}$$

$$\sigma_c^2 = \frac{N}{n-1}\sum\left(\log\frac{C_i}{O_i} - \mu_c\right)^2$$

$$\mu_c = \frac{1}{n}\sum\log\frac{C_i}{O_i}$$

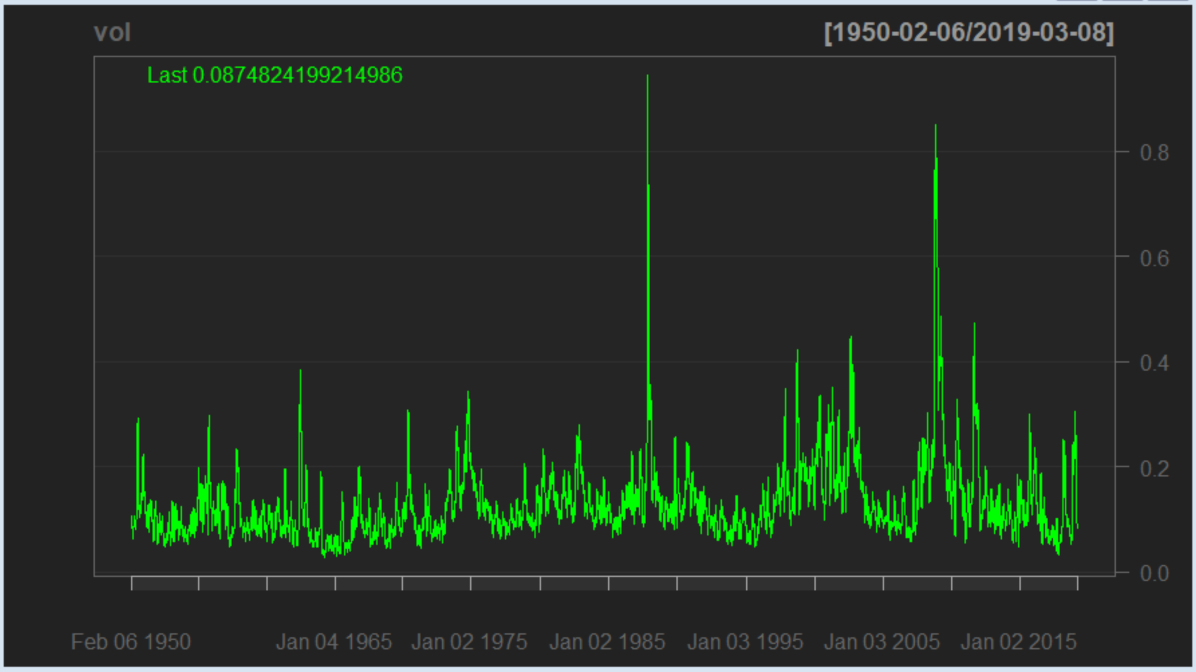
$$\sigma_{rs}^2 = \frac{N}{n}\sum\left(\log\frac{H_i}{C_i}\times\log\frac{H_i}{O_i} + \log\frac{L_i}{C_i}\times\log\frac{L_i}{O_i}\right)$$

$$k = \frac{\alpha - 1}{\alpha + \frac{n+1}{n-1}}$$

-

Voyons d'abord comment obtenir la volatilité historique cloture-cloture avec la fonction **volatility( )** du package **TTR** chargé automatique par **quantmod**:

```
> library("quantmod")
> SP500Quote<-getSymbols.yahoo('^GSPC',from='1950-01-01',verbose=F,auto.assign=F)
> head(SP500Quote)
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
1950-01-03     16.66     16.66    16.66      16.66     1260000         16.66
1950-01-04     16.85     16.85    16.85      16.85     1890000         16.85
1950-01-05     16.93     16.93    16.93      16.93     2550000         16.93
1950-01-06     16.98     16.98    16.98      16.98     2010000         16.98
1950-01-09     17.08     17.08    17.08      17.08     2520000         17.08
1950-01-10     17.03     17.03    17.03      17.03     2160000         17.03
> vol<-volatility(SP500Quote,n=25,N=252,calc="close")
> head(vol)
           [,1]
1950-01-03   NA
1950-01-04   NA
1950-01-05   NA
1950-01-06   NA
1950-01-09   NA
1950-01-10   NA
> chartSeries(vol)
>
```



Et la fonction **volatility( )** peut respectivement prendre comme paramètres calc, les valeurs suivantes: *close*, *garman.klass*, *parkinson*, *rogers.satchell*, *gk.yz*, *yang.zhang*.

Vérifier la Normalité des retours/rendements (return)

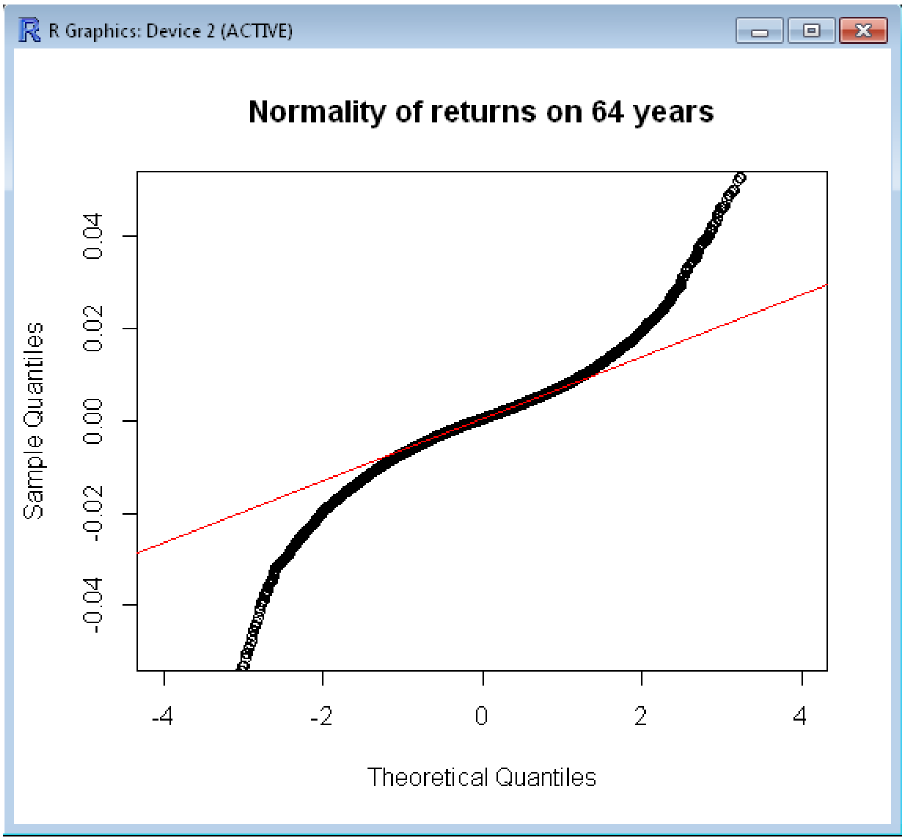
D'abord observons la normalité sur une longue période de l'indice S&P 500:

```
> library(quantmod)
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> x<-SP500Quote$GSPC.Close
> head(x)
           GSPC.Close
1950-01-03      16.66
1950-01-04      16.85
1950-01-05      16.93
1950-01-06      16.98
1950-01-09      17.08
1950-01-10      17.03
> x<-diff(log(x))
> head(x)
             GSPC.Close
1950-01-03           NA
1950-01-04  0.011340020
1950-01-05  0.004736539
1950-01-06  0.002948985
1950-01-09  0.005872007
1950-01-10 -0.002931694
> qqnorm(x[-1,],main="Normality of returns on 64 years",ylim=c(-0.05,0.05))
> qqline(x,col="red")
> 
```

Petit pic de rappel du cours de finance quantitative au passage:

$$\log\left(\frac{p_t}{p_{t-1}}\right)=\log\left(p_t\right)-\log\left(p_{t-1}\right)=\text{diff}\left(\log_i\right)$$

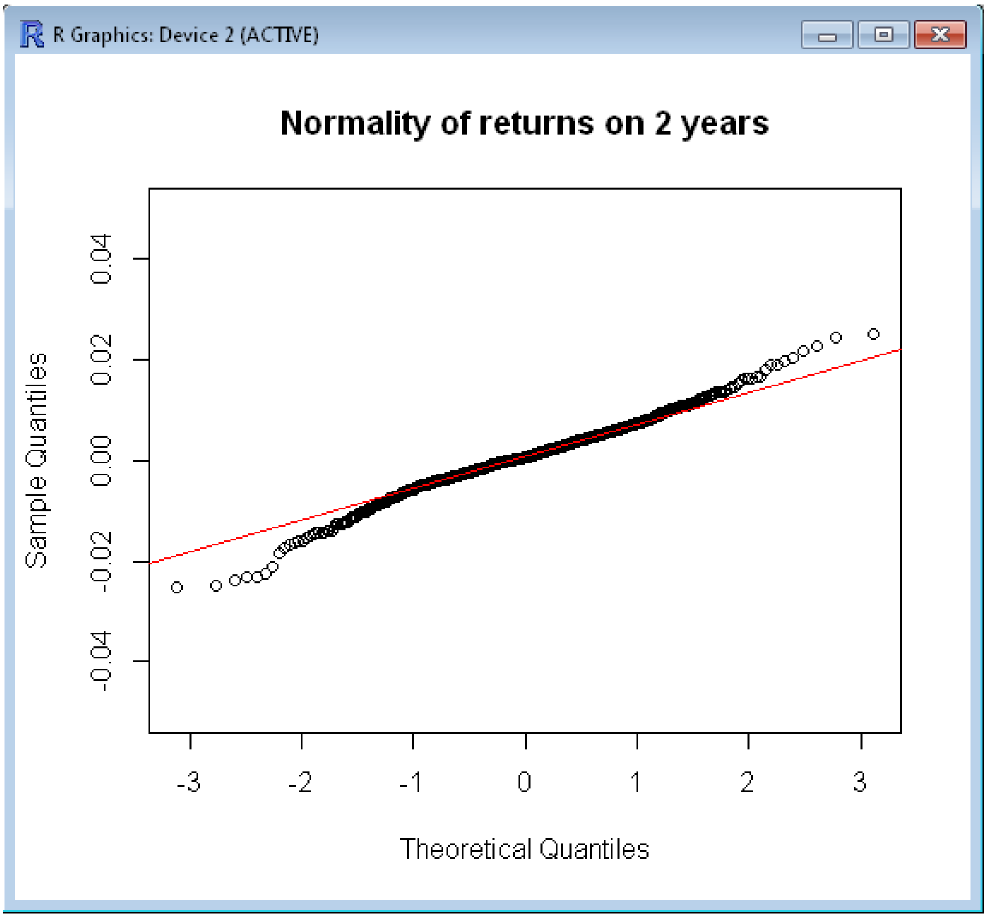
Pas convaincant comme le montre le résultat graphique ci-dessous:



Mais sur une plus courte période (en-dessous d'un certain nombre d'années il y a un décrochage et c'est assez drôle à voir), nous avons:


```
R Console
> library(quantmod)
> SP500Quote<-getSymbols.yahoo("^GSPC",from="2012-01-01",verbose=F,auto.assign=F)
> x<-SP500Quote$GSPC.Close
> head(x)
           GSPC.Close
2012-01-03    1277.06
2012-01-04    1277.30
2012-01-05    1281.06
2012-01-06    1277.81
2012-01-09    1280.70
2012-01-10    1292.08
> x<-diff(log(x))
> head(x)
             GSPC.Close
2012-01-03           NA
2012-01-04  0.000187914
2012-01-05  0.002939385
2012-01-06 -0.002540185
2012-01-09  0.002259128
2012-01-10  0.008846519
> qqnorm(x[-1,],main="Normality of returns on 2 years",ylim=c(-0.05,0.05))
> qqline(x,col="red")
> |
```

Ce qui donne:

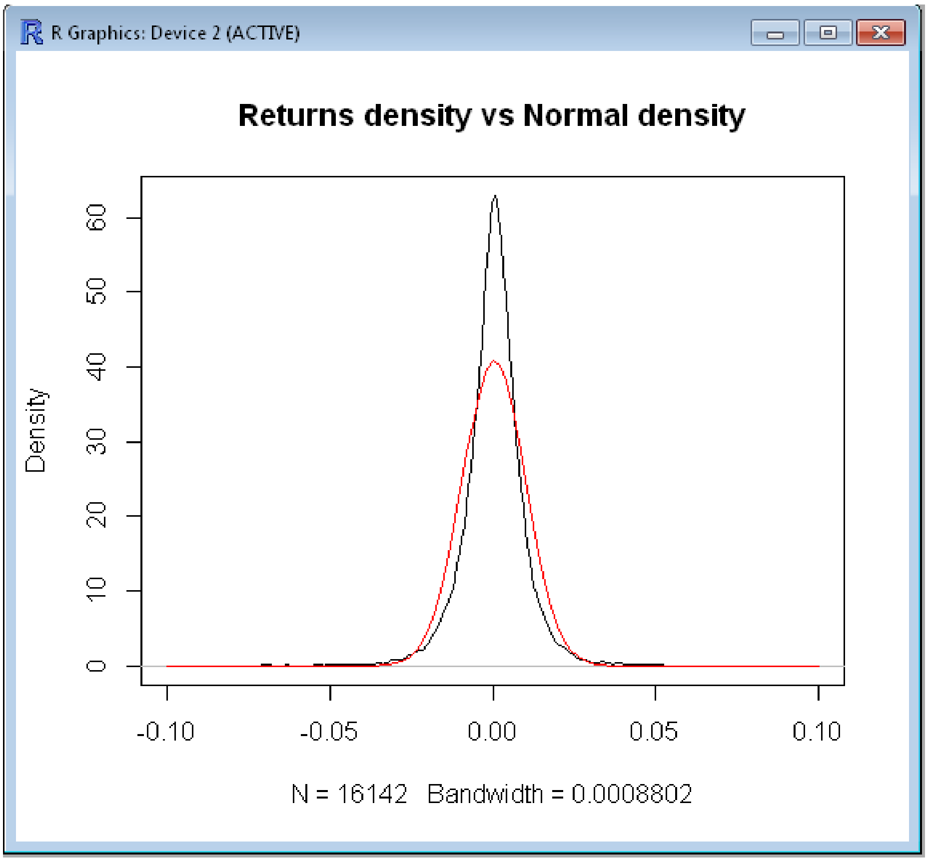


Comparons par rapport à une loi Normale:

R Console

```
> library(quantmod)
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> x<-SP500Quote$GSPC.Close
> x<-diff(log(x))
> plot(density(x[-1,]),main="Returns density vs Normal density",xlim=c(-0.1,0.1))
> m<-mean(x[-1,]);s<-sd(x[-1,])
> curve(dnorm(x,m,s),col="red",add=T)
> |
```

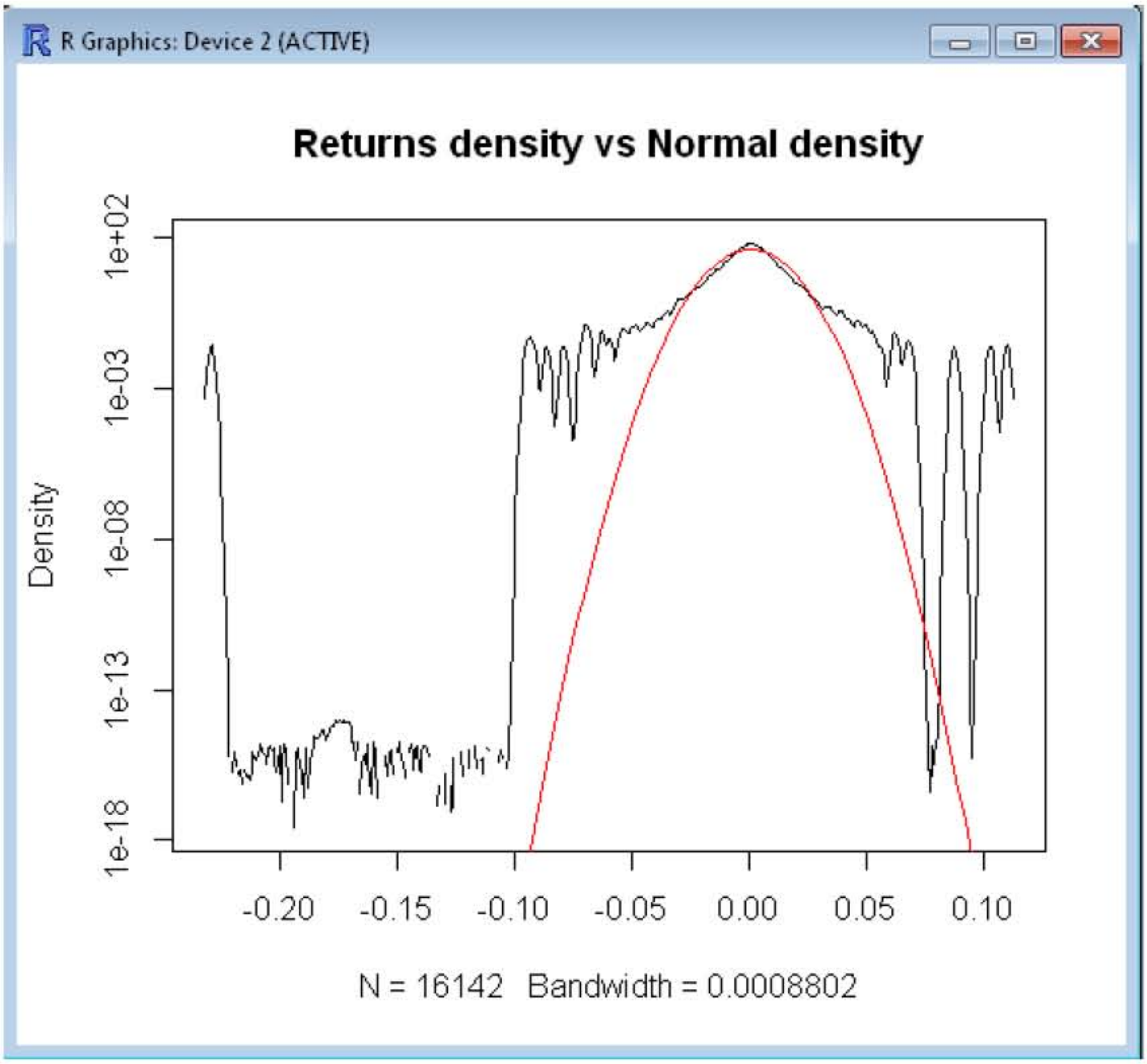
Ce qui donne:



Ce qui est intéressant c'est de plotter sur une échelle logarithmique:

```
> library(quantmod)
> SP500Quote<-getSymbols.yahoo("^GSPC",from="1950-01-01",verbose=F,auto.assign=F)
> x<-SP500Quote$GSPC.Close
> x<-diff(log(x))
> plot(density(x[-1,]),main="Returns density vs Normal density",log="y")
Message d'avis :
In xy.coords(x, y, xlabel, ylabel, log) :
  40 y values <= 0 omitted from logarithmic plot
> m<-mean(x[-1,]);s<-sd(x[-1,])
> curve(dnorm(x,m,s),col="red",add=T)
>
```

Comme quoi le résultat est intéressant:


R Graphics: Device 2 (ACTIVE)
Returns density vs Normal density
Density
1e+02
1e-03
1e-08
1e-13
1e-18
-0.20
-0.15
-0.10
-0.05
0.00
0.05
0.10
N = 16142 Bandwidth = 0.0008802

Heatmap des return mensuels

Ici nous allons utiliser un code R qui montre l'importance d'avoir acquis toutes les notions vues jusqu'à maintenant. Ce code (qui pourrait être amélioré pour devenir une fonction R) étant trop long pour en faire une capture d'écran nous allons directement le mettre en tant que texte brut ci-dessous.

Si jamais voici un aperçu du fichier *.csv utilisé qui provient directement de Yahoo (s'inspirer des exemples vus plus hauts pour importer des données de Yahoo!):

YahooSPY.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
"Date";"Open";"High";"Low";"Close";"Volume";"Adj.Close"
"44";2010-10-29;118.28;118.72;118.07;118.49;144305500;110.71
"45";2010-10-28;119.06;119.11;117.83;118.4;168576000;110.62
"46";2010-10-27;117.89;118.51;117.26;118.38;190024000;110.61
"47";2010-10-26;118.1;118.84;117.87;118.72;158982900;110.92
"48";2010-10-25;119.14;119.76;118.61;118.7;151145700;110.9
"49";2010-10-22;118.31;118.53;118;118.35;108212400;110.58
"50";2010-10-21;118.4;119.09;117.21;118.13;221585500;110.37
"51";2010-10-20;116.94;118.44;116.87;117.87;200051800;110.13
"52";2010-10-19;117.19;117.85;116.02;116.73;280604700;109.06
"53";2010-10-18;117.74;118.67;117.31;118.28;141204800;110.51
"54";2010-10-15;118.28;118.35;116.76;117.7;243705000;109.97
"55";2010-10-14;117.81;118.01;116.72;117.46;217764300;109.75
"56";2010-10-13;117.66;118.55;117.38;117.92;194347200;110.18
"57";2010-10-12;116.27;117.35;115.65;117.01;182210000;109.33
"58";2010-10-11;116.72;116.97;116.25;116.65;103098300;108.99
"59";2010-10-08;116.05;116.86;115.61;116.54;177760100;108.89
"60";2010-10-07;116.5;116.53;115.19;115.89;164860000;108.28
"61";2010-10-06;116.02;116.33;115.56;116.03;148626600;108.41
"62";2010-10-05;114.8;116.32;114.67;116.04;229634100;108.42
"63";2010-10-04;114.37;114.85;113.18;113.75;166153200;106.28
"64";2010-10-01;114.99;115.12;113.93;114.61;174638700;107.08
"65";2010-09-30;115.05;115.79;113.59;114.13;287106700;106.63
"66";2010-09-29;114.38;114.91;114.02;114.47;179665800;106.95
"67";2010-09-28;114.42;115.04;113.18;114.67;209207500;107.14
"68";2010-09-27;114.86;114.99;114.16;114.27;128761800;106.77
"69";2010-09-24;113.75;114.9;113.65;114.82;209671800;107.28
"70";2010-09-23;112.49;113.67;112.18;112.5;202354300;105.11
"71";2010-09-22;113.8;114.44;113.1;113.42;191322400;105.97
"72";2010-09-21;114.3;114.84;113.51;113.98;268389100;106.49
"73";2010-09-20;112.88;114.46;112.52;114.21;214555200;106.71
"74";2010-09-17;113.04;113.15;112.18;112.49;195836900;105.1
"75";2010-09-16;112.73;113.12;112.35;113.05;199962900;105.06
"76";2010-09-15;112.32;113.21;111.98;113.08;168608400;105.09
"77";2010-09-14;112.5;113.29;112.08;112.65;209823600;104.69
```

```
###################################################
# Édouard Tallent @ TaGoMa.Tech, November 2013    #
# QuantCorner @ http://quantcorner.wordpress.com  #
# Source: http://quantcorner.wordpress.com/page/2/#
###################################################
library(lubridate);library(xts);library(quantmod)
library(reshape2);library(ggplot2)
# Create a data object from the file object that actually is an object for the file
containing the data
data<-read.table("C:/YahooSPY.csv",sep=";",header=TRUE,na.strings="NA") # Create
the data object from file
#Analysis working constants (dates are already supposed to be in ISO format in the
first column)
StartYear<-year(data[dim(data)[1],1]);StartMonth<-month(data[dim(data)[1],1])
FinishYear<-year(data[1,1]);FinishMonth<-month(data[1,1])
# Remove rows containing NAs
row.has.NA <- apply(data, 1, function(x){any(is.na(x))})
data <- data[!row.has.NA,]
# Create a xts object from the data object (
ColumnValue<-5 #this is the colum of interest to be compute for the monthly return
data<-as.xts(data[,ColumnValue], order.by=as.Date(data[,1],format="%Y-%m-%d"))
# Compute the monthly log return
ret<-monthlyReturn(data,type="log",leading=TRUE)# Create a matrix mat object from
ret
# The original time series could not starts on a Jan month nor ends on Dec
#(necessary for our 12 witdth rectangle dashboard)
ret<-c(rep(NA,StartMonth-1),ret,rep(NA,12-FinishMonth)) # The original time series
nor starts on a Jan month nor ends on Dec
mat<-matrix(ret*100, ncol=12, byrow=TRUE, dimnames=NULL) # Create a R matrix
year<-seq(from=StartYear, to=FinishYear, by=1) # Vector containing the years
month<-c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec") #
Vector containing the year months
colnames(mat)<-month  # Give columns names
rownames(mat)<-year   # Give rows names
# Transform the data in mat into a long series using the melt() function
molten <- melt(mat) #Transform the series into a long series
# Define the color palette to be used in the hetmap
LtoM <-colorRampPalette(c("red","yellow")) #The spectrum of colors for the lowest
returns
Mid <- "snow3" # Snow3 is the color for the (approximatedly) median value
MtoH <-colorRampPalette(c("lightgreen","darkgreen")) # The spectrum of colors for
the highest values
# Plot the heatmap
hm<-ggplot(data=molten, aes(x=factor(Var2, levels=month.abb),
y=Var1, fill=value)) + geom_raster()  # Draw the heatmap using geom_raster()
hm<-hm+scale_fill_gradient2(low=LtoM(100), mid=Mid, high=MtoH(100)) # Colors,
please!
hm<-hm+labs(fill='Return (%)') # The legend
hm<-hm+geom_text(aes(label=paste(sprintf("%.1f %%", value))), size=4)   # Write the
monthly returns in the heatmap
hm<-hm+scale_y_continuous(breaks=StartYear:FinishYear)   # Deal with the y-axis
hm<-hm+xlab(label=NULL) + ylab(label=NULL) # No axis names
hm<-hm+theme_bw() # No background grey grid
hm<-hm+theme(axis.text.x=element_text(size=10, hjust=0, vjust=0.4,angle=90)) #
Beautify the x-axis
hm<-hm+ggtitle(label="Monthly return Dashboard")  # Add a title
hm
```

Ce qui donne (voir page suivante):

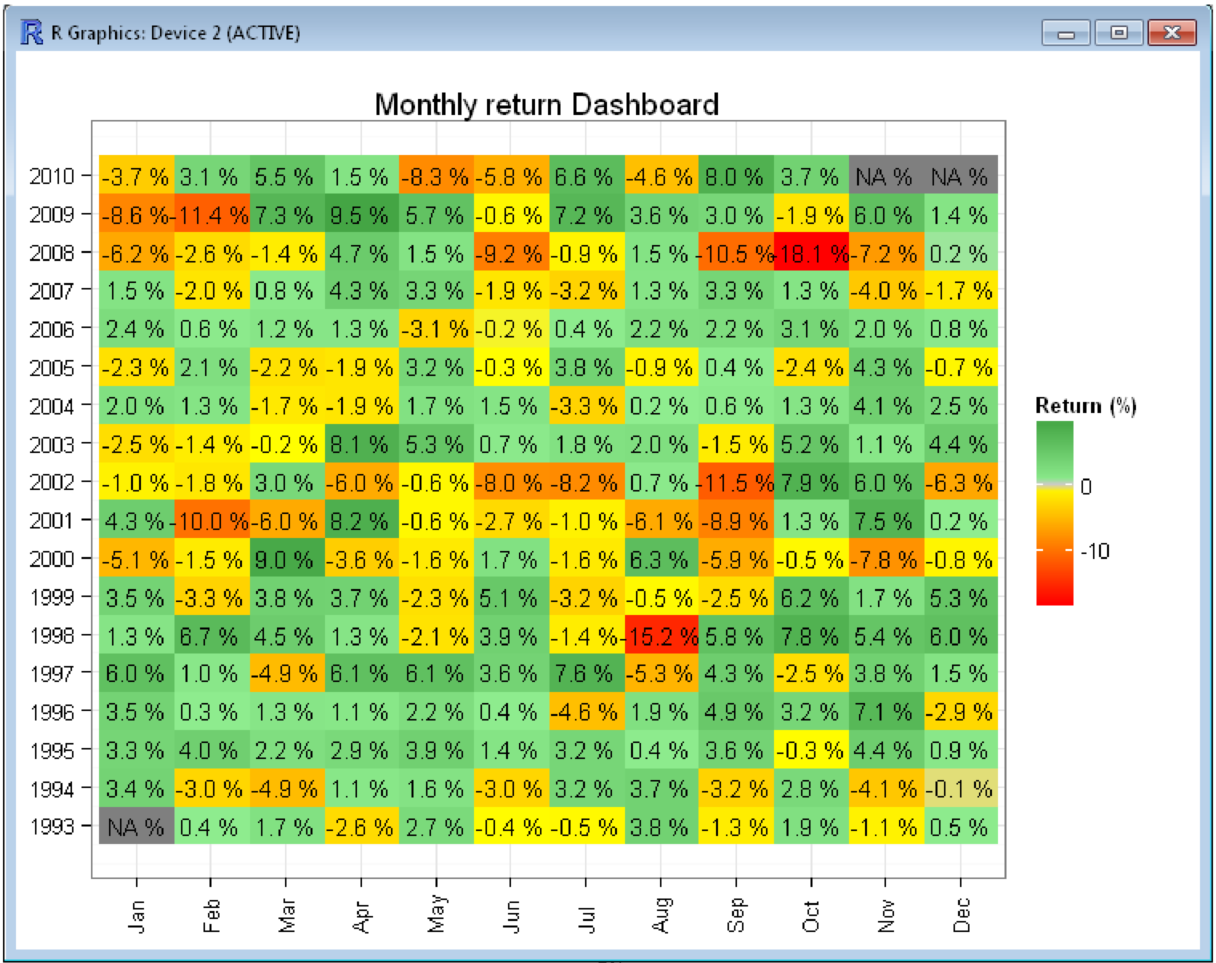
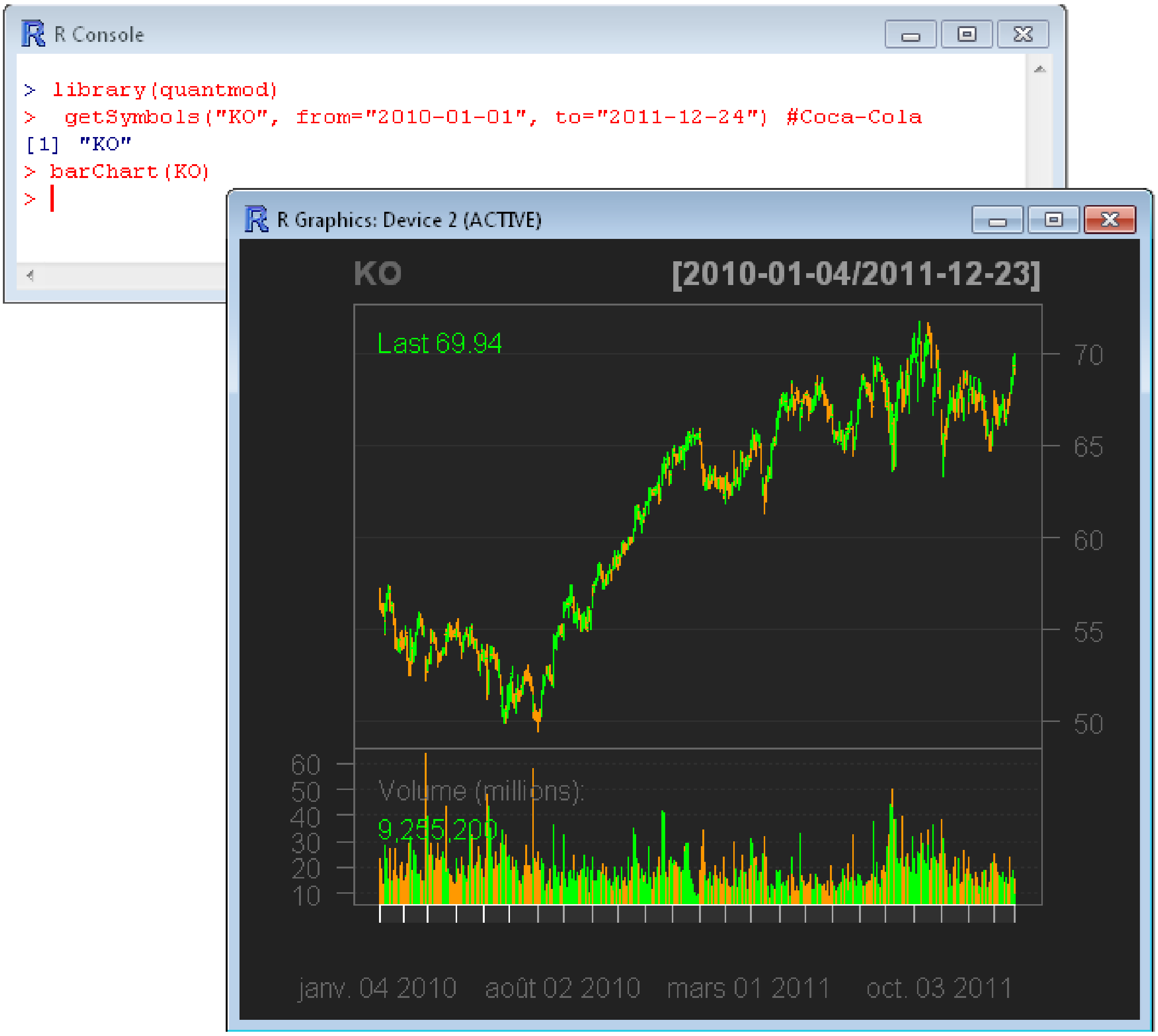


Figure 319 Graph de matrice des rendements sous forme de carte de chaleur

Diagrammes à barres et chandelles (OHLC)

Toujours avec le package **quantmod**, nous avons avec les commandes **getSymbols( )** et **barChart( )** :



Nous ne voyons pas bien les prix OHLC (Open High Low Close). En restreignant la plage de dates c'est mieux:

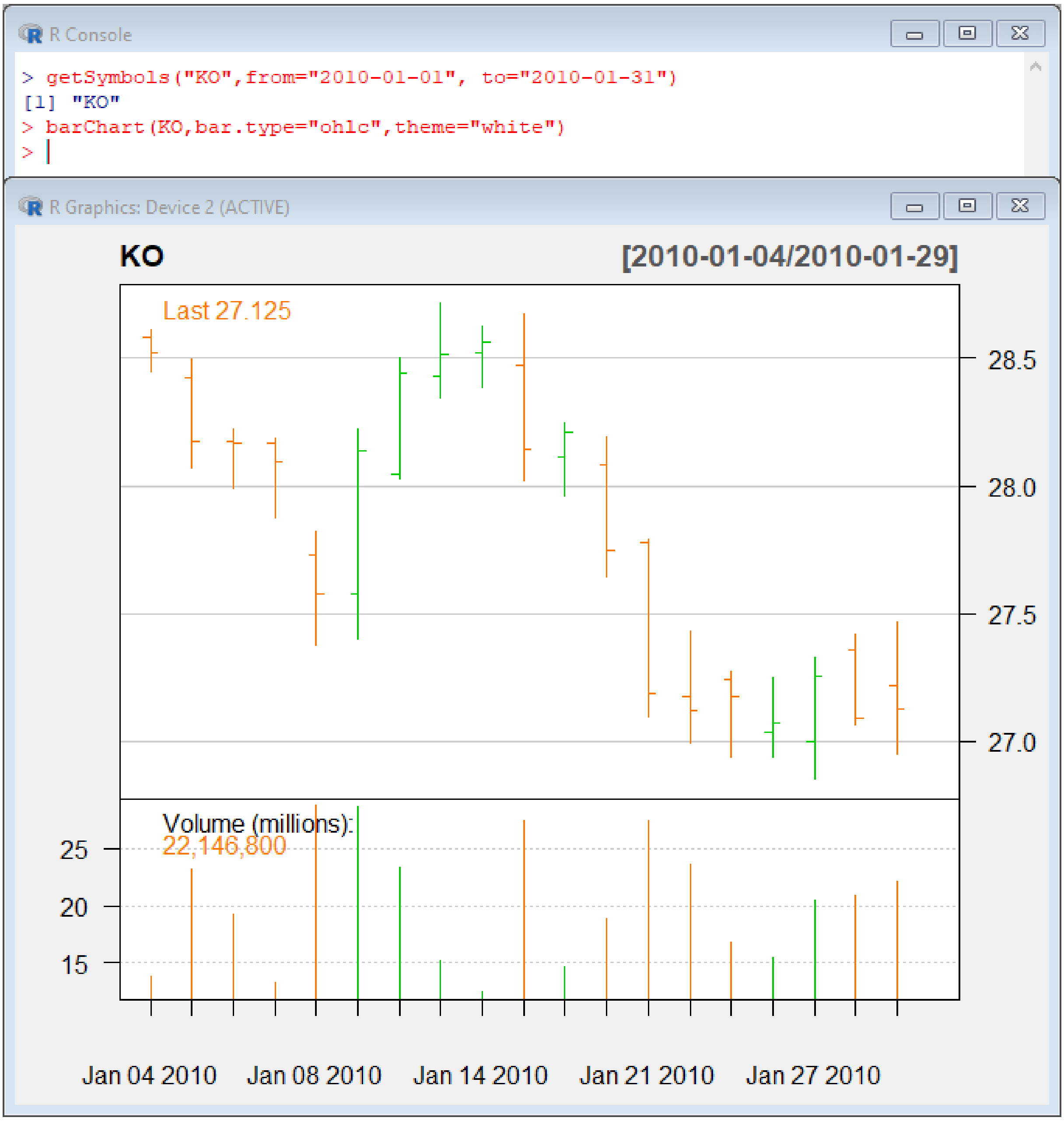
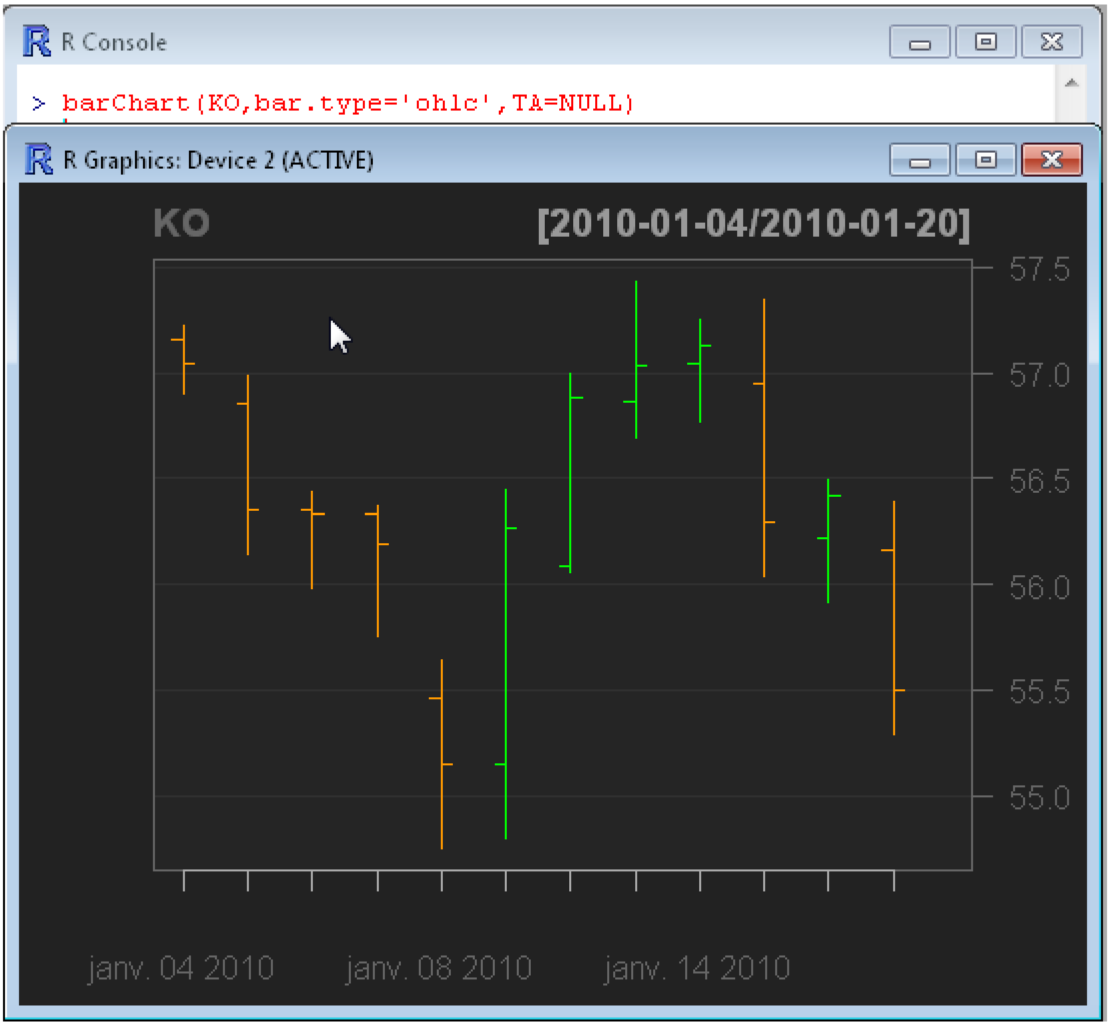


Figure 320 Graph à chandelles et volumétrie (finance)

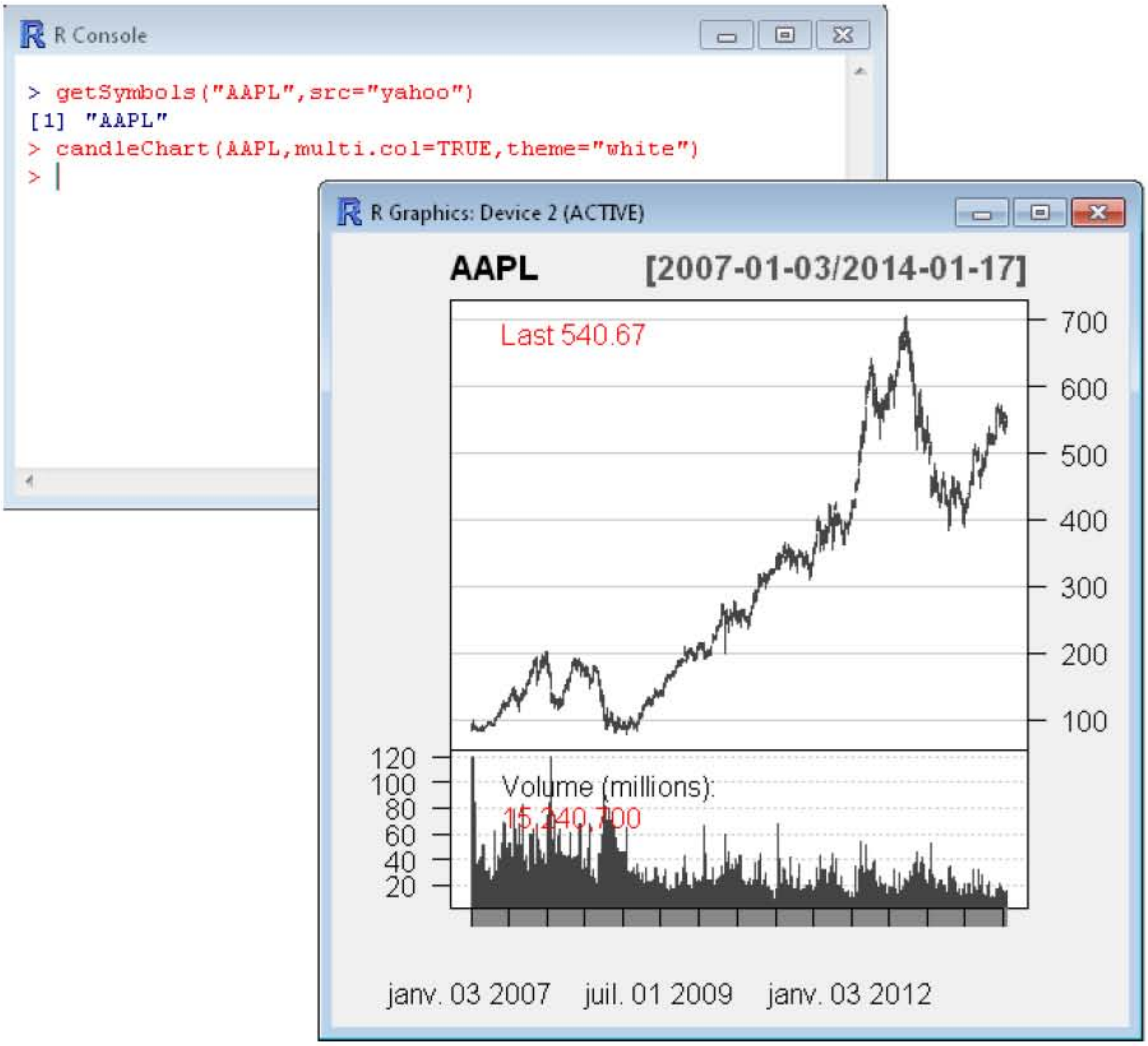
Si on veut enlever la partie concernant les volumes il suffit d'utiliser le paramètre *TA*:



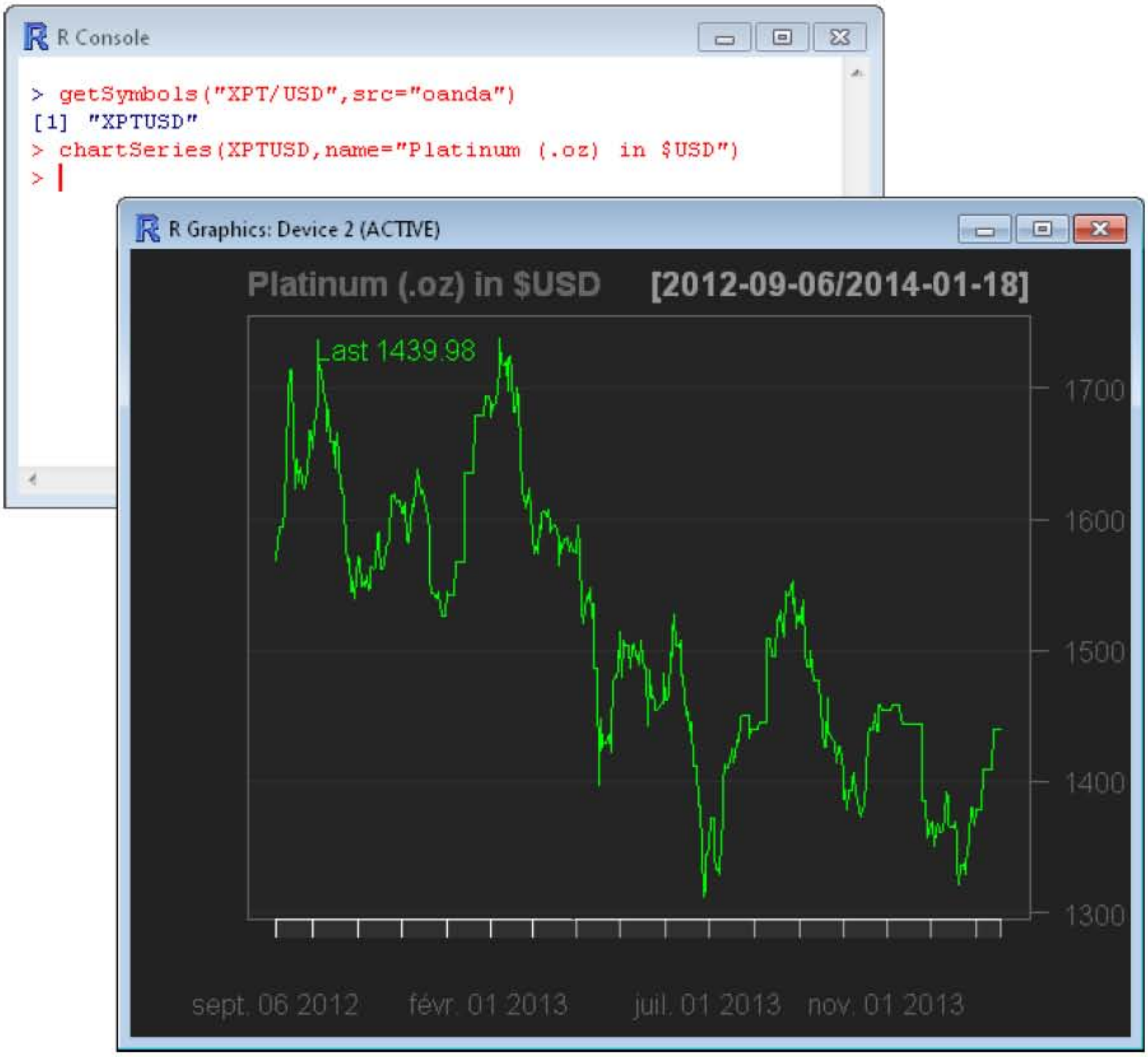
ou avec l'indice S&P 500:



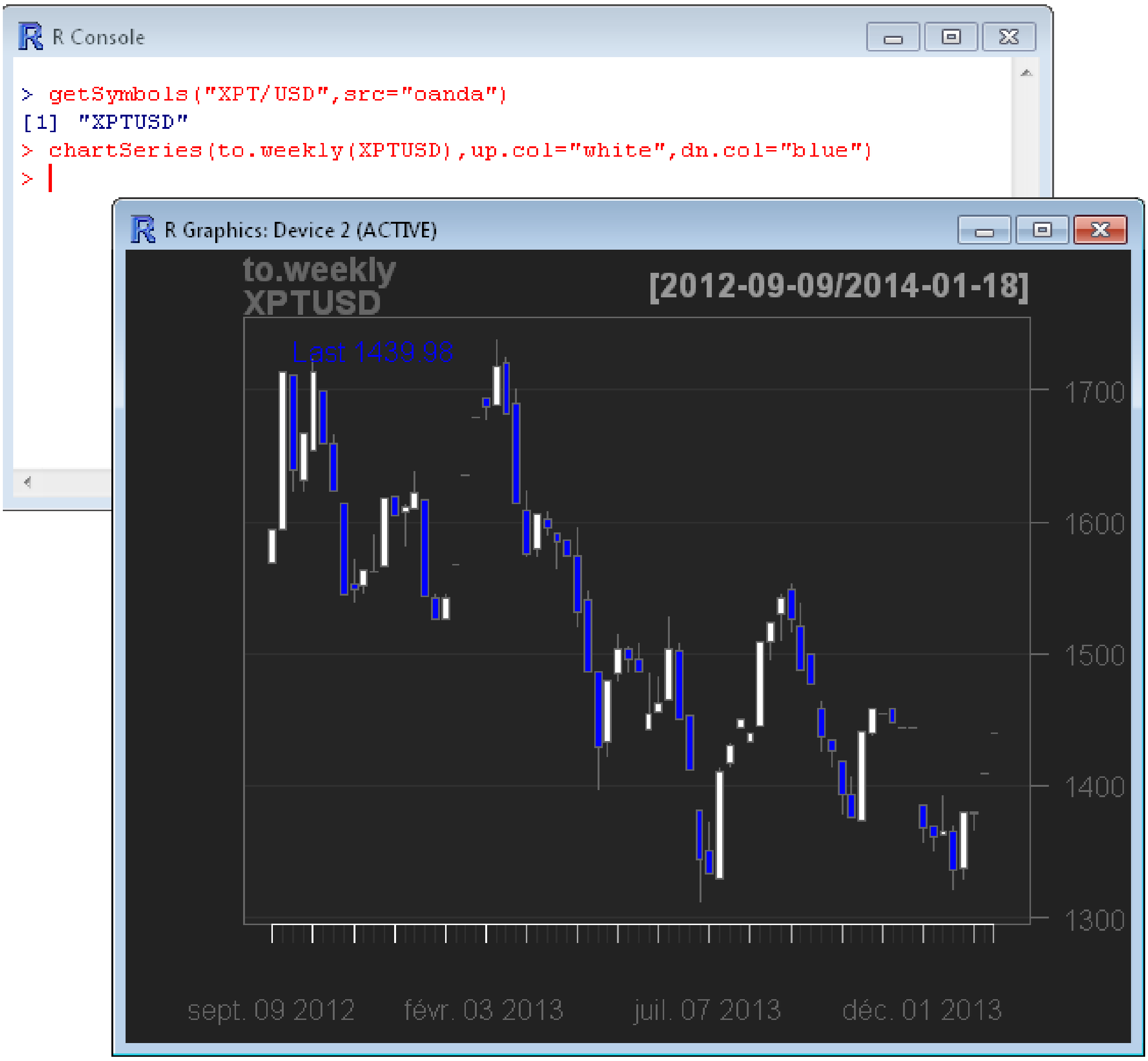
Dans un autre style visuel avec la commande **candleChart( )** :


R Console
> getSymbols("AAPL",src="yahoo")
[1] "AAPL"
> candleChart(AAPL,multi.col=TRUE,theme="white")
R Graphics: Device 2 (ACTIVE)
AAPL
[2007-01-03/2014-01-17]
Last 540.67
Volume (millions):
15,240,700
janv. 03 2007
juil. 01 2009
janv. 03 2012

ou encore:


R Console
> getSymbols("XPT/USD",src="oanda")
[1] "XPTUSD"
> chartSeries(XPTUSD,name="Platinum (.oz) in $USD")
R Graphics: Device 2 (ACTIVE)
Platinum (.oz) in $USD [2012-09-06/2014-01-18]
Last 1439.98
1700
1600
1500
1400
1300
sept. 06 2012
févr. 01 2013
juil. 01 2013
nov. 01 2013

ou encore:



ou encore en ajoutant des indicateurs empiriques à l'aide des commandes **MACD( )** et **addBBand( )** :

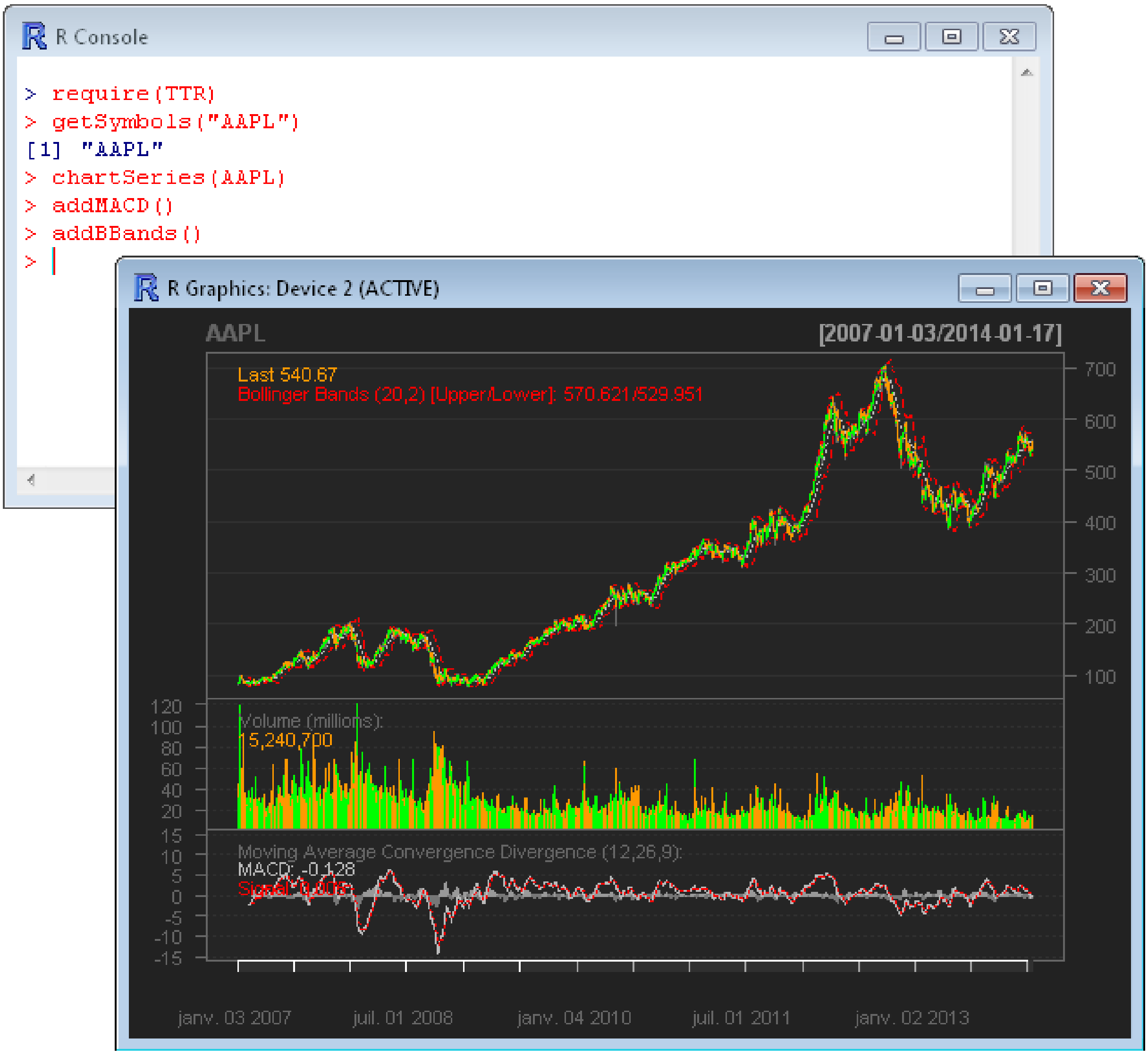


Figure 321 Graph de série temporelle avec bandes de Bollinger, volumétrie et moyenne mobile (finance)

avec un subset c'est mieux:

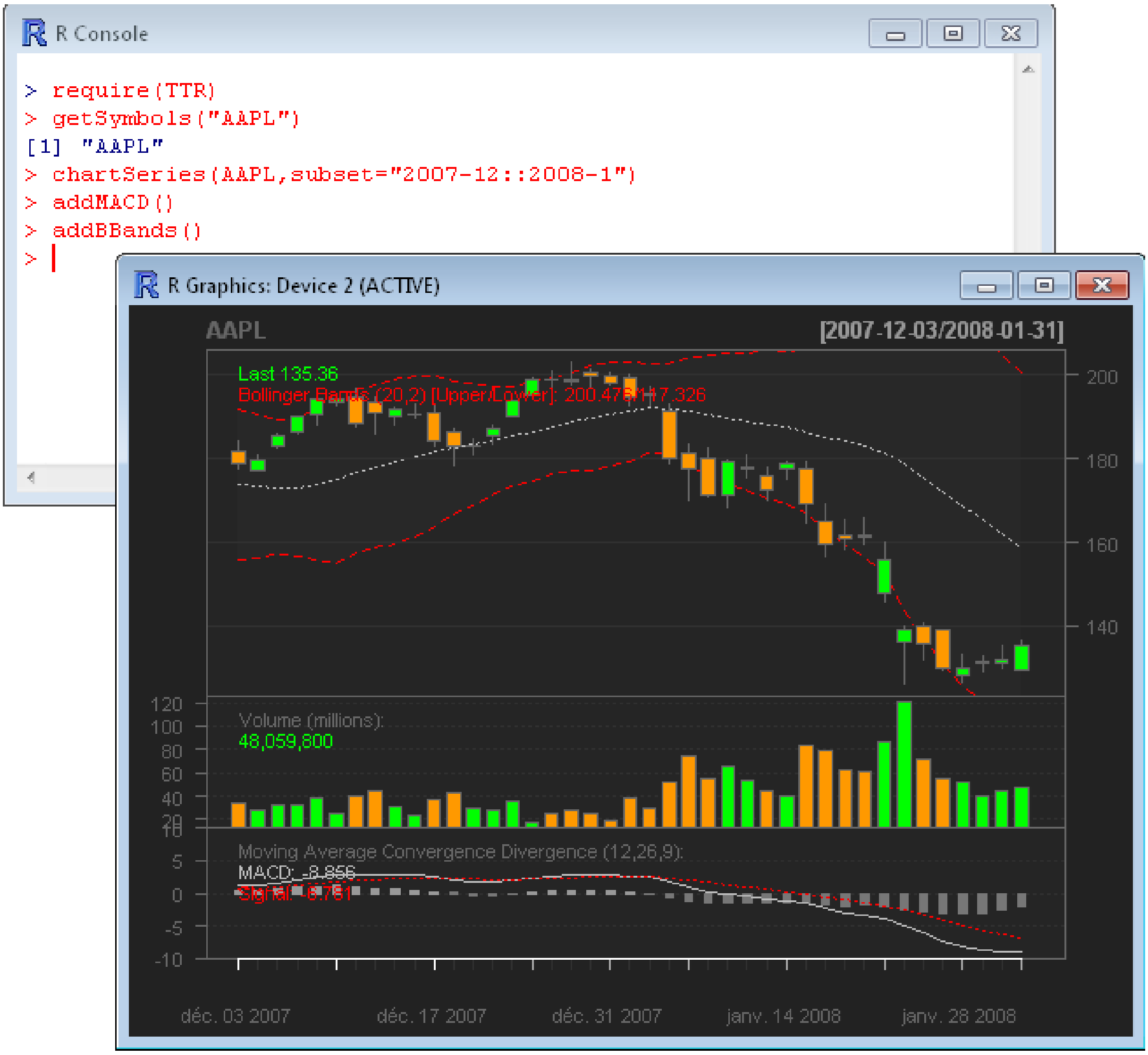
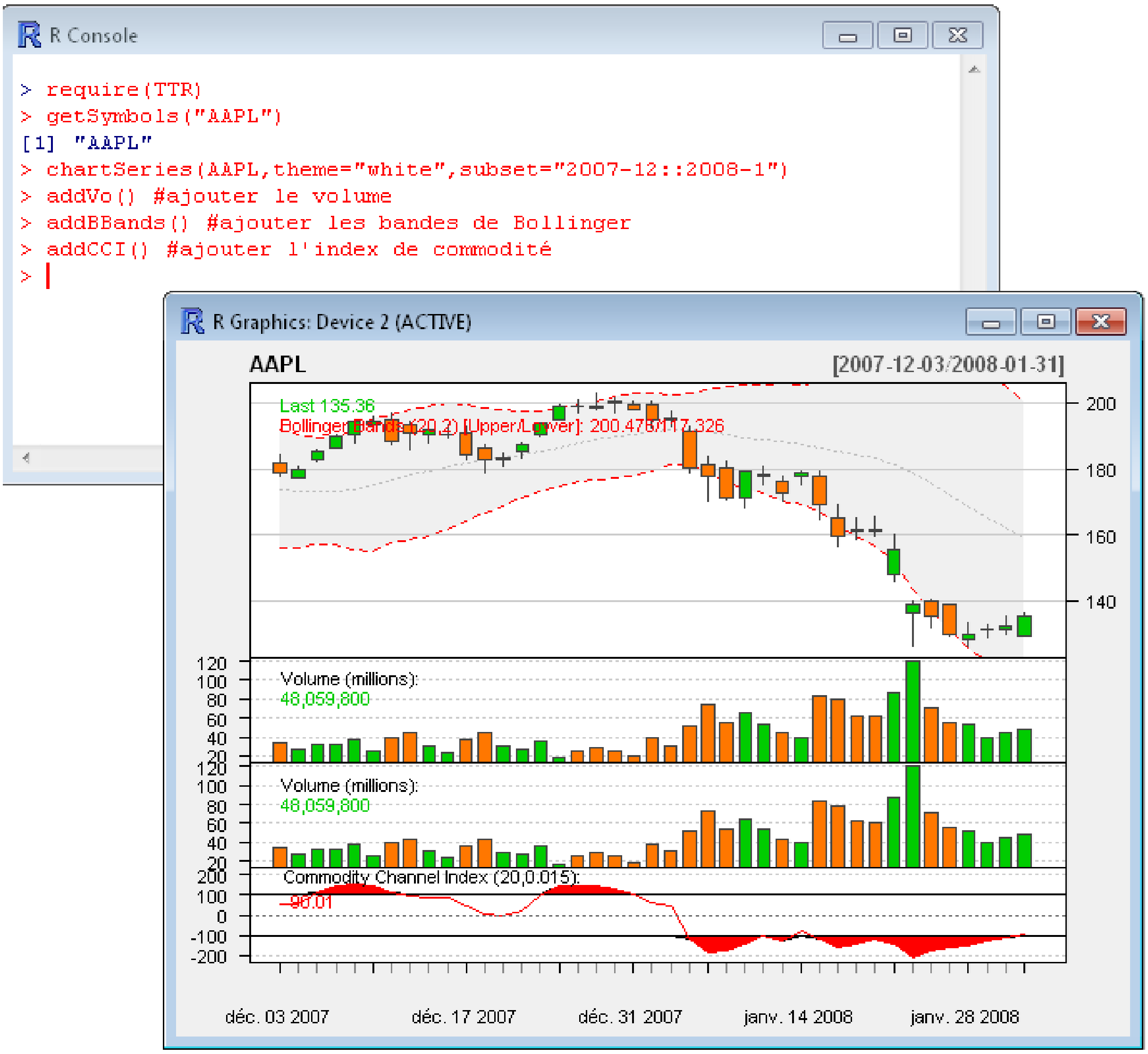
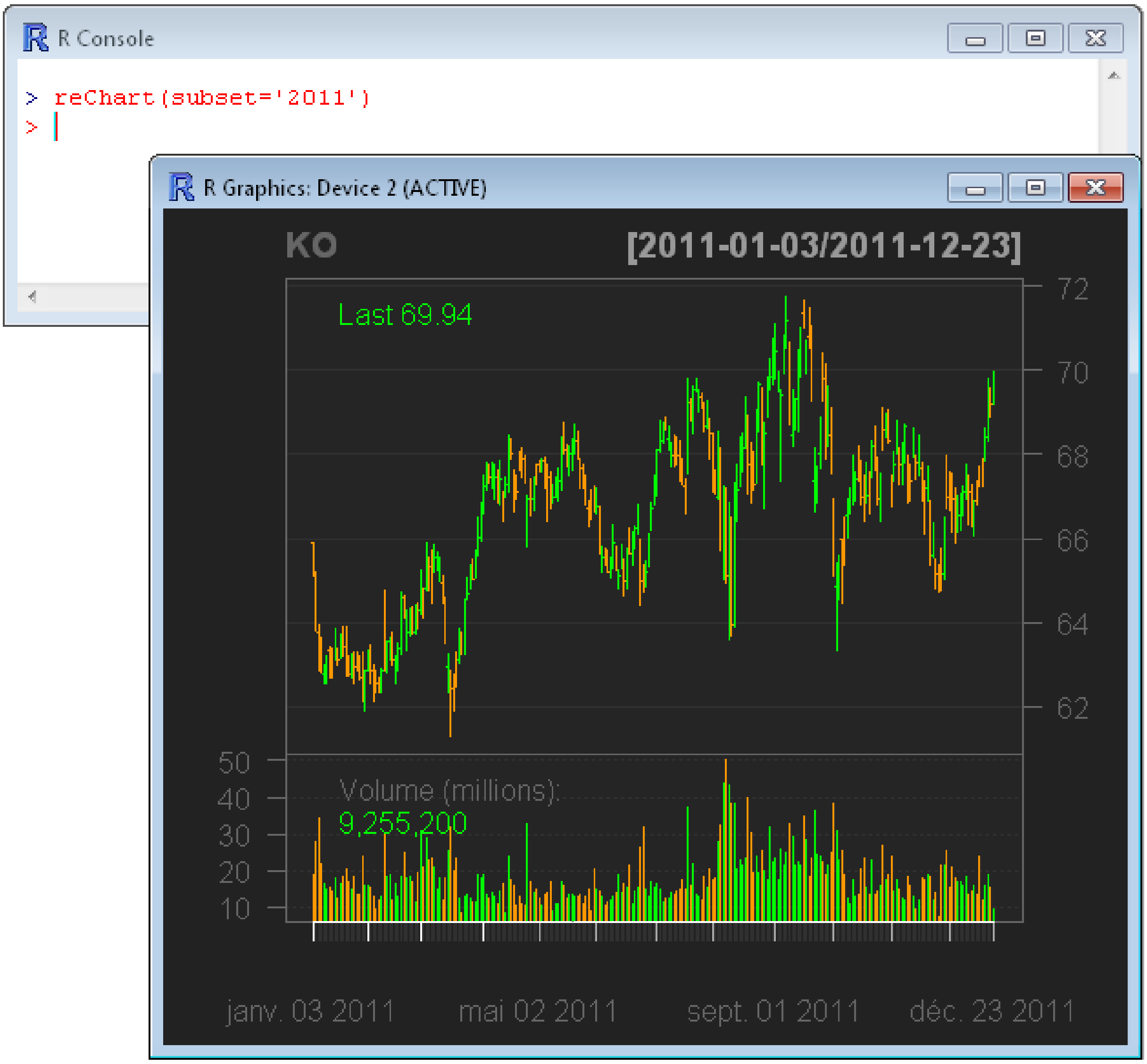


Figure 322 Graph à chandelles avec bandes de Bollinger, volumétrie et moyenne mobile (finance)

ou encore en ajoutant le volume avec la commande **addVo( )** ou l'index de commodité avec la commande **addCCI( )** :



Il est possible de rafraichir le graphique ou de le filtrer avec la commande **reChart( )** :



Pour avoir que février 2011, nous écririons **reChart(subset='201102')**, pour avoir l'année 2010 jusqu'à juin 2011 nous écririons **reChart(subset='2010/201106')**, les dix premières semaines de l'année choisie **reChart(subset='first 10 weeks')** ou les dix derniers mois **reChart(subset='last 10 months')** ou le dernier mois zoomChart('last 1 months')

Voici la liste des différents indicateurs empiriques disponibles à ce jour:

Indicateur	Nom TTR	Nom quantmod
Welles Wilder's Directional Movement Indicator	ADX	addADX
Average True Range	ATR	addATR
Bollinger Bands	BBands	addBBands
Bollinger Band Width	N/A	addBBands
Bollinger %b	N/A	addBBands
Commodity Channel Index	CCI	addCCI
Chaiken Money Flow	CMF	addCMF
Chande Momentum Oscillator	CMO	addCMO
Double Exponential Moving Average	DEMA	addDEMA
Detrended Price Oscillator	DPO	addDPO
Exponential Moving Average	EMA	addEMA
Price Envelope	N/A	addEnvelope
Exponential Volume Weigthed Moving Average	EVWMA	addEVWMA
Options and Futures Expiration	N/A	addExpiry
Moving Average Convergence Divergence	MACD	addMACD
Momentum	momentum	addMomentum
Rate of Change	ROC	addROC
Relative Strength Indicator	RSI	addRSI
Parabolic Stop and Reverse	SAR	addSAR
Simple Moving Average	SMA	addSMA
Stocastic Momentum Index	SMI	addSMI
Triple Smoothed Exponential Oscillator	TRIX	addTRIX
Volume	N/A	addVo
Weighted Moving Average	WMA	addWMA
Williams %R	WPR	addWPR
ZLEMA	ZLEMA	addZLEMA

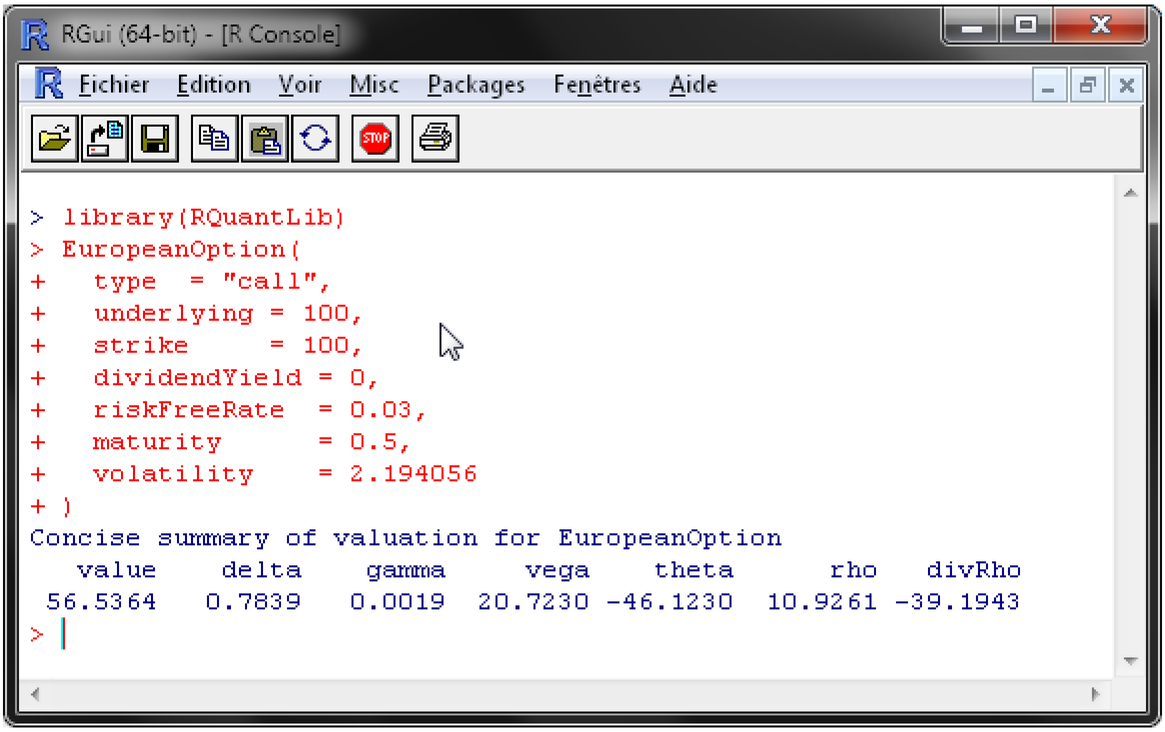
Exemples 400.: Options vanilla et exotiques

R 3.0.2

Évaluation d'options Vanilla Européennes ou Américaines, avec grecques et volatilité implicite (VI) selon B&S-Merton (BSM)

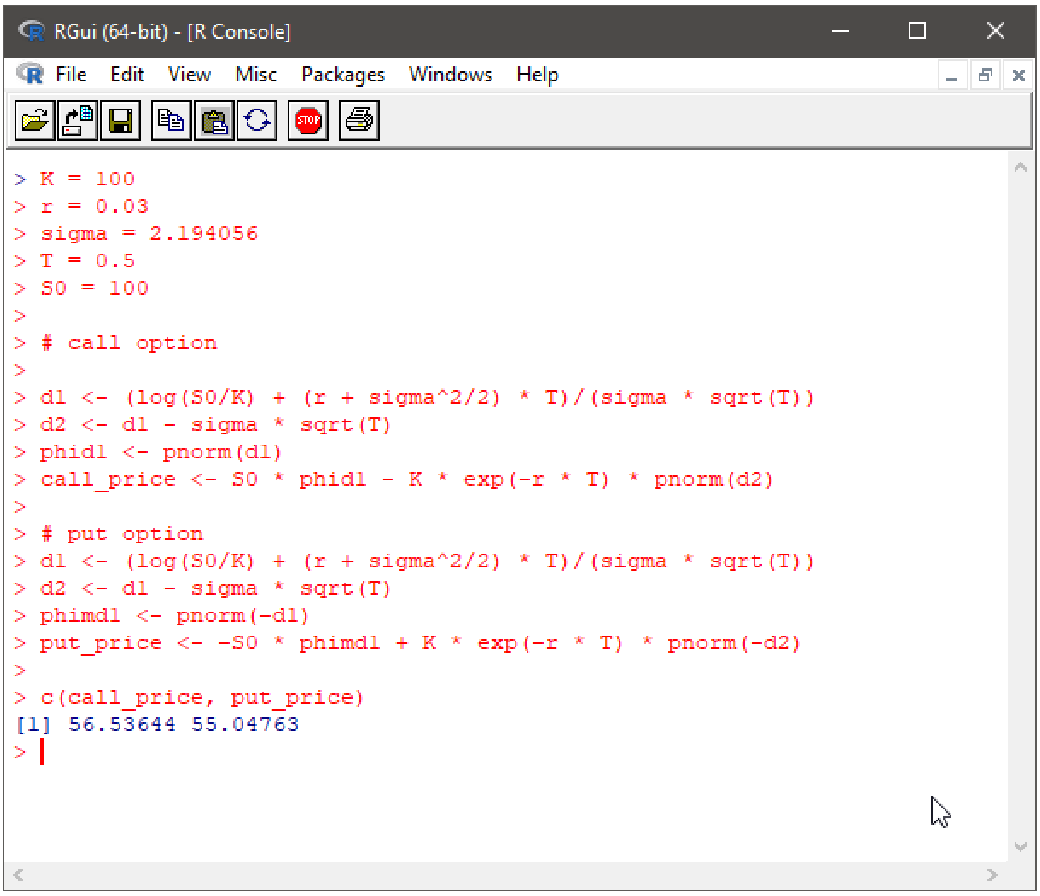
Valuation (pricing) d'options européennes, américaines et calculs des grecques

Pricing et grecques d'une option européenne de type Call (car ce sont avec les options Put les seuls types d'options dont nous avons fait les démonstrations mathématiques détaillées dans le cours théorique), nous utilisons la commande **EuropeanOption( )** ou **AmericanOption( )** du package **RQuantLib** qui utiliseraient les relations fermées du modèle de Black & Scholes:



Si nous comparons le résultat avec le code VBA que nous avions écrit dans MS Excel, nous voyons que nous avons pour les mêmes valeurs d'entrées la même valeur de sortie.

Sinon nous pouvons calculer facilement nous-même explicitement:

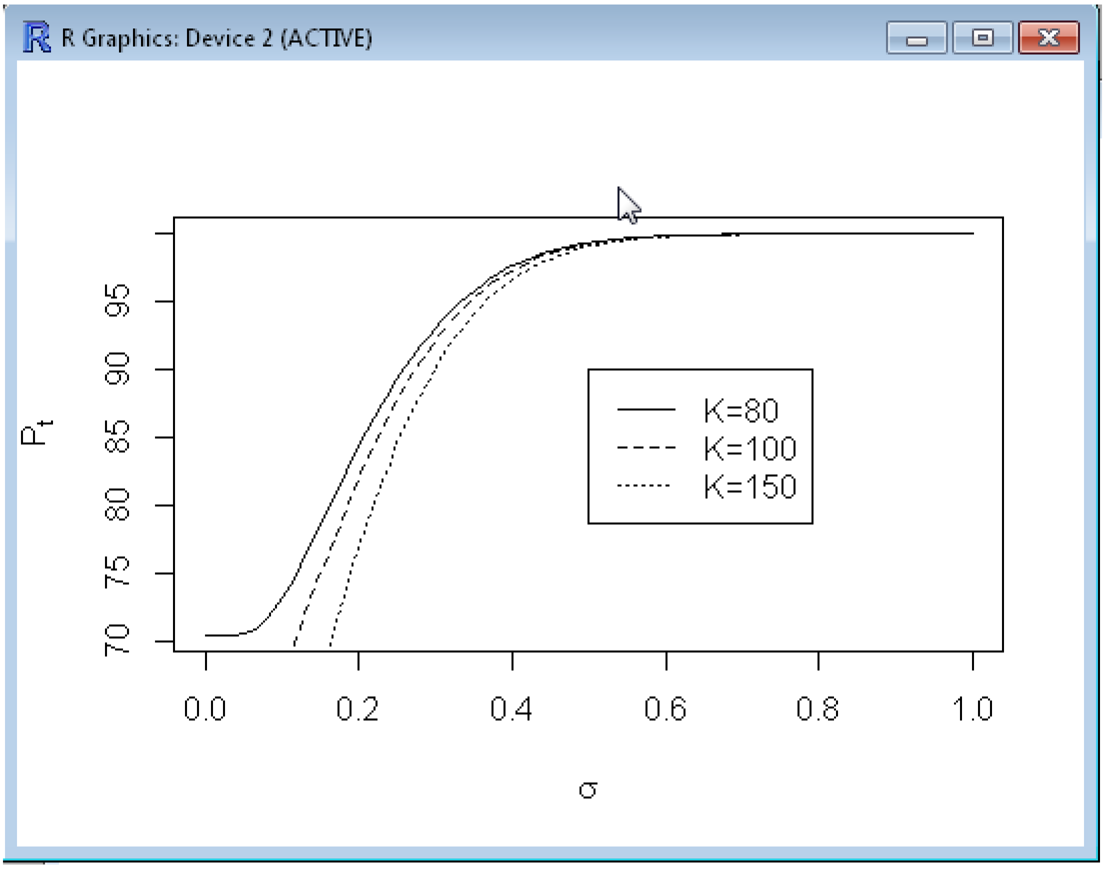


Remarque: Nous n'avons pas démontré dans le cours théorique, ni n'avons fourni de référence bibliographique contenant la mathématique détaillée du modèle de BSM avec sauts (jump) de type événements rares de Poisson, nous ne nous y arrêtons donc pas (même si R contient les packages qu'il faut pour ce type de modèles)!

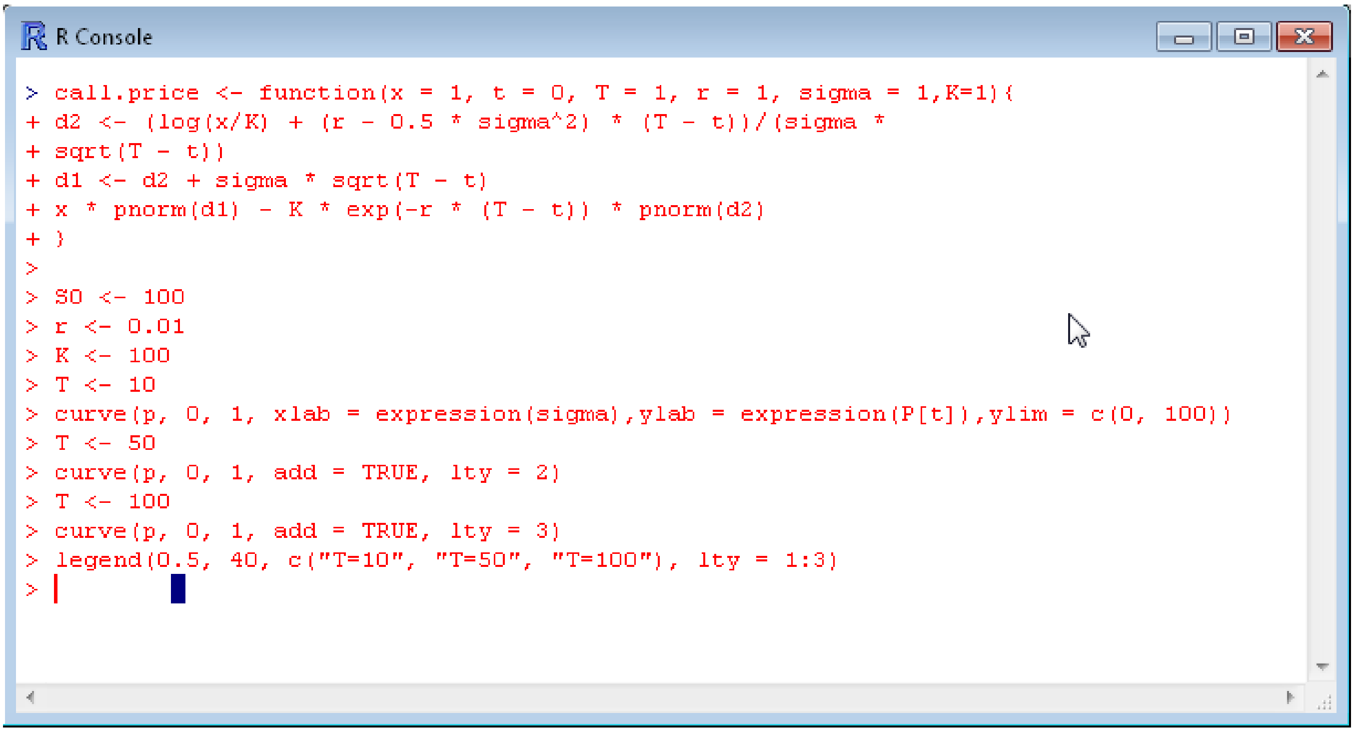
Voyons maintenant que le prix d'un Call pour une volatilité donnée est bien une fonction décroissante du strike (ce qui est assez logique):

```
> call.price <- function(x = 1, t = 0, T = 1, r = 1, sigma = 1,K=1){
+ d2 <- (log(x/K) + (r - 0.5 * sigma^2) * (T - t))/(sigma *
+ sqrt(T - t))
+ d1 <- d2 + sigma * sqrt(T - t)
+ x * pnorm(d1) - K * exp(-r * (T - t)) * pnorm(d2)
+ }
>
> S0 <- 100
> r <- 0.01
> T <- 100
> p <- function(sigma) call.price(x = S0, t = 0, T = T, r = r, K = K, sigma = sigma)
> K <- 80
> curve(p, 0, 1, xlab = expression(sigma),ylab = expression(P[t]))
> K <- 100
> curve(p, 0, 1, add = TRUE, lty = 2)
> K <- 150
> curve(p, 0, 1, add = TRUE, lty = 3)
> legend(0.5, 90, c("K=80", "K=100", "K=150"), lty = 1:3)
>
```

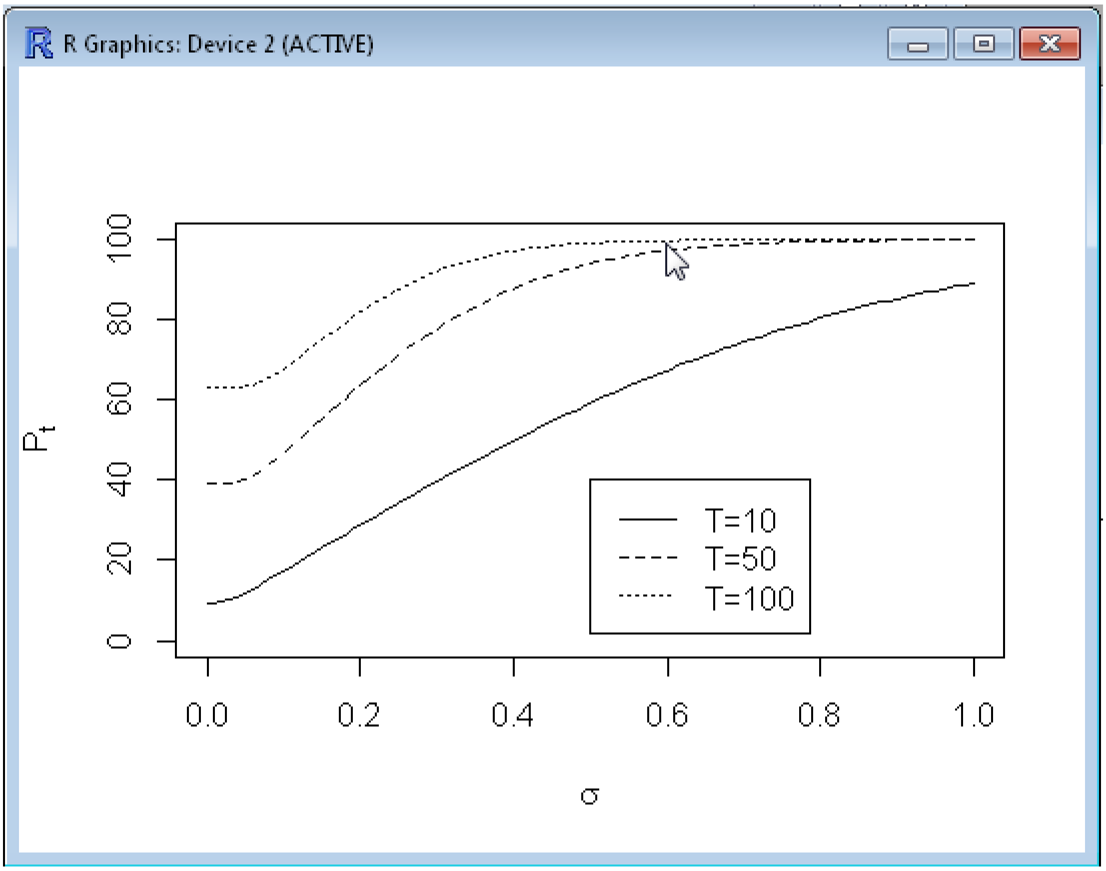
Ce qui donne:



Et nous savons aussi que le prix d'un Call est aussi fonction croissante du temps à maturité:



Ce qui nous donne:



Valuation (pricing) d'options européennes avec méthode de Monte Carlo

Nous appliquons simplement ce que nous avons démontré dans le cours théorique:

```
> # call put option monte carlo
> call_put_mc<-function(nSim=1000000, tau, r, sigma, S0, K) {
+
+   Z <- rnorm(nSim, mean=0, sd=1)
+   WT <- sqrt(tau) * Z
+   ST = S0*exp((r - 0.5*sigma^2)*tau + sigma*WT)
+
+   # price and standard error of call option
+   simulated_call_payoffs <- exp(-r*tau)*pmax(ST-K,0)
+   price_call <- mean(simulated_call_payoffs)
+   sterr_call <- sd(simulated_call_payoffs)/sqrt(nSim)
+   # price and standard error of put option
+   simulated_put_payoffs <- exp(-r*tau)*pmax(K-ST,0)
+   price_put <- mean(simulated_put_payoffs)
+   sterr_put <- sd(simulated_put_payoffs)/sqrt(nSim)
+
+   output<-list(price_call=price_call, sterr_call=sterr_call,
+                price_put=price_put, sterr_put=sterr_put)
+   return(output)
+
+ }
> set.seed(1)
> call_put_mc(n=1000000, tau=0.5, r=0.03, sigma=2.194056, S0=100, K=100)
$price_call
[1] 56.4103

$sterr_call
[1] 0.2980037

$price_put
[1] 55.04051

$sterr_put
[1] 0.03665131

>
```

Voyons maintenant les applications avec quelques techniques de réduction de la variance. Les codes ci-dessous sont reprise de:

https://www.r-bloggers.com/2020/12/pricing-of-european-options-with-monte-carlo/

Voyons maintenant l'application de deux techniques de réduction de la variance que nous avons démontré dans le cours théorique.

Méthode des variables antithétiques (réduction de la variance)

Voyons donc l'application des variables antithétiques pour le pricing des options vanilla européennes:

```
> antithetic_call_put_mc<-function(nSim, tau, r, sigma, S0, K) {
+
+   Z <- rnorm(nSim, mean=0, sd=1)
+
+   WT <- sqrt(tau) * Z
+   # ST1 and ST2 and the antithetic variates
+   ST1 = (S0*exp((r - 0.5*sigma^2)*tau + sigma*WT))
+   ST2 = (S0*exp((r - 0.5*sigma^2)*tau + sigma*(-WT)))
+
+   # call option price and standard error
+   simulated_call_payoffs1 <- exp(-r*tau)*pmax(ST1-K,0)
+   simulated_call_payoffs2 <- exp(-r*tau)*pmax(ST2-K,0)
+   # get the average
+   simulated_call_payoffs <- ( simulated_call_payoffs1 + simulated_call_payoffs2)/2
+   price_call <- mean(simulated_call_payoffs)
+   sterr_call <- sd(simulated_call_payoffs)/sqrt(nSim)
+
+   # put option price and standard error
+   simulated_put_payoffs1 <- exp(-r*tau)*pmax(K-ST1,0)
+   simulated_put_payoffs2 <- exp(-r*tau)*pmax(K-ST2,0)
+   # get the average
+   simulated_put_payoffs <- (simulated_put_payoffs1+simulated_put_payoffs2)/2
+   price_put <- mean(simulated_put_payoffs)
+   sterr_put <- sd(simulated_put_payoffs)/sqrt(nSim)
+
+   output<-list(price_call=price_call, sterr_call=sterr_call,
+                price_put=price_put, sterr_put=sterr_put )
+   return(output)
+
+ }
> antithetic_call_put_mc(n=1000000, tau=0.5, r=0.03, sigma=2.194056, S0=100, K=100)
$price_call
[1] 56.57819

$sterr_call
[1] 0.208214

$price_put
[1] 55.03211

$sterr_put
[1] 0.008853721

>
```

Méthode de l'échantillonnage préférentiel (réduction de la variance)

Nous avons vu dans le cours théorique que l'échantillonnage préférentiel est assez subtil a mettre en place car il nécessite le plus souvent de faire usage d'astuces mathématiques non-triviales. L'application de cette technique au pricing des options vanilla européennes comme ci-dessous en est encore une fois un bon exemple:

```
> importance_call_put_mc<-function(nSim, tau, r, sigma, S0, K) {
+
+   Z <- rnorm(nSim, mean=0, sd=1)
+   WT <- sqrt(tau) * Z
+   ST = S0*exp((r - 0.5*sigma^2)*tau + sigma*WT)
+
+   # call option price and standard error
+   simulated_call_payoffs <- (exp(-r*tau)*pmax(ST-K,0))[ST>K]
+   price_call <- mean(simulated_call_payoffs*mean(ST>K))
+   sterr_call <- sd(simulated_call_payoffs*mean(ST>K))/sqrt(nSim)
+
+   # put option price and standard error
+   simulated_put_payoffs <- (exp(-r*tau)*pmax(K-ST,0))[ST<K]
+   price_put <- mean(simulated_put_payoffs*mean(ST<K))
+   sterr_put <- sd(simulated_put_payoffs*mean(ST<K))/sqrt(nSim)
+
+   output<-list(price_call=price_call, sterr_call=sterr_call,
+                price_put=price_put, sterr_put=sterr_put )
+   return(output)
+
+ }
> antithetic_call_put_mc(n=1000000, tau=0.5, r=0.03, sigma=2.194056, S0=100, K=100)
$price_call
[1] 56.78135

$sterr_call
[1] 0.2171424

$price_put
[1] 55.04753

$sterr_put
[1] 0.008857805

>
```

Volatilité implicite d'options européennes ou américaines

Et avec les commandes **EuropeanOptionImpliedVolatility( )** ou respectivement **AmericanOptionImpliedVolatility( )** ,, nous pouvons chercher (entre autres), la volatilité implicite (ce qui est même temps est un preuve expérimentale ponctuelle que la fonction est écrite juste puisque nous retrouvons la volatilité ci-dessus):

```
RGui (64-bit) - [R Console]
> library(RQuantLib)
> EuropeanOptionImpliedVolatility(
+   type   = "call",
+   value = 56.5364,
+   underlying = 100,
+   strike       = 100,
+   dividendYield = 0,
+   riskFreeRate  = 0.03,
+   maturity      = 0.5,
+   volatility    = 0.01 #valeur initiale nécessaire cherche la VI
+ )$impliedVol
[1] 2.194054
>
```

Diagrammes Payoff (stratégies d'options)

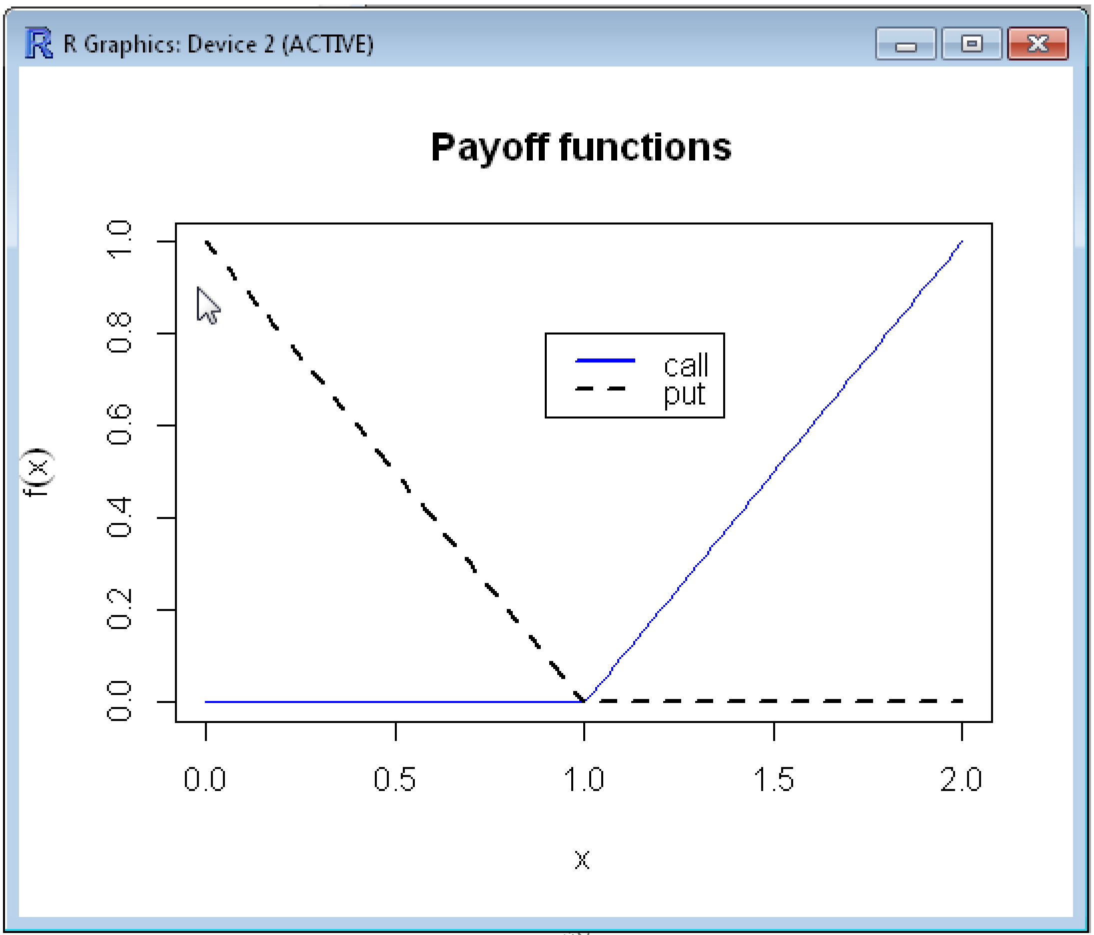
Nous avons étudié dans le cours théorique les diagrammes payoff des Call et Put ainsi de de plusieurs stratégies de portefeuilles d'options. Nous allons voir ici que ce n'est pas forcément trivial à écrire.

Diagramme Payoff Call et Put simple (sans package)

La stratégie d'options avec le diagramme payoff le plus simple que nous puissions imaginer est la stratégie consistant à posséder en même temps un call et un put. Pour construire le diagramme de payoff correspondant, la syntaxe n'est pas forcément triviale mais reste toutefois instructive:

```
R Console
> f.call<-function(x){ sapply(x, function(x) max(c(x - K, 0)))}
> f.put<-function(x){ sapply(x, function(x) max(c(K - x, 0)))}
> K<-1
> curve(f.call,0,2,main="Payoff functions",col="blue",lty=1,lwd=1,ylab=expression(f(x)))
> curve(f.put,0,2,col="black",add=TRUE,lty=2,lwd=2)
> legend(0.9,0.8,c("call","put"),lty=c(1,2),col=c("blue","black"),lwd=c(2,2))
>
```

Ce qui donne:



Plusieurs années après avoir écrit le script de diagramme payoff, un package a été publié en proposant un plus grand nombre et ce de façon quasi automatique. Voyons cela:

Autres diagrammes payoff (bull, bear, butterfly, covered, collar, protective,…)

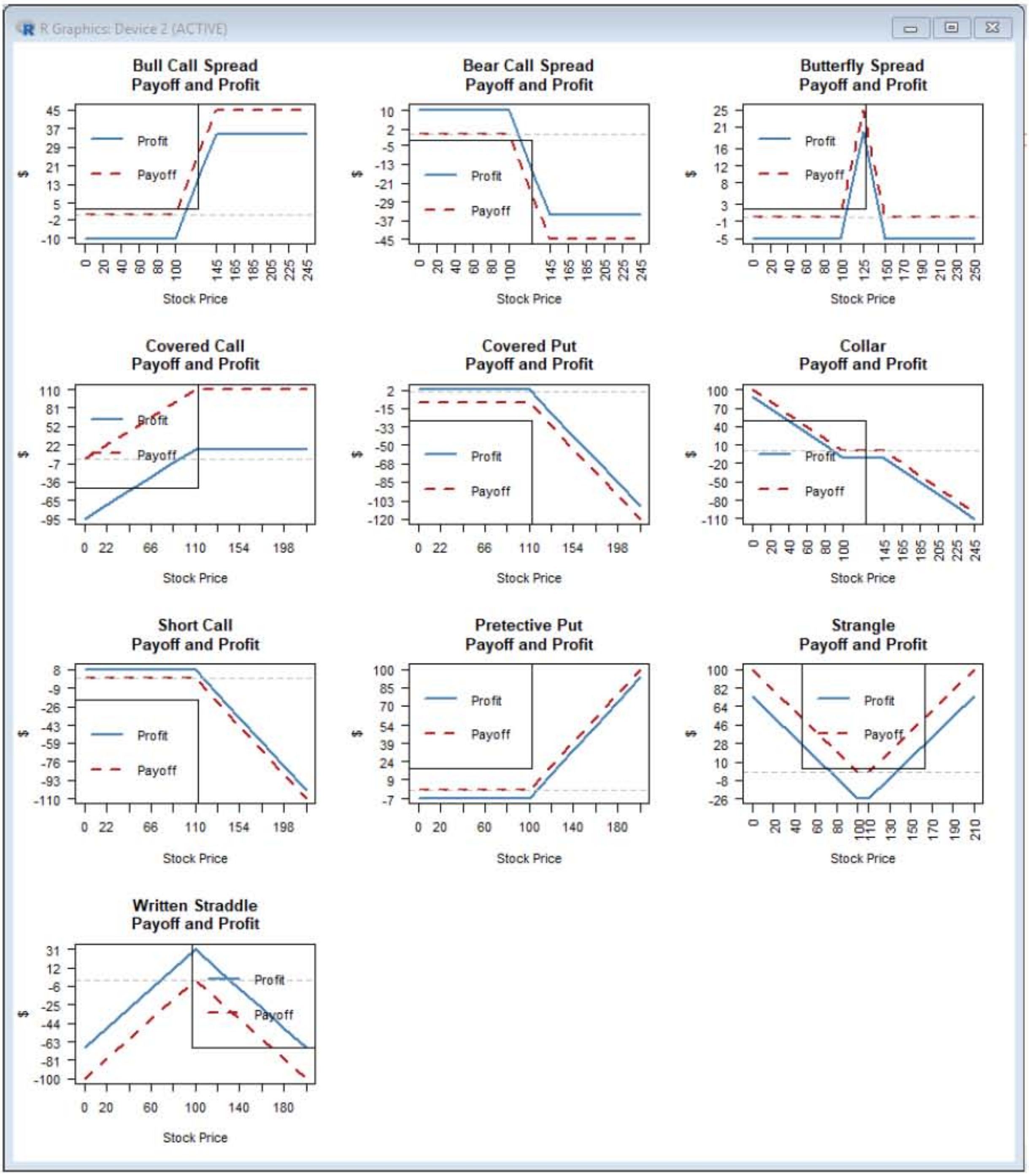
Nous avons vu dans le cours théorique qu'il existe au bas mot une vingtaine de stratégies d'options. C'est un peu la jungle et malheureusement, à notre connaissance, il n'existe aucun package les proposant tous. Le plus complet à ce jour semblerait être le package **FinancialMath** (la documentation est de plus très bien faite car elle rappelle les équations en LaTeX – c'est assez rare pour le souligner… – sous-jacentes de la stratégie).

Voyons un exemple d'abord des graphiques que ce package peut générer où on fournit exceptionnellement le script à copier/coller (car le récrire n'apporte pas grand-chose pédagogiquement parlant dans ce cas particulier):

```
library("FinancialMath")
par(mfrow=c(4,3))
bull.call(S=115,K1=100,K2=145,r=.03,t=1,price1=20,price2=10,plot=TRUE)
bear.call(S=115,K1=100,K2=145,r=.03,t=1,price1=20,price2=10,plot=TRUE)
butterfly.spread(S=125,K1=100,K2=125,K3=150,r=.03,t=1,price1=20,price2=10,p
rice3=5,plot=TRUE)
covered.call(S=100,K=110,r=.03,t=1,sd=.2,plot=TRUE)
covered.put(S=100,K=110,r=.03,t=1,sd=.2,plot=TRUE)
collar(S=115,K1=100,K2=145,r=.03,t=1,price1=20,price2=10,plot=TRUE)
option.call(S=100,K=110,r=.03,t=1.5,sd=.2,price=NA,position="short",plot=TR
UE)
option.put(S=100,K=110,r=.03,t=1,sd=.2,price=NA,position="long")
```

```
protective.put(S=100,K=100,r=.03,t=1,sd=.2,plot=TRUE)
strangle(S=105,K1=100,K2=110,r=.03,t=1,price1=10,price2=15,plot=TRUE)
straddle(S=115,K=100,r=.03,t=1,price1=20,price2=10,position="short",plot=TR
UE)
```

Ce qui donnera:



Voyons maintenant typiquement le type de données que cela renvoie:

```
> bc<-bull.call(S=115,K1=100,K2=145,r=.03,t=1,price1=20,price2=10,plot=TRUE)
> bc
$`Payoff`
   Stock Price Payoff     Profit
1            0      0 -10.304545
2           20      0 -10.304545
3           40      0 -10.304545
4           60      0 -10.304545
5           80      0 -10.304545
6          100      0 -10.304545
7          115     15   4.695455
8          130     30  19.695455
9          145     45  34.695455
10         165     45  34.695455
11         185     45  34.695455
12         205     45  34.695455
13         225     45  34.695455
14         245     45  34.695455

$Premiums
          Premiums
Call (K1)       20
Call (K2)       10
Net Cost        10

> bc$Payoff$Profit
 [1] -10.304545 -10.304545 -10.304545 -10.304545 -10.304545 -10.304545   4.695455
 [8]  19.695455  34.695455  34.695455  34.695455  34.695455  34.695455  34.695455
> 
```

Simulation de Monte-Carlo pour options Vanilla avec mouvement géométrique brownien

Nous renvoyons d'abord le lecteur à la lecture du chapitre sur les simulations de Monte Carlo à la page 1944 dans un premier temps.

Voyons maintenant comment faire une simulation de Monte Carlo avec R:

```
> #le classique d'abord
> stock=100
> sigma=2.194056
> strike=100
> TTM=0.5
> rf=0.03
>
> #le nombre de simulations
> num.sim<-1000000
>
> #on calcule certaines variables
> R<-(rf-0.5*sigma^2)*TTM
> R
[1] -1.18847
> SD<-sigma*sqrt(TTM)
> SD
[1] 1.551432
>
> #on fait les simulations
> TTM.price<-stock*exp(R+SD*rnorm(num.sim,0,1))
>
> #et donc pour un Call Vanilla on obtient
> TTM.call<-pmax(0,TTM.price-strike)
> PV.call<-TTM.call*(exp(-rf*TTM))
> mean(PV.call)
[1] 56.62282
>
> #et pour un Put Vanilla
> TTM.put<-pmax(0,strike-TTM.price)
> PV.put<-TTM.put*(exp(-rf*TTM))
> mean(PV.put)
[1] 55.02435
>
```

Nous retombons donc bien sur nos pattes!

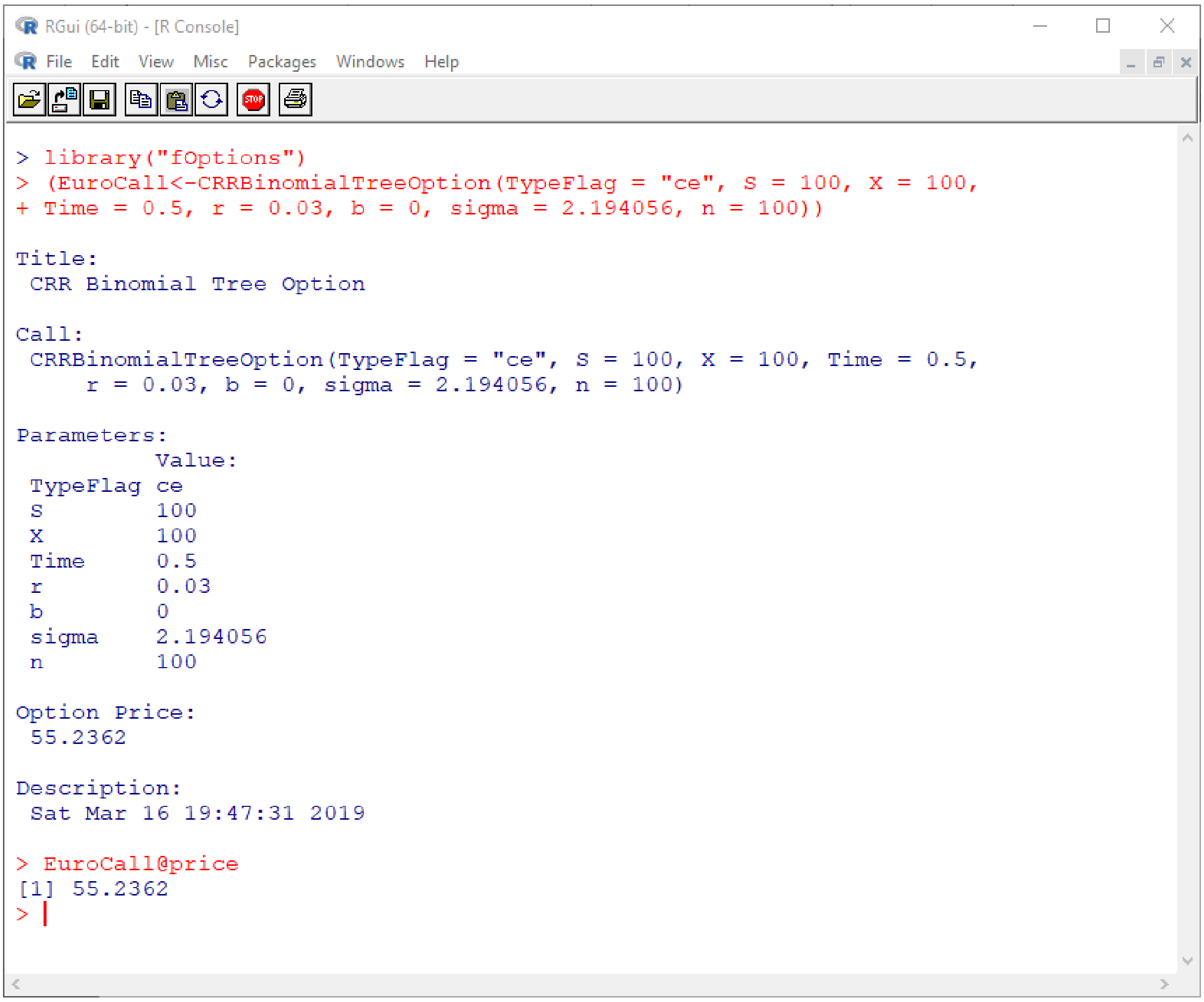
Arbre binomiaux alternatifs

Commençon d'abord par l'arbre binomial CRR classique

Arbre binomial de Cox-Ross-Rubinstein

Nous allons vérifier ici que nous obtenons bien avec R les calculs effectués à la main dans le cours théorique concernant l'arbre binomial de Cox-Ross-Rubinstein.

Pour cela nous installons le package **fOption** et utilisons la commande **CRRBinomialTreeOption( )** :



Nous retrouvons à peu près la valeur calculée dans le cours théorique. Et cette valeur est approximativement égale à celle que nous avons obtenu plus haut pour le modèle "exact" (environ 2% de différence)!

Pour information: "ce"/"pe" = call/put européen, "ca"/"pa" = call/put américain

Ensuite, nous pouvons faire un plot de l'arbre avec la commande **BinomialTreeOption( )** du même package:

```
> CRRTree<-BinomialTreeOption(TypeFlag = "ce", S = 100, X = 100,
+ Time = 0.5, r = 0.03, b = 0, sigma = 2.194056, n = 5)
> BinomialTreePlot(CRRTree, dy = 1, cex = 0.8, ylim = c(-6, 7),
+ xlab = "n", ylab = "Option Value")
>
```

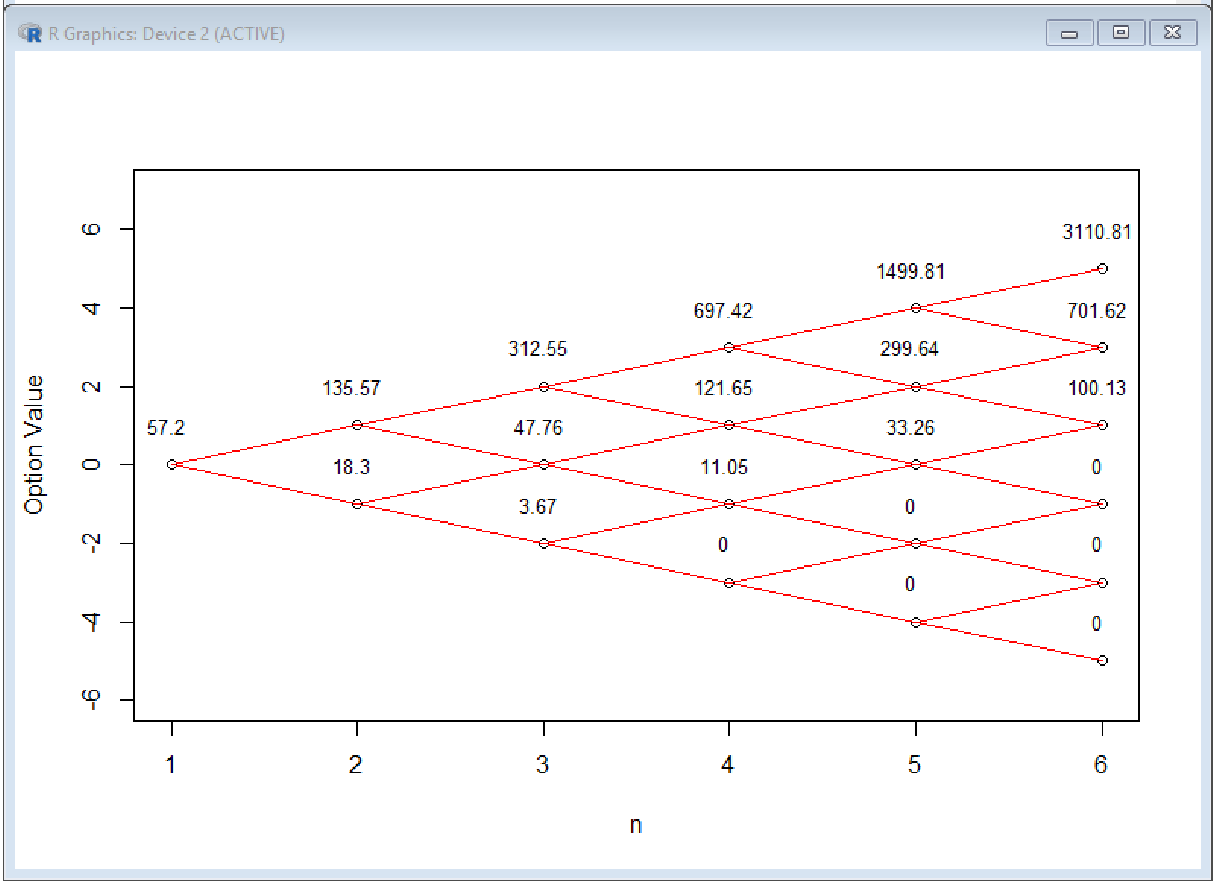


Figure 323 Graph arbre binomial

Nous avons d'autres modèles de valuations basés sur des arbres binomiaux.

Arbres de Tian et Jarrow-Rudd

Le package **fOptions** en contient deux autres (Jarrow-Rudd et Tian). Comparons-les tous:

```
> library("fOptions")
> CRR_EuroCall<-CRRBinomialTreeOption(TypeFlag = "ce", S = 100, X = 100,
+ Time = 0.5, r = 0.03, b = 0, sigma = 2.194056, n = 1000)
> CRR_EuroCall@price
[1] 55.35795
>
> Tian_EuroCall<-TIANBinomialTreeOption(TypeFlag = "ce", S = 100, X = 100,
+ Time = 0.5, r = 0.03, b = 0, sigma = 2.194056, n = 1000)
> Tian_EuroCall@price
[1] 55.34881
>
> JR_EuroCall<-JRBinomialTreeOption(TypeFlag = "ce", S = 100, X = 100,
+ Time = 0.5, r = 0.03, b = 0, sigma = 2.194056, n = 1000)
> JR_EuroCall@price
[1] 55.32485
>
```

Je n'ai pas encore été capable à ce jour de trouver comment reproduire les exemples faits dans le cours MATLAB – exemples différents de ceux ci-dessus - concernant les arbres JR et Tian (sachant que nous savons que les résultats renvoyés par MATLAB étaient corrects!).

Smile de volatilité

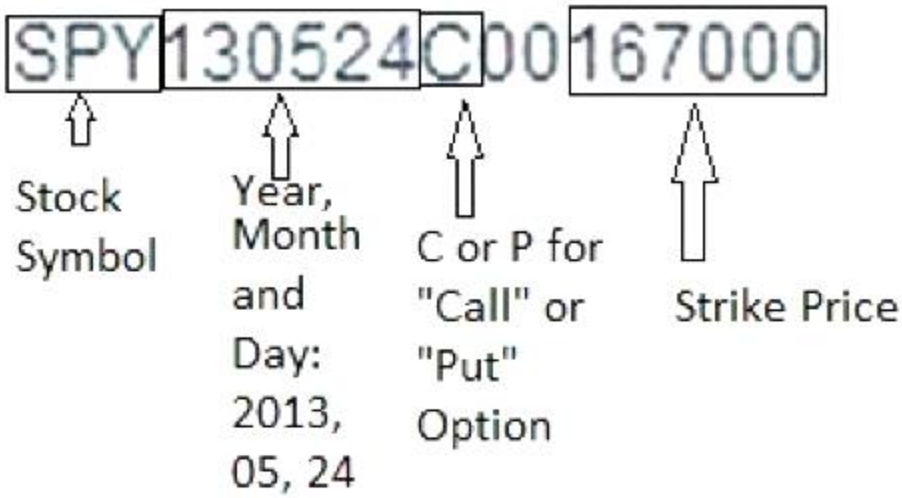
Malheureusement, il n'existe pas à notre connaissance et à ce jour de package faisant un smile de volatilité (ceci s'explique aisément vu l'aspect pratique de la chose…).

Pour voir toutefois comment écrire notre propose script, rappatrions les données des options Microsoft négociables sur le marché de gré à gré (1 option est pour 100 sous-jacents normalement dans les exports Yahoo/Google) dont la date d'expiration est avant le 30 Juillet 2014 et dans l'ordre décroissant des dates d'expiration (comme le lecteur pourra le voir ci-dessous il n'y a pas beaucoup d'options qui vont à une date ultérieure au 19 Juillet 2014 sachant que ce script a été écrit le 1er Juillet 2014... donc il y peu d'agents prenant le risque de proposer des options au-delà de 19 jours pour les sous-jacents Microsoft):

```
> library(quantmod)
> library(RQuantLib)
> options("getSymbols.warning4.0"=FALSE)
> x<-getOptionChain("MSFT",Exp ="2014-07-30")
> head(x)
$calls
                    Strike  Last  Chg   Bid   Ask Vol   OI
MSFT140719C00018000   18.0 21.60 0.00 23.45 24.80  72  104
MSFT140719C00020000   20.0 19.55 0.00 21.45 22.80  20  132
MSFT140719C00025000   25.0 16.70 0.00 16.65 17.10   1  305
MSFT140719C00026000   26.0 15.46 0.00 15.55 16.05   4    4
MSFT140719C00028000   28.0 13.68 0.00 13.70 14.10   2   35
MSFT140719C00029000   29.0 12.55 0.00 12.65 13.30  15   26
MSFT140719C00030000   30.0 11.80 0.00 11.70 12.10  54  267
MSFT140719C00031000   31.0 10.75 1.65 10.70 10.85  20   86
MSFT140703C00032000   32.0  9.30 0.00  9.65 10.05  55  135
MSFT140719C00032000   32.0 10.10 0.50  9.70  9.85 144  146
MSFT140719C00033000   33.0  9.00 0.40  8.70  8.85 174  305
MSFT140719C00034000   34.0  7.90 0.10  7.70  7.85   9  300
MSFT140703C00034500   34.5  7.20 0.00  7.15  7.30  30   84
MSFT140703C00035000   35.0  6.70 0.00  6.65  6.80  24   58
MSFT140719C00035000   35.0  7.00 0.41  6.70  6.85   6  785
MSFT140703C00035500   35.5  6.35 0.00  6.15  6.30   1    1
MSFT140703C00036000   36.0  5.75 0.00  5.65  5.80  40   40
MSFT140711C00036000   36.0  6.00 0.22  5.70  5.90  40   10
MSFT140719C00036000   36.0  6.00 0.00  5.70  5.85  25 1369
MSFT140725C00036000   36.0  5.42 0.00  5.75  6.35   5    5
MSFT140725C00036500   36.5  5.00 0.00  5.20  5.60  41   41
MSFT140711C00037000   37.0  4.23 0.00  4.70  4.85   5   45
```

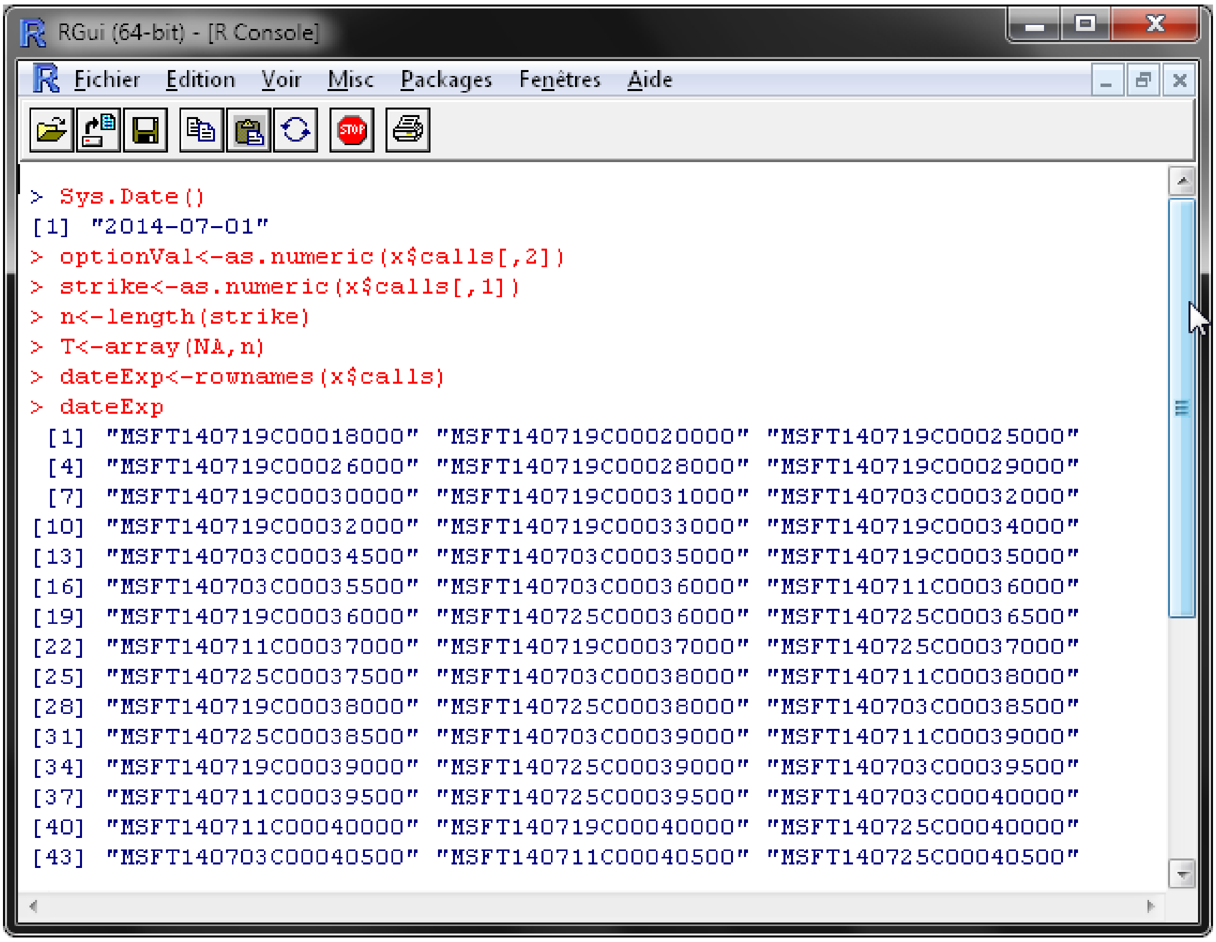
Alors que la colonne *Strike* donne le... strike du sous-jacent (logique...), la colonne *Last* donne le dernier prix de trade du Call. Nous nous baserons sur cette valeur pour déterminer la volatilité implicite mais nous pourrions tout aussi bien prendre la valeur *Bid* (qui est pour rappel le prix auquel les acheteurs essaient d'acheter l'option) si nous sommes acheteurs de Call.

La première colonne contient les dates d'expiration des Call (ou des Puts si vous descendez plus bas dans la liste...) mais voici comment la décoder en détails:

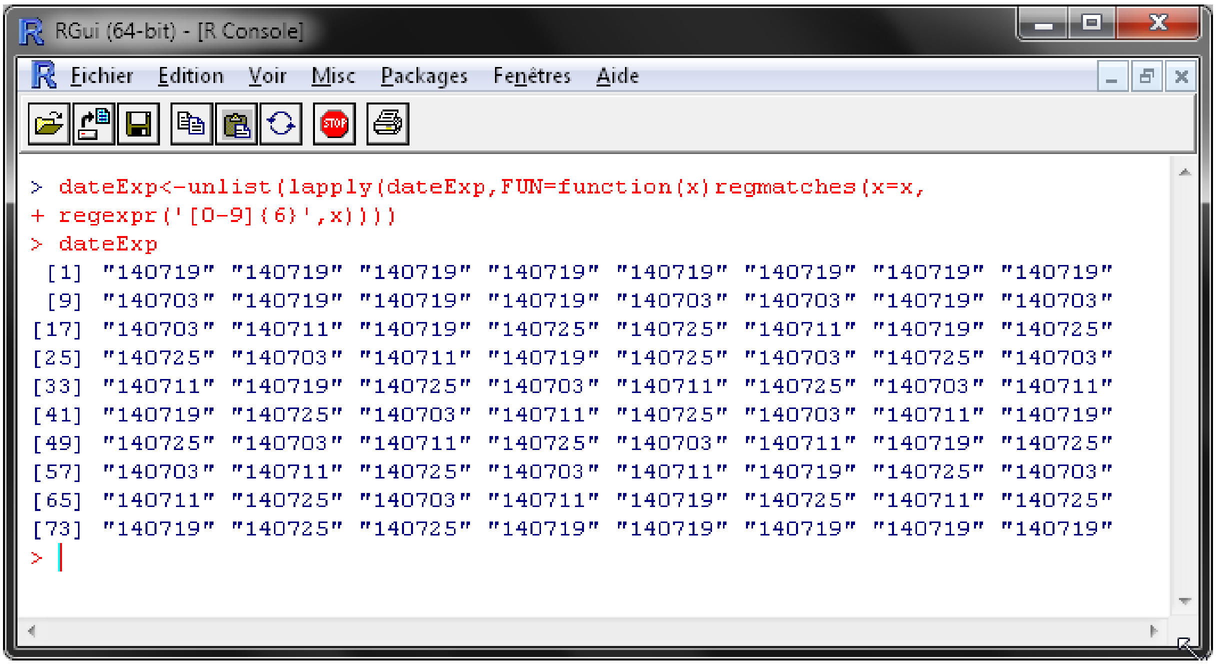


Nous voyons que pour une même date d'émission des agents proposent des options d'achats avec des strikes donnés variés (ce qui est normal puisque chacun fait des projections à sa manière).

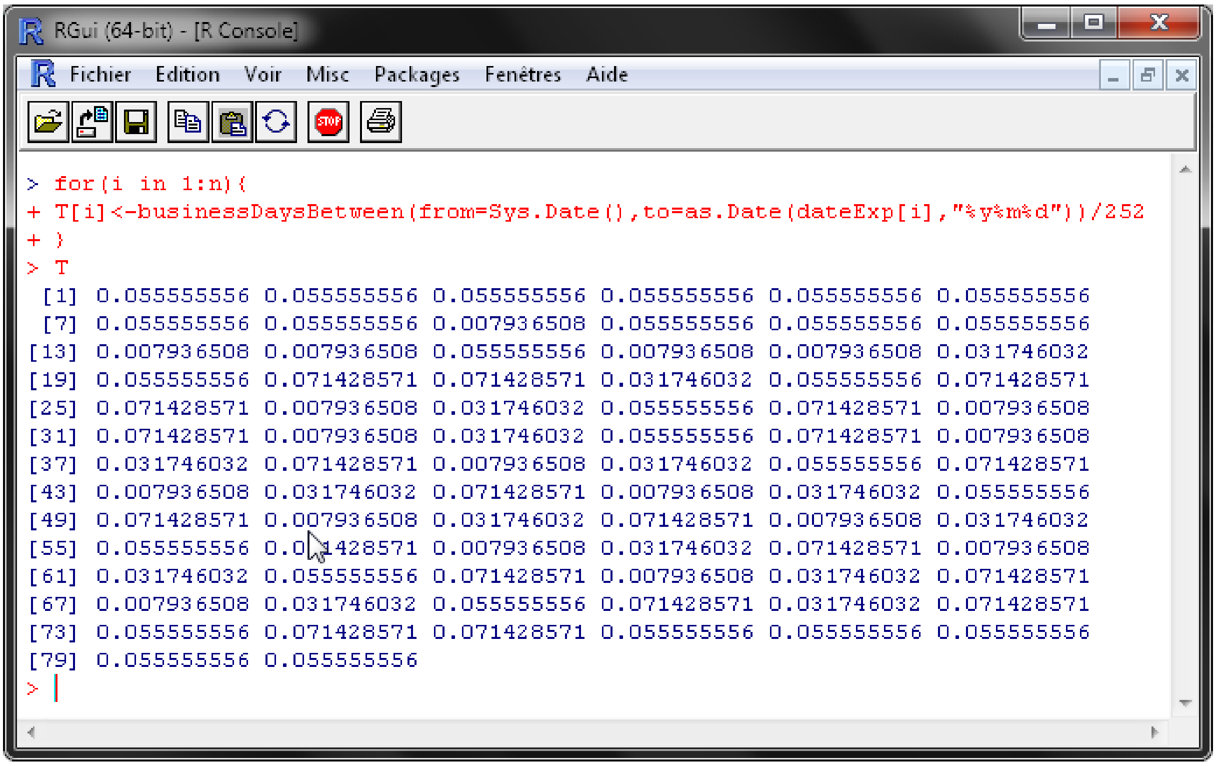
Nous allons pour poursuivre extraire la colonne des strikes, la convertir en valeurs numériques, compter les nombres d'options disponibles dans cette extraction et déterminer et extraire les noms de lignes pour ensuite les traiter afin d'obtenir les dates qui sont codées:



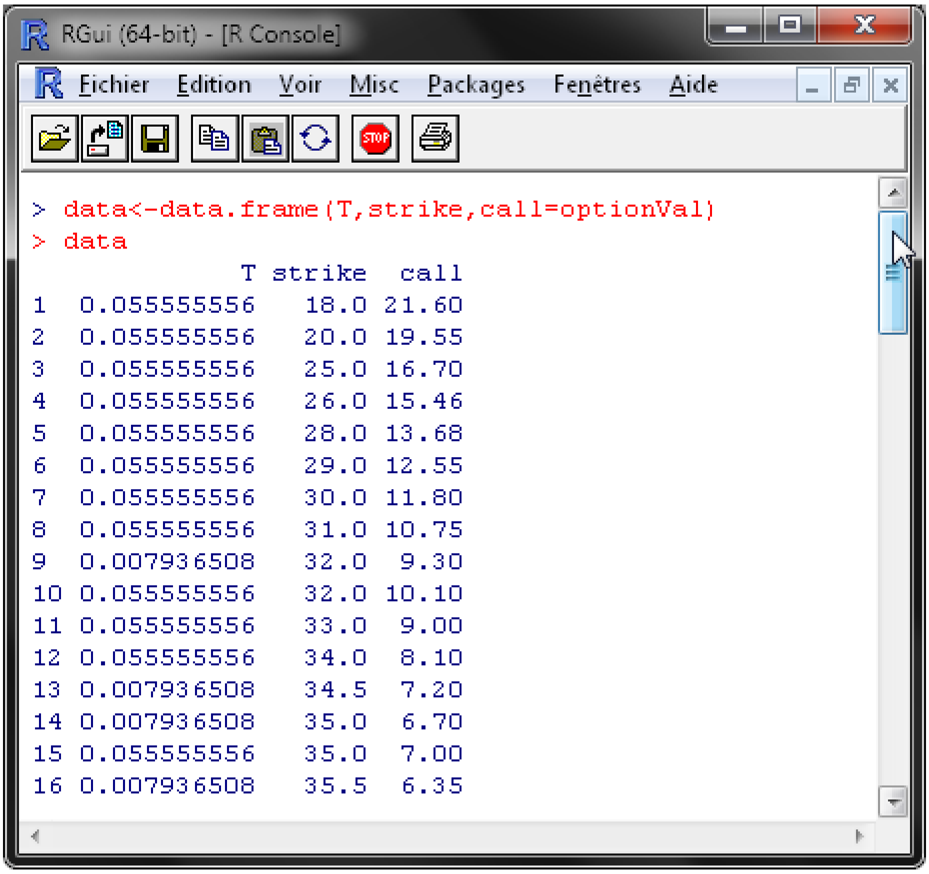
Donc maintenant nous extrayons les dates:



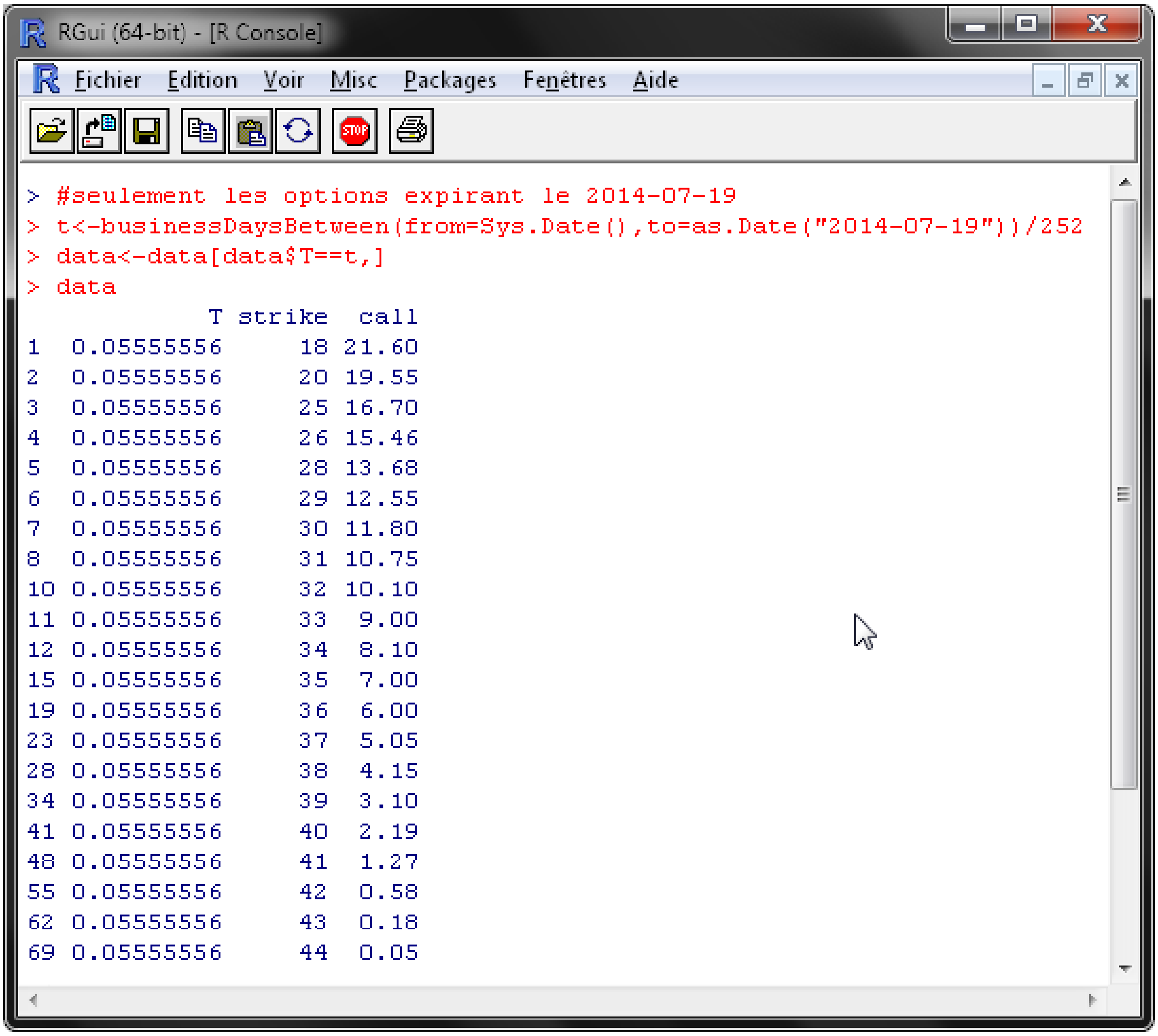
Ensuite nous calculons la durée jusqu'à maturité entre le jour de rédaction (exécution) du script et la date d'expiration de chaque option (je n'ai pas trouvé autre chose qu'une boucle...):



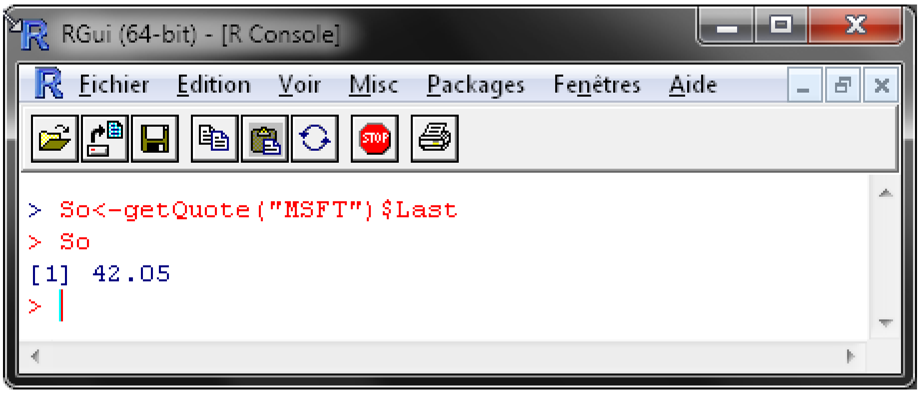
Nous regroupons le tout dans un data frame:



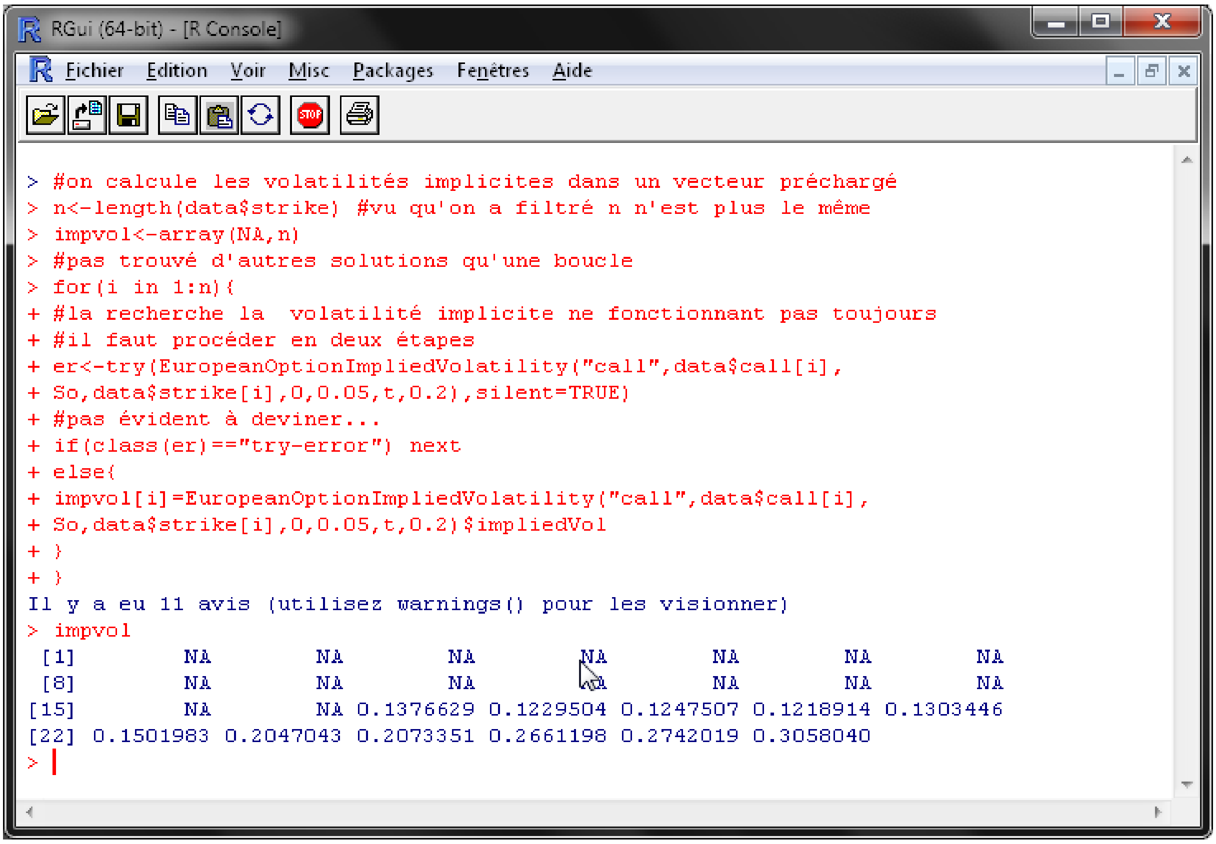
Ensuite puisque par définition le smile de volatilité est pour des options de maturité égales, nous allons filtrer sur un choix particulier arbitraire pour avoir que des options de même maturité:



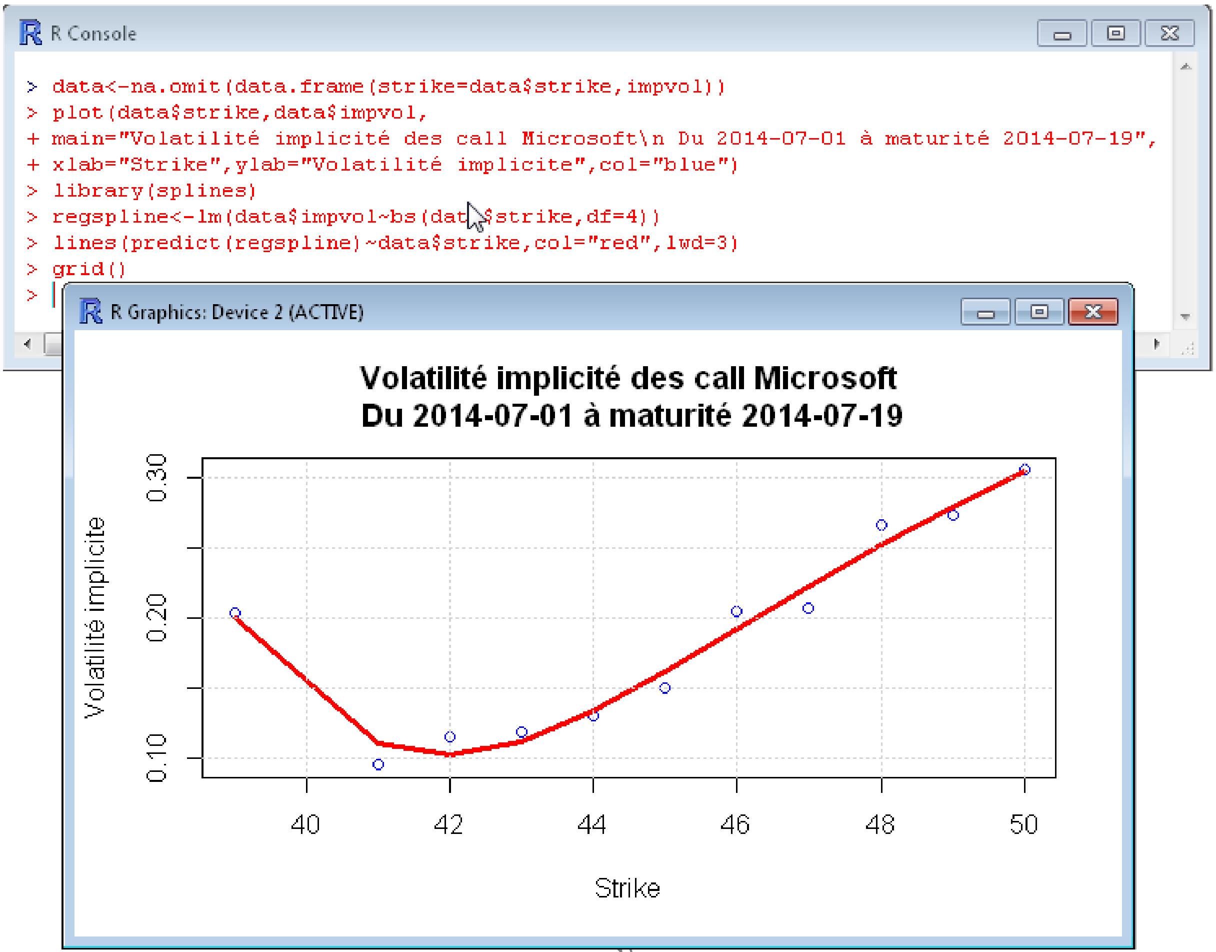
Ensuite, il nous faut le prix actuel du sous-jacent (spot price):



Et ensuite de calculer les volatilités implicites:



Il ne nous reste plus qu'à faire le graphique avec une interpolation de type spline:



Nous voyons que la smile (la volatilité implicite) est proche d'un minima pour la valeur du cours du jour du sous-jacent. L'interprétation de cette courbe est très intéressante mais si nous ne possédons pas toutes les informations du marché, nous pouvons a posteriori conclure que le sous-jacent a une plus forte probabilité d'être à l'avenir au-dessus du cours du jour qu'en dessous (c'est quand même des sous-jacents Microsoft....).

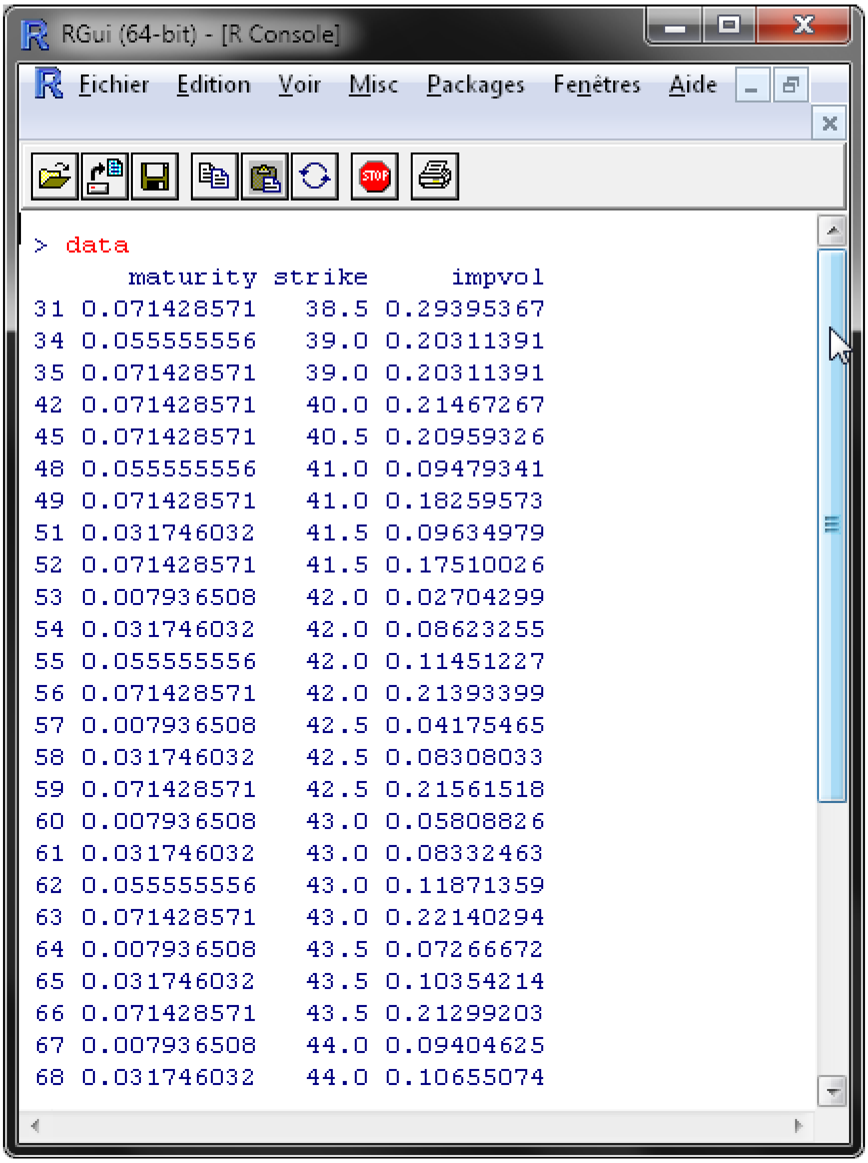
Surface de volatilité

Malheureusement, il n'existe pas non plus à notre connaissance et à ce jour de package faisant une surface de volatilité (ceci s'explique aisément vu l'aspect pratique de la chose…).

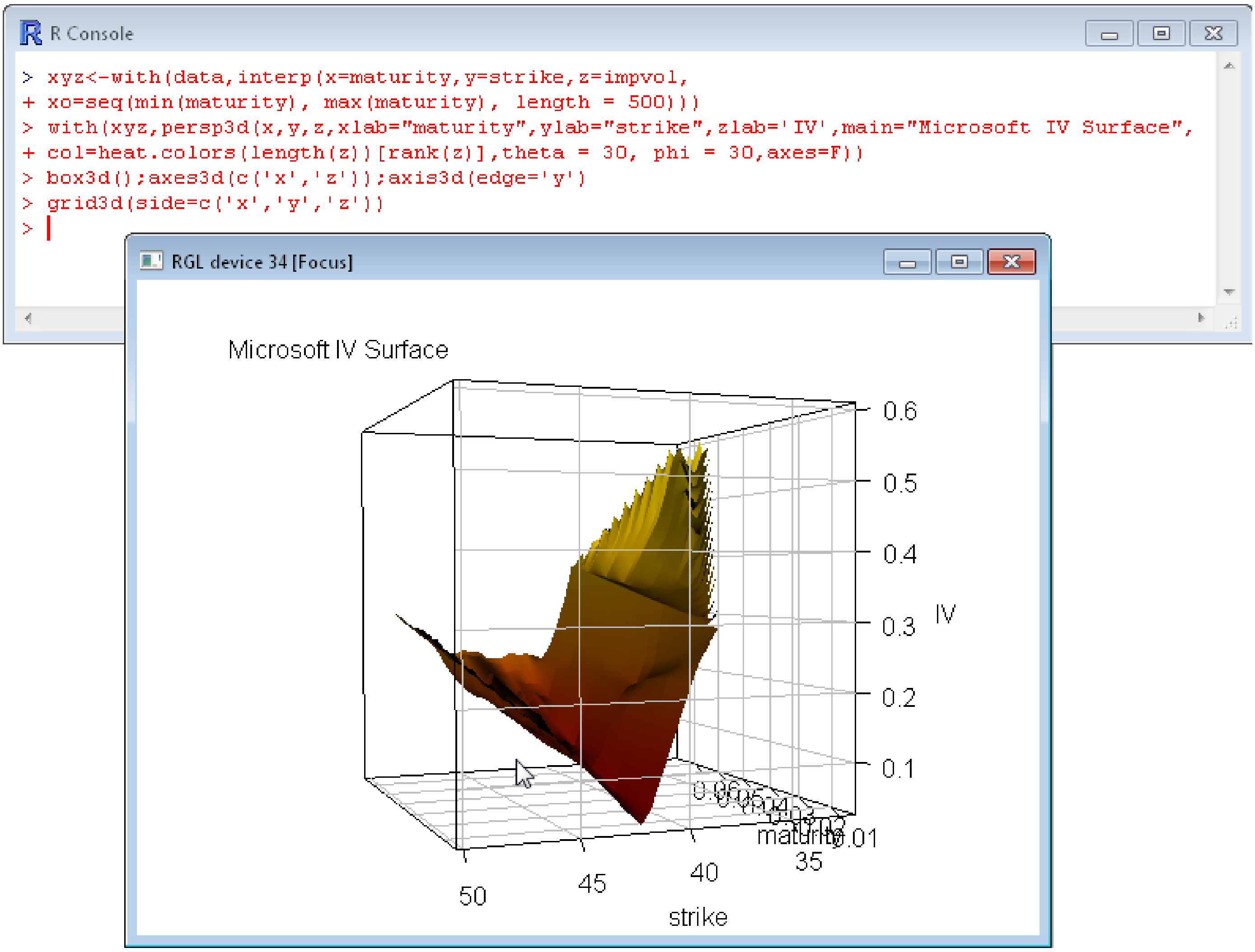
L'extension naturelle du smile de volatilité est la surface de volatilité. Le code est quasiment identique à avant à la différence que nous ne filtrons pas pour une seule date de maturité donnée et aussi à la différence qu'il s'agit d'un plot 3D plutôt que 2D.

```
> library(quantmod);library(RQuantLib)
> x<-getOptionChain("MSFT")
> optionVal<-as.numeric(x$calls[,2])
> strike<-as.numeric(x$calls[,1])
> n<-length(strike)
> T<-array(NA,n)
> dateExp<-rownames(x$calls)
> dateExp<-unlist(lapply(dateExp,FUN=function(x)regmatches(x=x,
+ regexpr('[0-9]{6}',x))))
> for(i in 1:n){
+ T[i]<-businessDaysBetween(from=Sys.Date(),to=as.Date(dateExp[i],"%y%m%d"))/252
+ }
> data<-data.frame(T,strike,call=optionVal)
> t<-businessDaysBetween(from=Sys.Date(),to=as.Date("2014-07-19"))/252
> So<-getQuote("MSFT")$Last
> impvol<-array(NA,n)
> for(i in 1:n){
+ er<-try(EuropeanOptionImpliedVolatility("call",data$call[i],
+ So,data$strike[i],0,0.05,t,0.2),silent=TRUE)
+ if(class(er)=="try-error") next
+ else{
+ impvol[i]=EuropeanOptionImpliedVolatility("call",data$call[i],
+ So,data$strike[i],0,0.05,t,0.2)$impliedVol
+ }
+ }
Il y a eu 48 avis (utilisez warnings() pour les visionner)
> data<-na.omit(data.frame(maturity=T,strike=data$strike,impvol))
> data<-data[order(data$maturity),]
>
```

Ce qui donne:



Après ne reste plus qu'à faire un plot mais vu la faible quantité de données il ne va pas être extra...:



Nous voyons donc bien ci-dessus la forme en "U" du smile de volatilité pour chaque maturité. Bon cela doit être possible de faire plus joli... mais après quand on réfléchit à l'utilité de ce type de graphiques 3D....

Volatilité implicité

Pour calculer la volatilité implicité nous importons d'abord des données de Yahoo:

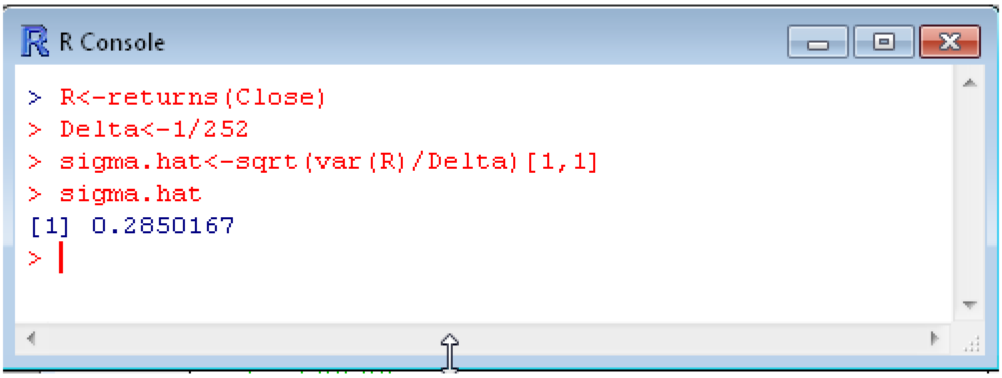
```
> library(fImport)
> S<-yahooSeries("ATL.MI",from="2004-07-23",to="2005-05-13")
essai de l'URL 'http://chart.yahoo.com/table.csv?s=ATL.MI&a=&b=&c=&d=11&e=31&f=2014&g=d&x=.csv'
Content type 'text/csv' length unknown
URL ouverte
downloaded 139 Kb

> head(S)
GMT
           ATL.MI.Open ATL.MI.High ATL.MI.Low ATL.MI.Close ATL.MI.Volume ATL.MI.Adj.Close
2005-05-13       22.95       23.01      22.60        22.66       5944700            14.21
2005-05-12       23.02       23.26      22.74        22.90       3324700            14.36
2005-05-11       22.94       23.16      22.78        23.00       7415700            14.42
2005-05-10       22.77       23.13      22.72        22.93       2357700            14.38
2005-05-09       22.49       22.78      22.30        22.71       4171500            14.24
2005-05-06       22.38       22.80      22.14        22.60       3038800            14.17
>
```

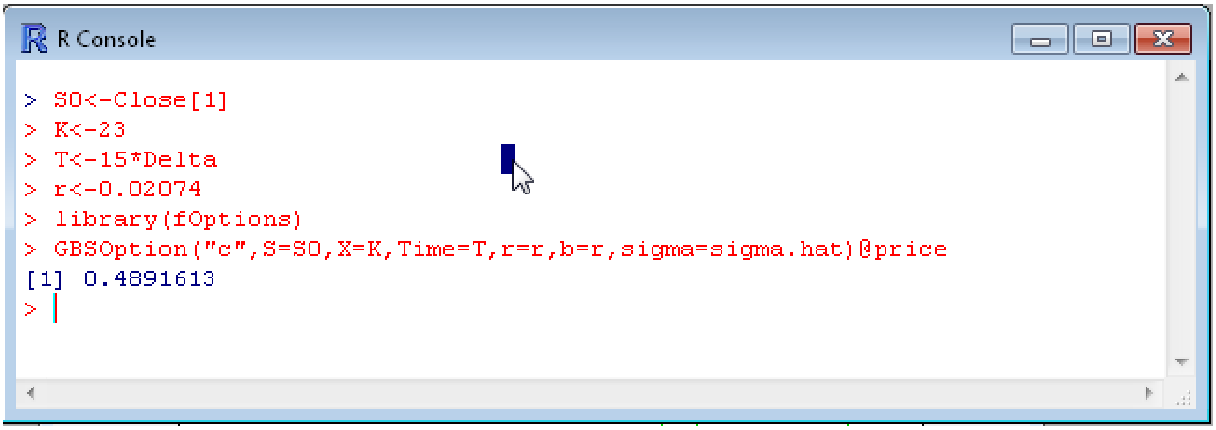
Faisons d'abord un graph du type habituel pour analyser les prix à la fermeture:

```
> Close<-S[,"ATL.MI.Close"]
> library(quantmod)
> chartSeries(Close,theme="white")
>
```

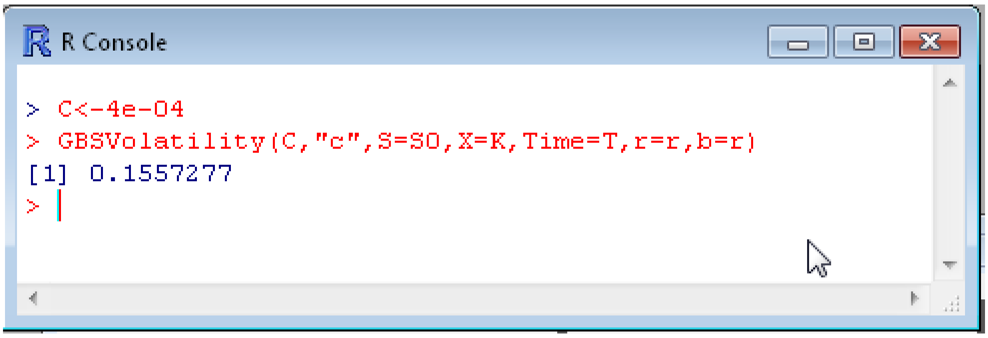




Considérons maintenant une option Call qui le 13 Mai 2005 était évaluée à 0.0004 pour un prix d'exercice $K = 23$, avec un spot $S_0 = 22.66$ (visible sur le graphique ci-dessus en haut à gauche). La date d'expiration était au 3 Juin 2005, ce qui correspond à 15 jours, dès lors rapporté à une fraction d'année $T = 15 \cdot \Delta$ et le taux sans risque du marché était de $r = 0.02074$. Dès lors, le prix de l'option était en utilisant un autre package que RQuantLib (comme cela vous pouvez voir qu'il y en a plusieurs qui font la même chose):



La différence entre le prix du Call sur le marché et selon le modèle de Black & Scholes-Merton est très différent. Voyons donc la volatilité implicite du marché:



Il y a presque un facteur 2 entre la volatilité historique de 0.285 et celle de la volatilité implicite de 0.155.

Évaluation de deux options Vanilla corrélées

Dans le cours théorique nous avons démontré comment valuer un panier de deux options européennes de type call à l'aide de la fonction **TwoAssetCorrelationOption( )** du package **fExoticOptions**:

```
> library("fExoticOptions")
> TwoAssetCorrelationOption(TypeFlag = "c", S1 = 52, S2 = 65,
+ X1 = 50, X2 = 70, Time = 0.5, r = 0.10, b1 = 0, b2 = 0,
+ sigma1 = 0.2, sigma2 = 0.3, rho = 0.75)

Title:
 Two Asset Correlation Option

Call:
 TwoAssetCorrelationOption(TypeFlag = "c", S1 = 52, S2 = 65, X1 = 50,
     X2 = 70, Time = 0.5, r = 0.1, b1 = 0, b2 = 0, sigma1 = 0.2,
     sigma2 = 0.3, rho = 0.75)

Parameters:
          Value:
 TypeFlag c
 S1       52
 S2       65
 X1       50
 X2       70
 Time     0.5
 r        0.1
 b1       0
 b2       0
 sigma1   0.2
 sigma2   0.3
 rho      0.75

Option Price:
 3.242528

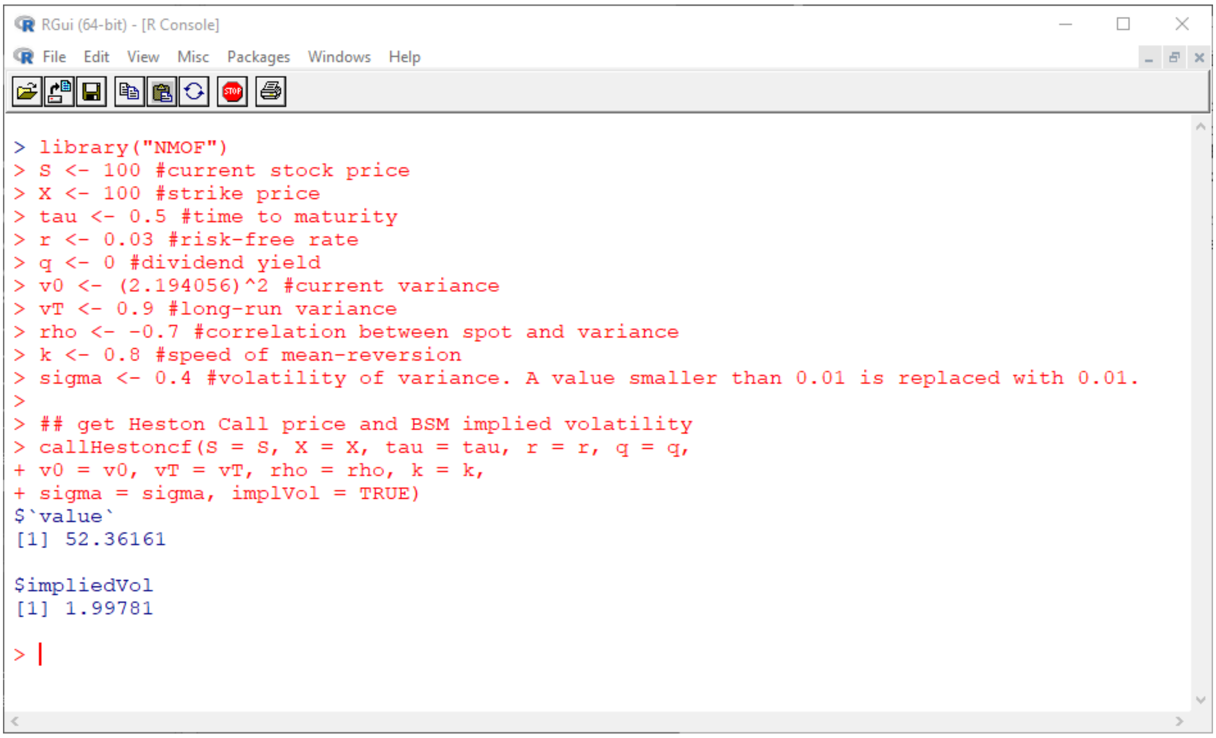
Description:
 Sun Mar 17 14:26:38 2019

>
```

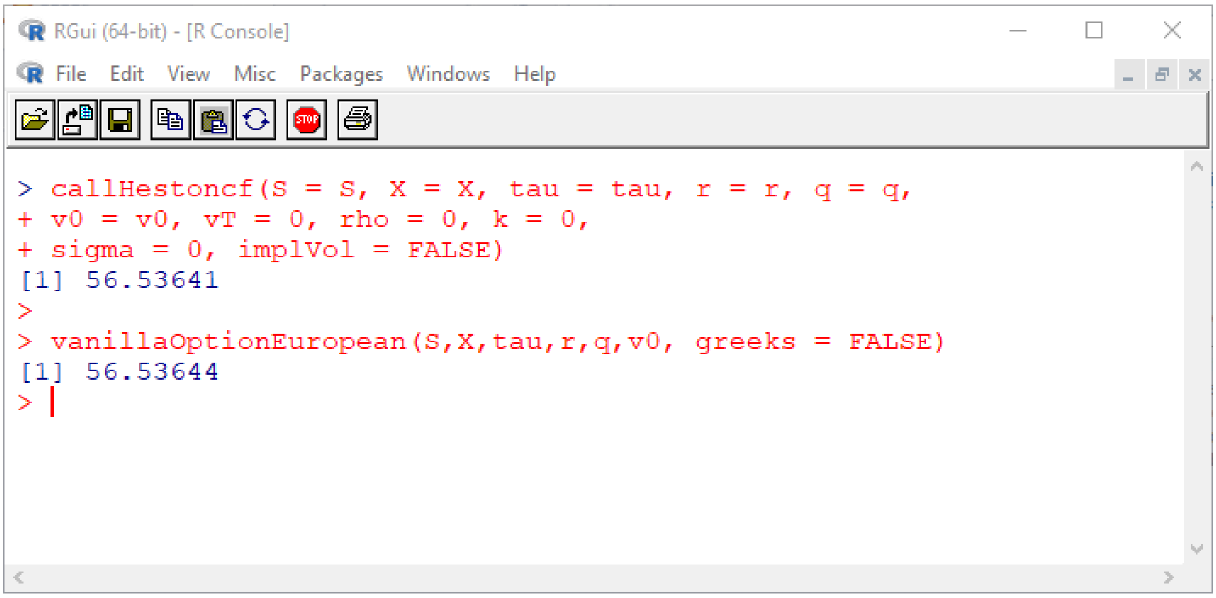
Évaluation d'options Vanilla selon modèle de Heston

Dans le cours théorique nous avons mentionné le modèle de Heston sans toutefois faire la démonstration mathématique puisque cette dernière si détaillée fait bien au minimum entre 70 et 150 pages A4 recto-verso… Nous avons cependant fourni une référence qui contient la démonstration détaillée. Cela légitime donc le fait que nous puissions nous y attarder.

Voyons cela en reprenant l'exemple de la documentation du package **NMOF** et sa fonction **callHestoncf( )** mais en changeant les valeurs pour les faire correspondre à notre cas de calcul du modèle de Black & Scholes-Merton (BMS):



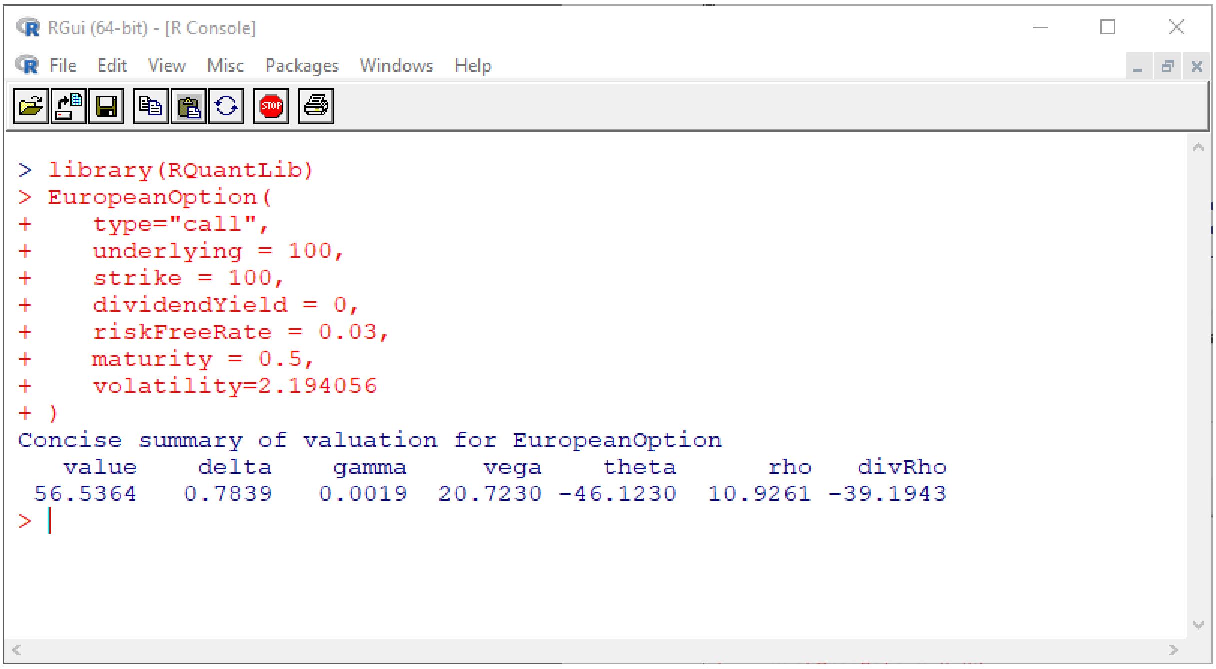
Et maintenant pour le fun regardons qu'en annulant certains paramètres du modèle de Heston, que nous retombons bien sur le modèle de pricing de Call plain vanilla de BSM:



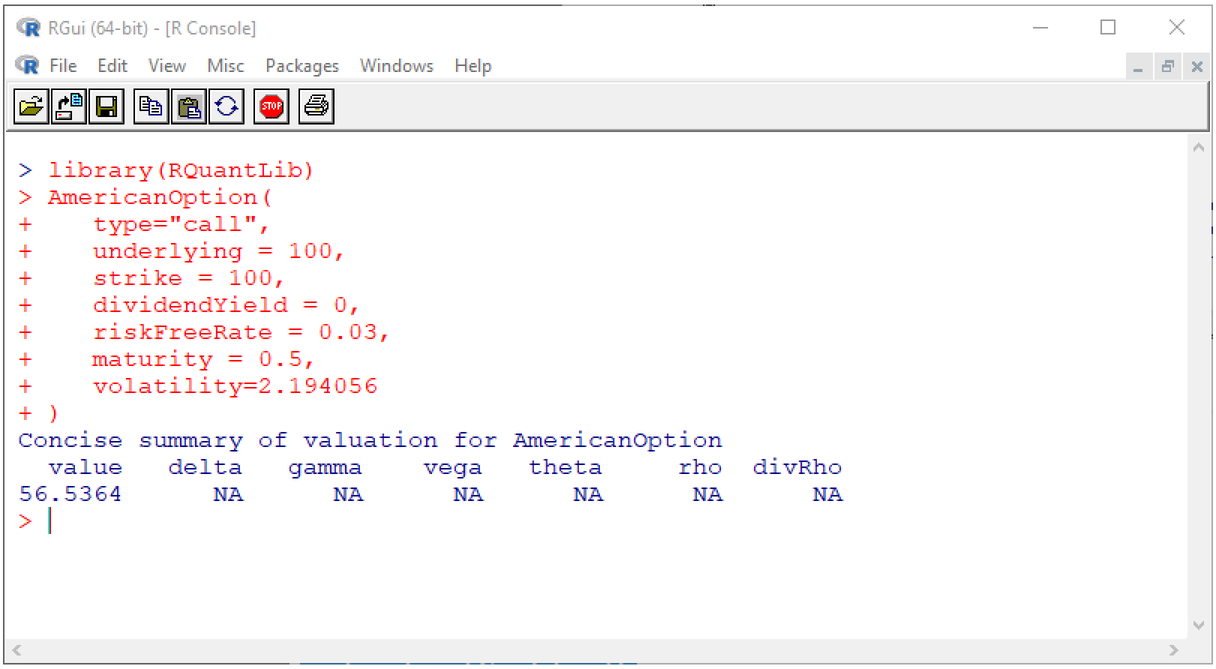
Évaluation d'options Asiatiques avec moyenne arithmétique ou géométrique

Nous avons fourni dans le cours théorique les références vers les démonstrations mathématiques détaillées des options asiatiques. Cela nous légitime donc pour aborder le sujet!

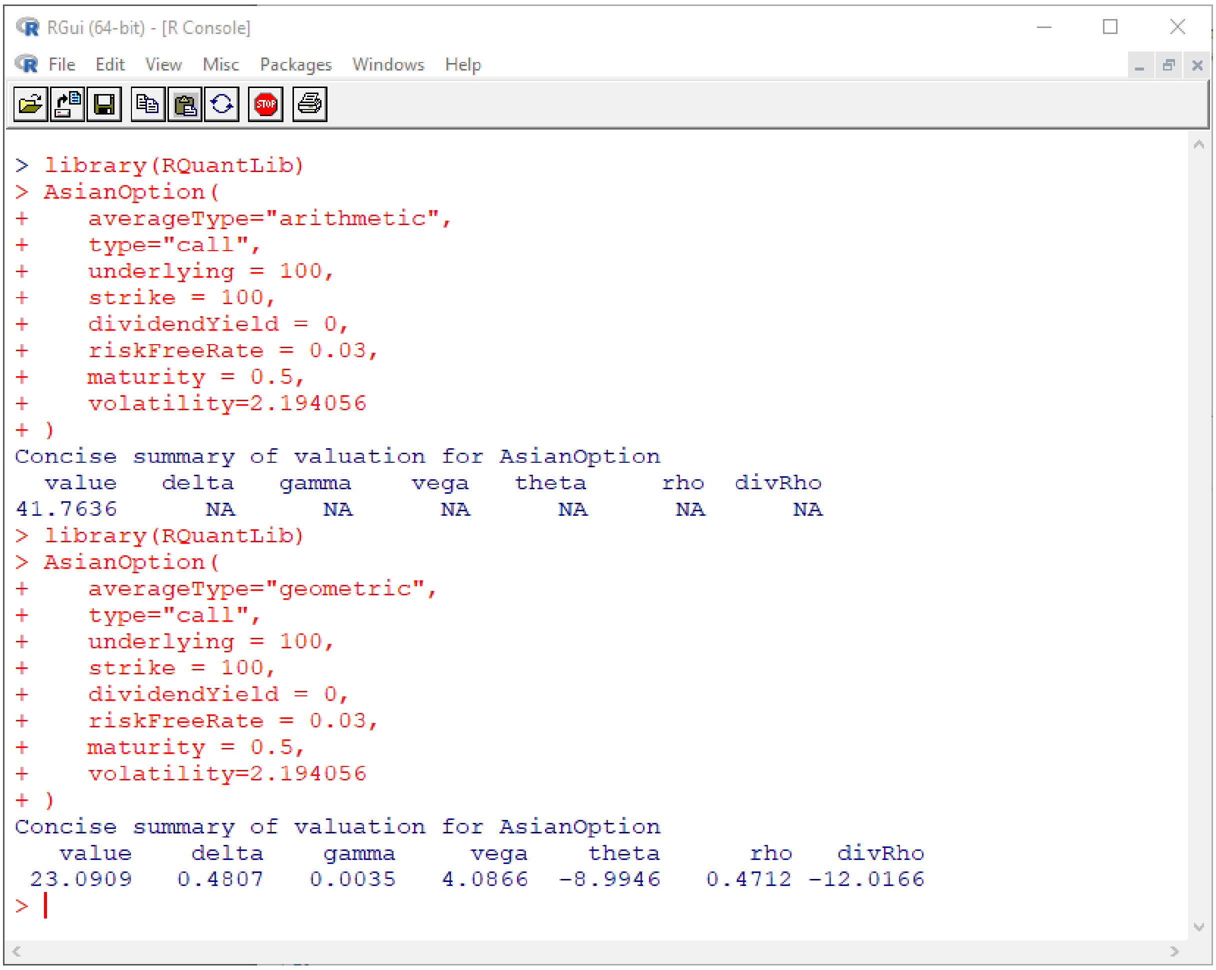
Rappelons d'abord que plus haut nous avons vu le pricing d'une option européenne de type Call avec le package **RQuantLib**:



Et une option américaine du même type:



Et enfin comparons cela à une option asiatique du même type à l'aide de la fonction **AsianOption( )**:



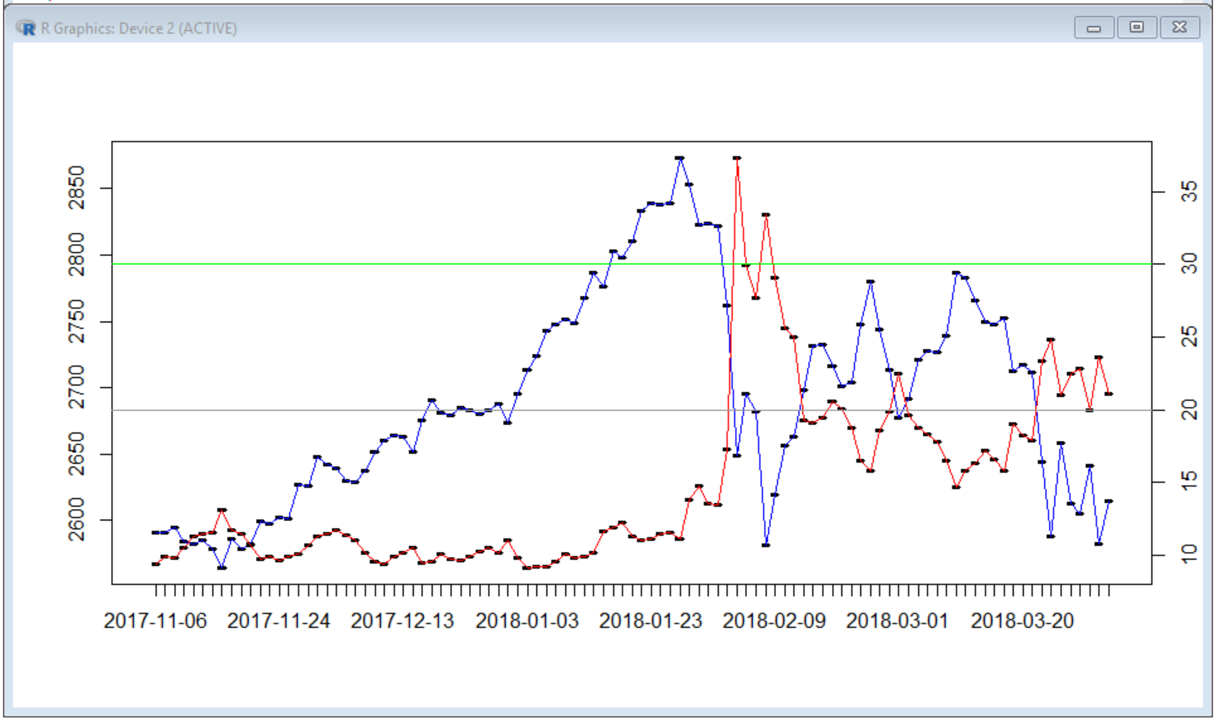
VIX (Index de Volatilité)

Nous avons (non sans mal) démontré dans le cours théorique l'origine d'une des relations permettant de calculer le VIX (Variance Inflation Index).

Voyons comment mettre ce dernier en pratique avec R!

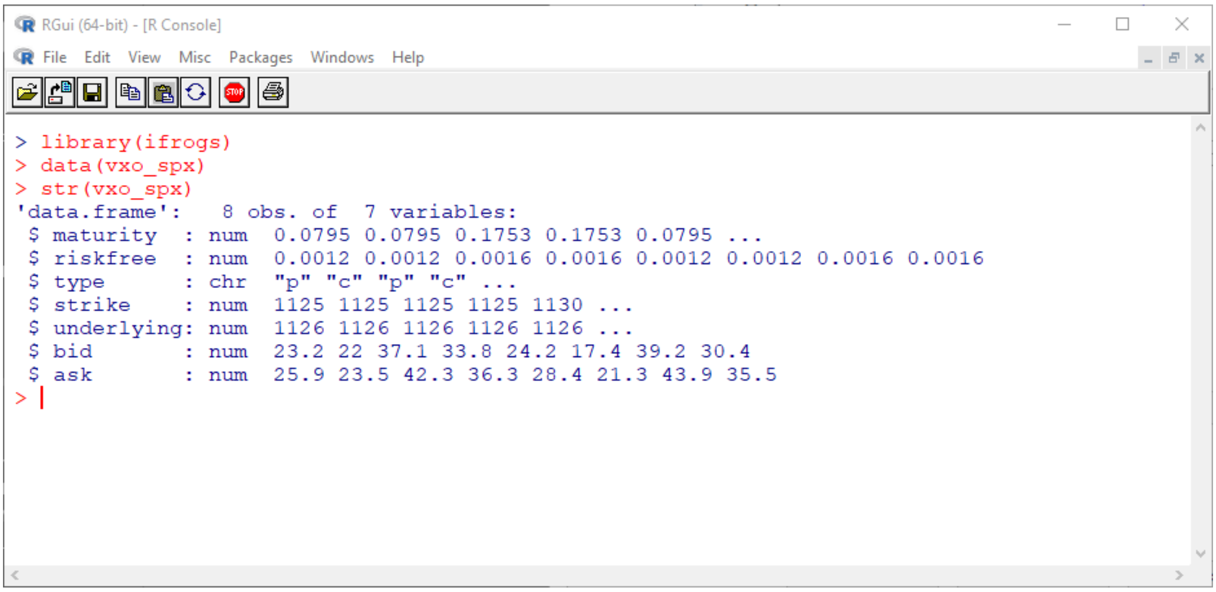
D'abord voyons comment extraire les valeurs précalculées du VIX du S&P 500:

```
> library("xts")
> library("quantmod")
> # S&P 500 vs VIX
> getSymbols(c("^GSPC","^VIX"),from = as.Date("2017-11-04"), to = as.Date("2018-04-04"))
[1] "GSPC" "VIX"
>
> df = as.data.frame(merge(GSPC,VIX))
> df <- data.frame(date = row.names(df), df, row.names = NULL)
>
> plot (df$date,df$GSPC.Adjusted)
> lines(df$date,df$GSPC.Adjusted,col="blue")
> par(new = TRUE)
> plot (df$date,df$VIX.Adjusted,type = "l", axes = FALSE, bty = "n", xlab = "", ylab = "")
> lines(df$date,df$VIX.Adjusted,col="red")
> axis(side=4, at = pretty(range(df$VIX.Adjusted)))
> abline(h=20,col = "gray60")
> abline(h=30,col = "green")
> |
```



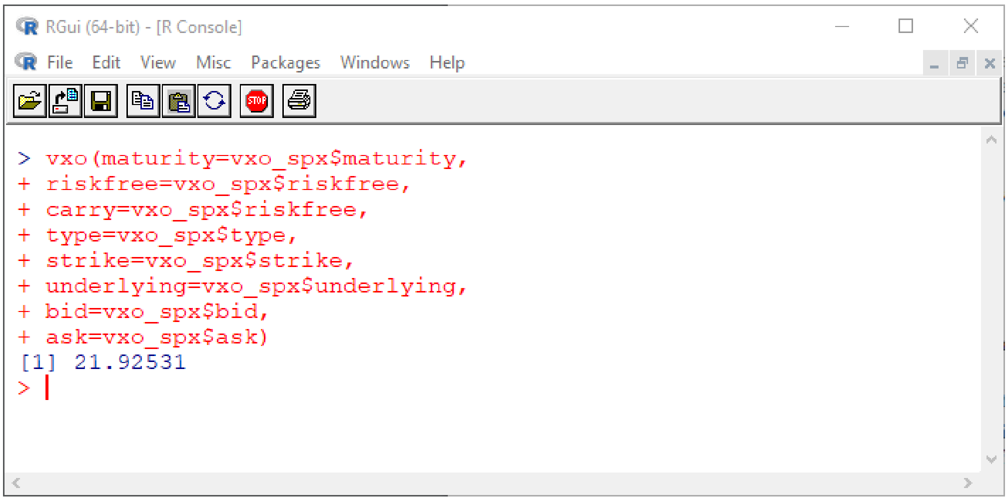
Maintenant, pour le VIX nous allons installer le package **ifrogs** qui n'est plus maintenu depuis 2012 mais qui est disponible en téléchargement en fichier *.zip. Nous avons déjà vu au début de livre comment installer des packages dans des fichiers *.zip, donc nous ne reviendrons pas sur ce point.

Donc d'abord nous chargeons le package avec le jeu de données fourni avec:



Comme nous pouvons le constater, c'est un tout petit jeu de données pour pouvoir faire le calcul à la main si besoin est (pour comparer avec le cours théorique!).

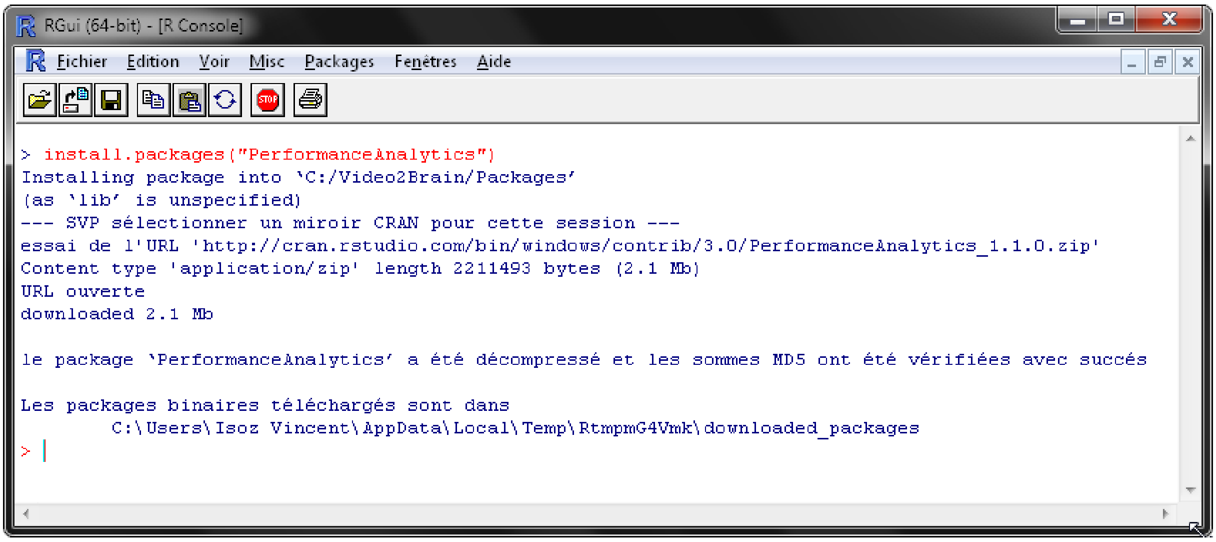
Voyons maintenant comment calculer le VXO:



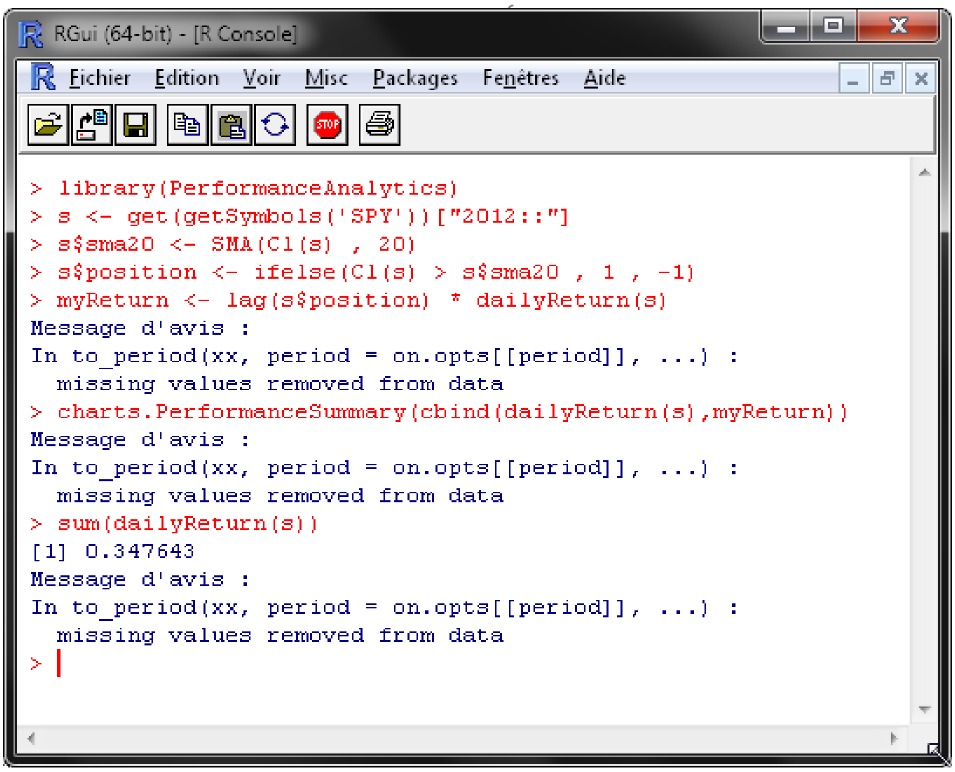
Backtesting de stratégies d'investissement

Backtesting sur modèle avec moyenne mobile (MM20)

Pour faire du backtesting, R peut aussi nous aider avec des packages déjà tout faits. Voici un exemple simple utilisant le package **PerformanceAnalytics**:



Et ensuite, nous lançons un code de backtesting de stratégie (achat lorsque la valeur un peu supérieure au SMA20):



ce qui nous donne:

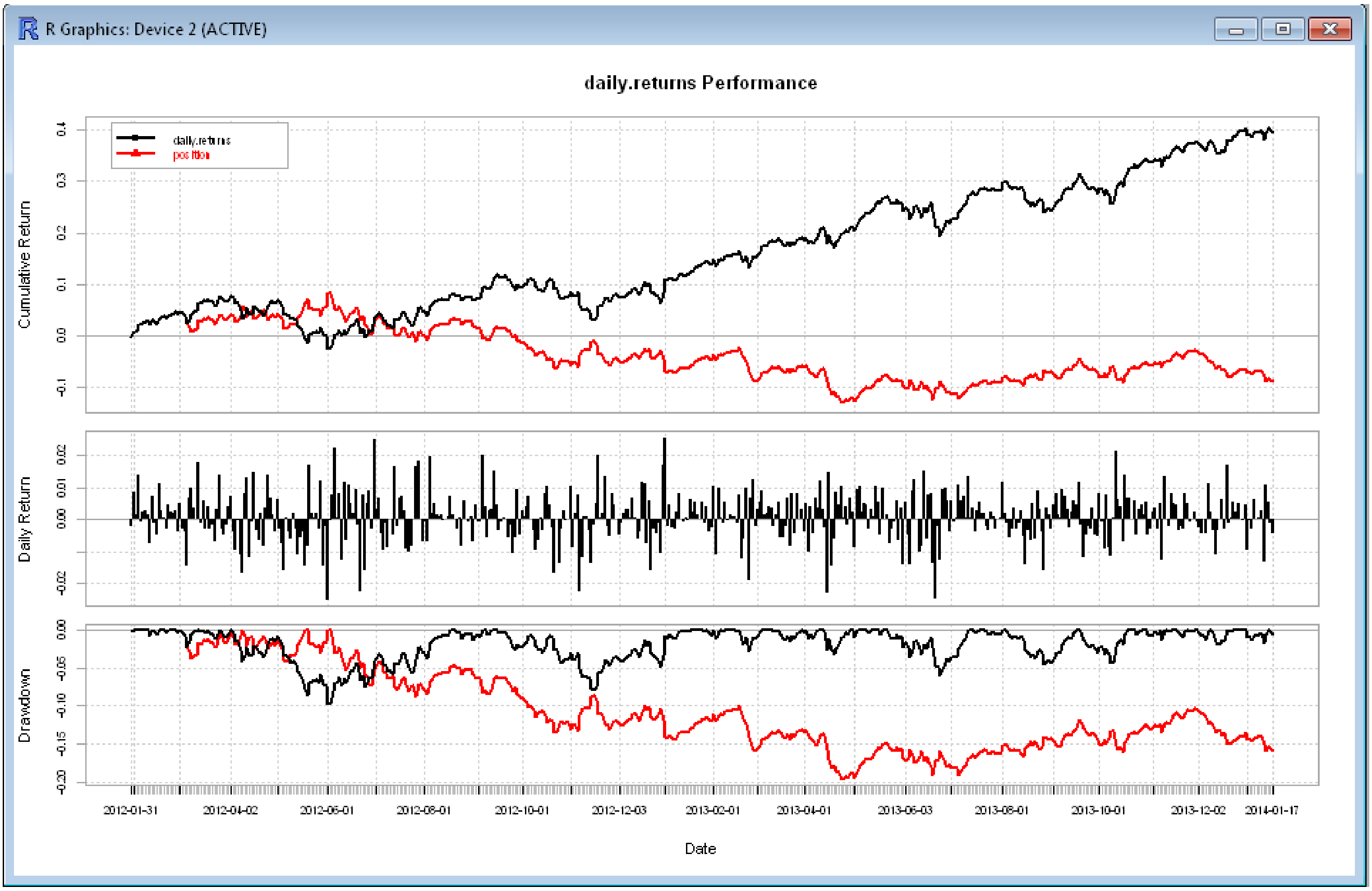


Figure 324 Graph des retours, retours cumulés et Drawdown

Exemples 401.: Analyse et Gestion de portefeuilles

R 3.0.2

Nous allons ici pouvoir appliquer ce que nous avons étudié dans le cours théorique et comparer les résultats – donc avec les mêmes données brutes – par rapport à l'approche approximative que nous avions mis en place avec le solveur du tableur Microsoft Excel.

QEPM (Quantitative Equity Portfolio Management)

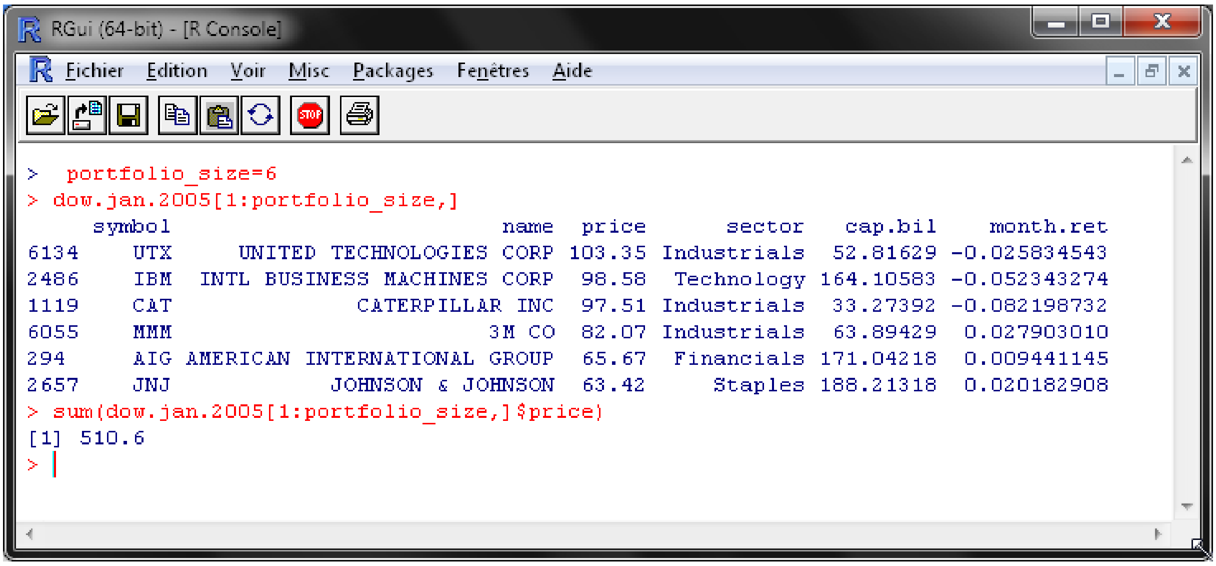
Bon commençons pas les modèles quantitatifs de gestion de portefeuilles d'actions (equity) vraiment les plus triviaux et vérifions au fur et à mesure de notre avancement les résultats obtenus de chacun (ça va être long vu la quantité de modèles théoriques et d'indicateurs de performances empiriques existants...).

Analyse élémentaire a posteriori de portefeuille (position longue sur 1 période)

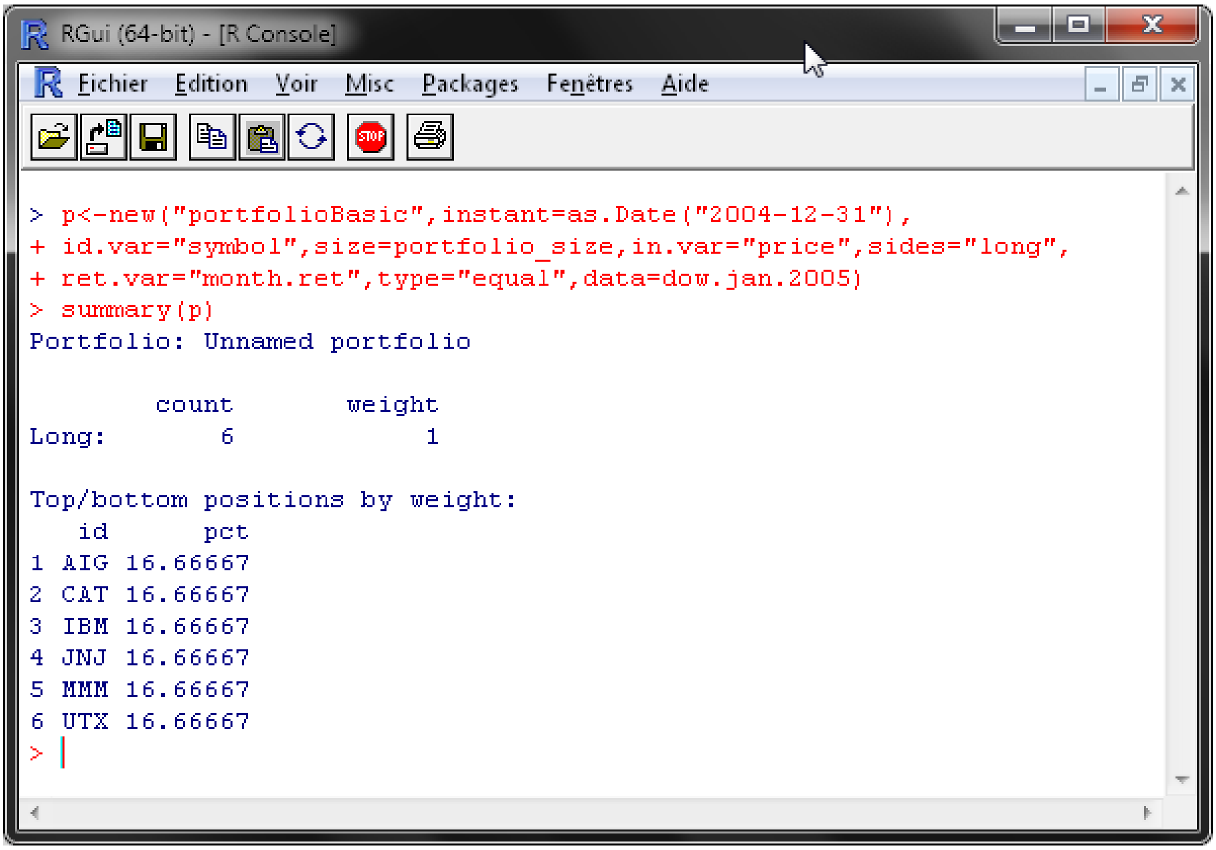
Nous allons commencer très scolairement par charger le package **Portfolio** et un jeu de données par défaut associé:

```
> library(portfolio)
> data(dow.jan.2005)
> dow.jan.2005<-dow.jan.2005[order(dow.jan.2005$price,decreasing=TRUE),]
> dow.jan.2005
     symbol                            name  price        sector   cap.bil    month.ret
6134    UTX      UNITED TECHNOLOGIES CORP 103.35   Industrials  52.81629 -0.025834543
2486    IBM   INTL BUSINESS MACHINES CORP  98.58    Technology 164.10583 -0.052343274
1119    CAT               CATERPILLAR INC  97.51   Industrials  33.27392 -0.082198732
6055    MMM                          3M CO  82.07   Industrials  63.89429  0.027903010
294     AIG AMERICAN INTERNATIONAL GROUP  65.67    Financials 171.04218  0.009441145
2657    JNJ               JOHNSON & JOHNSON  63.42       Staples 188.21318  0.020182908
214      MO               ALTRIA GROUP INC  61.10       Staples 125.41258  0.044680851
270     AXP            AMERICAN EXPRESS CO  56.37    Financials  70.75495 -0.051488427
5581     PG             PROCTER & GAMBLE CO  55.08       Staples 139.72050 -0.029324928
6225    WMT                WAL-MART STORES  52.82     Cyclicals 223.68567 -0.007951534
946      BA                      BOEING CO  51.77   Industrials  43.46594 -0.022599961
1613    XOM               EXXON MOBIL CORP  51.26        Energy 330.69343  0.006632852
1488     DD      DU PONT (E I) DE NEMOURS  49.05     Materials  52.98553 -0.030377166
1190      C                  CITIGROUP INC  48.18    Financials 250.04230  0.018057285
2378     HD                 HOME DEPOT INC  42.74     Cyclicals  93.85512 -0.034627983
1229     KO                   COCA-COLA CO  41.64       Staples 100.74532 -0.003602305
6186     VZ     VERIZON COMMUNICATIONS INC  40.51 Communications 112.17049 -0.112899251
1958     GM              GENERAL MOTORS CORP  40.06     Cyclicals  22.62693 -0.081128308
```

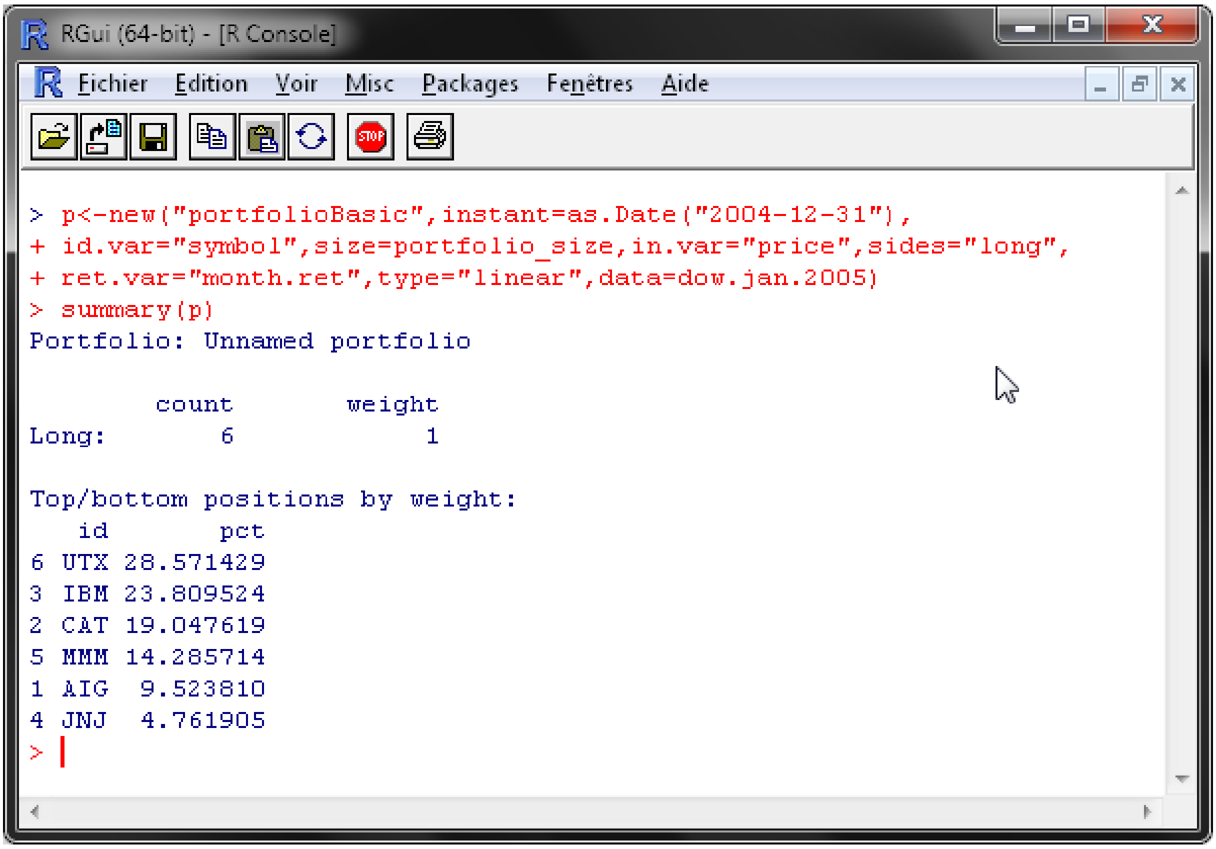
Ensuite, nous définissons combien de type d'actions nous voulons dans le portefeuille et prenons empiriquement les 6 qui ont la plus grande valeur nominale et calculons la valeur globale du portefeuille:



Nous construisons ensuite empiriquement un portefeuille avec les actions à poids égal (remarquez bien la propriété *type="equal"*):

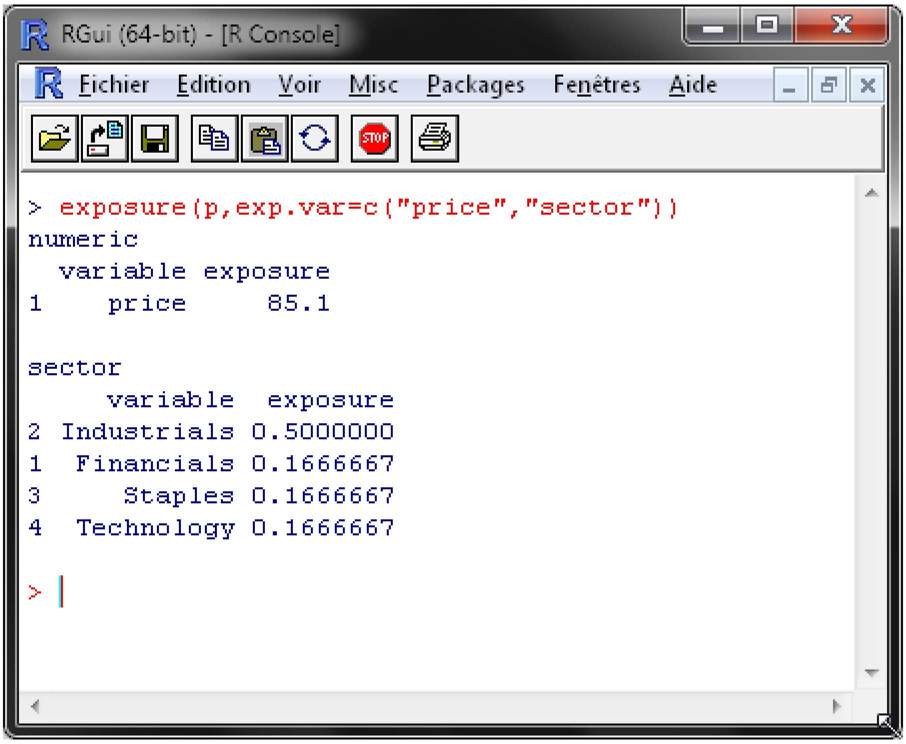


Nous pourrions également tout aussi bêtement choisir des poids proportionnels à la valeur de chaque action dans le portefeuille en mettant la propriété type à *linear*:

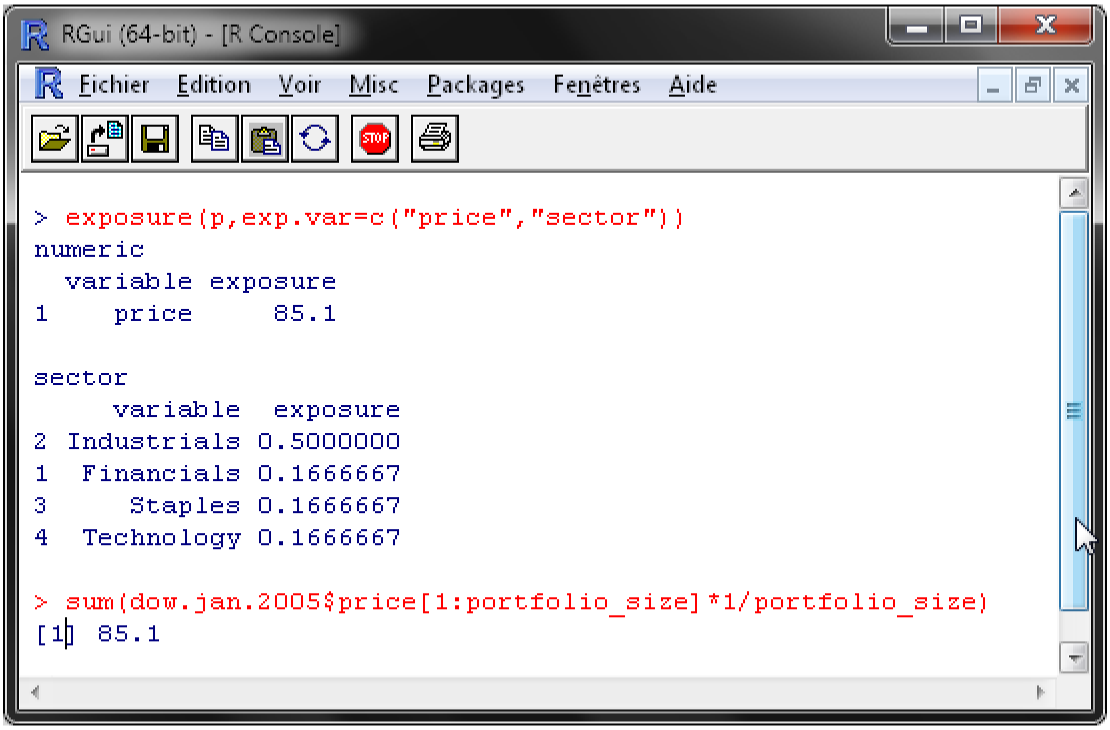


Bref rien d'extraordinaire jusqu'ici... c'est de l'arithmétique élémentaire! Nous poursuivons pour des raisons pédagogiques avec le cas des poids égaux!

Nous regardons les poids y relatifs regroupés par secteurs d'actions:



Rien d'extraordinaire mais contrôlons quand même l'exposition du prix:



Pour ceux qui aiment bien les graphes nous pouvons sortir un plot du résumé ci-dessus:

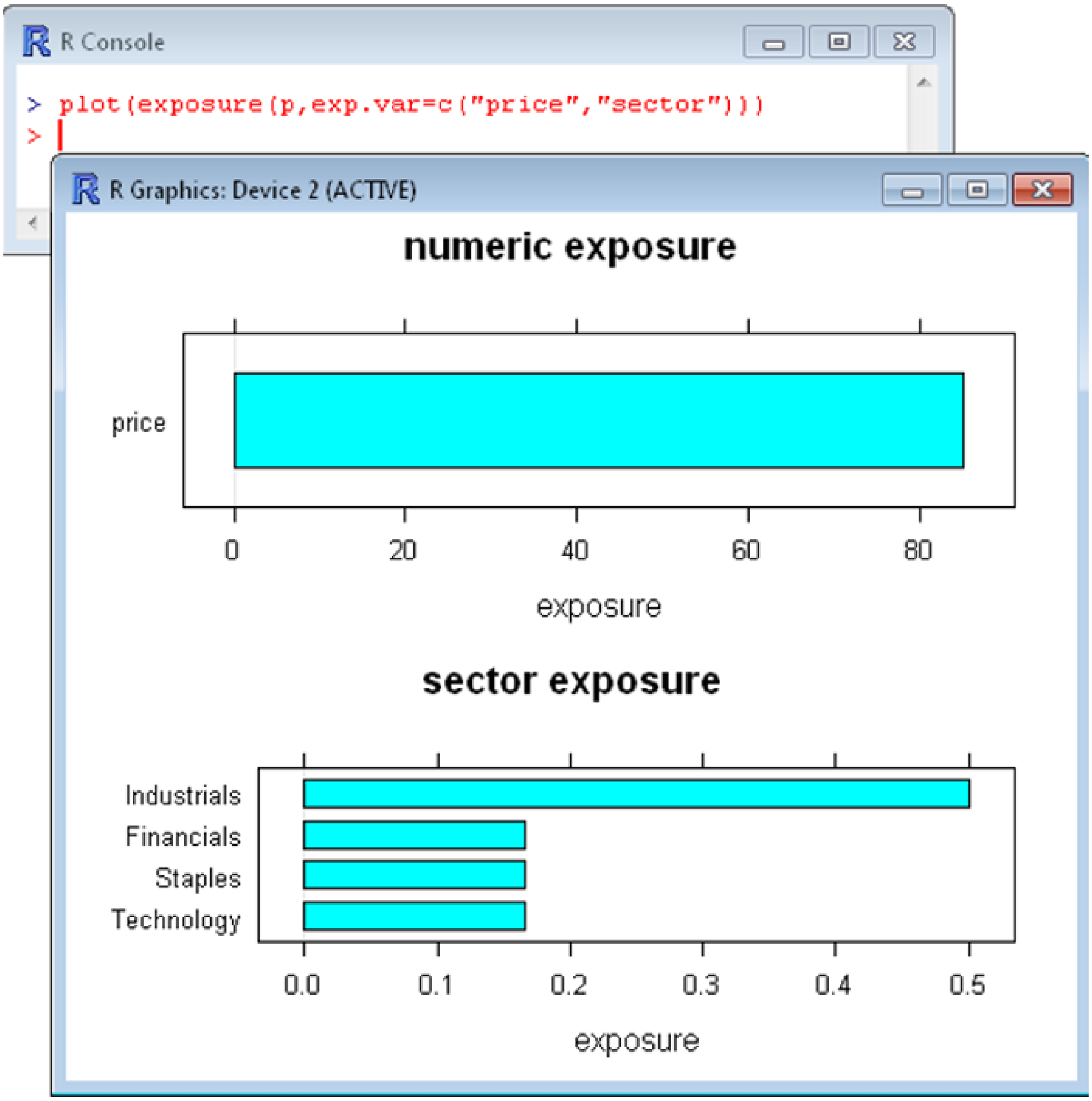
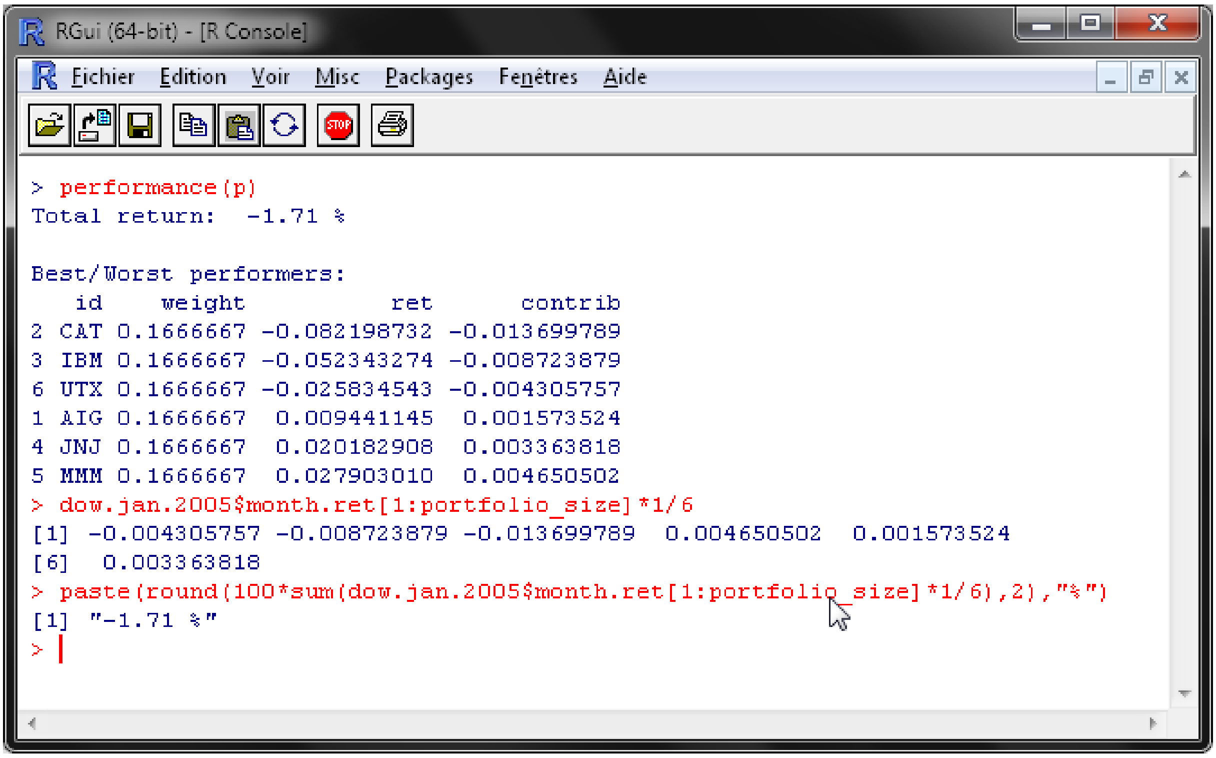
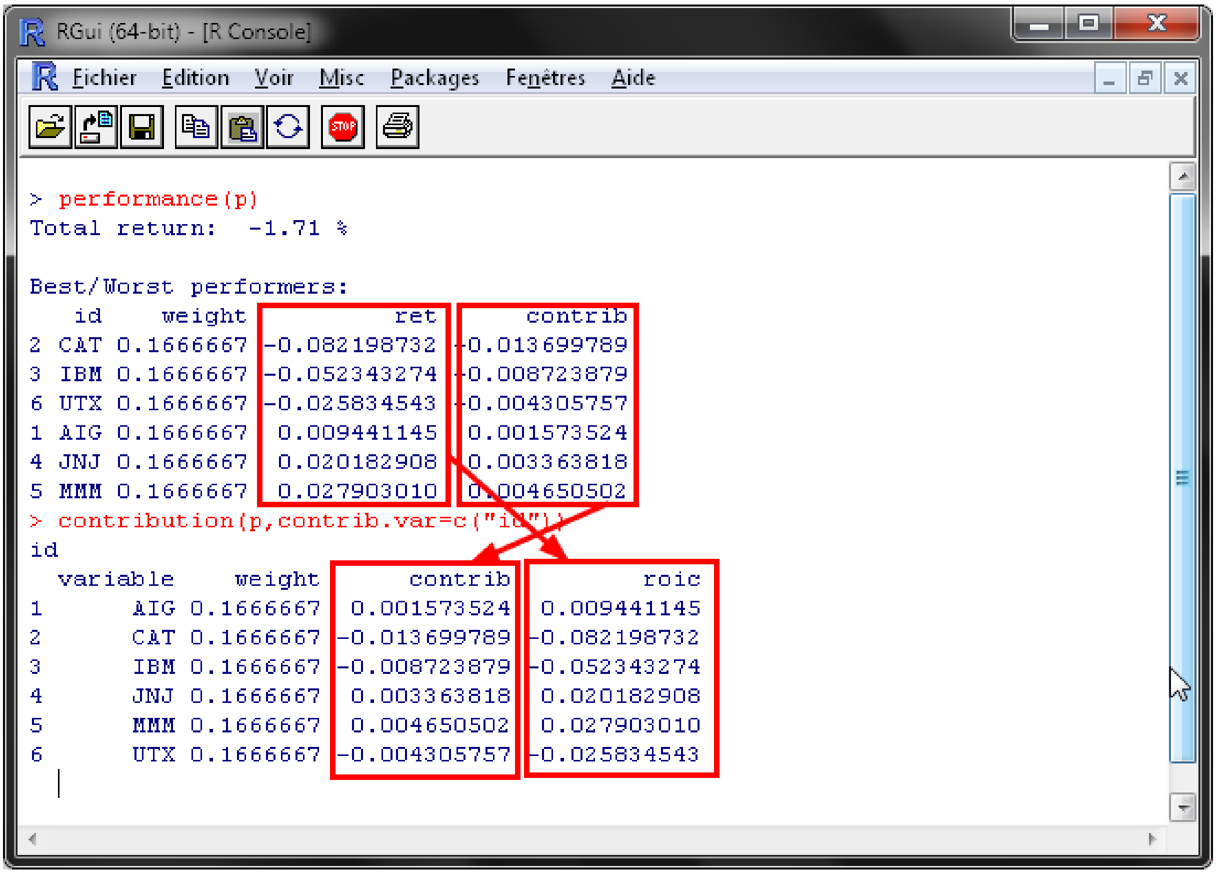


Figure 325 Graph répartition des poids d'un portefeuille avec exposition

Ensuite, nous analysons la performance avec la commande **performance( )** et vérifions en même temps les calculs renvoyés:



Affichons ensuite les contributions à la performance avec la commande **contribution( )** d'abord trivialement par actions:



Et pour ceux qui aiment avoir le roic (Return On Invested Capital) sous forme de graph tornado...:

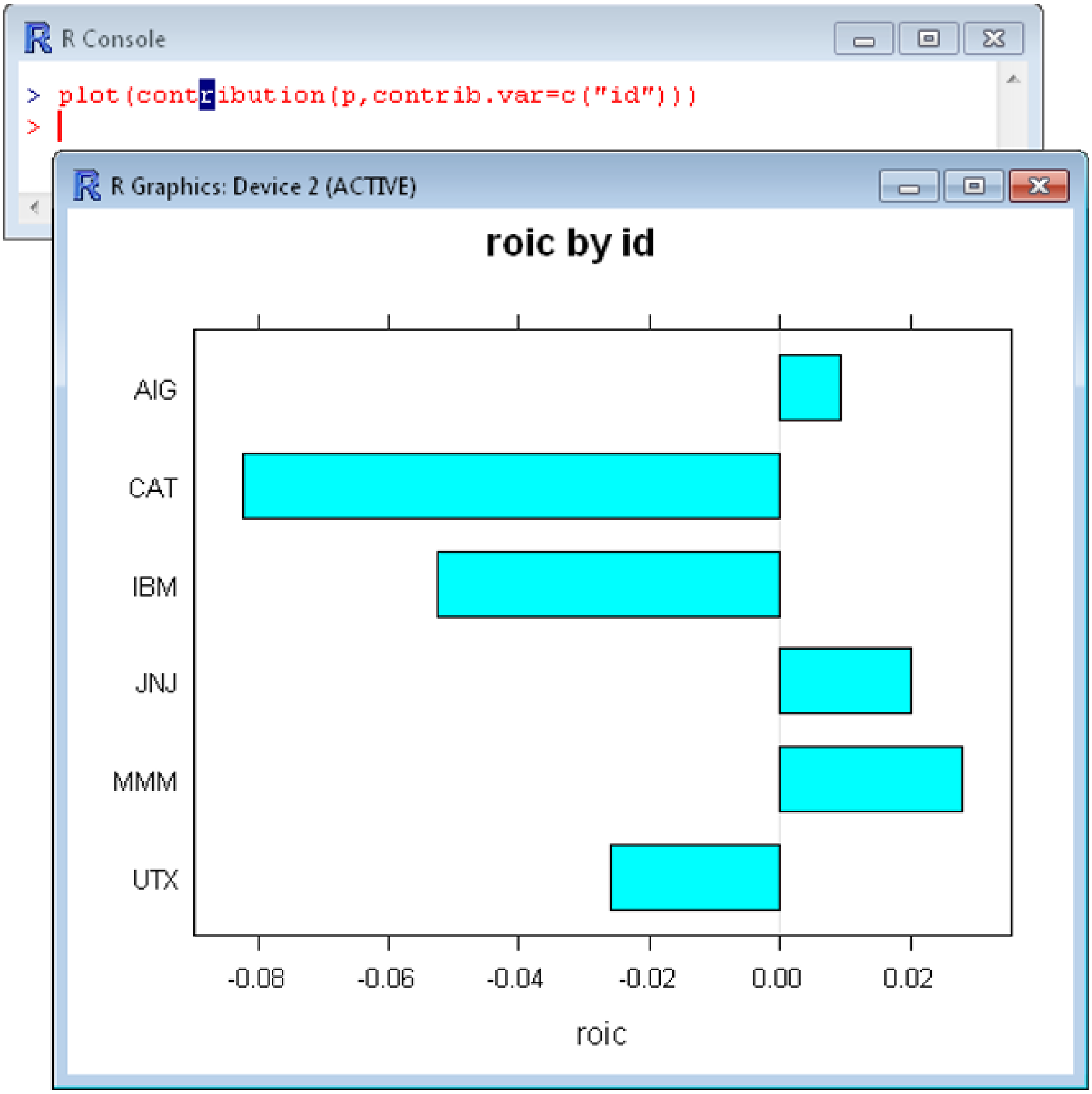
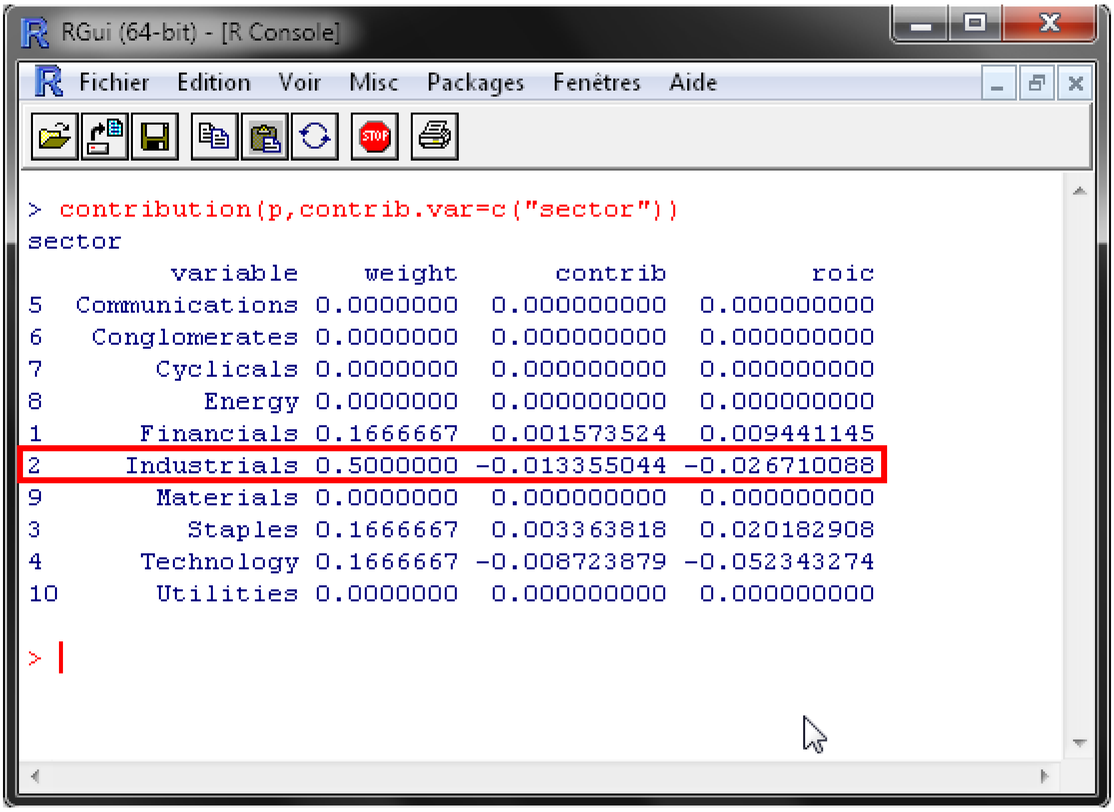


Figure 326 Graph Tornado des performances (ROIC) d'un placement d'un portefeuille

Maintenant observons la contribution à la performance par secteurs:



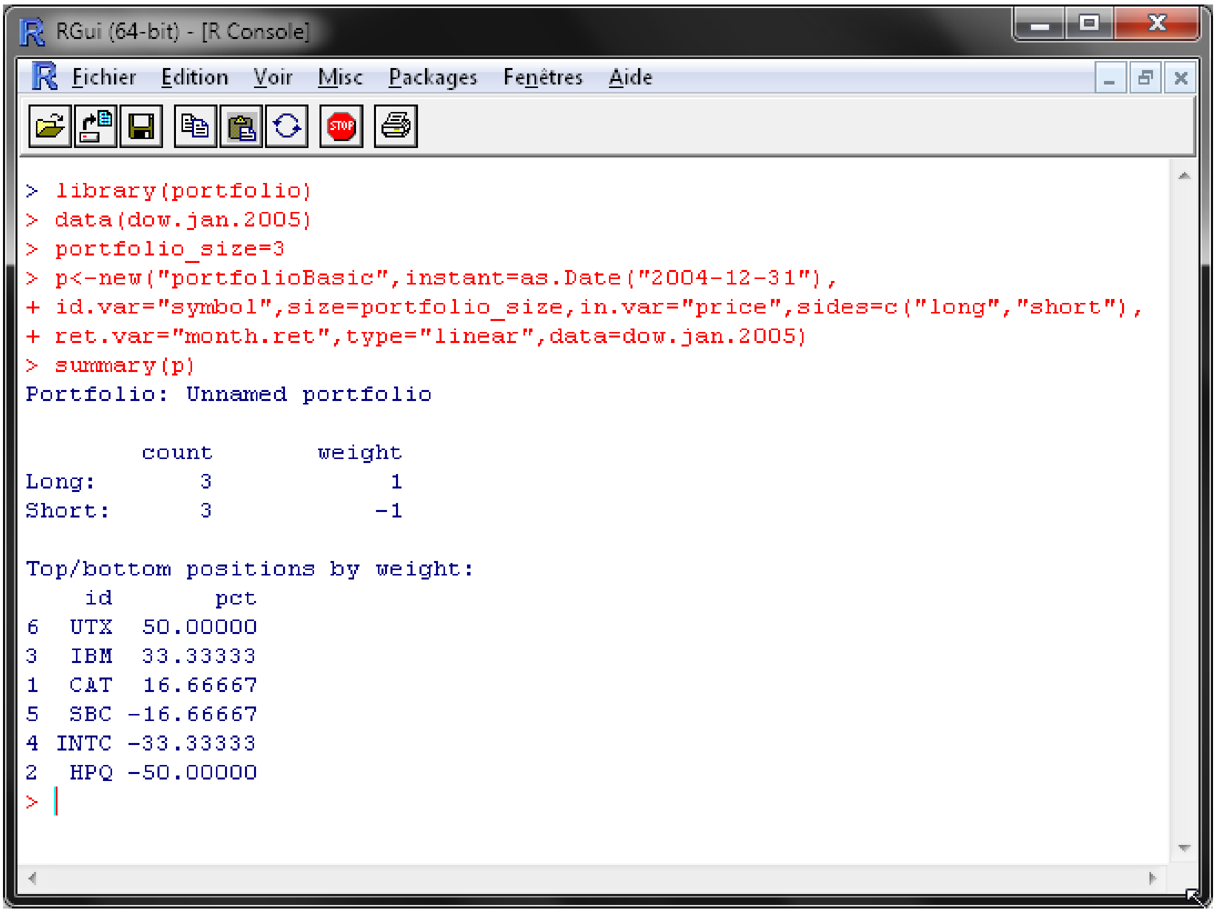
Et concentrons-nous sur la ligne *Technology*. La colonne *weight* est triviale. Retrouvons alors la valeur de la colonne *contrib*. Le secteur *Industrials* est composé des trois actions suivantes:

```
   id    weight          ret      contrib
2 CAT 0.1666667 -0.082198732 -0.013699789
3 IBM 0.1666667 -0.052343274 -0.008723879
6 UTX 0.1666667 -0.025834543 -0.004305757
1 AIG 0.1666667  0.009441145  0.001573524
4 JNJ 0.1666667  0.020182908  0.003363818
5 MMM 0.1666667  0.027903010  0.004650502
```

Si nous faisons la somme de leurs contributions, nous retrouvons la contribution du secteur *Industrials* (logique...). Ensuite, la colonne *roic* pour l'industrie est simplement sa contribution divisée par son poids (donc -0.013355044/0.5).

Analyse élémentaire a posteriori de portefeuille (position long-short sur 1 période)

Pour cet exemple scolaire, nous repartons des mêmes données mais avec un portefeuille long-short avec des poids proportionnels au nominal des actions:

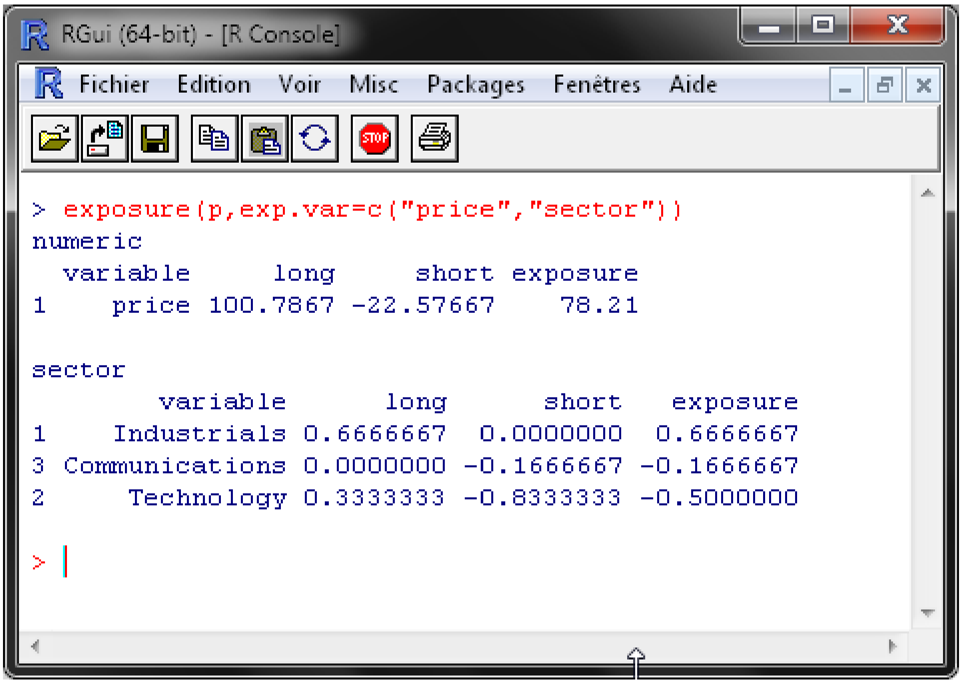


```
> library(portfolio)
> data(dow.jan.2005)
> portfolio_size=3
> p<-new("portfolioBasic",instant=as.Date("2004-12-31"),
+ id.var="symbol",size=portfolio_size,in.var="price",sides=c("long","short"),
+ ret.var="month.ret",type="linear",data=dow.jan.2005)
> summary(p)
Portfolio: Unnamed portfolio

        count       weight
Long:       3            1
Short:      3           -1

Top/bottom positions by weight:
    id        pct
6  UTX   50.00000
3  IBM   33.33333
1  CAT   16.66667
5  SBC  -16.66667
4 INTC  -33.33333
2  HPQ  -50.00000
>
```

Ensuite, nous regardons aussi l'exposition des différentes stratégies (on laisse le soin à l'étudiant de vérifier que les calculs sont justes):



et pour ceux qui préfèrent avoir cela sous forme graphique:

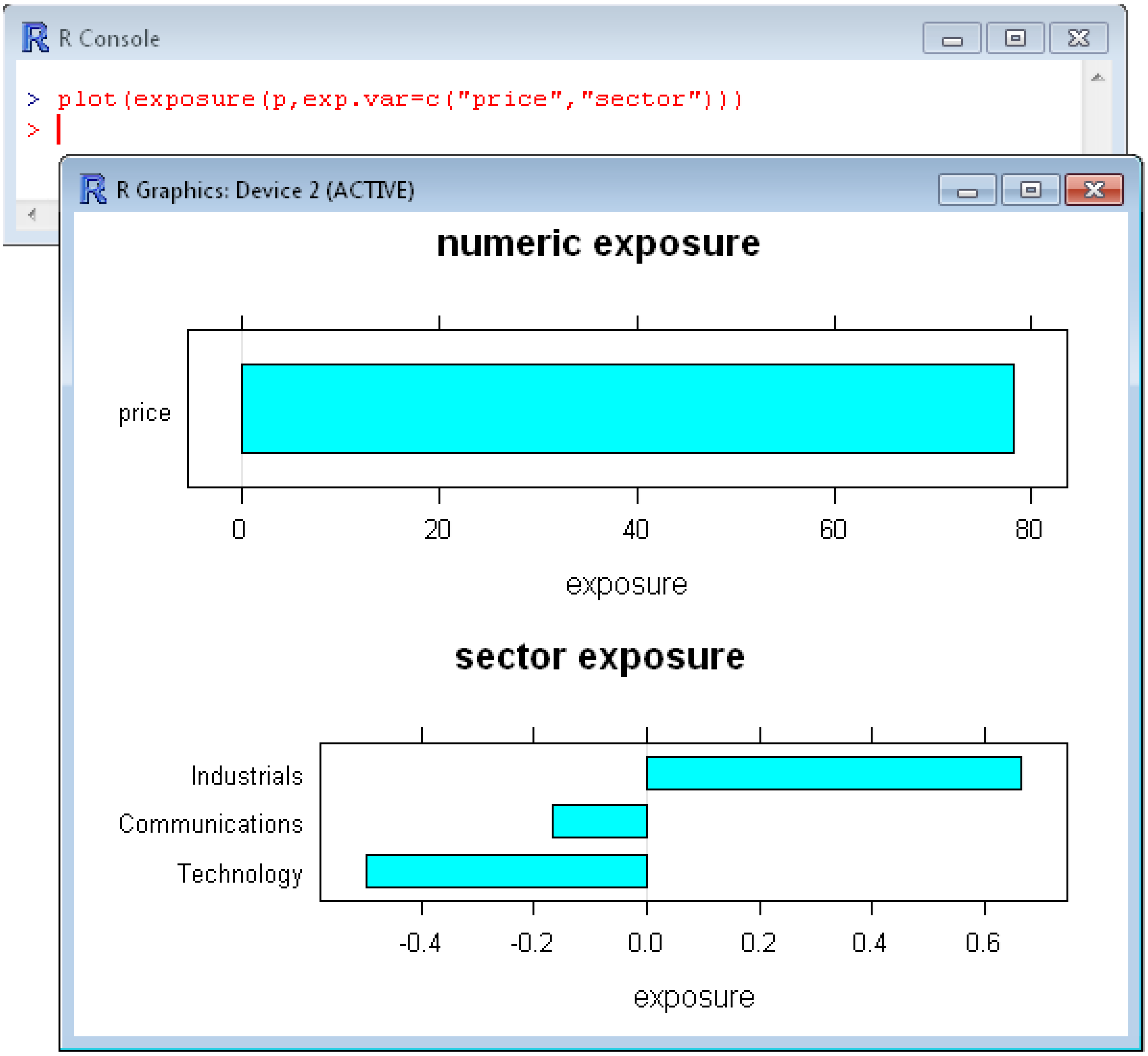
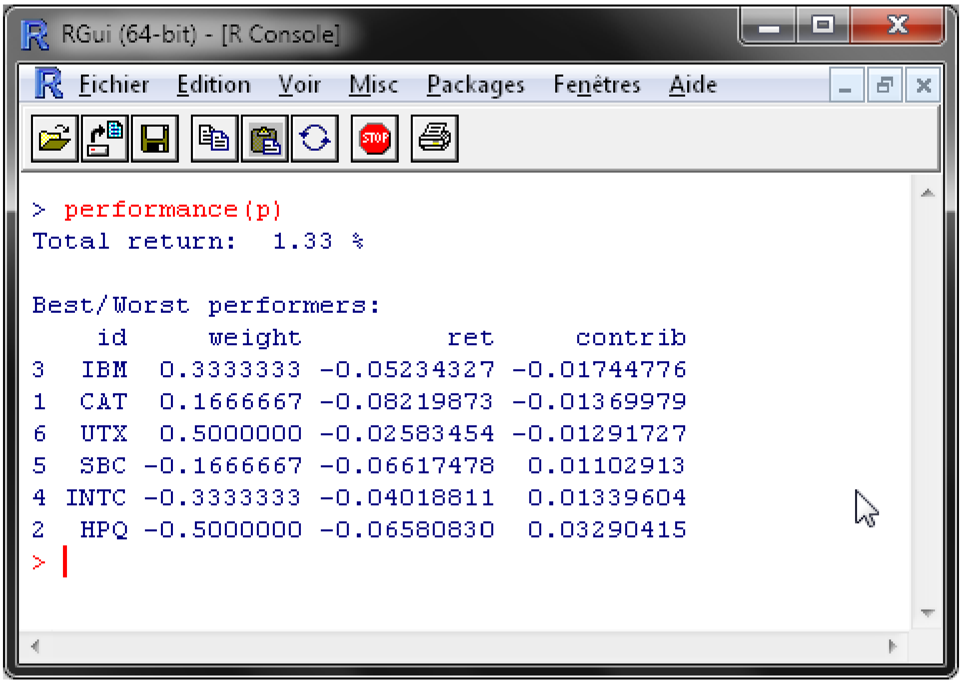
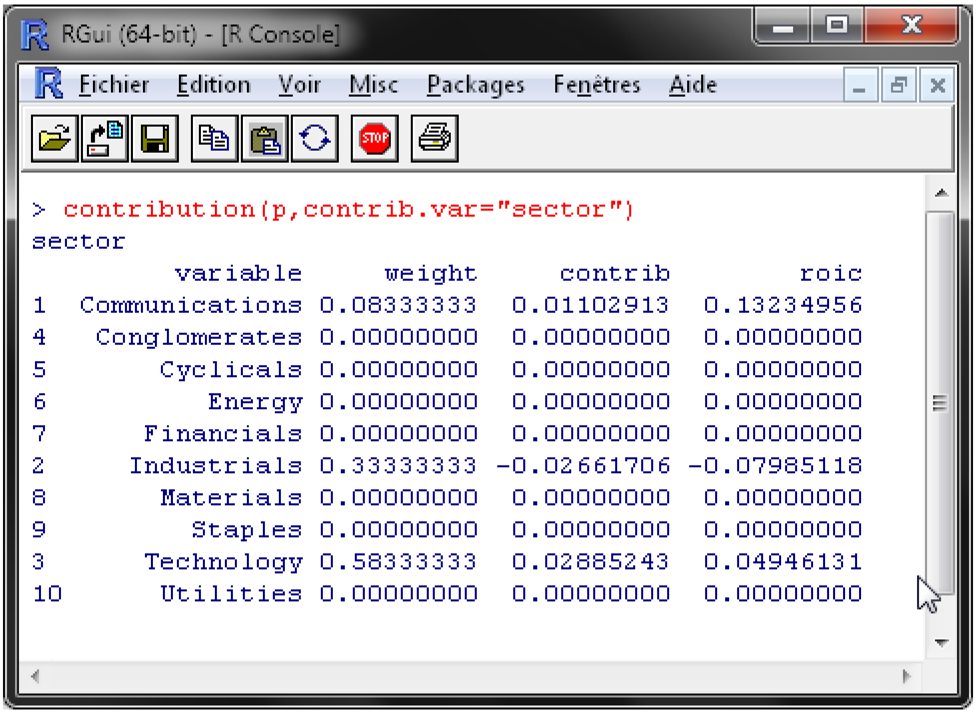


Figure 327 Graph répartition des positions short et long d'un portefeuille avec exposition

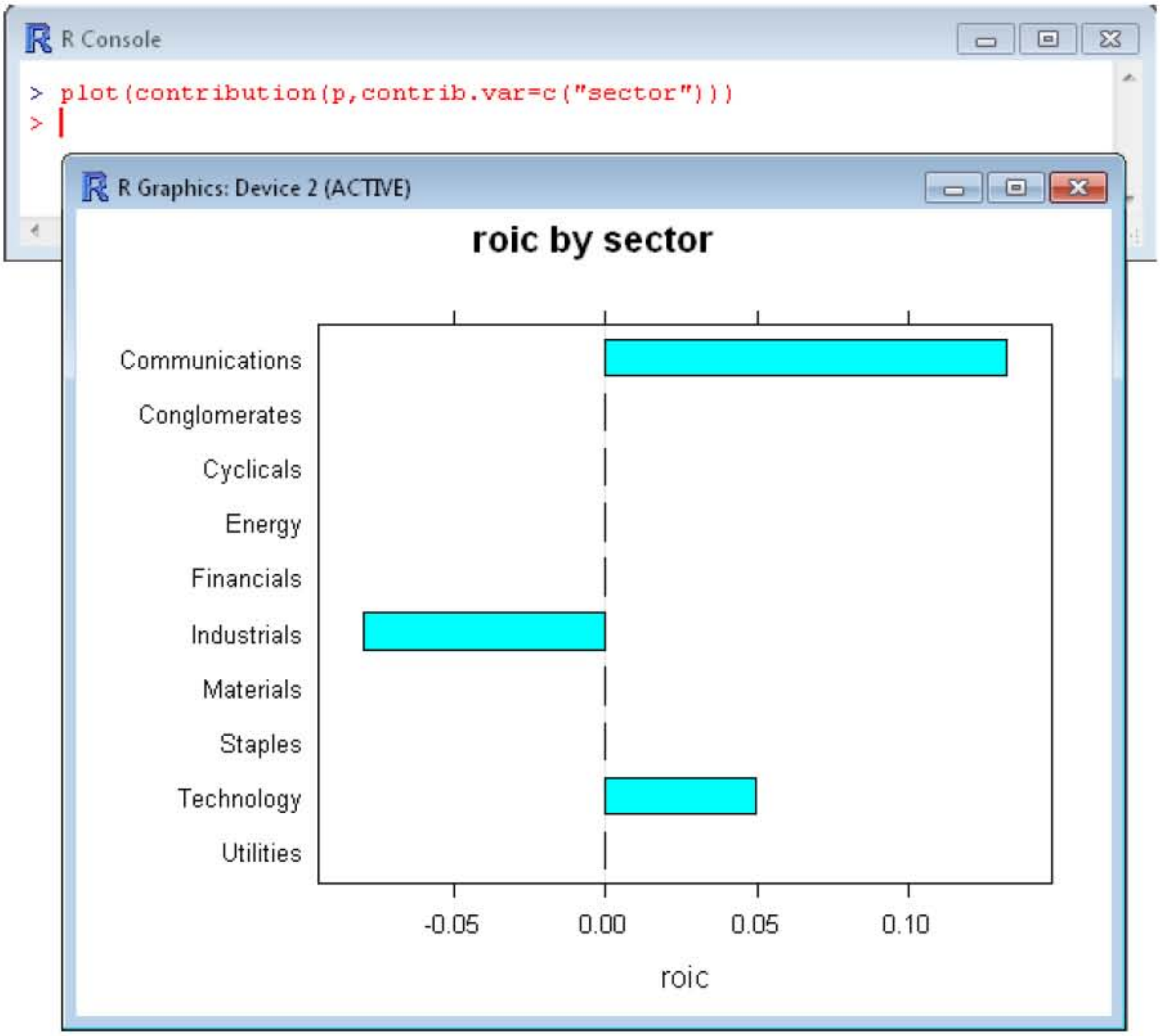
Nous pouvons aussi avoir la performance de ce portefeuille long-short:



et la contribution à la performance par secteur:



et nous pouvons aussi représenter cela sous forme graphique:


R Console
> plot(contribution(p,contrib.var=c("sector")))
>
R Graphics: Device 2 (ACTIVE)
roic by sector
Communications
Conglomerates
Cyclicals
Energy
Financials
Industrials
Materials
Staples
Technology
Utilities
-0.05
0.00
0.05
0.10
roic

Analyse de Brinson d'un portefeuille

Nous allons ici utiliser le package **pa** (portfolio analysis) pour faire une analyse empirique selon le modèle de Brinson d'un portefeuille. Comme il s'agit d'arithmétique élémentaire, nous n'avons pas étudié ce modèle dans le cours théorique et donc nous allons en présenter le principe (ultra simpliste) ici.

Considérons un gestionnaire de portefeuille d'actions qui utilise le S&P 500 comme indice de référence. Un mois donné, il surperforme le S&P 500 de 3% (le "rendement actif" pour rappel qui est la différence entre le rendement du portefeuille et l'indice). Une partie de cette performance s'explique par le fait qu'il a alloué plus de poids du portefeuille à certains secteurs qui ont bien performé. Appelez cet effet: "l'effet d'allocation". Une partie de sa surperformance s'explique par le fait que certains des stocks sélectionnés dans un secteur fait mieux que le même secteur dans l'indice. Appelez cet effet: "l'effet de sélection". Le résidu (la différence de performance entre le rendement actif et les deux effets) peut alors être attribué à une interaction entre répartition et la sélection: "l'effet d'interaction".

Le modèle de Brinson fournit des définitions mathématiques et des méthodes pour calculer ces différents éléments.

Remarque: L'exemple ci-dessus utilise le concept de variable "secteur" comme système de classification lors du calcul de l'effet d'allocation. Mais la même approche peut fonctionner avec toute autre variable qui place chaque titre dans une seule catégorie distincte : pays, industrie et ainsi de suit... En réalité, une approche similaire de classification peut se faire sur la base de variables continues de classification qui sont répartis en gammes distinctes: le quartile supérieur de la capitalisation boursière, le quartile le plus élevé et ainsi de suite... Pour généraliser, nous allons utiliser le terme "catégorie" pour décrire tout schéma de classification qui range un actif dans une et une seule unique catégorie.

Notations utilisées pour la suite:

- $R_{actif} = R_P - R_B$ différence entre rendement du portefeuille dynamique et de l'indice
- w_i^B est le poids de chaque actif *i* dans le benchmark (la somme étant égale à 1)
- w_i^P est le poids de chaque actif *i* dans le portefeuille (la somme étant égale à 1)
- W_j^B est le poids de chaque catégorie *j* dans le benchmark:

  $W_j^B = \sum_i w_i^B , i \in j$ (la somme des W_j^B étant égale à 1)
- W_j^P est le poids de chaque catégorie *j* dans le portefeuille dynamique:

  $W_j^P = \sum_i w_i^P , i \in j$ (la somme des W_j^P étant égale à 1)
- r_i est le rendement de l'actif *i*
- R_j^B est le rendement de la catégorie *j* dans le benchmark: $R_j^B = \sum_i w_i^B r_i , i \in j$
- R_j^P est le rendement de la catégorie *j* dans le portefeuille dynamique:

  $R_j^P = \sum_i w_i^P r_i , i \in j$

- Le rendement du portefeuille actif peut évidemment être calculé de deux manières différentes:

  $$R_P = \sum_{i=1}^{n} w_i^P r_i = \sum_{j=1}^{N} W_j^P R_j^P$$

- Le rendement de l'indice peut évidemment être calculé de deux manières différentes:

  $$R_B = \sum_{i=1}^{n} w_i^B r_i = \sum_{j=1}^{N} W_j^B R_j^B$$

Comme les poids des catégories du portefeuille dynamique sont généralement différents de l'indice (portefeuille dit "benchmark" pour rappel!), l'allocation joue un rôle dans le rendement actif comme nous l'avons déjà mentionné. La même règle au niveau de la sélection des actifs dans une catégorie.

Donc l'intuition derrière l'allocation c'est d'écrire:

$$R_{allocation} = \sum_{j=1}^{N} \left(W_j^P - W_j^B\right) R_j^B$$

où la multiplication par R_j^B se justifie par le fait qu'indirectement nous supposons que dans l'allocation que c'est uniquement le poids des catégories qui joue un rôle et non la sélection des actifs dans chacune des catégories (donc le poids des actifs pour chaque catégorie est supposé alors comme étant le même pour chaque portefeuille), raison pour laquelle pour chacun des portefeuilles et pour une catégorie donnée, le rendement est alors supposé le même et le rendement de référence est celui de l'indice!

L'intuition derrière la sélection c'est d'écrire:

$$R_{sélection} = \sum_{j=1}^{N} \left(R_j^P - R_j^B\right) W_j^B$$

et là l'idée est similaire à celle du dessus à la différence que ce qui nous intéresse c'est pour un poids de catégorie (en prenant celui de l'indice comme référence) d'analyser comment la sélection des actifs à l'intérieur de celle-ci influence le rendement et donc in extenso c'est le rendement de la catégorie qui va différer entre les deux portefeuilles.

Le rendement d'interaction se définit alors comme étant la part inexpliquée tel que:

$$R_{interaction} = R_{actif} - R_{allocation} - R_{selection}$$

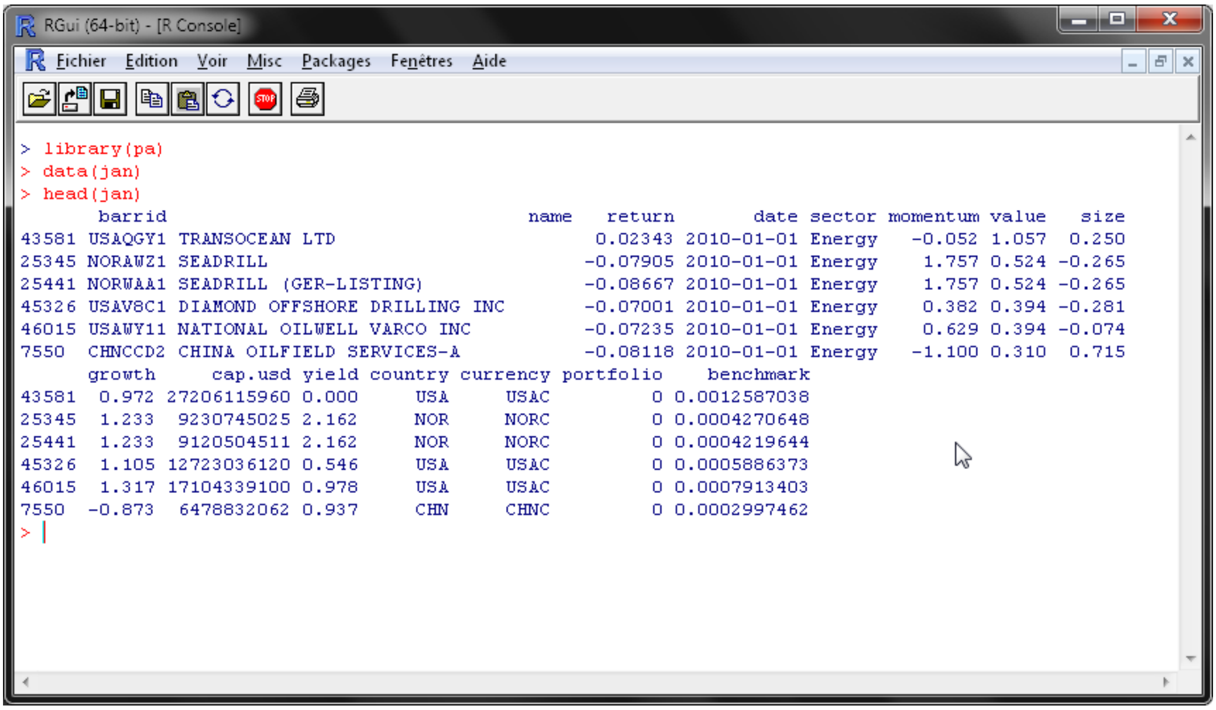
parfois notée:

$$R_{interaction} = R_{actif} - AE - SSE$$

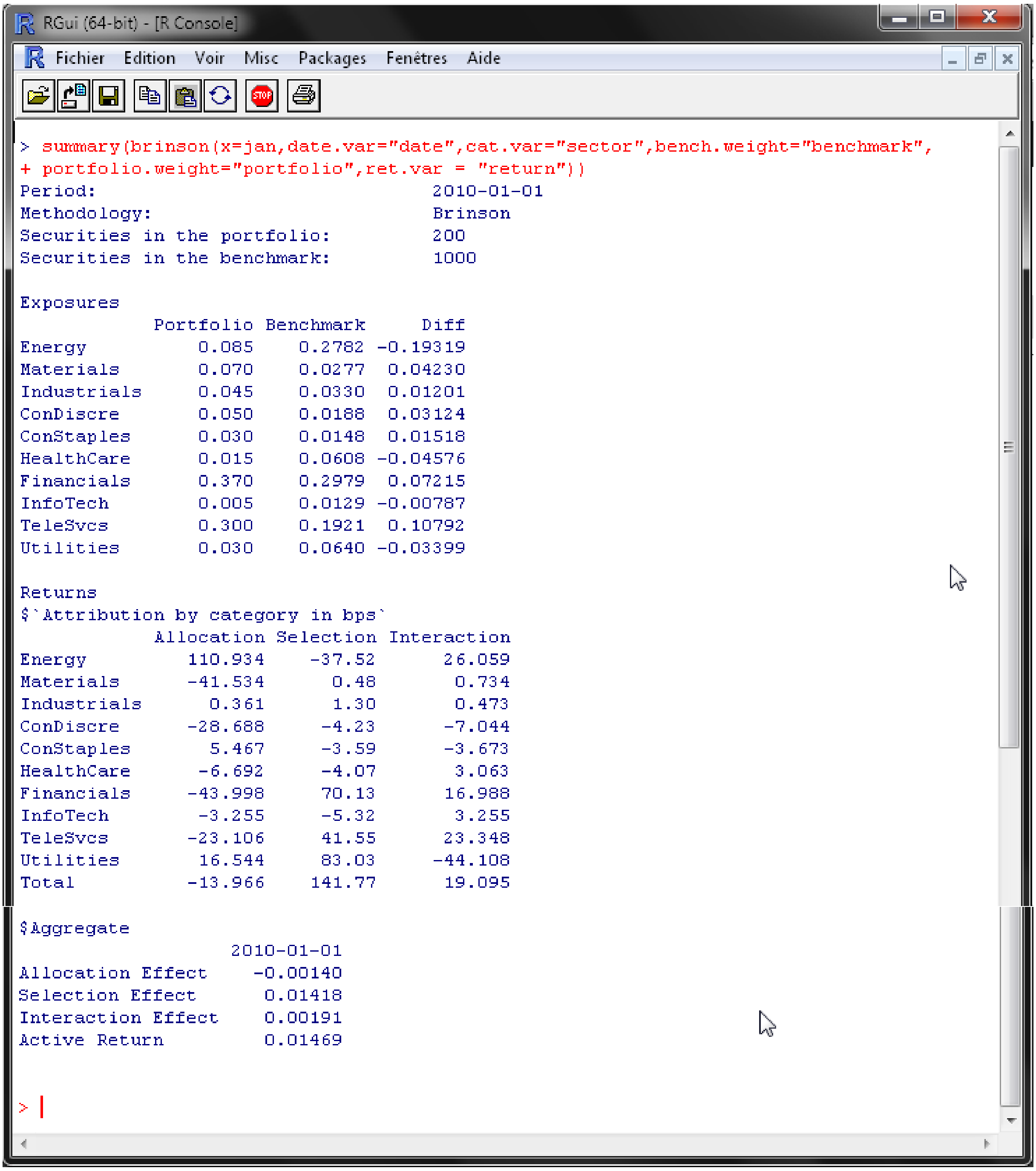
où "AE" signifie "Allocation Effect" et SSE "Security-selection effect".

Faisons un exemple:

D'abord nous chargeons donc le package **pa** et le montrons une partie du jeu *jan* (qui signifie "janvier")



Ensuite, nous utilisons la fonction **brinson( )** pour obtenir un résumé de ce dont nous avons parlé:



```
> summary(brinson(x=jan,date.var="date",cat.var="sector",bench.weight="benchmark",
+ portfolio.weight="portfolio",ret.var = "return"))
Period:                              2010-01-01
Methodology:                         Brinson
Securities in the portfolio:         200
Securities in the benchmark:         1000

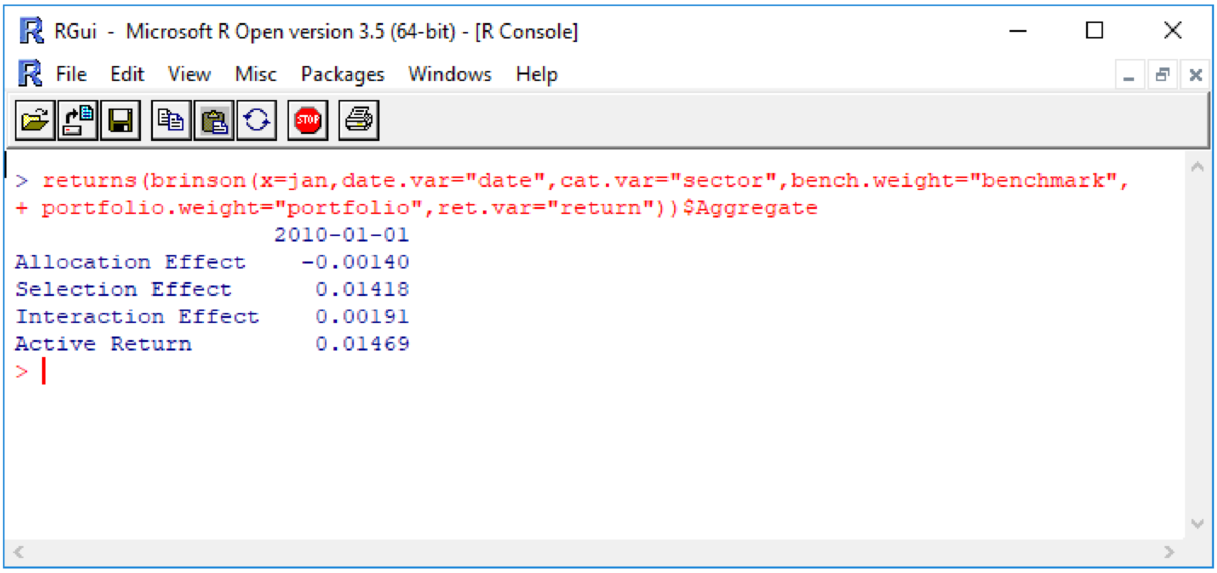
Exposures
            Portfolio Benchmark     Diff
Energy          0.085    0.2782 -0.19319
Materials       0.070    0.0277  0.04230
Industrials     0.045    0.0330  0.01201
ConDiscre       0.050    0.0188  0.03124
ConStaples      0.030    0.0148  0.01518
HealthCare      0.015    0.0608 -0.04576
Financials      0.370    0.2979  0.07215
InfoTech        0.005    0.0129 -0.00787
TeleSvcs        0.300    0.1921  0.10792
Utilities       0.030    0.0640 -0.03399

Returns
$`Attribution by category in bps`
            Allocation Selection Interaction
Energy         110.934    -37.52      26.059
Materials      -41.534      0.48       0.734
Industrials      0.361      1.30       0.473
ConDiscre      -28.688     -4.23      -7.044
ConStaples       5.467     -3.59      -3.673
HealthCare      -6.692     -4.07       3.063
Financials     -43.998     70.13      16.988
InfoTech        -3.255     -5.32       3.255
TeleSvcs       -23.106     41.55      23.348
Utilities       16.544     83.03     -44.108
Total          -13.966    141.77      19.095

$Aggregate
                   2010-01-01
Allocation Effect    -0.00140
Selection Effect      0.01418
Interaction Effect    0.00191
Active Return         0.01469

>
```

On peut extraire seulement une partie du rapport ci-dessus (sachant que le créateur du package **pa** a surchargé la commande **summary( )** et donc que cette dernière ne renvoie pas un objet "habituel"…):



Il est aussi possible de faire un graphique de la situation:

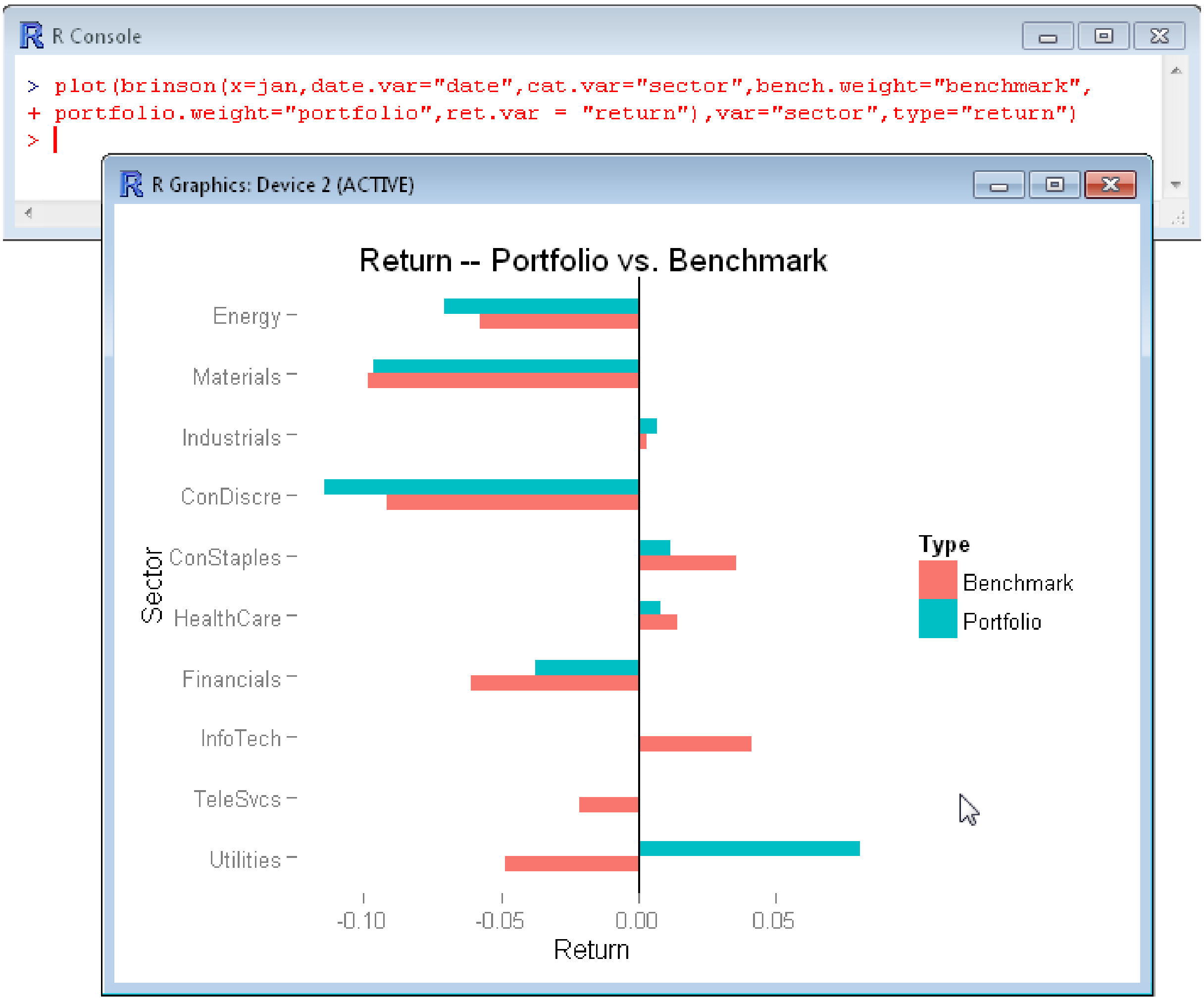
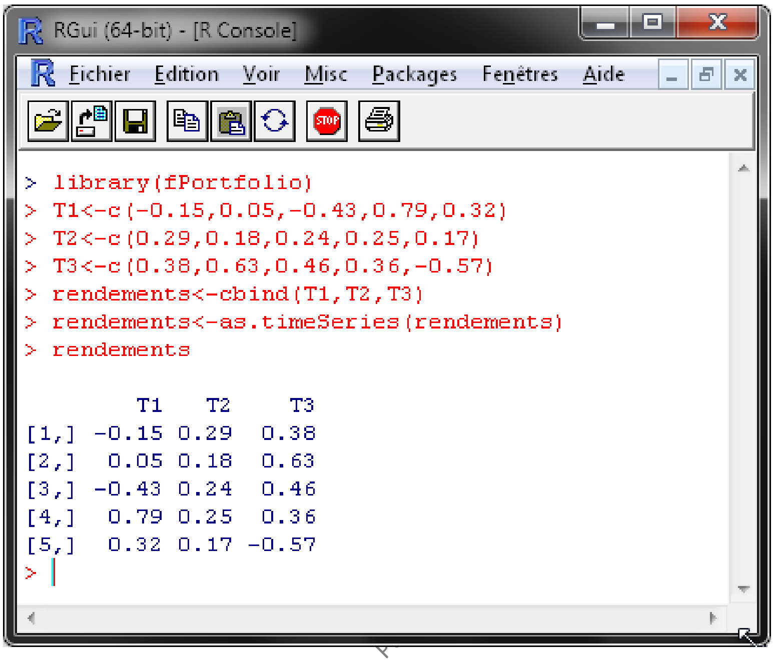


Figure 328 Graph Tornado des performances portefeuille vs benchmark

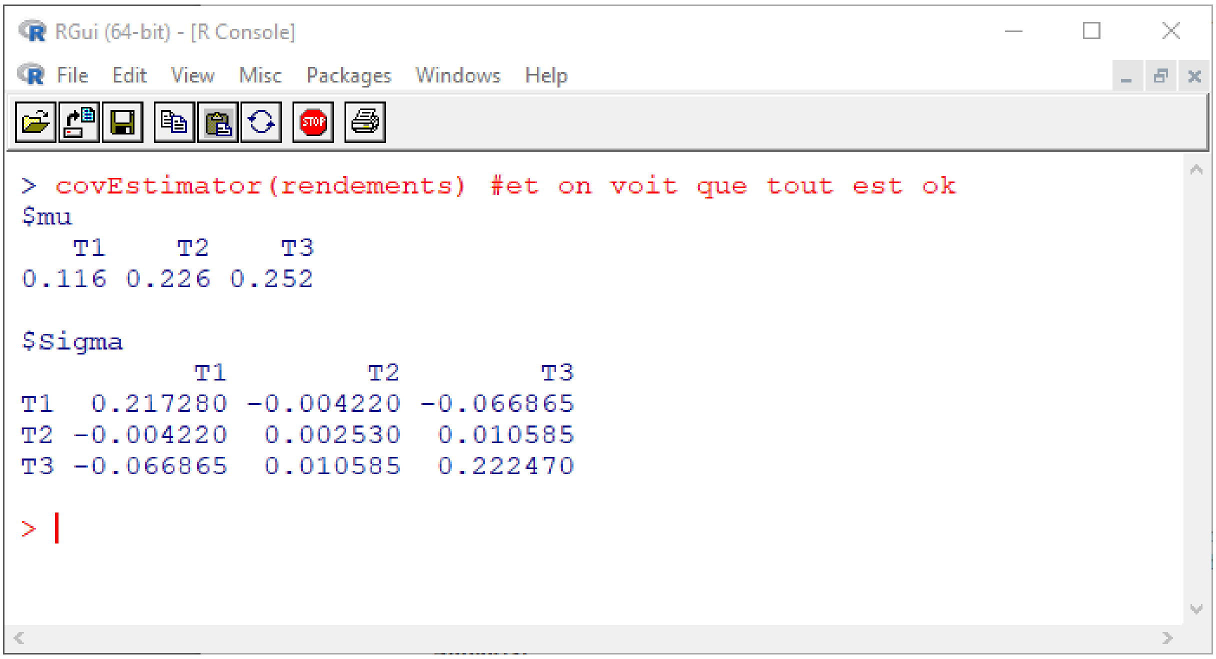
Modèles de portefeuilles de Markowitz et autres

Nous allons voir dans ce qui va suivre, 6 familles (modèles) types de portefeuilles d'actifs classiques parmi la petit vingtaine qui existe dans la pratique.

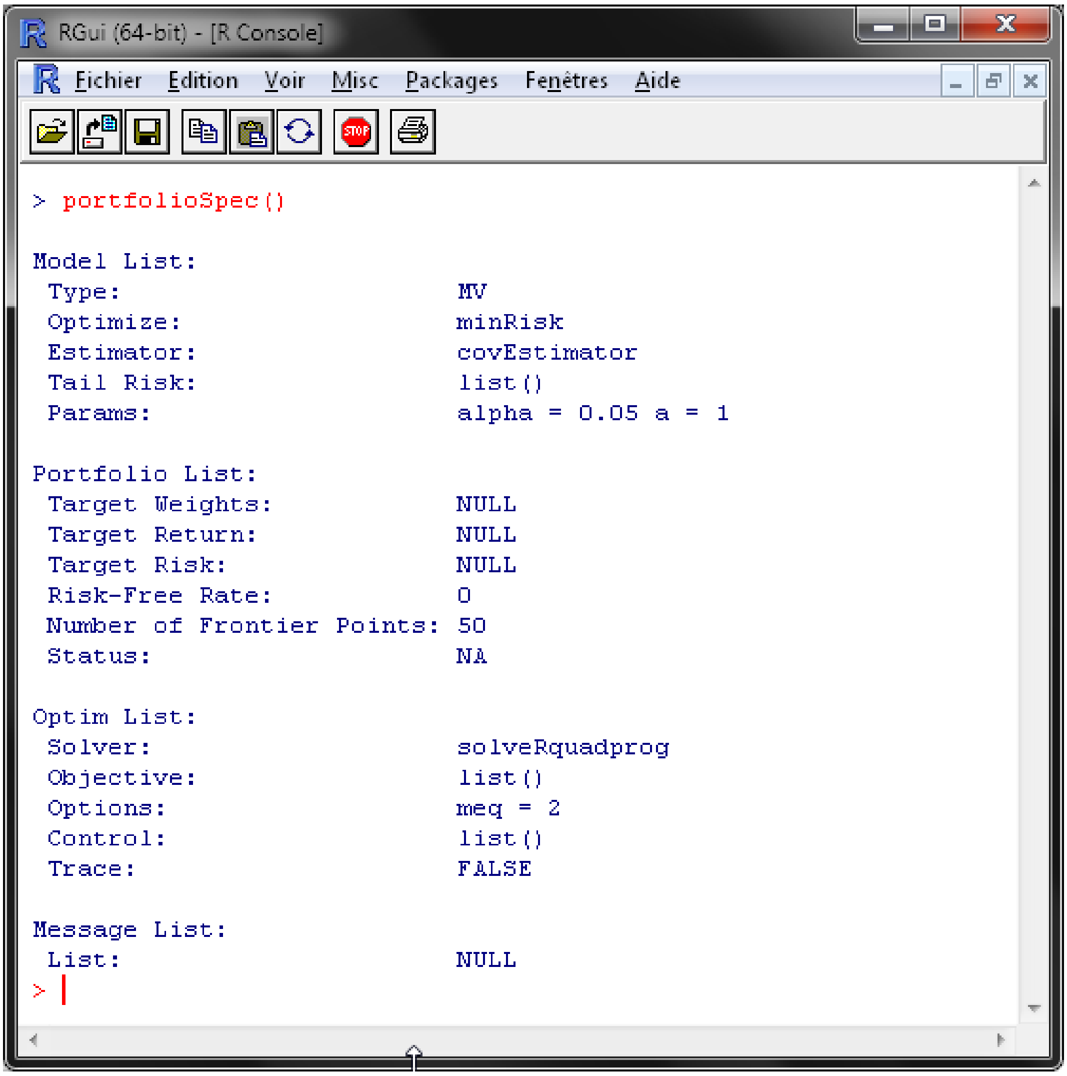
D'abord nous chargeons le package **fPortfolio** et nous construisons le même petit portefeuille de 3 actifs comme nous l'avons fait avec Microsoft Excel et ce exactement avec les mêmes données:



Ensuite, avec la commande **covEstimator( )** du package **fPortfolio**, nous vérifions que nous retrouvons les valeurs calculées à main pour les moyennes et la matrice des variances-covariances (nous voyons que tout est OK!!!):



Ensuite, nous allons spécifier quelques contraintes à notre portefeuille avec la commande **portfolioSpec( )** . Voyons ce qu'il est possible de spécifier:



Nous prendrons les contraintes suivantes en utilisant les commandes **setNFrontierPoints( )** , **setRiskFreeRate( )** et **setWeights( )** :

```
> covSpec<-portfolioSpec()
> setWeights(covSpec)<-c(1/3,1/3,1/3) #poids initial
> setNFrontierPoints(covSpec)<-10
> setRiskFreeRate(covSpec)<-22
> covSpec

Model List:
 Type:                        MV
 Optimize:                    minRisk
 Estimator:                   covEstimator
 Tail Risk:                   list()
 Params:                      alpha = 0.05 a = 1

Portfolio List:
 Portfolio Weights:           0.3333333 0.3333333 0.3333333
 Target Return:               NA
 Target Risk:                 NA
 Risk-Free Rate:              22
 Number of Frontier Points:   10
 Status:                      NA

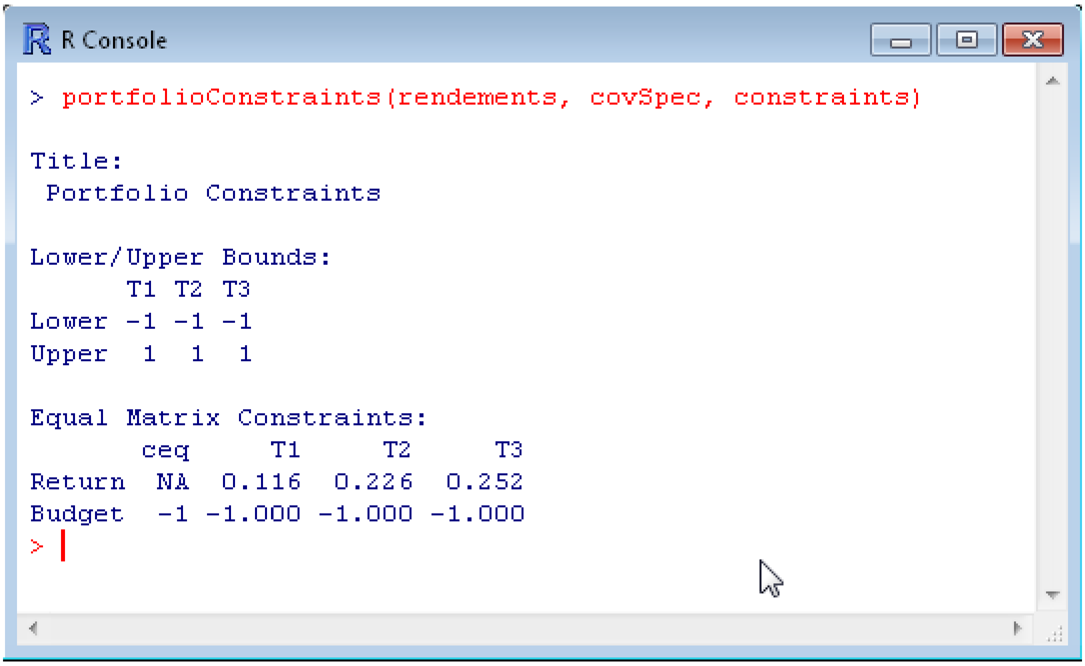
Optim List:
 Solver:                      solveRquadprog
 Objective:                   list()
 Options:                     meq = 2
 Control:                     list()
 Trace:                       FALSE

Message List:
 List:                        NULL
>
```

Une fois ces premières contraintes définies, nous allons spécifier déjà pour la suite si nous voulons une stratégie Long (*LongOnly*), Short (*Short*) ou mixe (en spécifiant les vecteurs comme ci-dessous):

```
> constraints <- c("minW[1:3] = -1", "maxW[1:3] = 1")
> #ou: constraints<-"Short"
> #ou: constraints<-"LongOnly"
> #ou: constraints <- c("minW[1:3] = c(-1,-1,-1)", "maxW[1:3] = c(1,1,1)")
>
```

Ensuite, avec la commande **portfolioConstraints( )** :



Jusque là tout est toujours conforme à ce que nous avons fait dans le cours théorique! Nous poursuivons donc!

Portefeuille tangentiel (maximisation ratio de Sharpe)

Nous utilisons maintenant la commande **tangencyPortfolio( )** du package **fPortfolio** pour avoir un résumé du portefeuille efficient qui utilise implicitement la maximisation du ratio de Sharpe:

```
R Console
> tangencyPortfolio(rendements, covSpec, constraints)

Title:
 MV Tangency Portfolio
 Estimator:         covEstimator
 Solver:            solveRquadprog
 Optimize:          minRisk
 Constraints:       minW maxW

Portfolio Weights:
     T1      T2      T3
 1.0000 -0.0018  0.0018

Covariance Risk Budgets:
     T1      T2      T3
 1.0005  0.0000 -0.0006

Target Return and Risks:
  mean     mu    Cov  Sigma   CVaR    VaR
0.1160 0.1160 0.4659 0.4659 0.4296 0.4296

Description:
 Sun Feb 23 12:25:24 2014 by user: Isoz Vincent
> 
```

Comme nous pouvons le voir ce n'est pas le même résultat que celui obtenu dans le cours théorique avec les algorithmes évolutionnaires où nous avions trouvé un portefeuille qui performait 100% plus. Il faut savoir que les auteurs de package **fPortfolio** ne garantissent en **aucun cas** l'efficacité de leur solution!

Portefeuille efficient (rendement portefeuille fixé)

Dans le cours théorique nous nous étions fixés un rendement précis à atteindre qui était de 20%. Pour trouver les poids actifs permettant d'atteindre cet objectif, nous utiliserons la commande **efficientPortfolio( )** du package **fPortfolio**:

```
> covSpec<-portfolioSpec()
> setWeights(covSpec)<-c(1/3,1/3,1/3) #poids initial
> setNFrontierPoints(covSpec)<-10
> setRiskFreeRate(covSpec)<-22
> setTargetReturn(covSpec)<-0.20
> efficientPortfolio(rendements,covSpec,constraints)

Title:
 MV Efficient Portfolio
 Estimator:          covEstimator
 Solver:             solveRquadprog
 Optimize:           minRisk
 Constraints:        minW maxW

Portfolio Weights:
     T1      T2      T3
 0.2331  0.7805 -0.0137

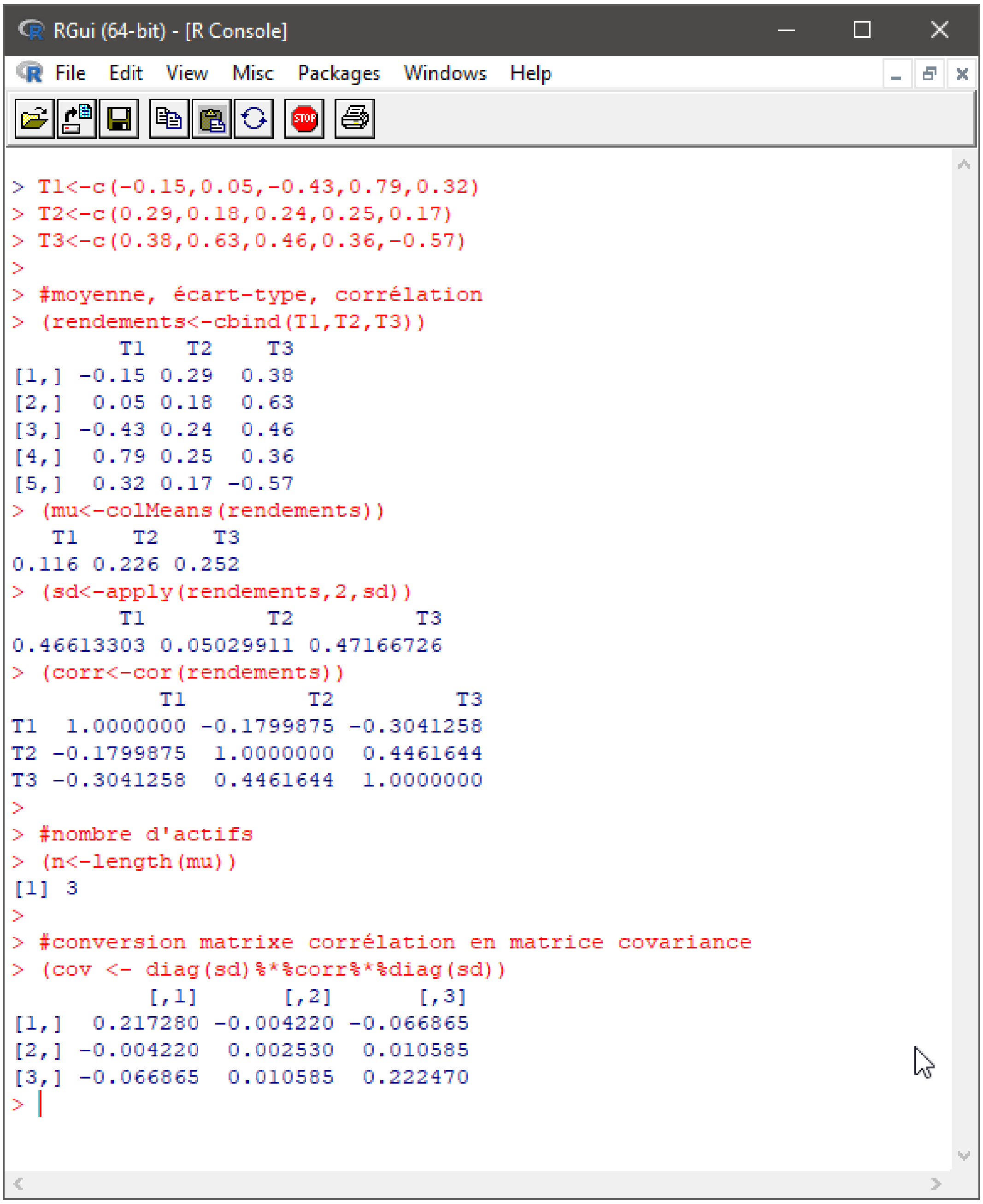
Covariance Risk Budgets:
    T1     T2     T3
0.9335 0.0548 0.0117

Target Returns and Risks:
   mean     Cov    CVaR     VaR
 0.2000  0.1098 -0.0808 -0.0808

Description:
 Sun Feb 23 12:49:28 2014 by user: Isoz Vincent
>
```

Comme nous pouvons le voir, les résultats des poids est proche des calculs effectués à la main dans le cours théorique mais pas identique.

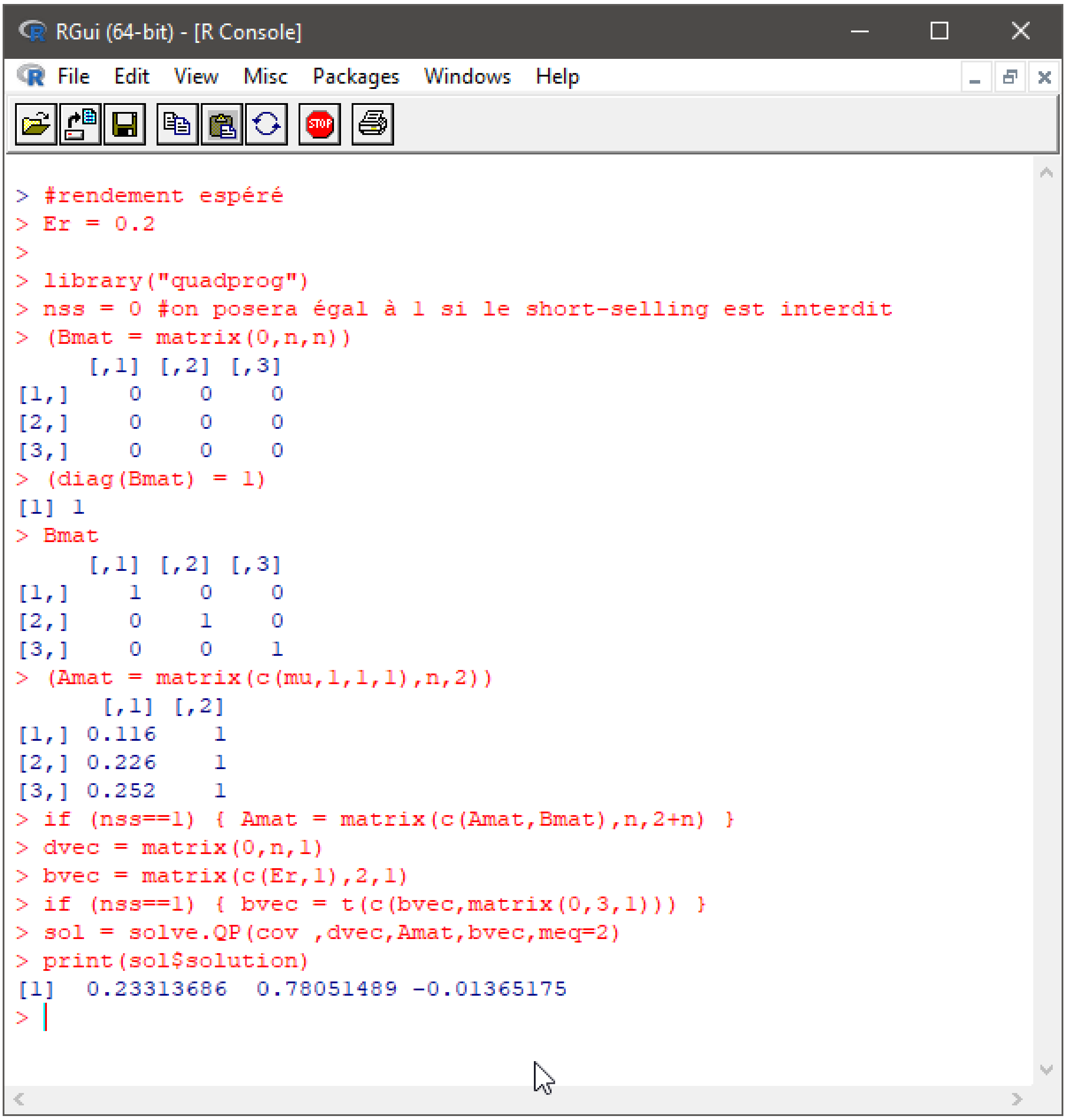
Refaisons l'exemple cela avec un package plus fondamental qu'est **quadprog** (programmation quadratique). Nous préparons d'abord la donnée de base:



```
> T1<-c(-0.15,0.05,-0.43,0.79,0.32)
> T2<-c(0.29,0.18,0.24,0.25,0.17)
> T3<-c(0.38,0.63,0.46,0.36,-0.57)
>
> #moyenne, écart-type, corrélation
> (rendements<-cbind(T1,T2,T3))
        T1   T2    T3
[1,] -0.15 0.29  0.38
[2,]  0.05 0.18  0.63
[3,] -0.43 0.24  0.46
[4,]  0.79 0.25  0.36
[5,]  0.32 0.17 -0.57
> (mu<-colMeans(rendements))
   T1    T2    T3
0.116 0.226 0.252
> (sd<-apply(rendements,2,sd))
        T1         T2         T3
0.46613303 0.05029911 0.47166726
> (corr<-cor(rendements))
           T1         T2         T3
T1  1.0000000 -0.1799875 -0.3041258
T2 -0.1799875  1.0000000  0.4461644
T3 -0.3041258  0.4461644  1.0000000
>
> #nombre d'actifs
> (n<-length(mu))
[1] 3
>
> #conversion matrixe corrélation en matrice covariance
> (cov <- diag(sd)%*%corr%*%diag(sd))
          [,1]      [,2]      [,3]
[1,]  0.217280 -0.004220 -0.066865
[2,] -0.004220  0.002530  0.010585
[3,] -0.066865  0.010585  0.222470
>
```

Et nous utilisons la fameuse fonction **solve.QP( )** de **quadprog**:

```
> #rendement espéré
> Er = 0.2
>
> library("quadprog")
> nss = 0 #on posera égal à 1 si le short-selling est interdit
> (Bmat = matrix(0,n,n))
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
> (diag(Bmat) = 1)
[1] 1
> Bmat
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> (Amat = matrix(c(mu,1,1,1),n,2))
      [,1] [,2]
[1,] 0.116    1
[2,] 0.226    1
[3,] 0.252    1
> if (nss==1) { Amat = matrix(c(Amat,Bmat),n,2+n) }
> dvec = matrix(0,n,1)
> bvec = matrix(c(Er,1),2,1)
> if (nss==1) { bvec = t(c(bvec,matrix(0,3,1))) }
> sol = solve.QP(cov ,dvec,Amat,bvec,meq=2)
> print(sol$solution)
[1]  0.23313686  0.78051489 -0.01365175
> |
```



Nous retrouvons donc bien le résultat de la méthode précédente mais aussi pour le coup avec les mêmes différences qu'avec l'exemple fait dans le cours théorique avec Microsoft Excel!

Portefeuille de variance minimale (MV)

Là encore nous allons vérifier si nous obtenons les mêmes poids que les calculs effectués dans le cours théorique. Nous utilisons la commande **minvariancePortfolio( )** du package **fPortfolio** (cette dernière fonction ignore au besoin automatique le rendement cible que nous avions défini juste plus haut!):

```
R Console
> covSpec<-portfolioSpec()
> setWeights(covSpec)<-c(1/3,1/3,1/3) #poids initial
> setNFrontierPoints(covSpec)<-10
> minvariancePortfolio(rendements,covSpec,constraints)

Title:
 MV Minimum Variance Portfolio
 Estimator:         covEstimator
 Solver:            solveRquadprog
 Optimize:          minRisk
 Constraints:       minW maxW

Portfolio Weights:
     T1      T2      T3
 0.0258  1.0000 -0.0258

Covariance Risk Budgets:
     T1      T2      T3
 0.0374  1.0000 -0.0374

Target Return and Risks:
   mean      mu     Cov   Sigma    CVaR     VaR
 0.2225  0.2225  0.0463  0.0463 -0.1650 -0.1650

Description:
 Sun Feb 23 12:58:39 2014 by user: Isoz Vincent
>
```

Nous retrouvons donc exactement les mêmes poids que ceux calculés dans le cours théorique!

Portefeuille de Michaud

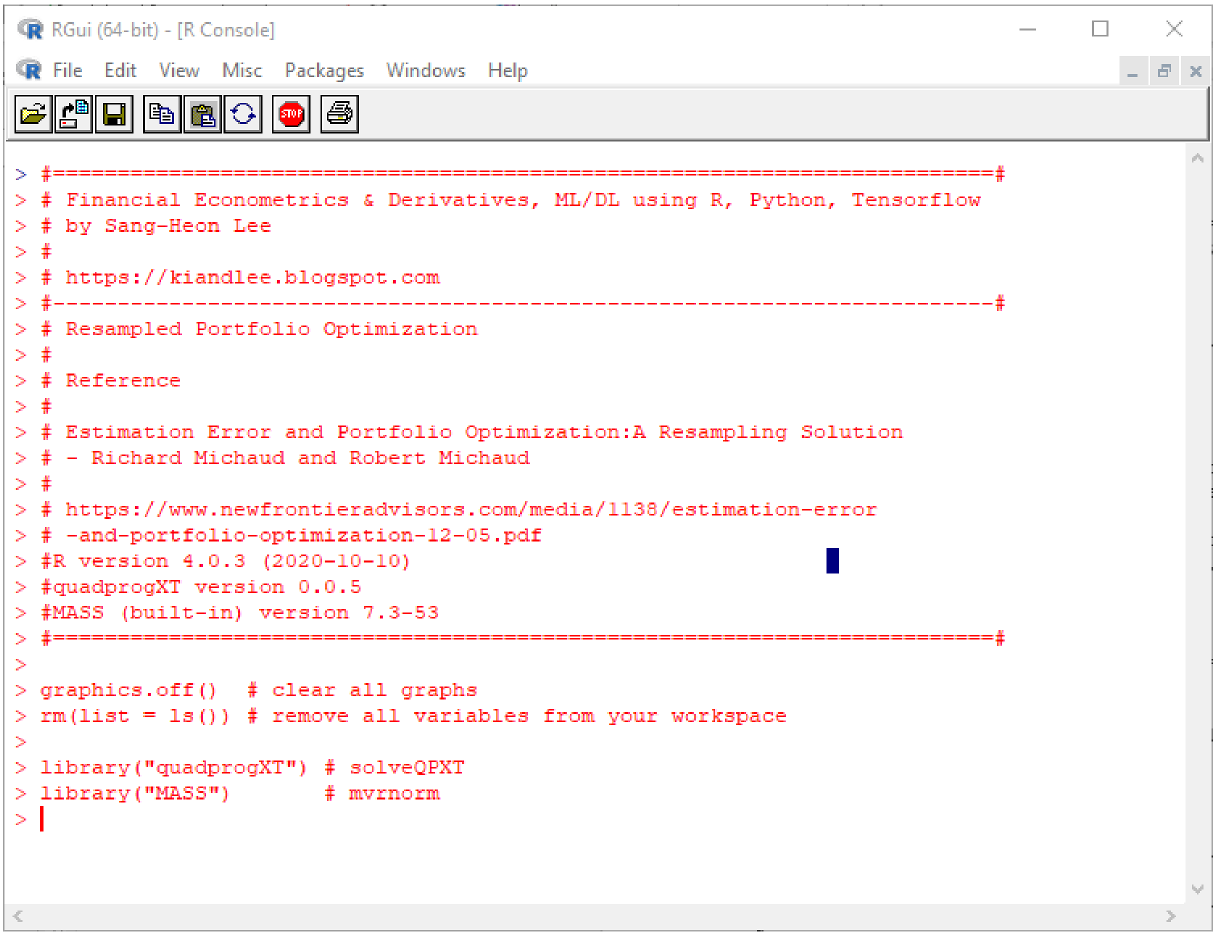
Comme nous l'avons vu dans le cours théorique, le but du portefeuille de Michaud est de rééchantilloner le portefeuille pour contrecarrer le fait que travailler avec des estimateurs ponctuels de la moyenne des rendements et de la matrice des covariances peut être peu réaliste dans certaines situations.

Nous avons vu dans le support de cours MATLAB comment implémenter ce modèle, nous allons refaire ici de même en reprenant la magnifique code de Sang-Heon Lee mis à dispostion à l'origine ici:

https://kiandlee.blogspot.com/2021/05/markowitz-vs-michaud-portfolio.html

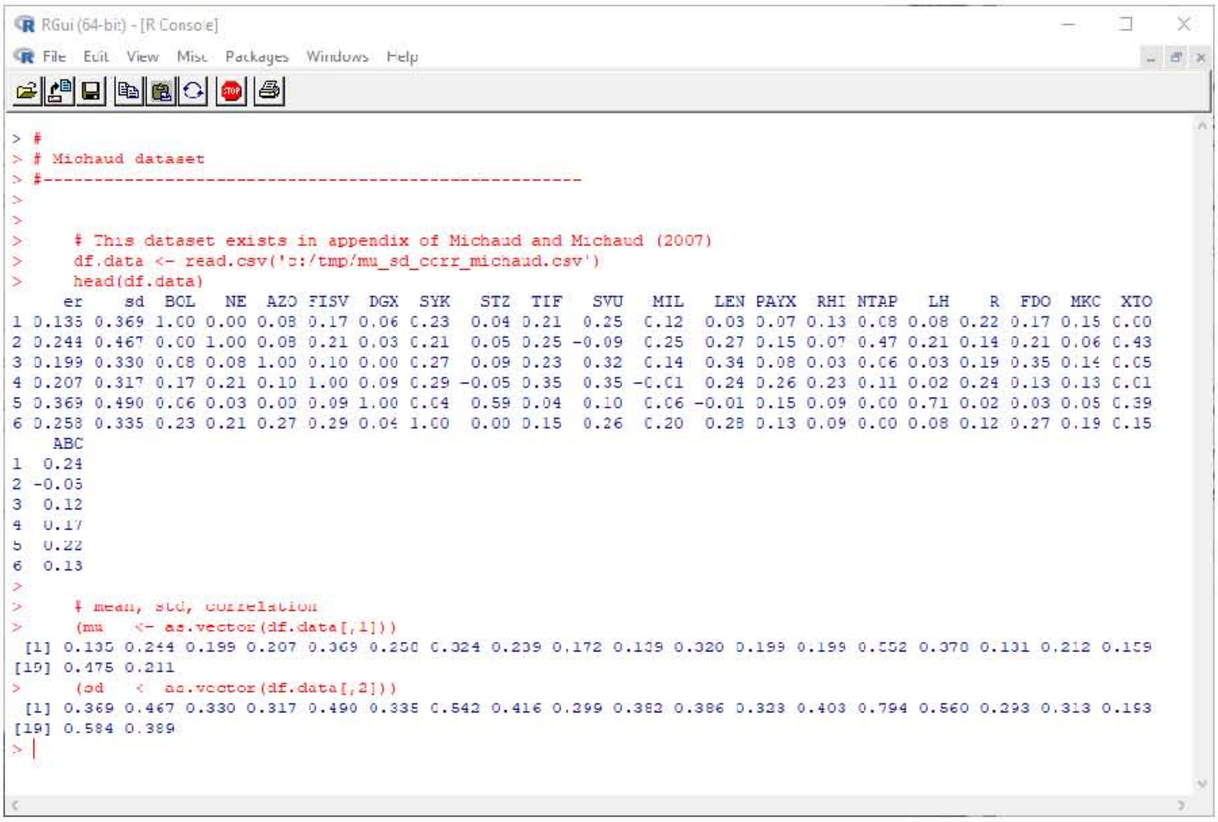
Et nous allons dans un premier temps faire comme il l'a fournit: avec les données du livre de Michaud et Michaud (2007), et seulement ensuite nous ferons le cas avec les mêmes données que dans le cours MATLAB (du moins quand nous aurons le temps de le rédiger).

Nous préparons d'above l'environnement et chargeons les libraries ad hoc:

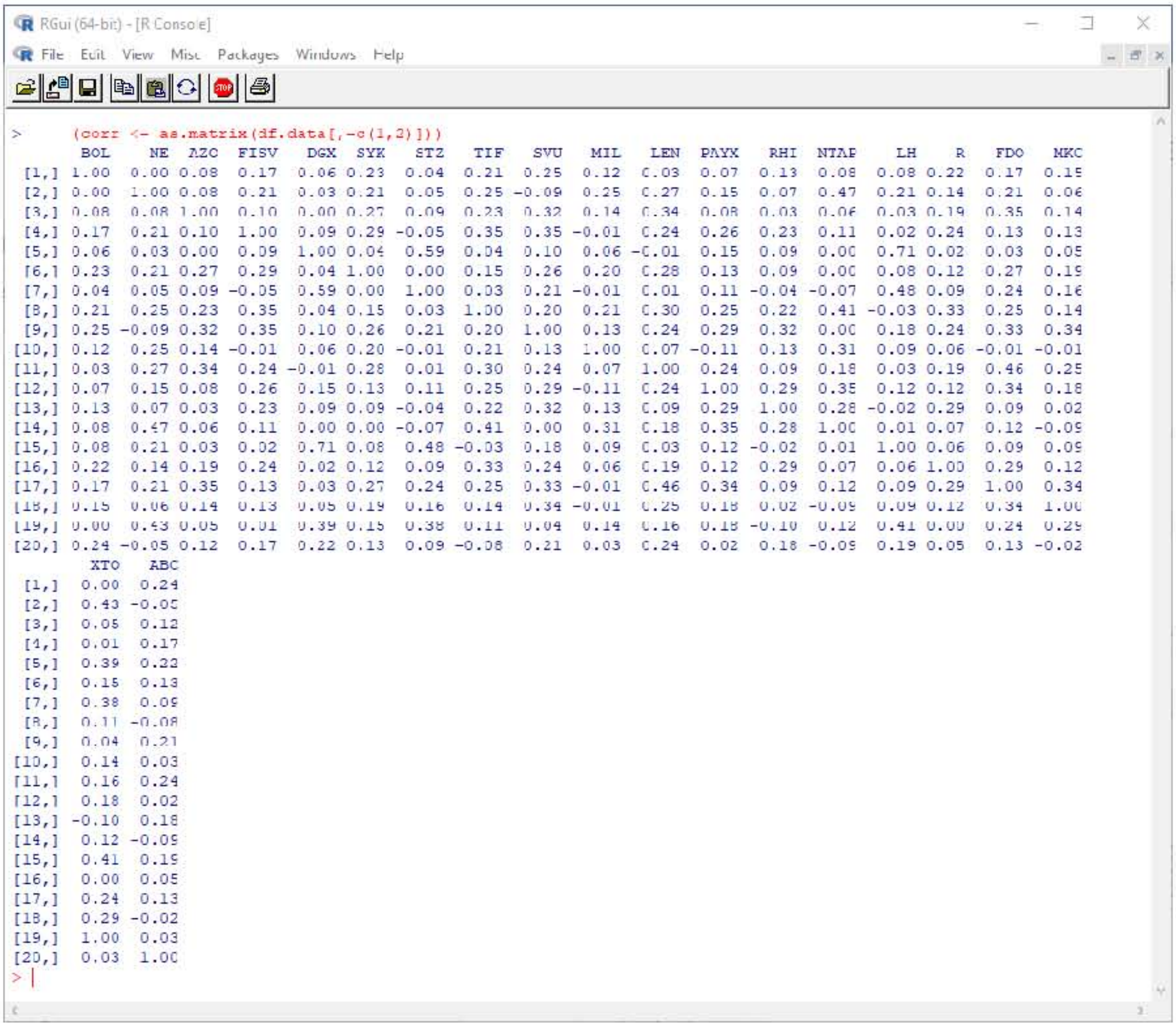


```
> #=========================================================================#
> # Financial Econometrics & Derivatives, ML/DL using R, Python, Tensorflow
> # by Sang-Heon Lee
> #
> # https://kiandlee.blogspot.com
> #-------------------------------------------------------------------------#
> # Resampled Portfolio Optimization
> #
> # Reference
> #
> # Estimation Error and Portfolio Optimization:A Resampling Solution
> # - Richard Michaud and Robert Michaud
> #
> # https://www.newfrontieradvisors.com/media/1138/estimation-error
> # -and-portfolio-optimization-12-05.pdf
> #R version 4.0.3 (2020-10-10)
> #quadprogXT version 0.0.5
> #MASS (built-in) version 7.3-53
> #=========================================================================#
>
> graphics.off()  # clear all graphs
> rm(list = ls()) # remove all variables from your workspace
>
> library("quadprogXT") # solveQPXT
> library("MASS")       # mvrnorm
>
```

On charge les données et on prépare les vecteurs nécessaires pour la suite:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> #
> # Michaud dataset
> #--
>
>
> # This dataset exists in appendix of Michaud and Michaud (2007)
> df.data <- read.csv('c:/tmp/mu_sd_corr_michaud.csv')
> head(df.data)
er sd BOL NE AZO FISV DGX SYK STZ TIF SVU MIL LEN PAYX RHI NTAP LH R FDO MKC XTO
1 0.135 0.369 1.00 0.00 0.08 0.17 0.06 0.23 0.04 0.21 0.25 0.12 0.03 0.07 0.13 0.08 0.08 0.22 0.17 0.15 0.00
2 0.244 0.467 0.00 1.00 0.08 0.21 0.03 0.21 0.05 0.25 -0.09 0.25 0.27 0.15 0.07 0.47 0.21 0.14 0.21 0.06 0.43
3 0.199 0.330 0.08 0.08 1.00 0.10 0.00 0.27 0.09 0.23 0.32 0.14 0.34 0.08 0.03 0.06 0.03 0.19 0.35 0.14 0.05
4 0.207 0.317 0.17 0.21 0.10 1.00 0.09 0.29 -0.05 0.35 0.35 -0.01 0.24 0.26 0.23 0.11 0.02 0.24 0.13 0.13 0.01
5 0.369 0.490 0.06 0.03 0.00 0.09 1.00 0.04 0.59 0.04 0.10 0.06 -0.01 0.15 0.09 0.00 0.71 0.02 0.03 0.05 0.39
6 0.258 0.335 0.23 0.21 0.27 0.29 0.04 1.00 0.00 0.15 0.26 0.20 0.28 0.13 0.09 0.00 0.08 0.12 0.27 0.19 0.15
ABC
1 0.24
2 -0.05
3 0.12
4 0.17
5 0.22
6 0.13
>
> # mean, std, correlation
> (mu <- as.vector(df.data[,1]))
[1] 0.135 0.244 0.199 0.207 0.369 0.258 0.324 0.239 0.172 0.139 0.320 0.199 0.199 0.552 0.370 0.131 0.212 0.159
[19] 0.475 0.211
> (sd <- as.vector(df.data[,2]))
[1] 0.369 0.467 0.330 0.317 0.490 0.335 0.542 0.416 0.299 0.382 0.386 0.323 0.403 0.794 0.560 0.293 0.313 0.193
[19] 0.584 0.389
>

Et:




```
>      (corr <- as.matrix(df.data[,-c(1,2)]))
       BOL    NE  AZO  FISV   DGX  SYK   STZ   TIF   SVU   MIL   LEN  PAYX   RHI  NTAP    LH    R   FDO   MKC
 [1,] 1.00  0.00 0.08  0.17  0.06 0.23  0.04  0.21  0.25  0.12  0.03  0.07  0.13  0.08  0.08 0.22  0.17  0.15
 [2,] 0.00  1.00 0.08  0.21  0.03 0.21  0.05  0.25 -0.09  0.25  0.27  0.15  0.07  0.47  0.21 0.14  0.21  0.06
 [3,] 0.08  0.08 1.00  0.10  0.00 0.27  0.09  0.23  0.32  0.14  0.34  0.08  0.03  0.06  0.03 0.19  0.35  0.14
 [4,] 0.17  0.21 0.10  1.00  0.09 0.29 -0.05  0.35  0.35 -0.01  0.24  0.26  0.23  0.11  0.02 0.24  0.13  0.13
 [5,] 0.06  0.03 0.00  0.09  1.00 0.04  0.59  0.04  0.10  0.06 -0.01  0.15  0.09  0.00  0.71 0.02  0.03  0.05
 [6,] 0.23  0.21 0.27  0.29  0.04 1.00  0.00  0.15  0.26  0.20  0.28  0.13  0.09  0.00  0.08 0.12  0.27  0.19
 [7,] 0.04  0.05 0.09 -0.05  0.59 0.00  1.00  0.03  0.21 -0.01  0.01  0.11 -0.04 -0.07  0.48 0.09  0.24  0.16
 [8,] 0.21  0.25 0.23  0.35  0.04 0.15  0.03  1.00  0.20  0.21  0.30  0.25  0.22  0.41 -0.03 0.33  0.25  0.14
 [9,] 0.25 -0.09 0.32  0.35  0.10 0.26  0.21  0.20  1.00  0.13  0.24  0.29  0.32  0.00  0.18 0.24  0.33  0.34
[10,] 0.12  0.25 0.14 -0.01  0.06 0.20 -0.01  0.21  0.13  1.00  0.07 -0.11  0.13  0.31  0.09 0.06 -0.01 -0.01
[11,] 0.03  0.27 0.34  0.24 -0.01 0.28  0.01  0.30  0.24  0.07  1.00  0.24  0.09  0.18  0.03 0.19  0.46  0.25
[12,] 0.07  0.15 0.08  0.26  0.15 0.13  0.11  0.25  0.29 -0.11  0.24  1.00  0.29  0.35  0.12 0.12  0.34  0.18
[13,] 0.13  0.07 0.03  0.23  0.09 0.09 -0.04  0.22  0.32  0.13  0.09  0.29  1.00  0.28 -0.02 0.29  0.09  0.02
[14,] 0.08  0.47 0.06  0.11  0.00 0.00 -0.07  0.41  0.00  0.31  0.18  0.35  0.28  1.00  0.01 0.07  0.12 -0.09
[15,] 0.08  0.21 0.03  0.02  0.71 0.08  0.48 -0.03  0.18  0.09  0.03  0.12 -0.02  0.01  1.00 0.06  0.09  0.09
[16,] 0.22  0.14 0.19  0.24  0.02 0.12  0.09  0.33  0.24  0.06  0.19  0.12  0.29  0.07  0.06 1.00  0.29  0.12
[17,] 0.17  0.21 0.35  0.13  0.03 0.27  0.24  0.25  0.33 -0.01  0.46  0.34  0.09  0.12  0.09 0.29  1.00  0.34
[18,] 0.15  0.06 0.14  0.13  0.05 0.19  0.16  0.14  0.34 -0.01  0.25  0.18  0.02 -0.09  0.09 0.12  0.34  1.00
[19,] 0.00  0.43 0.05  0.01  0.39 0.15  0.38  0.11  0.04  0.14  0.16  0.18 -0.10  0.12  0.41 0.00  0.24  0.29
[20,] 0.24 -0.05 0.12  0.17  0.22 0.13  0.09 -0.08  0.21  0.03  0.24  0.02  0.18 -0.09  0.19 0.05  0.13 -0.02
        XTO   ABC
 [1,]  0.00  0.24
 [2,]  0.43 -0.05
 [3,]  0.05  0.12
 [4,]  0.01  0.17
 [5,]  0.39  0.22
 [6,]  0.15  0.13
 [7,]  0.38  0.09
 [8,]  0.11 -0.08
 [9,]  0.04  0.21
[10,]  0.14  0.03
[11,]  0.16  0.24
[12,]  0.18  0.02
[13,] -0.10  0.18
[14,]  0.12 -0.09
[15,]  0.41  0.19
[16,]  0.00  0.05
[17,]  0.24  0.13
[18,]  0.29 -0.02
[19,]  1.00  0.03
[20,]  0.03  1.00
>
```

Et:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

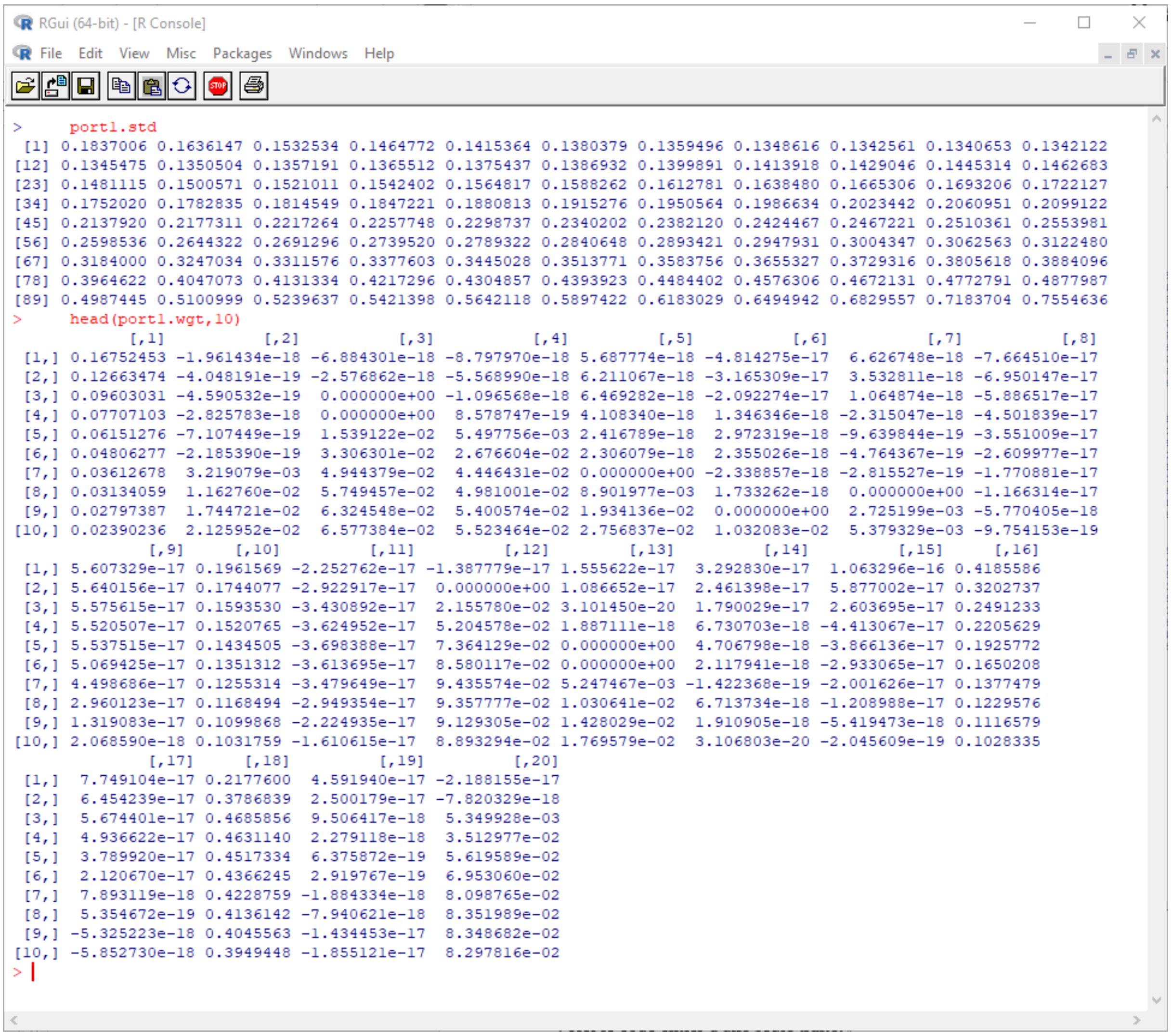
>     # number of asset
>     (nvar <- length(mu))
[1] 20
>     (var.name <- colnames(df.data[,-c(1,2)]))
 [1] "BOL"  "NE"   "AZO"  "FISV" "DGX"  "SYK"  "STZ"  "TIF"  "SVU"  "MIL"  "LEN"  "PAYX" "RHI"  "NTAP" "LH"
[16] "R"    "FDO"  "MKC"  "XTO"  "ABC"
>
>     # convert correlation to covariance matrix
>     (cov <- diag(sd)%*%corr%*%diag(sd))
           [,1]        [,2]      [,3]        [,4]       [,5]       [,6]        [,7]        [,8]        [,9]
 [1,] 0.13616100  0.00000000 0.0097416  0.01988541  0.0108486 0.02843145  0.00799992  0.03223584  0.02758275
 [2,] 0.00000000  0.21808900 0.0123288  0.03108819  0.0068649 0.03285345  0.01265570  0.04856800 -0.01256697
 [3,] 0.00974160  0.01232880 0.1089000  0.01046100  0.0000000 0.02984850  0.01609740  0.03157440  0.03157440
 [4,] 0.01988541  0.03108819 0.0104610  0.10048900  0.0139797 0.03079655 -0.00859070  0.04615520  0.03317405
 [5,] 0.01084860  0.00686490 0.0000000  0.01397970  0.2401000 0.00656600  0.15669220  0.00815360  0.01465100
 [6,] 0.02843145  0.03285345 0.0298485  0.03079655  0.0065660 0.11222500  0.00000000  0.02090400  0.02604290
 [7,] 0.00799992  0.01265570 0.0160974 -0.00859070  0.1566922 0.00000000  0.29376400  0.00676416  0.03403218
 [8,] 0.03223584  0.04856800 0.0315744  0.04615520  0.0081536 0.02090400  0.00676416  0.17305600  0.02487680
 [9,] 0.02758275 -0.01256697 0.0315744  0.03317405  0.0146510 0.02604290  0.03403218  0.02487680  0.08940100
[10,] 0.01691496  0.04459850 0.0176484 -0.00121094  0.0112308 0.02559400 -0.00207044  0.03337152  0.01484834
[11,] 0.00427302  0.04867074 0.0433092  0.02936688 -0.0018914 0.03620680  0.00209212  0.04817280  0.02769936
[12,] 0.00834309  0.02262615 0.0085272  0.02662166  0.0237405 0.01406665  0.01925726  0.03359200  0.02800733
[13,] 0.01933191  0.01317407 0.0039897  0.02938273  0.0177723 0.01215045 -0.00873704  0.03688256  0.03855904
[14,] 0.02343888  0.17427506 0.0157212  0.02768678  0.0000000 0.00000000 -0.03012436  0.13542464  0.00000000
[15,] 0.01653120  0.05491920 0.0055440  0.00355040  0.1948240 0.01500800  0.14568960 -0.00698880  0.03013920
[16,] 0.02378574  0.01915634 0.0183711  0.02229144  0.0028714 0.01177860  0.01429254  0.04022304  0.02102568
[17,] 0.01963449  0.03069591 0.0361515  0.01289873  0.0046011 0.02831085  0.04071504  0.03255200  0.03088371
[18,] 0.01068255  0.00540786 0.0089166  0.00795353  0.0047285 0.01228445  0.01673696  0.01124032  0.01962038
[19,] 0.00000000  0.11727304 0.0096360  0.00185128  0.1116024 0.02934600  0.12028064  0.02672384  0.00698464
[20,] 0.03444984 -0.00908315 0.0154044  0.02096321  0.0419342 0.01694095  0.01897542 -0.01294592  0.02442531
            [,10]       [,11]       [,12]       [,13]       [,14]      [,15]      [,16]       [,17]       [,18]
 [1,]  0.01691496  0.00427302  0.00834309  0.01933191  0.02343888  0.0165312 0.02378574  0.01963449  0.01068255
 [2,]  0.04459850  0.04867074  0.02262615  0.01317407  0.17427506  0.0549192 0.01915634  0.03069591  0.00540786
 [3,]  0.01764840  0.04330920  0.00852720  0.00398970  0.01572120  0.0055440 0.01837110  0.03615150  0.00891660
 [4,] -0.00121094  0.02936688  0.02662166  0.02938273  0.02768678  0.0035504 0.02229144  0.01289873  0.00795353
 [5,]  0.01123080 -0.00189140  0.02374050  0.01777230  0.00000000  0.1948240 0.00287140  0.00460110  0.00472850
 [6,]  0.02559400  0.03620680  0.01406665  0.01215045  0.00000000  0.0150080 0.01177860  0.02831085  0.01228445
 [7,] -0.00207044  0.00209212  0.01925726 -0.00873704 -0.03012436  0.1456896 0.01429254  0.04071504  0.01673696
 [8,]  0.03337152  0.04817280  0.03359200  0.03688256  0.13542464 -0.0069888 0.04022304  0.03255200  0.01124032
 [9,]  0.01484834  0.02769936  0.02800733  0.03855904  0.00000000  0.0301392 0.02102568  0.03088371  0.01962038
[10,]  0.14592400  0.01032164 -0.01357246  0.02001298  0.09402548  0.0192528 0.00671556 -0.00119566 -0.00073726
[11,]  0.01032164  0.14899600  0.02992272  0.01400022  0.05516712  0.0064848 0.02148862  0.05557628  0.01862450
[12,] -0.01357246  0.02992272  0.10432900  0.03774901  0.08976170  0.0217056 0.01135668  0.03437366  0.01122102
[13,]  0.02001298  0.01400022  0.03774901  0.16240900  0.08959496 -0.0045136 0.03424291  0.01135251  0.00155558
[14,]  0.09402548  0.05516712  0.08976170  0.08959496  0.63043600  0.0044464 0.01628494  0.02982264 -0.01379178
[15,]  0.01925280  0.00648480  0.02170560 -0.00451360  0.00444640  0.3136000 0.00984480  0.01577520  0.00972720
[16,]  0.00671556  0.02148862  0.01135668  0.03424291  0.01628494  0.0098448 0.08584900  0.02659561  0.00678588
[17,] -0.00119566  0.05557628  0.03437366  0.01135251  0.02982264  0.0157752 0.02659561  0.09796900  0.02053906
[18,] -0.00073726  0.01862450  0.01122102  0.00155558 -0.01379178  0.0097272 0.00678588  0.02053906  0.03724900
[19,]  0.03123232  0.03606784  0.03395376 -0.02353520  0.05564352  0.1340864 0.00000000  0.04387008  0.03268648
[20,]  0.00445794  0.03603696  0.00251294  0.02821806 -0.02779794  0.0413896 0.00569885  0.01582841 -0.00150154
            [,19]       [,20]
 [1,]  0.00000000  0.03444984
 [2,]  0.11727304 -0.00908315
 [3,]  0.00963600  0.01540440
 [4,]  0.00185128  0.02096321
 [5,]  0.11160240  0.04193420
 [6,]  0.02934600  0.01694095
```

On fait une optimisation traditionnelle mais avec la fonction **solveQPXT( )** de la librairie **quadprogXT** que nous n'avons jamais utilisé jusqu'ici:

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> #-----------------------------------------------------
> # Traditional portfolio optimization
> #-----------------------------------------------------
>
>     n.er <- 100  # number of Efficient Frontier points
>     (rset <- seq(min(mu),max(mu),length=n.er+2))
  [1] 0.1310000 0.1351683 0.1393366 0.1435050 0.1476733 0.1518416 0.1560099 0.1601782 0.1643465 0.1685149
 [11] 0.1726832 0.1768515 0.1810198 0.1851881 0.1893564 0.1935248 0.1976931 0.2018614 0.2060297 0.2101980
 [21] 0.2143663 0.2185347 0.2227030 0.2268713 0.2310396 0.2352079 0.2393762 0.2435446 0.2477129 0.2518812
 [31] 0.2560495 0.2602178 0.2643861 0.2685545 0.2727228 0.2768911 0.2810594 0.2852277 0.2893960 0.2935644
 [41] 0.2977327 0.3019010 0.3060693 0.3102376 0.3144059 0.3185743 0.3227426 0.3269109 0.3310792 0.3352475
 [51] 0.3394158 0.3435842 0.3477525 0.3519208 0.3560891 0.3602574 0.3644257 0.3685941 0.3727624 0.3769307
 [61] 0.3810990 0.3852673 0.3894356 0.3936040 0.3977723 0.4019406 0.4061089 0.4102772 0.4144455 0.4186139
 [71] 0.4227822 0.4269505 0.4311188 0.4352871 0.4394554 0.4436238 0.4477921 0.4519604 0.4561287 0.4602970
 [81] 0.4644653 0.4686337 0.4728020 0.4769703 0.4811386 0.4853069 0.4894752 0.4936436 0.4978119 0.5019802
 [91] 0.5061485 0.5103168 0.5144851 0.5186535 0.5228218 0.5269901 0.5311584 0.5353267 0.5394950 0.5436634
[101] 0.5478317 0.5520000
>     (rset <- rset[2:n.er+1])
 [1] 0.1393366 0.1435050 0.1476733 0.1518416 0.1560099 0.1601782 0.1643465 0.1685149 0.1726832 0.1768515 0.1810198
[12] 0.1851881 0.1893564 0.1935248 0.1976931 0.2018614 0.2060297 0.2101980 0.2143663 0.2185347 0.2227030 0.2268713
[23] 0.2310396 0.2352079 0.2393762 0.2435446 0.2477129 0.2518812 0.2560495 0.2602178 0.2643861 0.2685545 0.2727228
[34] 0.2768911 0.2810594 0.2852277 0.2893960 0.2935644 0.2977327 0.3019010 0.3060693 0.3102376 0.3144059 0.3185743
[45] 0.3227426 0.3269109 0.3310792 0.3352475 0.3394158 0.3435842 0.3477525 0.3519208 0.3560891 0.3602574 0.3644257
[56] 0.3685941 0.3727624 0.3769307 0.3810990 0.3852673 0.3894356 0.3936040 0.3977723 0.4019406 0.4061089 0.4102772
[67] 0.4144455 0.4186139 0.4227822 0.4269505 0.4311188 0.4352871 0.4394554 0.4436238 0.4477921 0.4519604 0.4561287
[78] 0.4602970 0.4644653 0.4686337 0.4728020 0.4769703 0.4811386 0.4853069 0.4894752 0.4936436 0.4978119 0.5019802
[89] 0.5061485 0.5103168 0.5144851 0.5186535 0.5228218 0.5269901 0.5311584 0.5353267 0.5394950 0.5436634 0.5478317
>
>     (portl.ret <- rset)
 [1] 0.1393366 0.1435050 0.1476733 0.1518416 0.1560099 0.1601782 0.1643465 0.1685149 0.1726832 0.1768515 0.1810198
[12] 0.1851881 0.1893564 0.1935248 0.1976931 0.2018614 0.2060297 0.2101980 0.2143663 0.2185347 0.2227030 0.2268713
[23] 0.2310396 0.2352079 0.2393762 0.2435446 0.2477129 0.2518812 0.2560495 0.2602178 0.2643861 0.2685545 0.2727228
[34] 0.2768911 0.2810594 0.2852277 0.2893960 0.2935644 0.2977327 0.3019010 0.3060693 0.3102376 0.3144059 0.3185743
[45] 0.3227426 0.3269109 0.3310792 0.3352475 0.3394158 0.3435842 0.3477525 0.3519208 0.3560891 0.3602574 0.3644257
[56] 0.3685941 0.3727624 0.3769307 0.3810990 0.3852673 0.3894356 0.3936040 0.3977723 0.4019406 0.4061089 0.4102772
[67] 0.4144455 0.4186139 0.4227822 0.4269505 0.4311188 0.4352871 0.4394554 0.4436238 0.4477921 0.4519604 0.4561287
[78] 0.4602970 0.4644653 0.4686337 0.4728020 0.4769703 0.4811386 0.4853069 0.4894752 0.4936436 0.4978119 0.5019802
[89] 0.5061485 0.5103168 0.5144851 0.5186535 0.5228218 0.5269901 0.5311584 0.5353267 0.5394950 0.5436634 0.5478317
>     portl.std <- rset*0
>     portl.wgt <- matrix(0,n.er,nvar)
>
>     # calculate efficient frontier
>     for (i in 1:n.er) {
+         Dmat <- 2*cov
+         dvec <- rep(0,nvar) #c(0,0)
+         Amat <- t(rbind(t(rep(1,nvar)),t(mu),diag(nvar)))
+         bvec <- c(1,rset[i],rep(0,nvar))
+
+         # mean-variance optimization
+         m<-solveQPXT(Dmat,dvec,Amat,bvec,meq=2,factorized=FALSE)
+
+         # output
+         portl.std[i]  <- sqrt(m$value)
+         portl.wgt[i,] <- t(m$solution)
+     }
Error in (function (Dmat, dvec, Amat, Aind, bvec, meq = 0, factorized = FALSE)  :
  NA/NaN/Inf in foreign function call (arg 10)
>
```

Ce qui donne:



```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

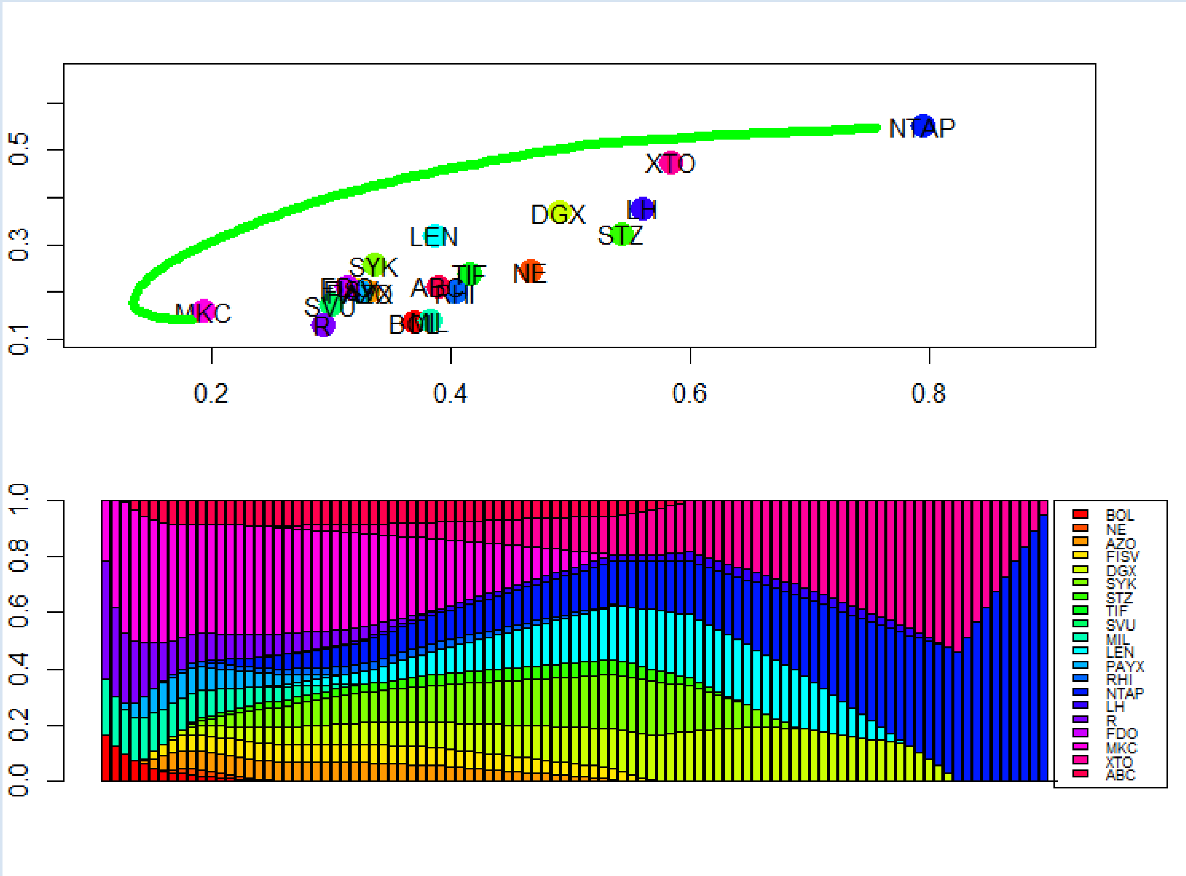
>     portl.std
 [1] 0.1837006 0.1636147 0.1532534 0.1464772 0.1415364 0.1380379 0.1359496 0.1348616 0.1342561 0.1340653 0.1342122
[12] 0.1345475 0.1350504 0.1357191 0.1365512 0.1375437 0.1386932 0.1399891 0.1413918 0.1429046 0.1445314 0.1462683
[23] 0.1481115 0.1500571 0.1521011 0.1542402 0.1564817 0.1588262 0.1612781 0.1638480 0.1665306 0.1693206 0.1722127
[34] 0.1752020 0.1782835 0.1814549 0.1847221 0.1880813 0.1915276 0.1950564 0.1986634 0.2023442 0.2060951 0.2099122
[45] 0.2137920 0.2177311 0.2217264 0.2257748 0.2298737 0.2340202 0.2382120 0.2424467 0.2467221 0.2510361 0.2553981
[56] 0.2598536 0.2644322 0.2691296 0.2739520 0.2789322 0.2840648 0.2893421 0.2947931 0.3004347 0.3062563 0.3122480
[67] 0.3184000 0.3247034 0.3311576 0.3377603 0.3445028 0.3513771 0.3583756 0.3655327 0.3729316 0.3805618 0.3884096
[78] 0.3964622 0.4047073 0.4131334 0.4217296 0.4304857 0.4393923 0.4484402 0.4576306 0.4672131 0.4772791 0.4877987
[89] 0.4987445 0.5100999 0.5239637 0.5421398 0.5642118 0.5897422 0.6183029 0.6494942 0.6829557 0.7183704 0.7554636
>     head(portl.wgt,10)
            [,1]          [,2]          [,3]          [,4]         [,5]          [,6]          [,7]          [,8]
 [1,] 0.16752453 -1.961434e-18 -6.884301e-18 -8.797970e-18 5.687774e-18 -4.814275e-17  6.626748e-18 -7.664510e-17
 [2,] 0.12663474 -4.048191e-19 -2.576862e-18 -5.568990e-18 6.211067e-18 -3.165309e-17  3.532811e-18 -6.950147e-17
 [3,] 0.09603031 -4.590532e-19  0.000000e+00 -1.096568e-18 6.469282e-18 -2.092274e-17  1.064874e-18 -5.886517e-17
 [4,] 0.07707103 -2.825783e-18  0.000000e+00  8.578747e-19 4.108340e-18  1.346346e-18 -2.315047e-18 -4.501839e-17
 [5,] 0.06151276 -7.107449e-19  1.539122e-02  5.497756e-03 2.416789e-18  2.972319e-18 -9.639844e-19 -3.551009e-17
 [6,] 0.04806277 -2.185390e-19  3.306301e-02  2.676604e-02 2.306079e-18  2.355026e-18 -4.764367e-19 -2.609977e-17
 [7,] 0.03612678  3.219079e-03  4.944379e-02  4.446431e-02 0.000000e+00 -2.338857e-18 -2.815527e-19 -1.770881e-17
 [8,] 0.03134059  1.162760e-02  5.749457e-02  4.981001e-02 8.901977e-03  1.733262e-18  0.000000e+00 -1.166314e-17
 [9,] 0.02797387  1.744721e-02  6.324548e-02  5.400574e-02 1.934136e-02  0.000000e+00  2.725199e-03 -5.770405e-18
[10,] 0.02390236  2.125952e-02  6.577384e-02  5.523464e-02 2.756837e-02  1.032083e-02  5.379329e-03 -9.754153e-19
              [,9]     [,10]         [,11]         [,12]        [,13]         [,14]         [,15]     [,16]
 [1,] 5.607329e-17 0.1961569 -2.252762e-17 -1.387779e-17 1.555622e-17  3.292830e-17  1.063296e-16 0.4185586
 [2,] 5.640156e-17 0.1744077 -2.922917e-17  0.000000e+00 1.086652e-17  2.461398e-17  5.877002e-17 0.3202737
 [3,] 5.575615e-17 0.1593530 -3.430892e-17  2.155780e-02 3.101450e-20  1.790029e-17  2.603695e-17 0.2491233
 [4,] 5.520507e-17 0.1520765 -3.624952e-17  5.204578e-02 1.887111e-18  6.730703e-18 -4.413067e-17 0.2205629
 [5,] 5.537515e-17 0.1434505 -3.698388e-17  7.364129e-02 0.000000e+00  4.706798e-18 -3.866136e-17 0.1925772
 [6,] 5.069425e-17 0.1351312 -3.613695e-17  8.580117e-02 0.000000e+00  2.117941e-18 -2.933065e-17 0.1650208
 [7,] 4.498686e-17 0.1255314 -3.479649e-17  9.435574e-02 5.247467e-03 -1.422368e-19 -2.001626e-17 0.1377479
 [8,] 2.960123e-17 0.1168494 -2.949354e-17  9.357777e-02 1.030641e-02  6.713734e-18 -1.208988e-17 0.1229576
 [9,] 1.319083e-17 0.1099868 -2.224935e-17  9.129305e-02 1.428029e-02  1.910905e-18 -5.419473e-18 0.1116579
[10,] 2.068590e-18 0.1031759 -1.610615e-17  8.893294e-02 1.769579e-02  3.106803e-20 -2.045609e-19 0.1028335
              [,17]     [,18]         [,19]         [,20]
 [1,]  7.749104e-17 0.2177600  4.591940e-17 -2.188155e-17
 [2,]  6.454239e-17 0.3786839  2.500179e-17 -7.820329e-18
 [3,]  5.674401e-17 0.4685856  9.506417e-18  5.349928e-03
 [4,]  4.936622e-17 0.4631140  2.279118e-18  3.512977e-02
 [5,]  3.789920e-17 0.4517334  6.375872e-19  5.619589e-02
 [6,]  2.120670e-17 0.4366245  2.919767e-19  6.953060e-02
 [7,]  7.893119e-18 0.4228759 -1.884334e-18  8.098765e-02
 [8,]  5.354672e-19 0.4136142 -7.940621e-18  8.351989e-02
 [9,] -5.325223e-18 0.4045563 -1.434453e-17  8.348682e-02
[10,] -5.852730e-18 0.3949448 -1.855121e-17  8.297816e-02
>
```

Avec le graphique correspondant:

R Console

```
>     # draw efficient fronter and allocation profile
>     x11(width=6); par(mfrow = c(2,1), mar = c(3,2,2,3), xpd=TRUE)
>
>         # individual asset
>         plot(sqrt(diag(cov)), mu,
+              xlim=c(0.8*min(port1.std),1.2*max(port1.std)),
+              ylim=c(0.8*min(mu),1.2*max(mu)),
+              col = rainbow(nvar), lwd = 10)
>         text(sqrt(diag(cov)), mu,
+              labels=var.name, cex= 1)
>
>         # efficient frontier
>         lines(port1.std,port1.ret,col = "green", lwd = 6)
>
>         # weight
>         barplot(t(port1.wgt),col=rainbow(nvar))
>         legend("topright", legend = var.name,
+                fill = rainbow(nvar), ncol = 1,
+                inset=c(-0.07,0), cex = 0.6)
>
> 
```

R Graphics: Device 2 (ACTIVE)



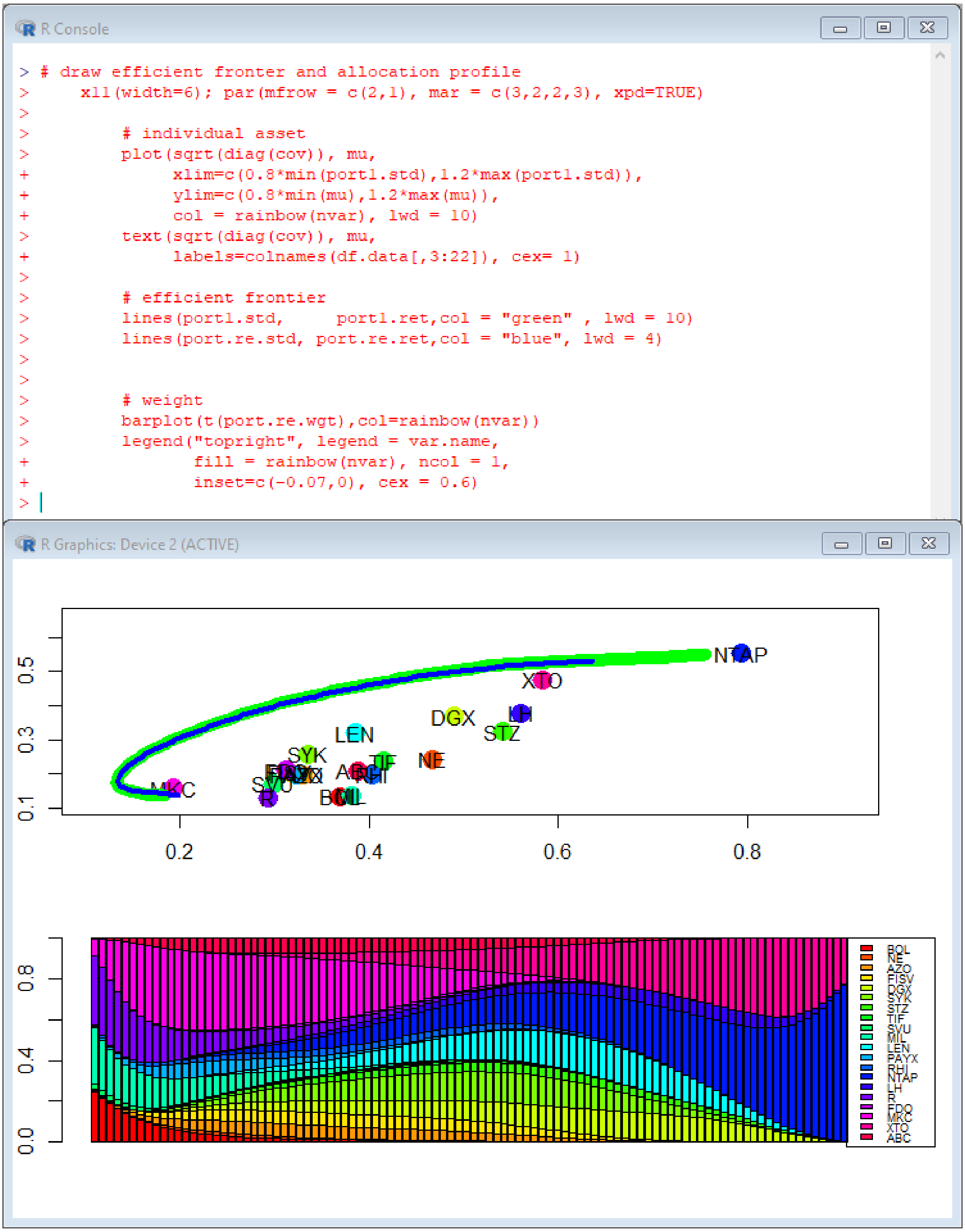
Maintenant nous faisons le ré-échantillonnage:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> #---------------------------------------------------
> # Resampled portfolio optimization
> #---------------------------------------------------
>     n.er <- 100  # number of EF points
>     rset <- seq(min(mu),max(mu),length=(n.er+2))
>     rset <- rset[2:(n.er+1)]
>     port.re.ret <- rset
>     port.re.std <- rset*0
>     port.re.wgt <- matrix(0,n.er,nvar)
>
>     n.rr = 1000  # number of resampling
>     for (rr in 1:n.rr) {
+         # simulated time series of assets
+         sim     <- mvrnorm(n = 120, mu, cov)
+
+         # simulated mu & cov
+         mu.sim  <- colMeans(sim)
+         cov.sim <- cov(sim)
+
+         rset.sim <- seq(min(mu.sim),max(mu.sim),length=(n.er+2))
+         rset.sim <- rset.sim[2:(n.er+1)]
+
+         port.sim.ret <- rset.sim
+         port.sim.std <- rset.sim*0
+         port.sim.wgt <- matrix(0,n.er,nvar)
+
+         # calculate efficient frontier
+         for (i in 1:n.er) {
+             Dmat <- 2*cov.sim
+             dvec <- rep(0,nvar)
+             Amat <- t(rbind(t(rep(1,nvar)),t(mu.sim),diag(nvar)))
+             bvec <- c(1,rset.sim[i],rep(0,nvar))
+
+             # mean-variance optimization
+             m<-solveQPXT(Dmat,dvec,Amat,bvec,meq=2,factorized=FALSE)
+
+             # output
+             port.sim.std[i]  <- sqrt(m$value)
+             port.sim.wgt[i,] <- t(m$solution)
+         }
+
+         # sum of resampling portfolios before average
+         port.re.ret <- port.re.ret + port.sim.ret
+         port.re.wgt <- port.re.wgt + port.sim.wgt
+     }
>
>     # average of resampling portfolios
>     port.re.wgt <- port.re.wgt/n.rr
>     port.re.ret <- port.re.ret/n.rr
>
>     # portfolio SD and Expexted Return
>     for (i in 1:n.er) {
+         port.re.ret[i] <- port.re.wgt[i,]%*%mu
+         port.re.std[i] <- sqrt(port.re.wgt[i,]%*%cov%*%port.re.wgt[i,])
+     }
> 
```

Et les graphiques correspondants:



Voici le code entier d'une seule traite:

```
#==========================================================================#
# Financial Econometrics & Derivatives, ML/DL using R, Python, Tensorflow
# by Sang-Heon Lee
#
# https://kiandlee.blogspot.com
#--------------------------------------------------------------------------#
# Resampled Portfolio Optimization
```

```
#
# Reference
#
# Estimation Error and Portfolio Optimization:A Resampling Solution
# - Richard Michaud and Robert Michaud
#
# https://www.newfrontieradvisors.com/media/1138/estimation-error
# -and-portfolio-optimization-12-05.pdf
#R version 4.0.3 (2020-10-10)
#quadprogXT version 0.0.5
#MASS (built-in) version 7.3-53
#=========================================================================#

graphics.off()  # clear all graphs
rm(list = ls()) # remove all variables from your workspace

library("quadprogXT") # solveQPXT
library("MASS")       # mvrnorm

#-----------------------------------------------------
# Michaud dataset
#-----------------------------------------------------


    # This dataset exists in appendix of Michaud and Michaud (2007)
    df.data <- read.csv('c:/tmp/mu_sd_corr_michaud.csv')
    head(df.data)

    # mean, std, correlation
    (mu   <- as.vector(df.data[,1]))
    (sd   <- as.vector(df.data[,2]))
    (corr <- as.matrix(df.data[,-c(1,2)]))

    # number of asset
    (nvar <- length(mu))
    (var.name <- colnames(df.data[,-c(1,2)]))

    # convert correlation to covariance matrix
    (cov <- diag(sd)%*%corr%*%diag(sd))

#-----------------------------------------------------
# Traditional portfolio optimization
#-----------------------------------------------------

    n.er <- 100  # number of Efficient Frontier points
    (rset <- seq(min(mu),max(mu),length=n.er+2))
    (rset <- rset[2:n.er+1])

    (port1.ret <- rset)
    port1.std <- rset*0
    port1.wgt <- matrix(0,n.er,nvar)

    # calculate efficient frontier
    for (i in 1:n.er) {
        Dmat <- 2*cov
        dvec <- rep(0,nvar) #c(0,0)
        Amat <- t(rbind(t(rep(1,nvar)),t(mu),diag(nvar)))
        bvec <- c(1,rset[i],rep(0,nvar))

        # mean-variance optimization
        m<-solveQPXT(Dmat,dvec,Amat,bvec,meq=2,factorized=FALSE)
```

```
        # output
        port1.std[i]  <- sqrt(m$value)
        port1.wgt[i,] <- t(m$solution)
    }


    # draw efficient fronter and allocation profile
    x11(width=6); par(mfrow = c(2,1), mar = c(3,2,2,3), xpd=TRUE)

        # individual asset
        plot(sqrt(diag(cov)), mu,
             xlim=c(0.8*min(port1.std),1.2*max(port1.std)),
             ylim=c(0.8*min(mu),1.2*max(mu)),
             col = rainbow(nvar), lwd = 10)
        text(sqrt(diag(cov)), mu,
             labels=var.name, cex= 1)

        # efficient frontier
        lines(port1.std,port1.ret,col = "green", lwd = 6)

        # weight
        barplot(t(port1.wgt),col=rainbow(nvar))
        legend("topright", legend = var.name,
               fill = rainbow(nvar), ncol = 1,
               inset=c(-0.07,0), cex = 0.6)

#-------------------------------------------------------
# Resampled portfolio optimization
#-------------------------------------------------------

    n.er <- 100  # number of EF points
    rset <- seq(min(mu),max(mu),length=(n.er+2))
    rset <- rset[2:(n.er+1)]

    port.re.ret <- rset
    port.re.std <- rset*0
    port.re.wgt <- matrix(0,n.er,nvar)

    n.rr = 1000  # number of resampling

    for (rr in 1:n.rr) {
        print(rr)

        # simulated time series of assets
        sim     <- mvrnorm(n = 120, mu, cov)

        # simulated mu & cov
        mu.sim  <- colMeans(sim)
        cov.sim <- cov(sim)

        rset.sim <- seq(min(mu.sim),max(mu.sim),length=(n.er+2))
        rset.sim <- rset.sim[2:(n.er+1)]

        port.sim.ret <- rset.sim
        port.sim.std <- rset.sim*0
        port.sim.wgt <- matrix(0,n.er,nvar)

        # calculate efficient frontier
        for (i in 1:n.er) {
            Dmat <- 2*cov.sim
```

```
            dvec <- rep(0,nvar)
            Amat <- t(rbind(t(rep(1,nvar)),t(mu.sim),diag(nvar)))
            bvec <- c(1,rset.sim[i],rep(0,nvar))

            # mean-variance optimization
            m<-solveQPXT(Dmat,dvec,Amat,bvec,meq=2,factorized=FALSE)

            # output
            port.sim.std[i]  <- sqrt(m$value)
            port.sim.wgt[i,] <- t(m$solution)
        }

        # sum of resampling portfolios before average
        port.re.ret <- port.re.ret + port.sim.ret
        port.re.wgt <- port.re.wgt + port.sim.wgt
    }

    # average of resampling portfolios
    port.re.wgt <- port.re.wgt/n.rr
    port.re.ret <- port.re.ret/n.rr

    # portfolio SD and Expexted Return
    for (i in 1:n.er) {
        port.re.ret[i] <- port.re.wgt[i,]%*%mu
        port.re.std[i] <- sqrt(port.re.wgt[i,]%*%cov%*%port.re.wgt[i,])
    }

    # draw efficient fronter and allocation profile
    x11(width=6); par(mfrow = c(2,1), mar = c(3,2,2,3), xpd=TRUE)

        # individual asset
        plot(sqrt(diag(cov)), mu,
             xlim=c(0.8*min(port1.std),1.2*max(port1.std)),
             ylim=c(0.8*min(mu),1.2*max(mu)),
             col = rainbow(nvar), lwd = 10)
        text(sqrt(diag(cov)), mu,
             labels=colnames(df.data[,3:22]), cex= 1)

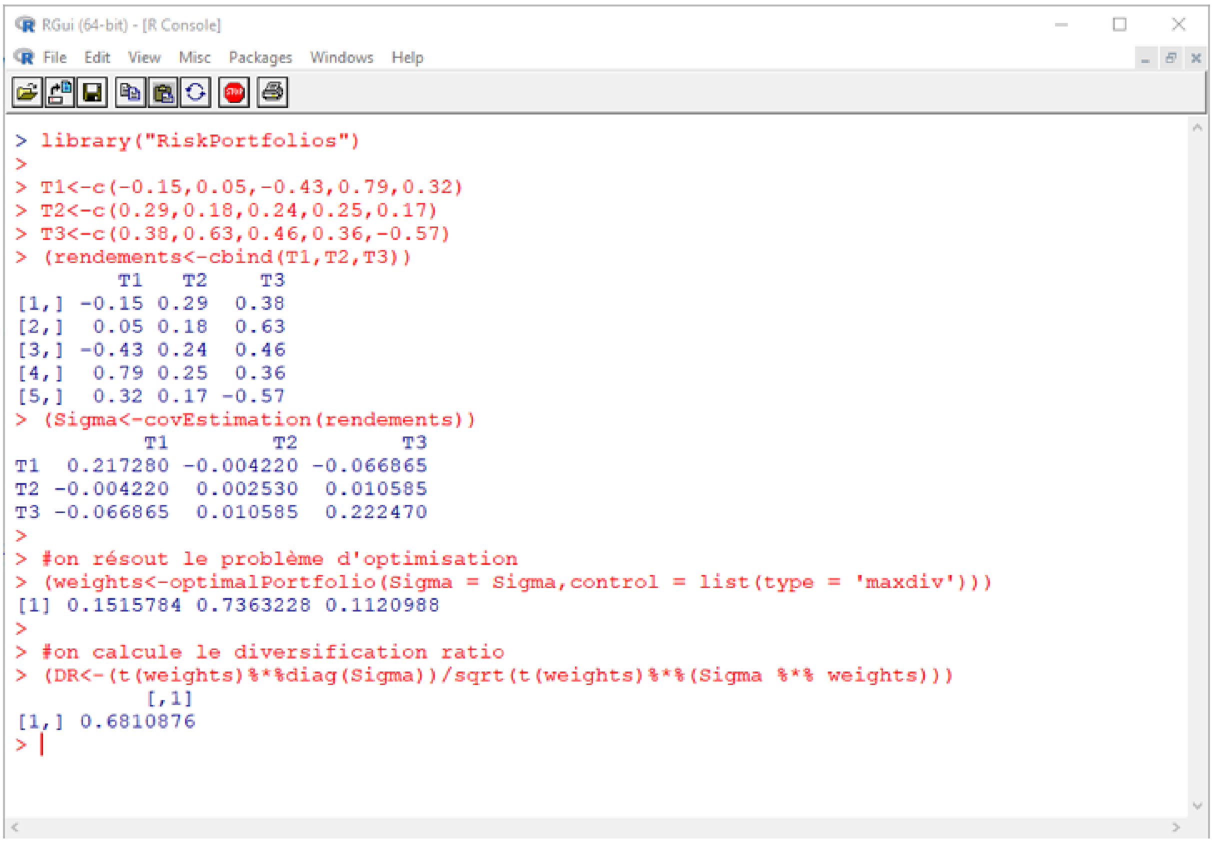
        # efficient frontier
        lines(port1.std,     port1.ret,col = "green" , lwd = 10)
        lines(port.re.std, port.re.ret,col = "blue", lwd = 4)


        # weight
        barplot(t(port.re.wgt),col=rainbow(nvar))
        legend("topright", legend = var.name,
               fill = rainbow(nvar), ncol = 1,
               inset=c(-0.07,0), cex = 0.6)
```

Portefeuille de maximum diversification

Là encore, voyons si nous arrivons à obtenir le même résultat que dans le cours théorique. Toujours en évitant d'utiliser le solveur interne à R. Aucun des packages que nous avons vu précédemment n'implémente à notre connaissance le portefeuille de maximum diversification.

Cependant, le package **RiskPortfolios** à une fonction **optimalPortfolio( )** avec de nombreux paramètres pour différents portefeuilles possibles, mais celui qui va nous intéresser ici est la valeur *maxdiv* du paramètre *control* comme l'illustre l'exemple suivant:

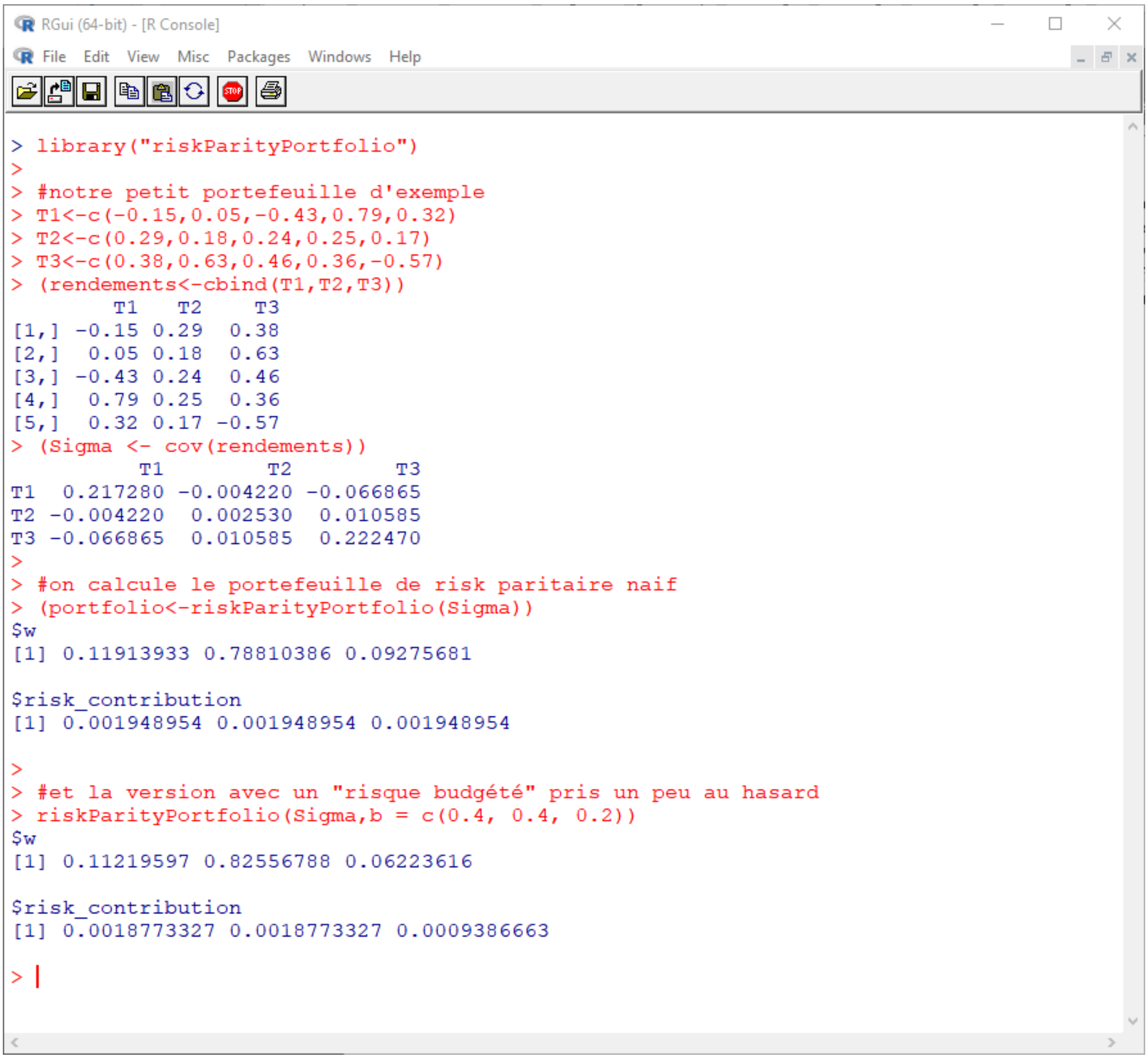


Nous retrouvons donc exactement les mêmes poids et le même DR que ceux calculés dans le cours théorique!

Portefeuille de risque paritaire

Nous allons ici utiliser le package **riskParityPortfolio** et sa fonction **riskParityPortfolio( )** qui implémente un ensemble de modèles empiriques basés sur le concept de risque paritaire. Rappelons que dans le cours théorique, nous avons survolé juste une formulation existante possible du problème d'optimisation de ce type de portefeuille, mais le package en contient plusieurs variantes très différentes (voir la documentation du package pour plus d'informations!).

Voyons donc l'application de deux cas simples avec notre portefeuille habituel:

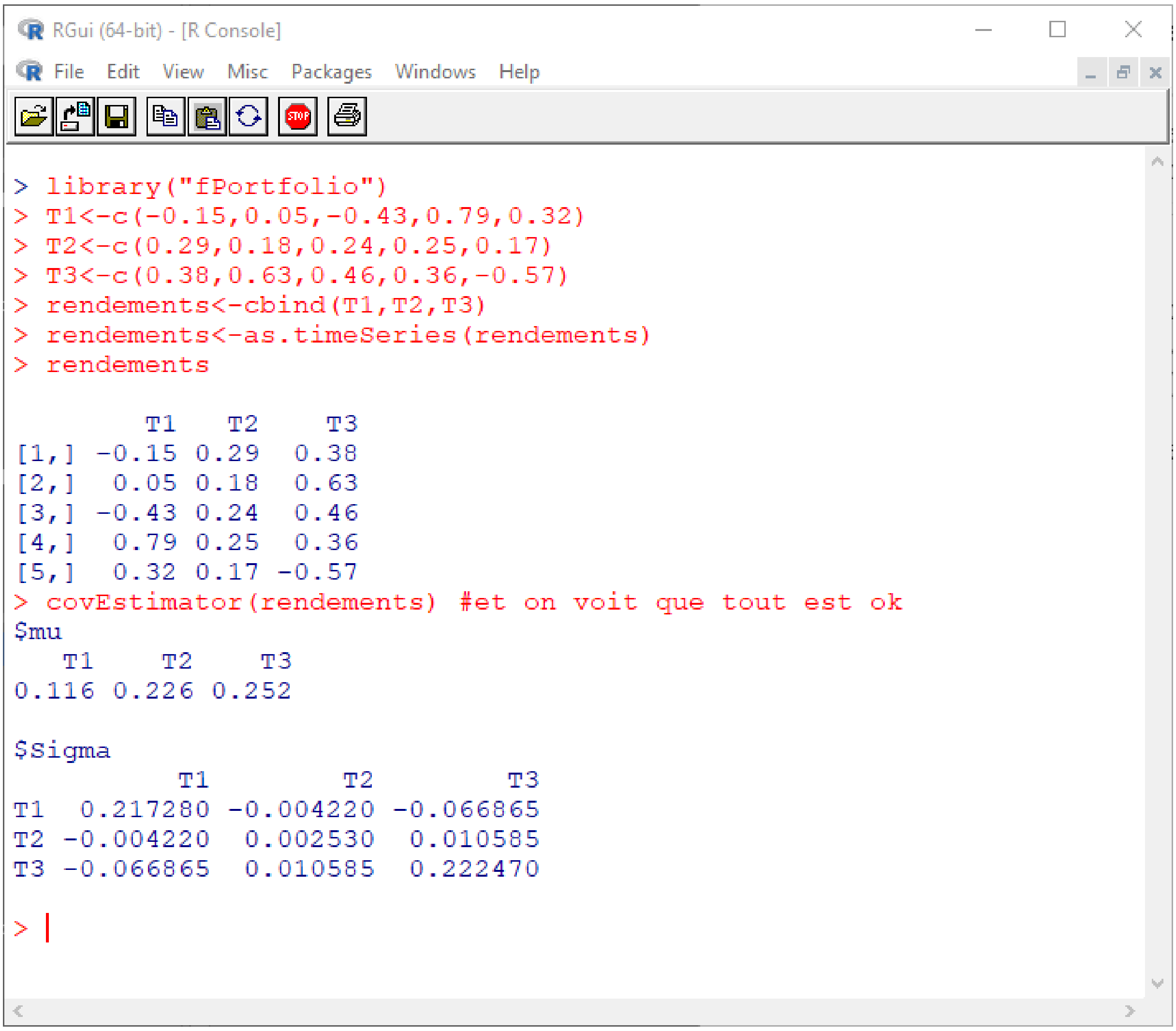


Portefeuille de variance conditionnelle

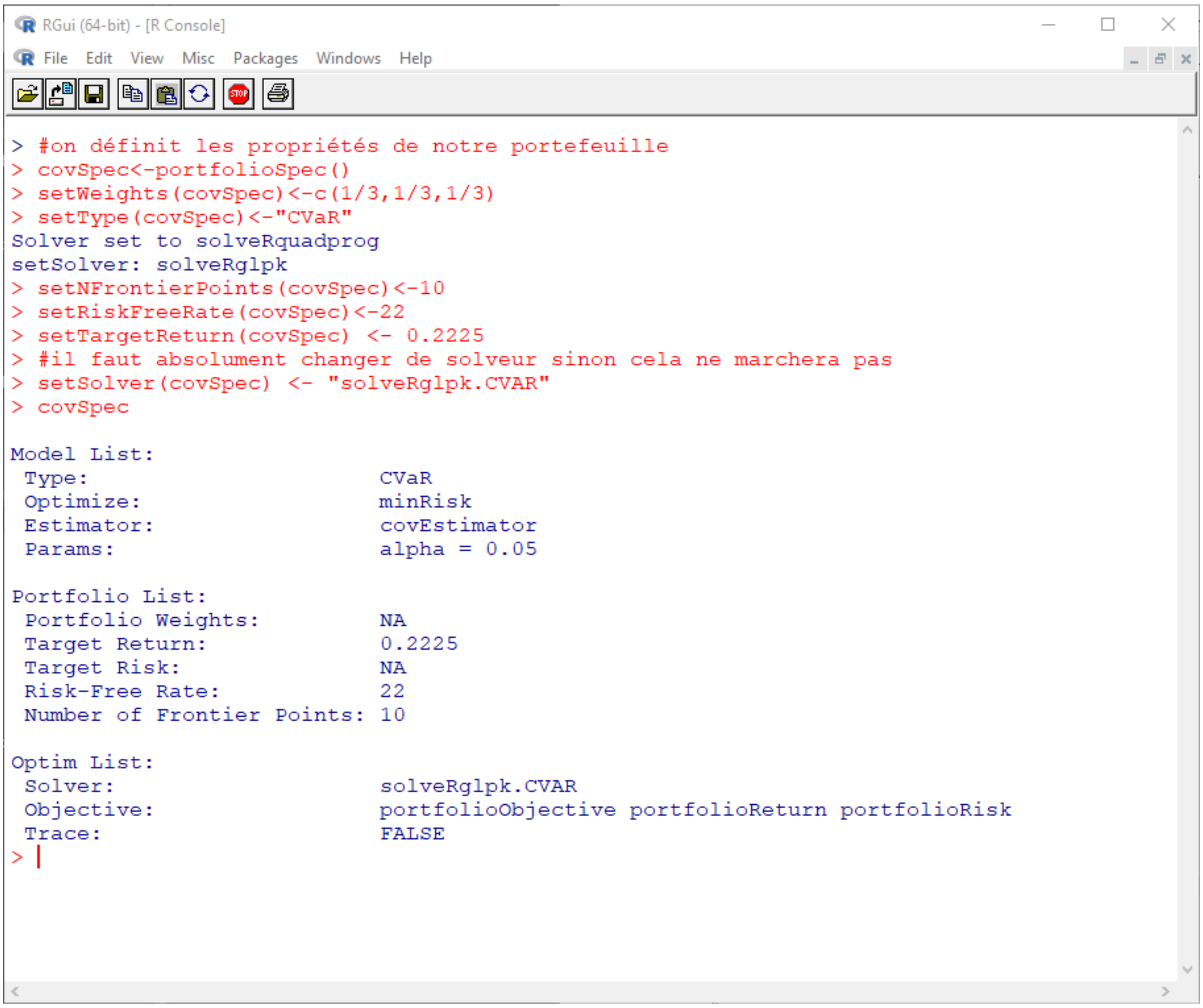
Nous avons vu dans le cours théorique ce qu'était la variance conditionnelle et comment l'appliquer à un portefeuille, sans toutefois faire les calculs à la main puisque a priori il n'existe pas d'expression analytique permettant d'obtenir les poids pour une CVaR donnée. De plus, les résultats varient assez fortement d'un logiciel à l'autre suivant le type de résolution numérique utilisée.

Remarque: Au jour auquel nous écrivons ces lignes, il semblerait que le package **fPortfolio** a un problème lorsqu'il s'agit de définir la valeur de alpha que nous voulons pour la CVaR (il y a un message d'erreur!).

Essayons de reproduire l'exemple que nous avions fait avec MATLAB:



Jusque là c'est du standard. Maintenant faisons ce qui est spécifique au CVaR:



Et on lance la définition des contraintes ainsi que l'analyse:

```
> constraints = "LongOnly"
> portfolioConstraints(rendements,covSpec,constraints)

Title:
 Portfolio Constraints

Lower/Upper Bounds:
      T1 T2 T3
Lower  0  0  0
Upper  1  1  1

Equal Matrix Constraints:
           ceq     T1     T2     T3
Return  0.2225  0.116  0.226  0.252
Budget -1.0000 -1.000 -1.000 -1.000

Cardinality Constraints:
      T1 T2 T3
Lower  0  0  0
Upper  1  1  1
> efficientPortfolio(rendements,covSpec,constraints)

Title:
 CVaR Efficient Portfolio
 Estimator:         covEstimator
 Solver:            solveRglpk.CVAR
 Optimize:          minRisk
 Constraints:       LongOnly
 VaR Alpha:         0.05

Portfolio Weights:
    T1     T2     T3
0.0318 0.9682 0.0000

Covariance Risk Budgets:
    T1     T2     T3
0.0386 0.9614 0.0000

Target Returns and Risks:
   mean     Cov    CVaR     VaR
 0.2225  0.0483 -0.1748 -0.1748

Description:
 Sun Mar 24 13:03:41 2019 by user: Vincent Isoz
> 
```

Frontière efficiente

Nous allons voir maintenant qu'il nous est possible de tracer la frontière efficiente avec la commande **frontierPlot( )** :

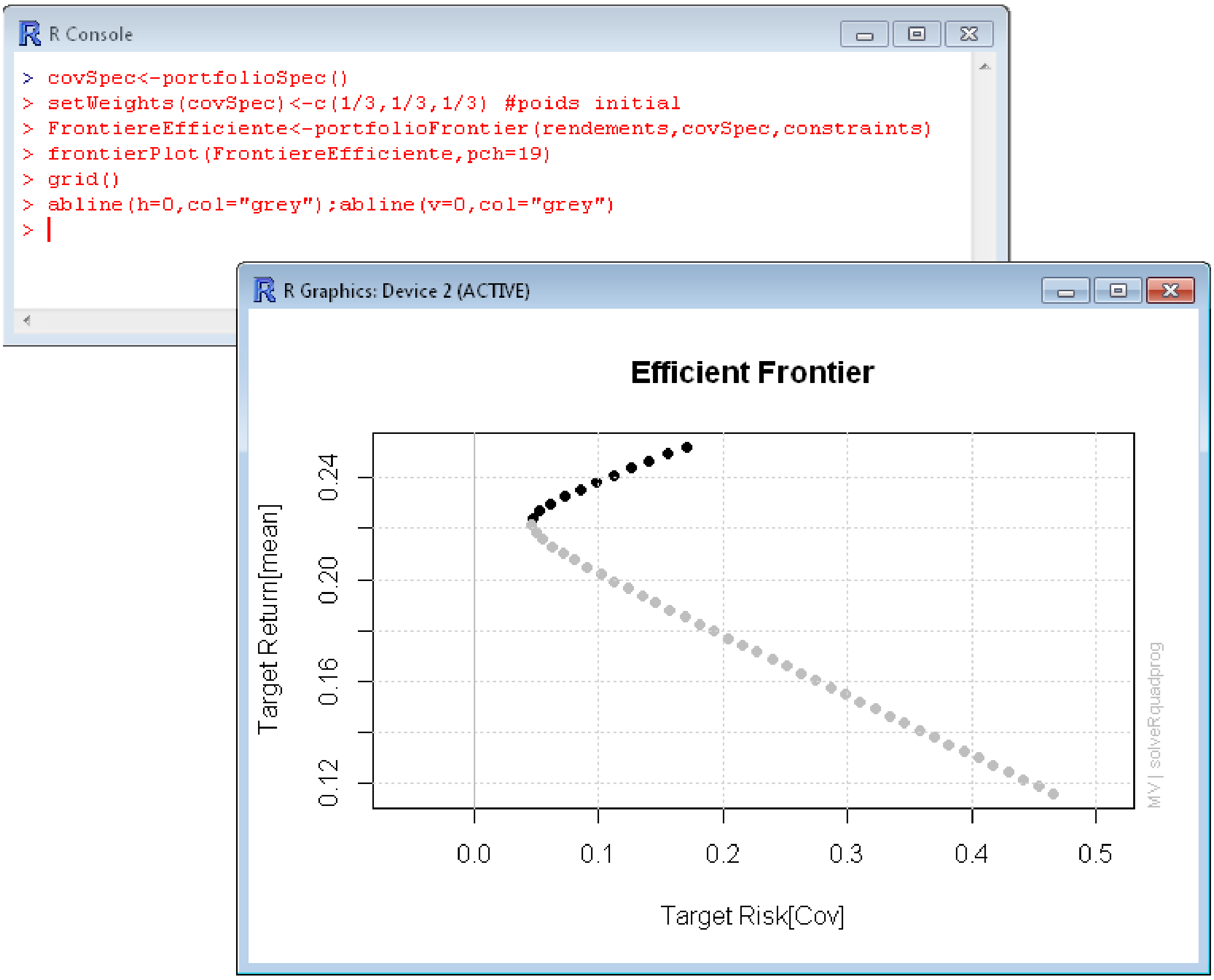
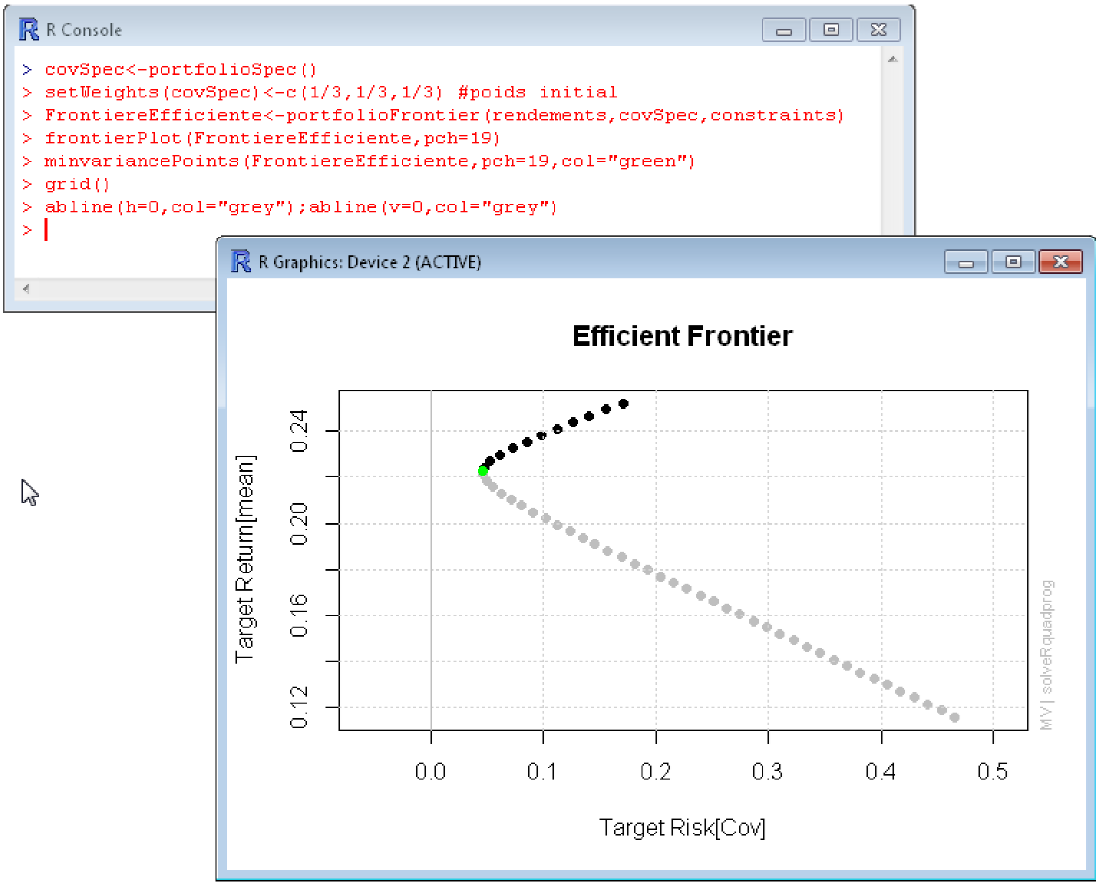


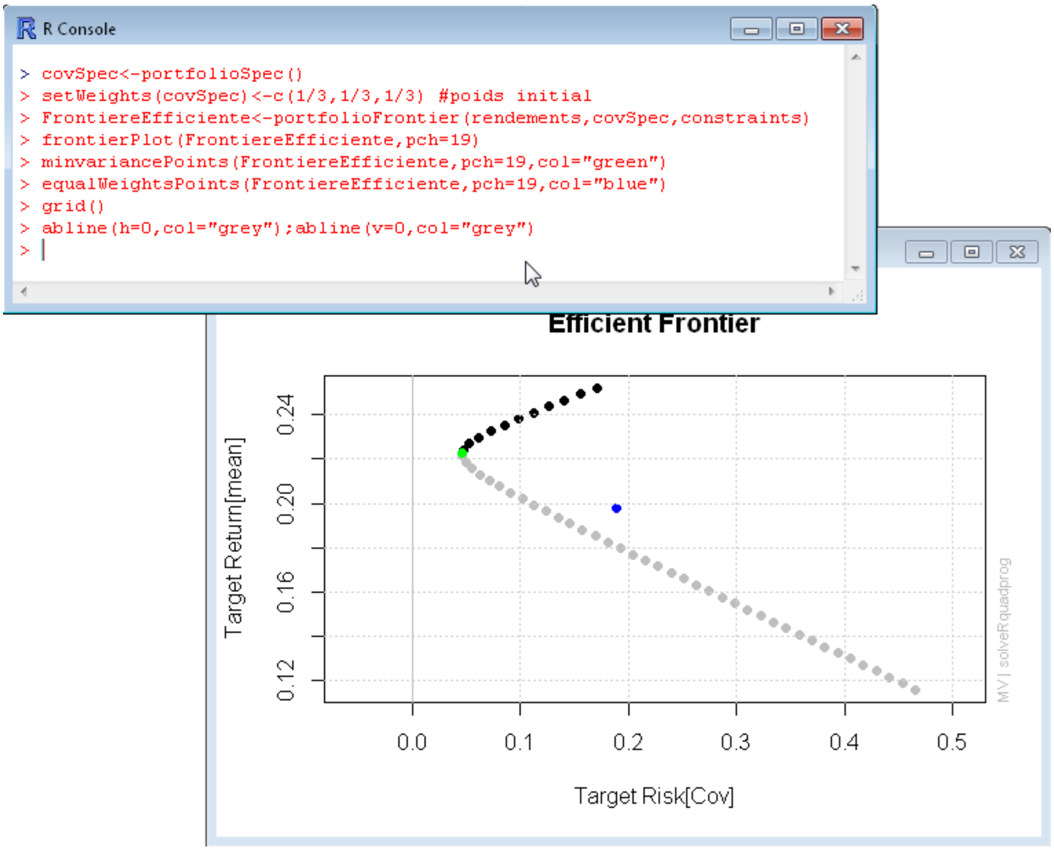
Figure 329 Graph frontière d'efficience de Markowitz

C'est la même que celle obtenue dans le cours théorique.

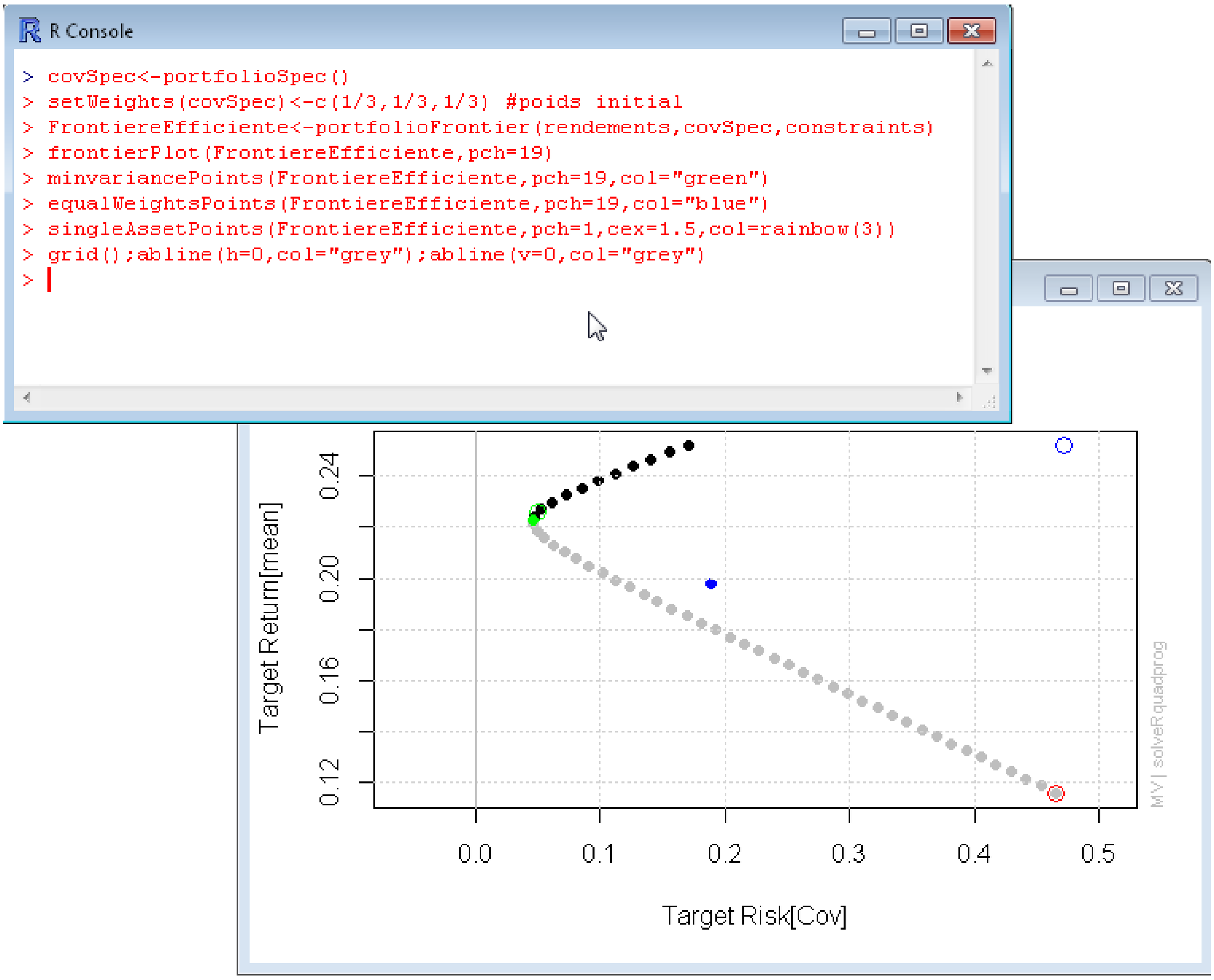
Avec la commande **minvariancePoints( )** nous pouvons afficher le point de variance minimale calculé plus haut:



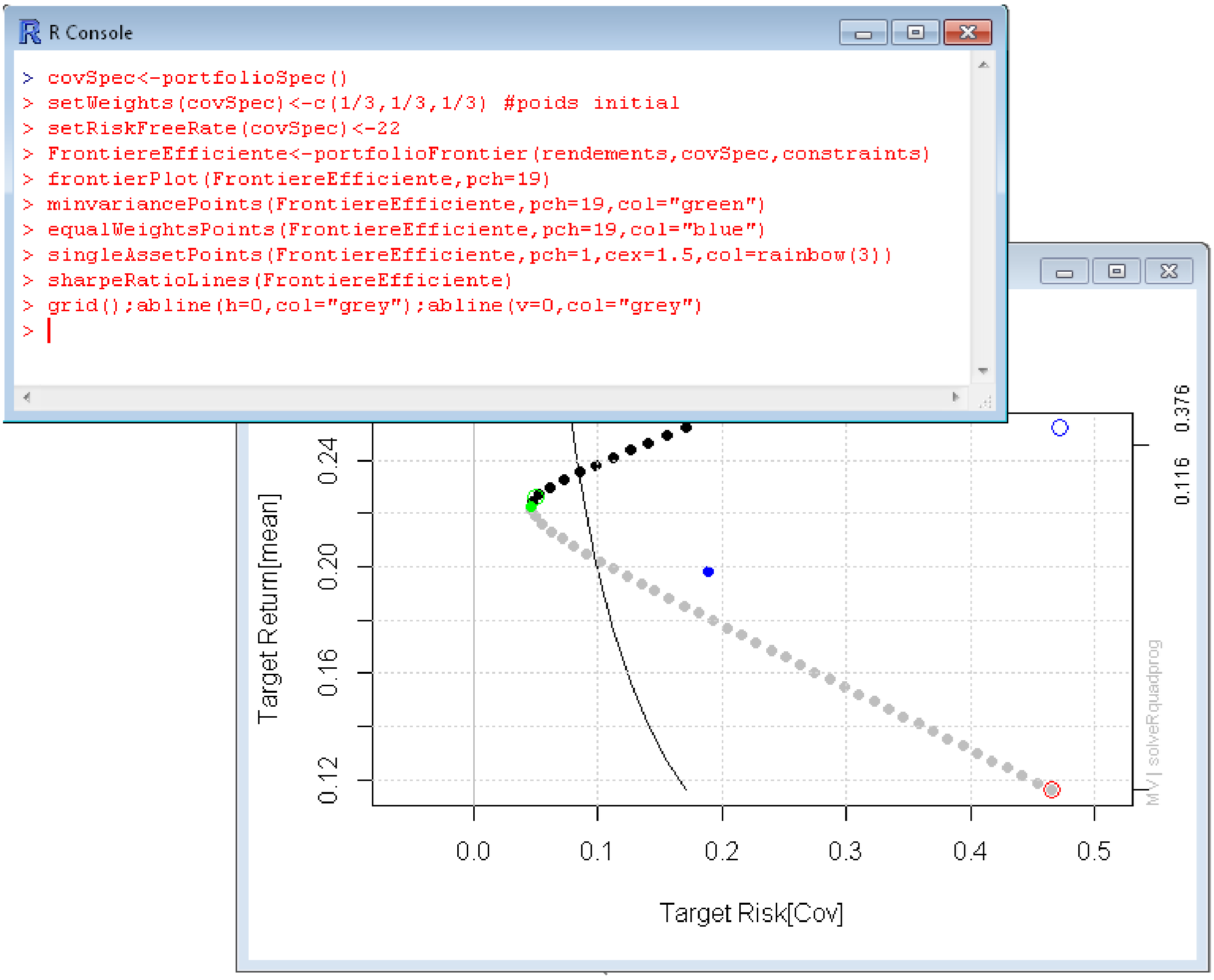
Nous pouvons avec **equalWeightsPoints( )** afficher le portefeuille équilibré:



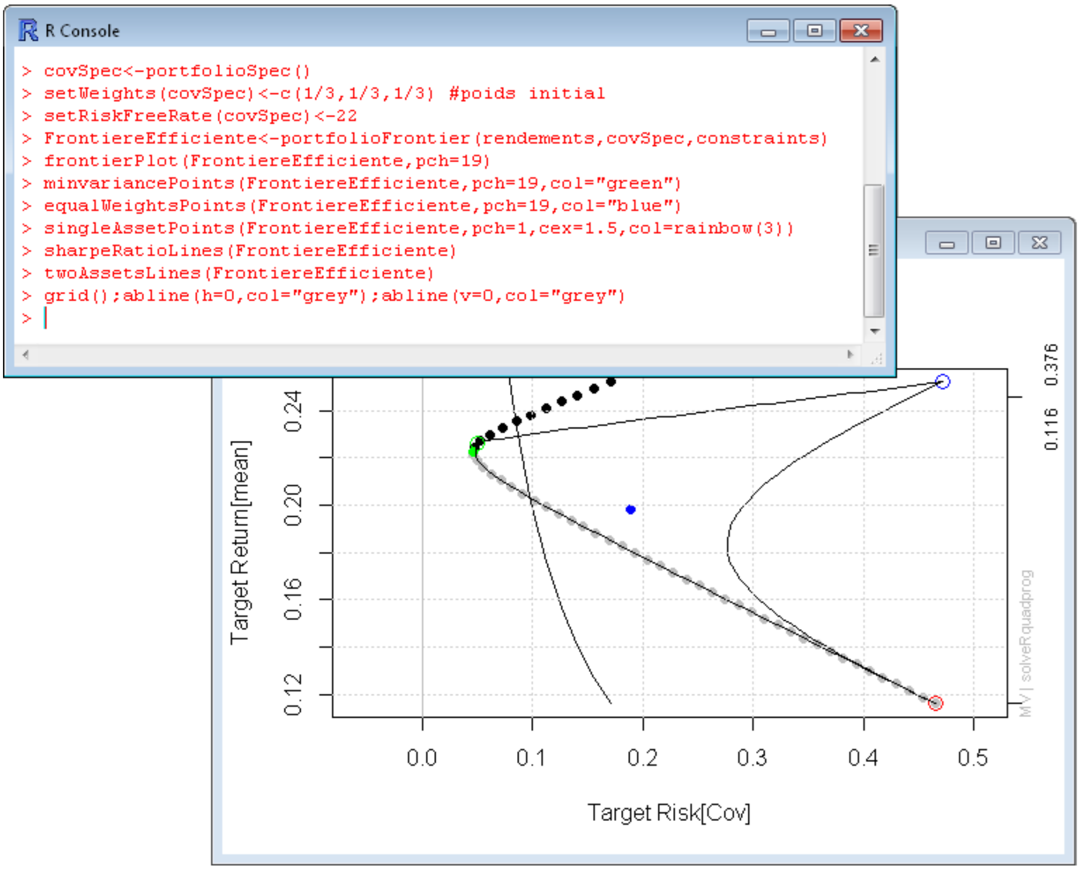
Nous pouvons afficher les trois portefeuilles avec a chaque fois un actif seul avec la commande **singleAssetPoints( )** :



En fixant le taux sans risque, nous pouvons pour tracer la courbe du ratio de Sharpe comme nous l'avons vu dans le cours théorique avec la commande **sharpeRatioLines( )** :



Avec la commande **twoAssetsLines( )** on peut voir la frontière efficiente de portefeuilles composés de deux actifs seulement:



Avec la commande **monteCarloPoints( )** on peut simuler quelques valeurs de portefeuilles réalisables:

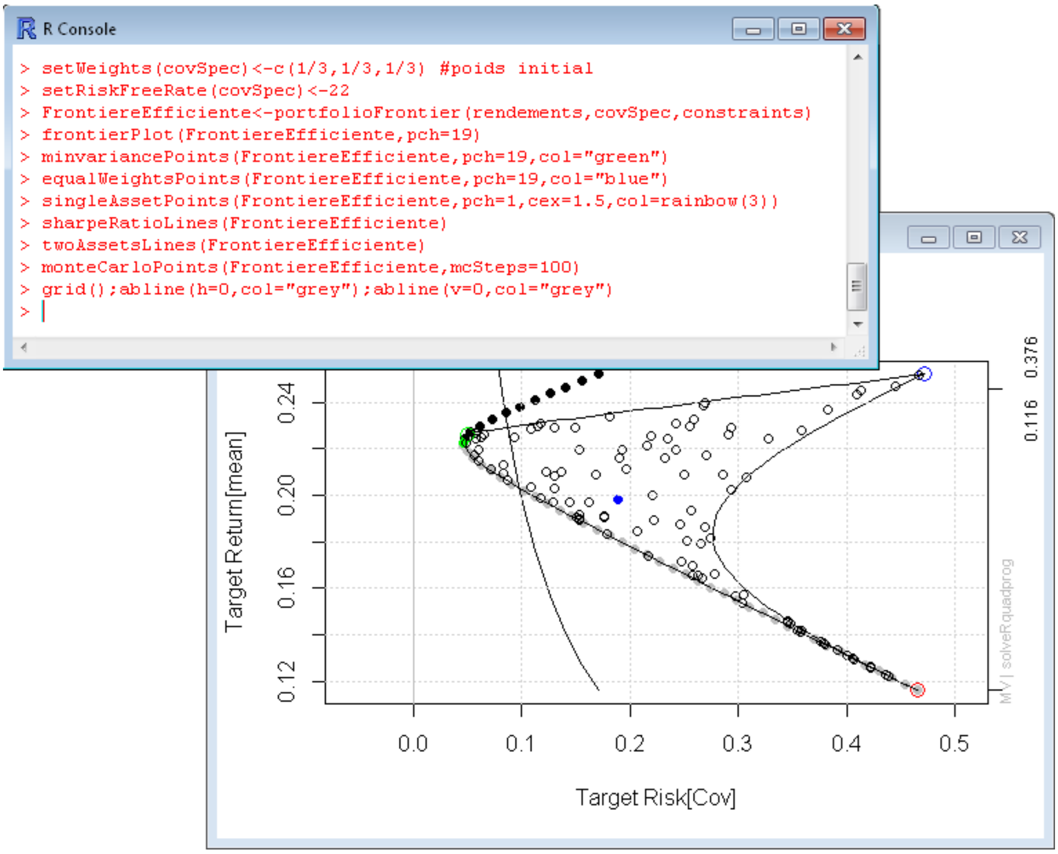


Figure 330 Graph frontière d'efficience de Markowitz avec courbe de Sharpe, portefeuilles simulés et autres

Nous allons maintenant déterminer la capital market line (CML) telle que faite approximativement et grossièrement dans le cours théorique. Nous allons voir ici, s'il y a une grande différence avec R en utilisant les commandes **tangencyLines( )** et **tangencyPoints( )** :

```
> covSpec<-portfolioSpec()
> setWeights(covSpec)<-c(1/3,1/3,1/3)
> setNFrontierPoints(covSpec)<-100
> setRiskFreeRate(covSpec)<-0.22
> FrontiereEfficiente<-portfolioFrontier(rendements,covSpec,constraints)
> frontierPlot(FrontiereEfficiente,pch=19,ylim=c(0.2,0.25),xlim=c(0,0.2))
> tangencyLines(FrontiereEfficiente,pch=19,col="blue")
> tangencyPoints(FrontiereEfficiente,pch=19,col="red",cex=1.5)
> abline(h=0.23167,col="red");abline(h=0.22,col="grey");abline(v=0,col="grey")
>
```

Nous avons mis en évidence en gris l'ordonnée 0.22 avec l'axe en 0 de l'actif sans risque que nous nous étions fixés dans le cours théorique (et là nous voyons que tout est bon puisque la tangente passe bien par le point de coordonnée (0,0.22)).

En rouge horizontal nous mis en évidence le rendement du portefeuille trouvé en conséquence dans le cours théorique avec nos approximations successives. Comme nous pouvons le voir, nous avons une différence d'environ 2% entre le point calculé dans le cours théorique de et par les méthodes numériques d'optimisation de R (soit environ 10% d'erreur!).

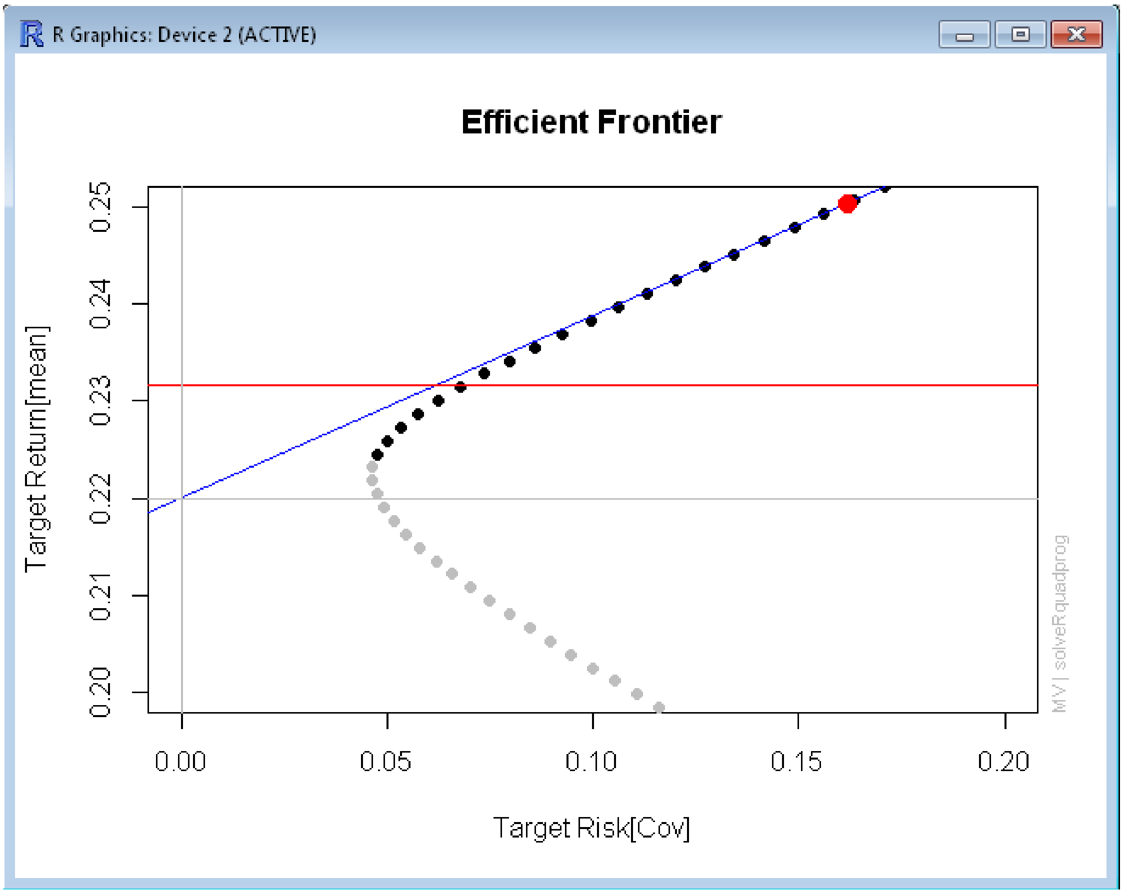
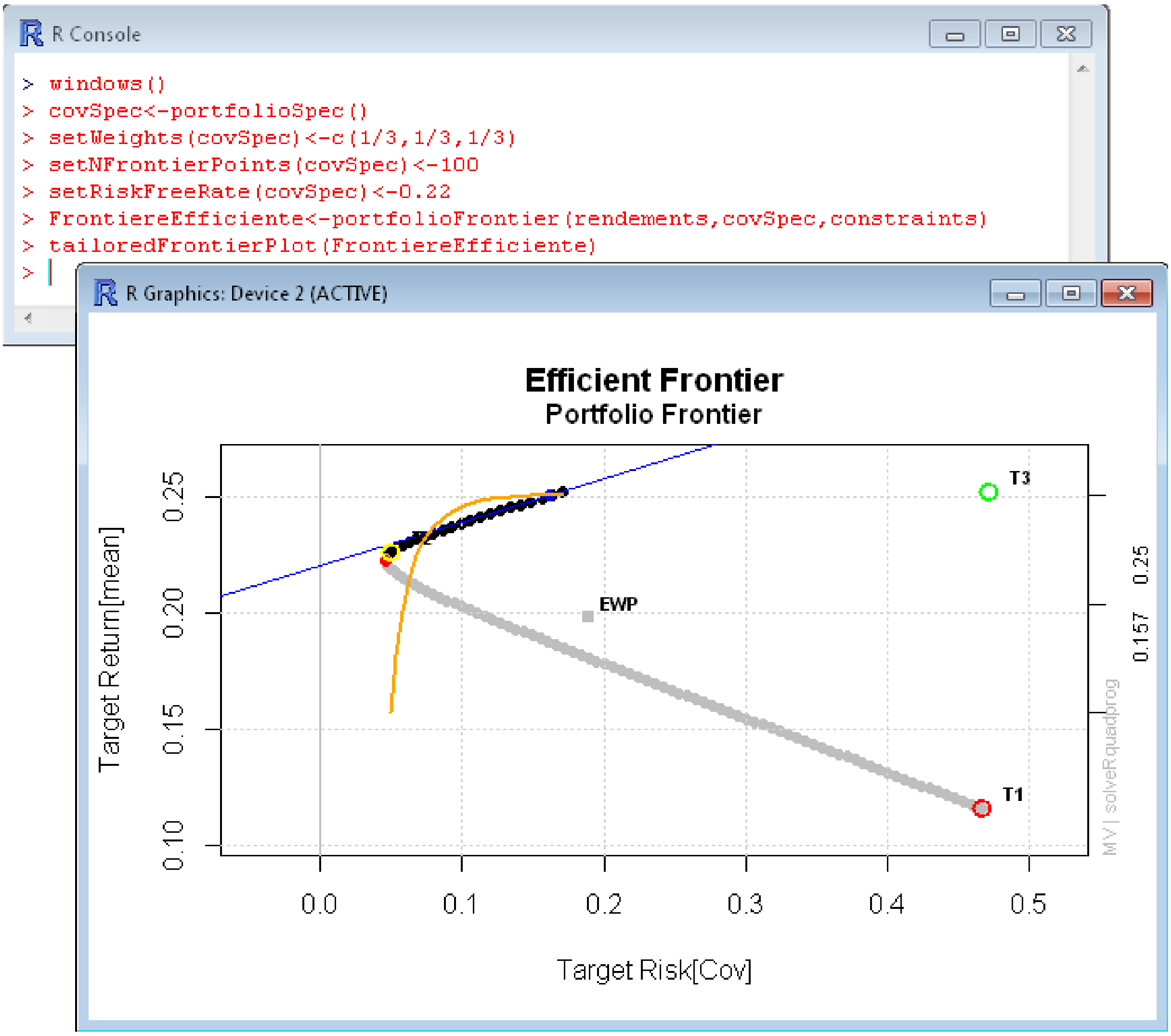


Figure 331 Graph frontière d'efficience de Markowitz avec CML

Je n'ai cependant pas encore trouvé à ce jour comment obtenir facilement avec la package **fPortfolio** le poids des 3 actifs correspondant au point de tangence en rouge ci-dessus. Si quelqu'un trouve qu'il n'hésite pas à nous le communiquer (nous mentionnerons son nom).

Nous pouvons avoir un autre visuel résumant dans les grandes lignes les propriétés de notre portefeuille en utilisant la commande **tailoredFrontierPlot( )** :

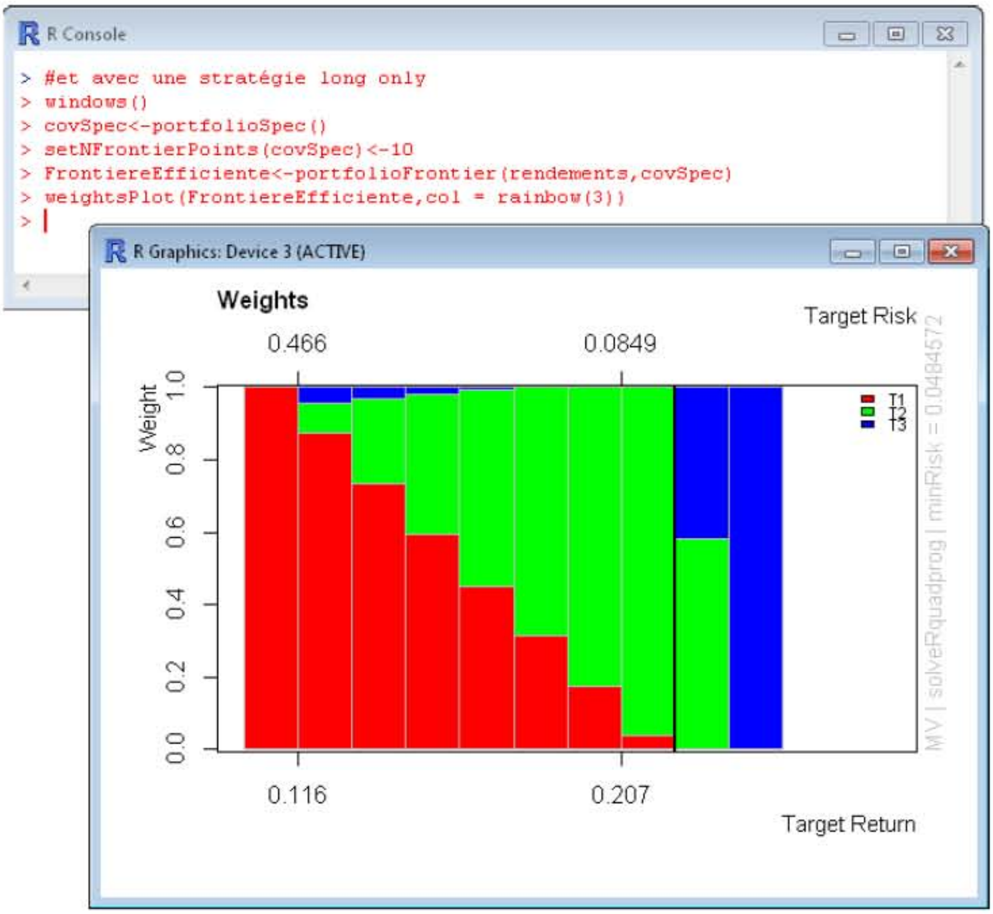


Ou encore un diagramme des poids avec la commande **weightsPlots( )** indiquant quelles combinaisons donnent quels rendements:



Figure 332 Graphique des poids de portefeuille avec rendement et risque d'un portefeuille

Une stratégie long only changera évidemment le graphique:


R Console
> #et avec une stratégie long only
> windows()
> covSpec<-portfolioSpec()
> setNFrontierPoints(covSpec)<-10
> FrontiereEfficiente<-portfolioFrontier(rendements,covSpec)
> weightsPlot(FrontiereEfficiente,col = rainbow(3))
R Graphics: Device 3 (ACTIVE)
Weights
Target Risk
0.466
0.0849
Weight
1.0
0.8
0.6
0.4
0.2
0.0
T1
T2
T3
MV | solveRquadprog | minRisk = 0.0484572
0.116
0.207
Target Return

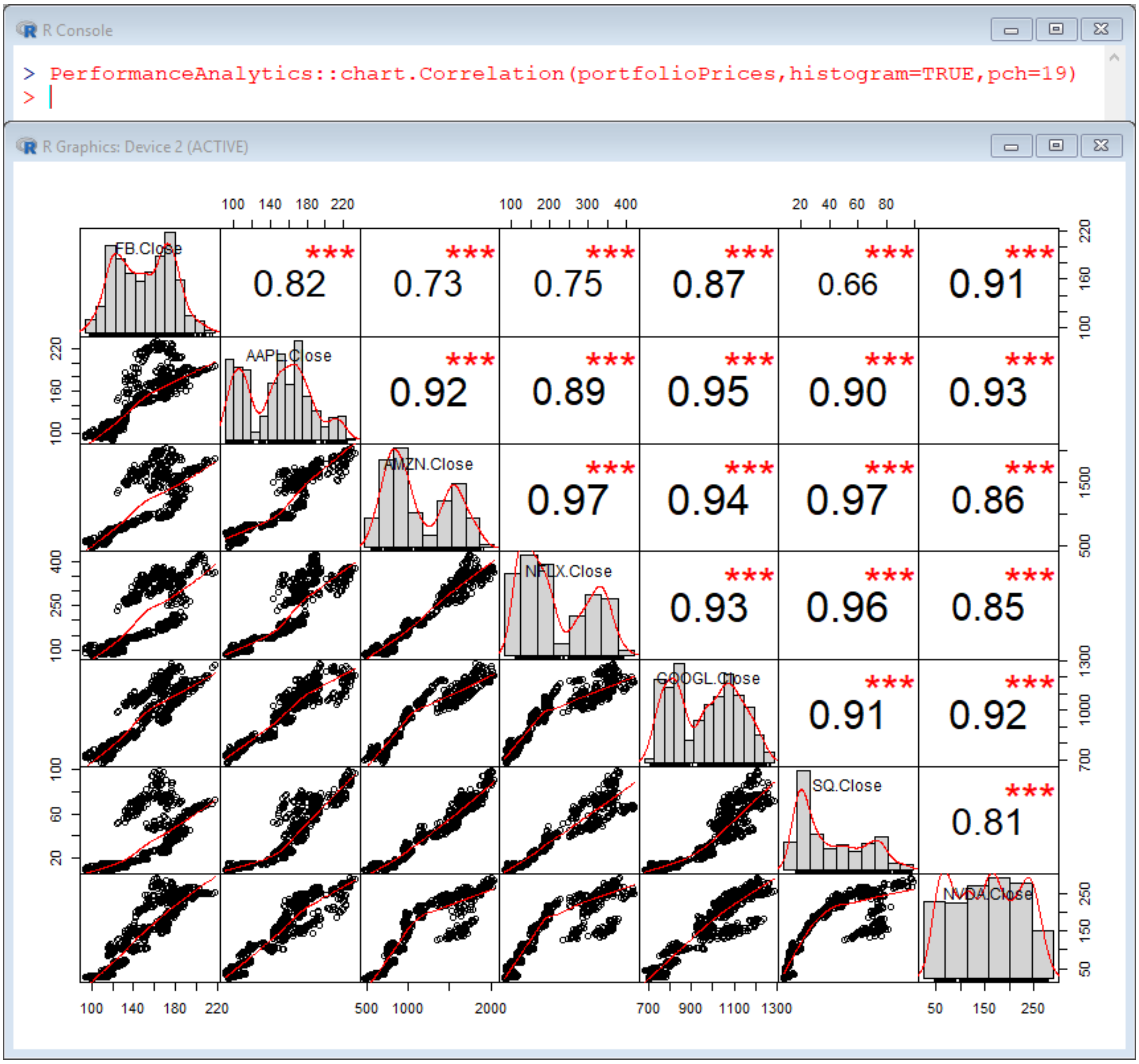
Backtesting de stratégie

Malheureusement nous ne sommes pas encore à ce jour arrivés à faire fonctionner les fonctions de backtesting du package **fPortfolio**. Nous allons donc nous reporter sur une série de packages permettant d'arriver à un résultat quasi-similaire mais pour lequel nous pensons personnellement que nous y gagnons au change!

Commençons donc par importer les données et à effectuer un premier traitement:

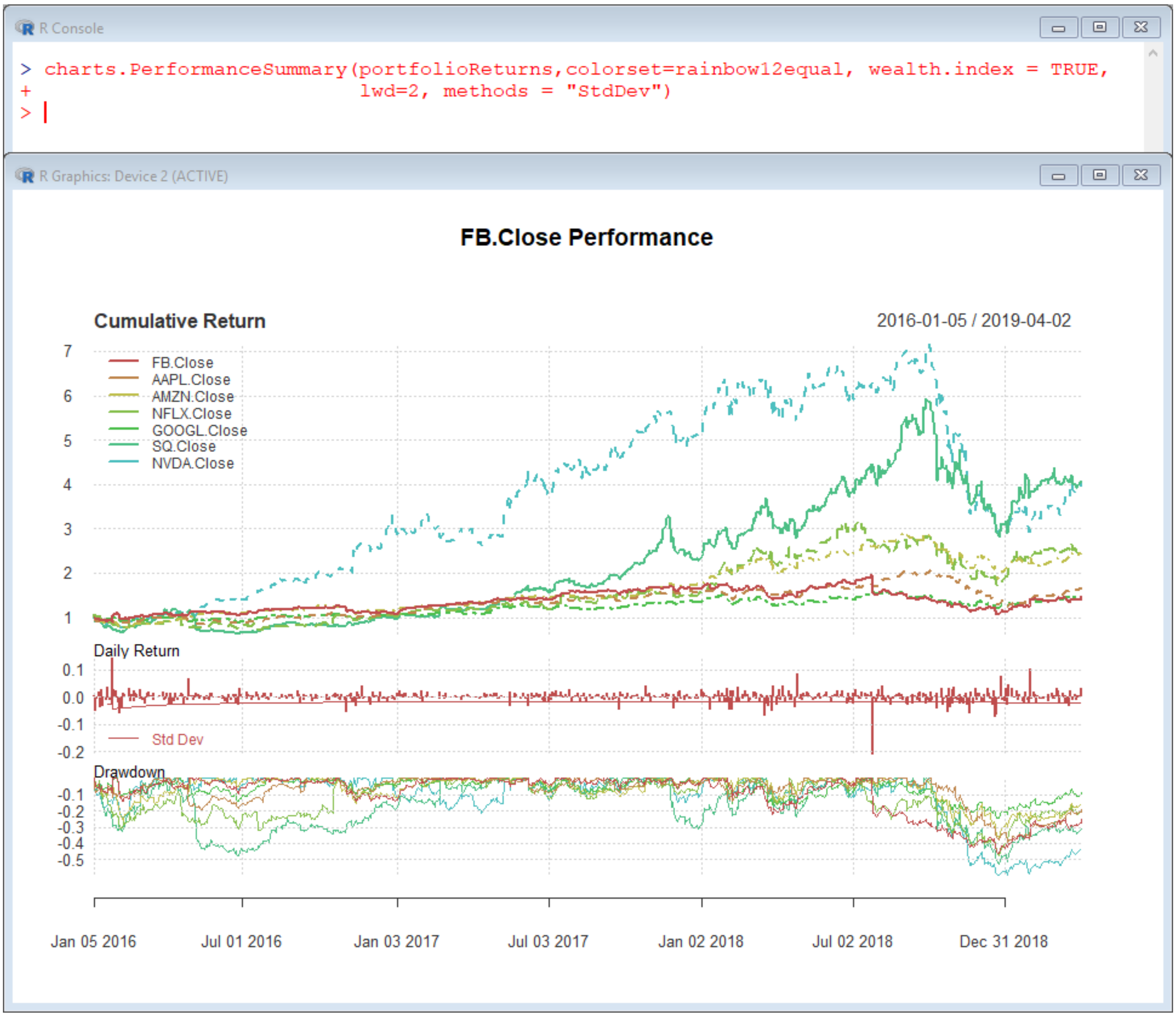
```
> library("quantmod")
> library("PerformanceAnalytics")
> library("PortfolioAnalytics")
> library("TTR")
>
> tickers <- c("FB", "AAPL", "AMZN", "NFLX", "GOOGL", "SQ", "NVDA")
>
> portfolioPrices <- NULL
> for(ticker in tickers) {
+   portfolioPrices <- cbind(portfolioPrices,
+   quantmod::getSymbols.yahoo(ticker, from='2016-01-03', periodicity = 'daily', auto.assign=FALSE)[,4])
+ }
> head(portfolioPrices)
           FB.Close AAPL.Close AMZN.Close NFLX.Close GOOGL.Close SQ.Close NVDA.Close
2016-01-04   102.22     105.35     636.99     109.96      759.44    12.16      32.37
2016-01-05   102.73     102.71     633.79     107.66      761.53    11.51      32.89
2016-01-06   102.97     100.70     632.65     117.68      759.33    11.52      31.53
2016-01-07    97.92      96.45     607.94     114.56      741.00    11.16      30.28
2016-01-08    97.33      96.96     607.05     111.39      730.91    11.31      29.63
2016-01-11    97.51      98.53     617.74     114.97      733.07    11.84      29.68
>
> #on supprime les valeurs NA et on calcule le Rate of Change (ROC)
> portfolioReturns <- na.omit(TTR::ROC(portfolioPrices))
> head(portfolioReturns)
               FB.Close   AAPL.Close    AMZN.Close  NFLX.Close  GOOGL.Close      SQ.Close
2016-01-05  0.004976853 -0.025378648 -0.0050363066 -0.02113850  0.002748283 -0.0549356538
2016-01-06  0.002333477 -0.019763704 -0.0018002500  0.08899093 -0.002893118  0.0008684325
2016-01-07 -0.050286895 -0.043121062 -0.0398411884 -0.02687039 -0.024435863 -0.0317486983
2016-01-08 -0.006043511  0.005273804 -0.0014650560 -0.02806114 -0.013710329  0.0133513332
2016-01-11  0.001847670  0.016062548  0.0174565005  0.03163369  0.002950909  0.0457963393
2016-01-12  0.018895330  0.014409036  0.0002428316  0.01390652  0.016599322  0.0208950352
             NVDA.Close
2016-01-05  0.01593659
2016-01-06 -0.04222912
2016-01-07 -0.04045204
2016-01-08 -0.02170013
2016-01-11  0.00168609
2016-01-12  0.01670604
>
```

On peut aussi afficher un graphique sympa avec la fonction **chart.Correlation( )**:



Les étoiles *** indiquent pour rappel que le test de Pearson de la corrélation est très significatif.

Et encore un autre avec la fonction **charts.PerformanceSummary( )**:



Maintenant on va construire les spécifications de notre portefeuille (notez que l'on crée deux portefeuilles: un nommé *portf.dn* dont les nombre de position est limité et l'autre nommé *portf* qui n'a pas de limites dans les positions!!!):

```
> #on définit un object de spécification  de portefeuilles
> portf <- PortfolioAnalytics::portfolio.spec(colnames(portfolioReturns))
> portf
**************************************************
PortfolioAnalytics Portfolio Specification
**************************************************

Call:
PortfolioAnalytics::portfolio.spec(assets = colnames(portfolioReturns))

Number of assets: 7
Asset Names
[1] "FB.Close"    "AAPL.Close"  "AMZN.Close"  "NFLX.Close"  "GOOGL.Close" "SQ.Close"
[7] "NVDA.Close"

>
> #on y met les quelques contraintes et objectifs suivants:
> portf <- add.constraint(portf, type="weight_sum", min_sum=0.99, max_sum=1.01)
> #on impose un coût de transaction
> portf <- add.constraint(portf, type="transaction_cost", ptc = 0.001)
> #on impose que chaque actif doit avoir un poids entre -10% et +40%
> portf <- add.constraint(portf, type="box", min=.10, max=.40)
> portf <- add.objective(portf, type="return", name="mean")
> portf <- add.objective(portf, type="risk", name="StdDev", target=0.005)
> #on ajoute une contrainte comme quoi il ne doit pas y avoir plus de 5 positions
> portf.dn <- add.constraint(portf, type="position_limit", max_pos=5)
> portf
**************************************************
PortfolioAnalytics Portfolio Specification
**************************************************

Call:
PortfolioAnalytics::portfolio.spec(assets = colnames(portfolioReturns))

Number of assets: 7
Asset Names
[1] "FB.Close"    "AAPL.Close"  "AMZN.Close"  "NFLX.Close"  "GOOGL.Close" "SQ.Close"
[7] "NVDA.Close"

Constraints
Enabled constraint types
                - weight_sum
                - transaction_cost
                - box

Objectives:
Enabled objective names
                - mean
                - StdDev

>
```

Notez qu'on peut aussi définir pour les poids:

```
> portf <- add.constraint(portf, type = "box", min=c(0.05,0.1,0.08,0.06,-0.1,0.3,-0.4),max=c(1,1,1,1,1,1,1))
>
```

Maintenant que les spécifications du portefeuille sont faites, on peut lancer l'optimisation par Monte Carlo :

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #on optimise pour avoir le plot de la frontière de Markowitz
> #on attends quelques secondes ou minutes
> opt_dn <- PortfolioAnalytics::optimize.portfolio(portfolioReturns,
+                                                  portf,
+                                                  optimize_method="random",
+
+                                                  rebalance_on="months",
+                                                  training_period=1,
+                                                  rolling_window=10,
+                                                  trace= TRUE)
>
> #et on affiche les portfeuilles
> plot(opt_dn, main="Markowith Portfolio", risk.col="StdDev", neighbors=20)
>
```

La sortie de session donnera:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> summary(opt_dn)
**************************************************
PortfolioAnalytics Optimization Summary
**************************************************

Call:
PortfolioAnalytics::optimize.portfolio(R = portfolioReturns,
    portfolio = portf, optimize_method = "random", trace = TRUE,
    rebalance_on = "months", training_period = 1, rolling_window = 10,
    rp = rp)

Optimal Weights:
   FB.Close   AAPL.Close   AMZN.Close   NFLX.Close GOOGL.Close     SQ.Close   NVDA.Close
      0.106        0.222        0.114        0.100        0.248        0.100        0.100

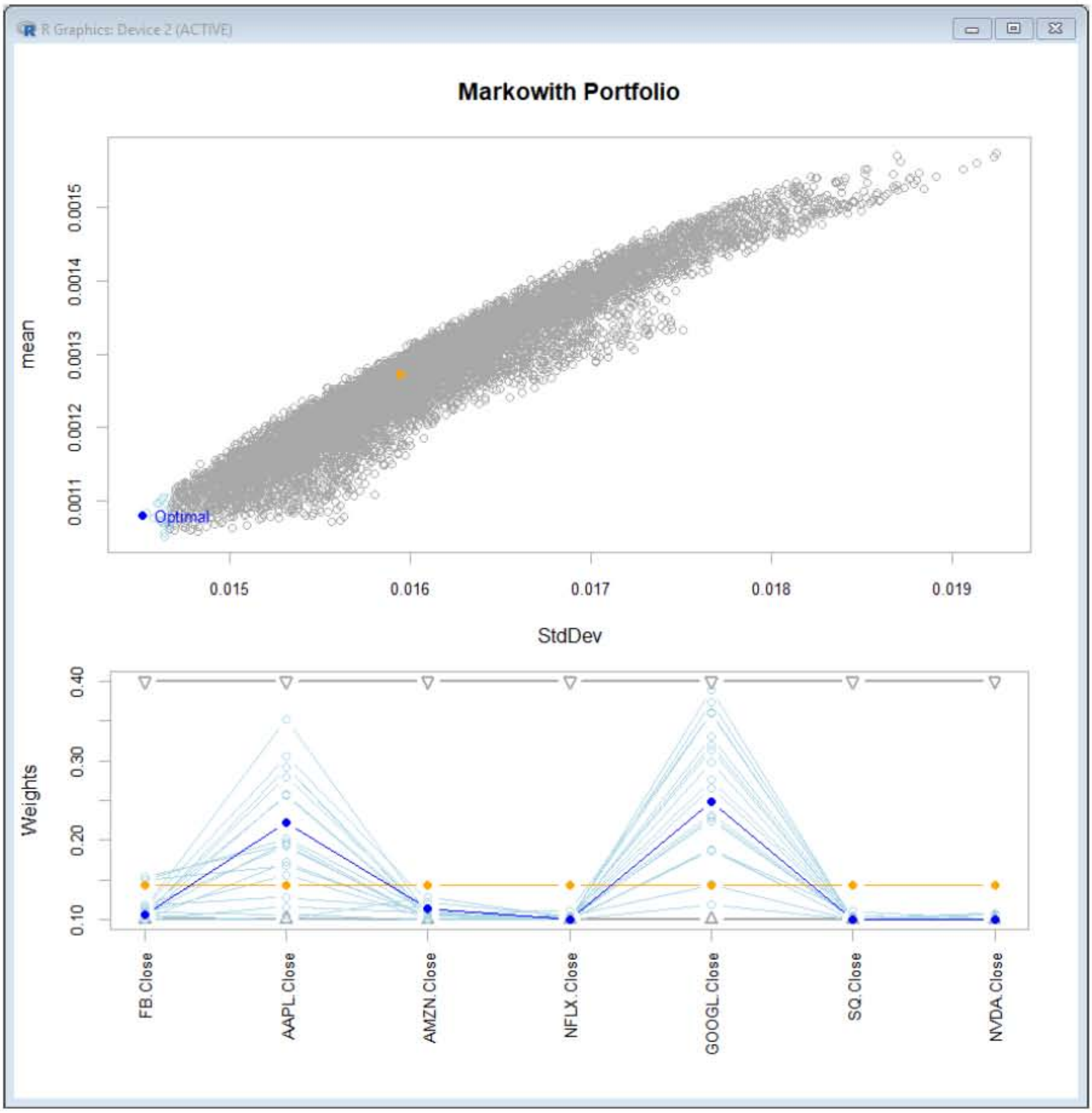
Objective Measures:
   mean
0.00108


 StdDev
0.01452


Portfolio Assets and Initial Weights:
   FB.Close   AAPL.Close   AMZN.Close   NFLX.Close GOOGL.Close     SQ.Close   NVDA.Close
  0.1428571    0.1428571    0.1428571    0.1428571    0.1428571    0.1428571    0.1428571

**************************************************
PortfolioAnalytics Portfolio Specification
**************************************************
```

Et le graphique:



Maintenant on va analyser et plotter le rebalancement du portefeuille par mois, pour cela on écrira:

```
> #on optimise pour analyser l'évolution des poids
> opt_rebal <- PortfolioAnalytics::optimize.portfolio.rebalancing(portfolioReturns,
+                                            portf,
+                                            optimize_method="random",
+                                            rp=rp,
+                                            rebalance_on="months",
+                                            training_period=1,
+                                            rolling_window=10)
>
> summary(opt_rebal)
**************************************************
PortfolioAnalytics Optimization with Rebalancing
**************************************************

Call:
PortfolioAnalytics::optimize.portfolio.rebalancing(R = portfolioReturns,
    portfolio = portf, optimize_method = "random", rp = rp, rebalance_on = "months",
    training_period = 1, rolling_window = 10)

First rebalance date:
[1] "2016-01-29"

Last rebalance date:
[1] "2019-03-29"

Annualized Portfolio Rebalancing Return:
[1] 0.2621835

Annualized Portfolio Standard Deviation:
[1] 0.2401668

Downside Risk Measures:
                              portfolio.returns
Semi Deviation                           0.0114
Gain Deviation                           0.0093
Loss Deviation                           0.0125
Downside Deviation (MAR=210%)            0.0154
Downside Deviation (Rf=0%)               0.0110
Downside Deviation (0%)                  0.0110
Maximum Drawdown                         0.4187
Historical VaR (95%)                    -0.0259
Historical ES (95%)                     -0.0386
Modified VaR (95%)                      -0.0253
Modified ES (95%)                       -0.0433
>
```

Maintenant affichons le graphique correspondant:



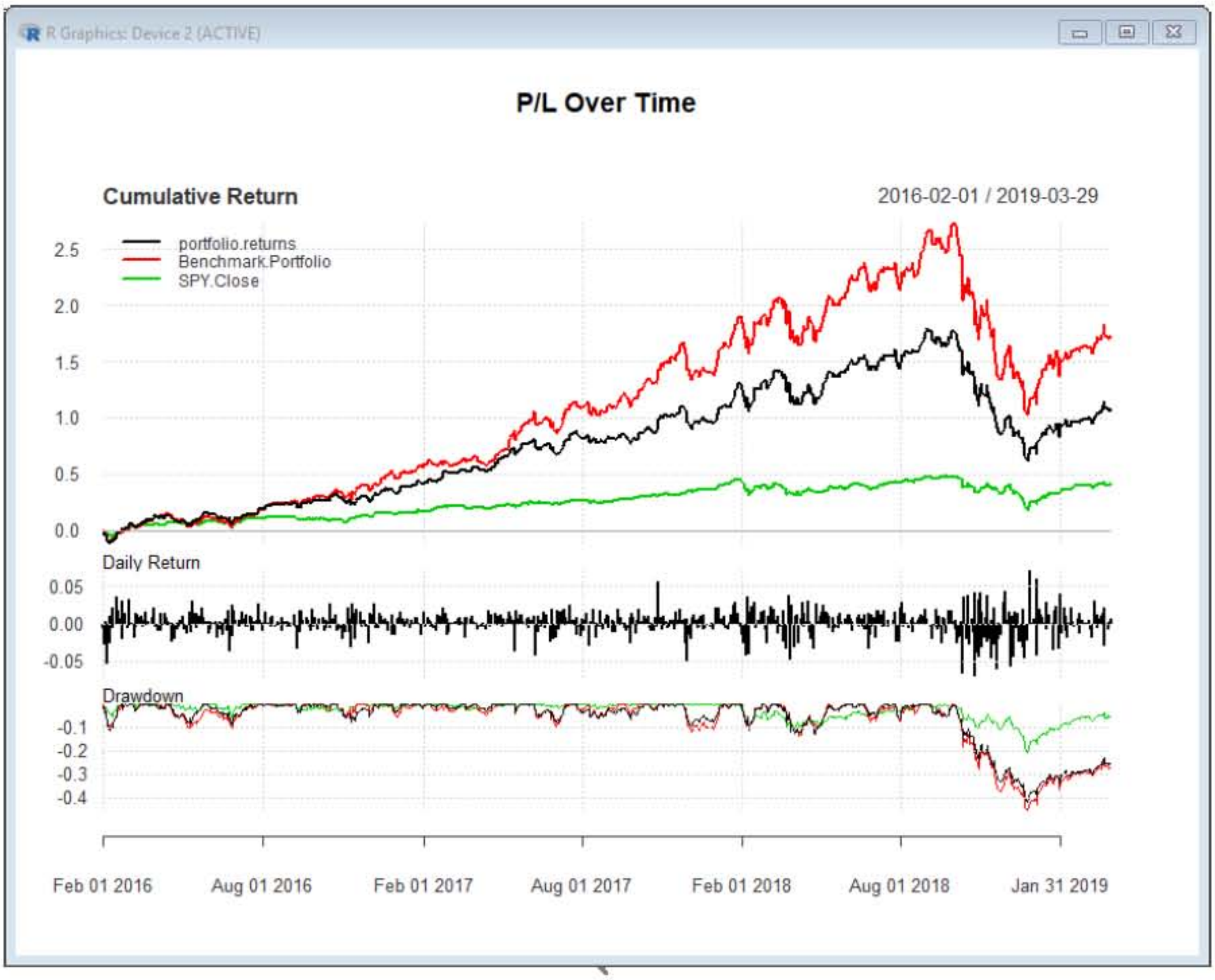
Maintenant pour faire du backtesting, il nous faut un ou des portefeuilles de benchmark… nous allons donc les construire:

```
> (equal_weight <- rep(1 / ncol(portfolioReturns), ncol(portfolioReturns)))
[1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571
> benchmark <- PerformanceAnalytics::Return.portfolio(portfolioReturns, weights = equal_weight)
> colnames(benchmark) <- "Benchmark Portfolio"
> head(benchmark)
           Benchmark Portfolio
2016-01-05        -0.011832483
2016-01-06         0.003385648
2016-01-07        -0.036580082
2016-01-08        -0.007907097
2016-01-11         0.016870462
2016-01-12         0.014488606
>
> sp500prices <- getSymbols.yahoo("SPY", from='2016-01-03', periodicity = 'daily', auto.assign=FALSE)[,4]
> sp500Rets <- na.omit(ROC(sp500prices))
> sp500Rets <- as.xts(sp500Rets)
> head(sp500Rets)
               SPY.Close
2016-01-05  0.0016899303
2016-01-06 -0.0126944278
2016-01-07 -0.0242840547
2016-01-08 -0.0110372652
2016-01-11  0.0009895217
2016-01-12  0.0080359350
>
```

Mais pour faire le backtesting avec ces portfolios, il nous faut aussi construire les rendements de nos portefeuilles optimisés et vue que ce que nous avons est uniquement leurs poids, nous allons devoir mettre un tout petit peu la main dans le cambouis pour en extraire les rendements correspondants:

```
> rebal_weights <- PortfolioAnalytics::extractWeights(opt_rebal)
> head(rebal_weights)
           FB.Close AAPL.Close AMZN.Close NFLX.Close GOOGL.Close SQ.Close NVDA.Close
2016-01-29    0.100      0.100      0.100      0.102       0.100    0.154      0.334
2016-02-29    0.100      0.100      0.100      0.100       0.336    0.100      0.158
2016-03-31    0.202      0.100      0.100      0.112       0.256    0.100      0.120
2016-04-29    0.162      0.222      0.100      0.100       0.100    0.100      0.220
2016-05-31    0.224      0.130      0.168      0.100       0.100    0.102      0.166
2016-06-30    0.100      0.386      0.104      0.100       0.104    0.100      0.100
> rebal_returns <- PerformanceAnalytics::Return.portfolio(portfolioReturns, weights=rebal_weights)
> head(rebal_returns)
           portfolio.returns
2016-02-01      -0.006866786
2016-02-02      -0.027820575
2016-02-03      -0.007920239
2016-02-04       0.003731294
2016-02-05      -0.053902943
2016-02-08      -0.025716076
>
> rets_df <- cbind(rebal_returns, benchmark, sp500Rets)
> head(rets_df,25)
           portfolio.returns Benchmark.Portfolio     SPY.Close
2016-01-05                NA       -1.183248e-02  0.0016899303
2016-01-06                NA        3.385648e-03 -0.0126944278
2016-01-07                NA       -3.658008e-02 -0.0242840547
2016-01-08                NA       -7.907097e-03 -0.0110372652
2016-01-11                NA        1.687046e-02  0.0009895217
2016-01-12                NA        1.448861e-02  0.0080359350
2016-01-13                NA       -4.681438e-02 -0.0252569149
2016-01-14                NA        4.209125e-06  0.0162835358
2016-01-15                NA       -3.741662e-02 -0.0216998829
2016-01-19                NA       -1.731043e-03  0.0013302474
2016-01-20                NA       -1.235290e-03 -0.0128979019
2016-01-21                NA        1.774835e-03  0.0055863497
2016-01-22                NA        2.098561e-02  0.0203076990
2016-01-25                NA       -9.701530e-03 -0.0152319688
2016-01-26                NA       -1.474812e-03  0.0135509062
2016-01-27                NA       -4.046526e-02 -0.0109428945
2016-01-28                NA        4.112542e-02  0.0051956218
2016-01-29                NA        2.323196e-03  0.0240849611
2016-02-01      -0.006866786        4.946499e-03 -0.0003614477
2016-02-02      -0.027820575       -2.076353e-02 -0.0181865309
2016-02-03      -0.007920239       -1.320037e-02  0.0059770480
2016-02-04       0.003731294       -1.414720e-03  0.0015670047
2016-02-05      -0.053902943       -5.220721e-02 -0.0192339428
2016-02-08      -0.025716076       -1.872914e-02 -0.0135524427
2016-02-09       0.009895919        6.474899e-03  0.0000539032
>
> PerformanceAnalytics::charts.PerformanceSummary(rets_df, main="P/L Over Time")
> |
```

Et donc le graphique de backtesting et benchmarking donnera:


R Graphics: Device 2 (ACTIVE)
P/L Over Time
Cumulative Return
2016-02-01 / 2019-03-29
portfolio.returns
Benchmark.Portfolio
SPY.Close
2.5
2.0
1.5
1.0
0.5
0.0
Daily Return
0.05
0.00
-0.05
Drawdown
-0.1
-0.2
-0.3
-0.4
Feb 01 2016
Aug 01 2016
Feb 01 2017
Aug 01 2017
Feb 01 2018
Aug 01 2018
Jan 31 2019

Exemples 402.: Analyse de la performance

R 3.0.2

Avant de commencer il faut savoir que ce package ne peut manipuler que des séries temporelles de type xts même si le contraire est écrit...

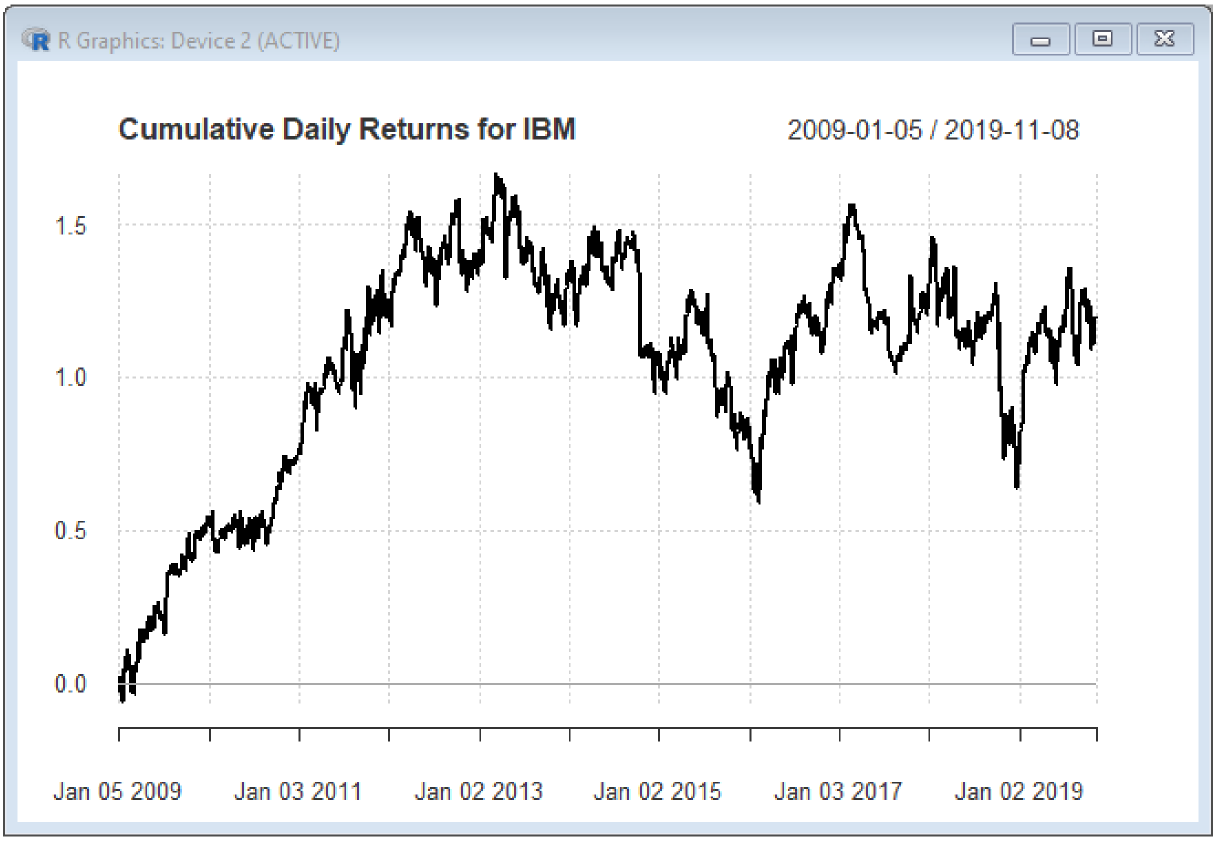
Nous nous concentrerons ici que sur des fonctionnalités du package dont les concepts sous-jacents ont été démontrés dans le cours théorique et sur certaines fonctionnalités ne nécessitant aucun package mathématique particulier mais qui sont visuellement élégantes.

Plotter les retours/rendements (return)

Avec la commande **chart.CumReturns( )** du package **PerformanceAnalytics** en association avec la commande **get.hist.quote( )** du package **tseries** nous pouvons plotter facilement le return en fonction du temps en utilisant les commandes **Return.calculate( )** et **chart.CumReturns( )** du package **PerformanceAnalytics**:

```
R Console
> library("PerformanceAnalytics")
> library("tseries")
> prices<-get.hist.quote("IBM",start="2009-01-01",quote="AdjClose",
+ compression="d")
time series starts 2009-01-02
time series ends   2019-11-08
> ReturnIBM<-Return.calculate(prices,method="simple")
> chart.CumReturns(ReturnIBM,main="Cumulative Daily Returns for IBM")
>
```

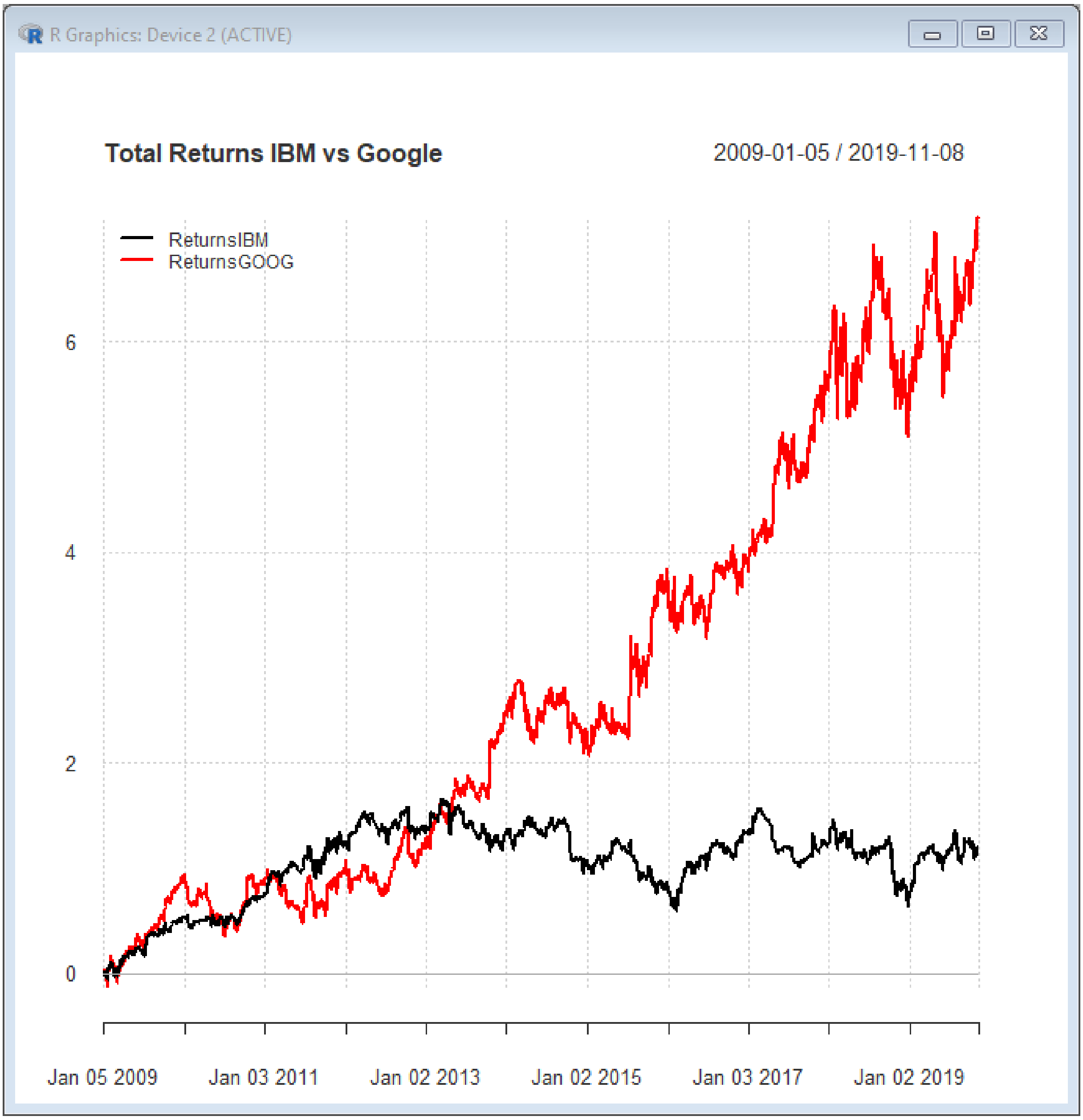
Ce qui nous donne:



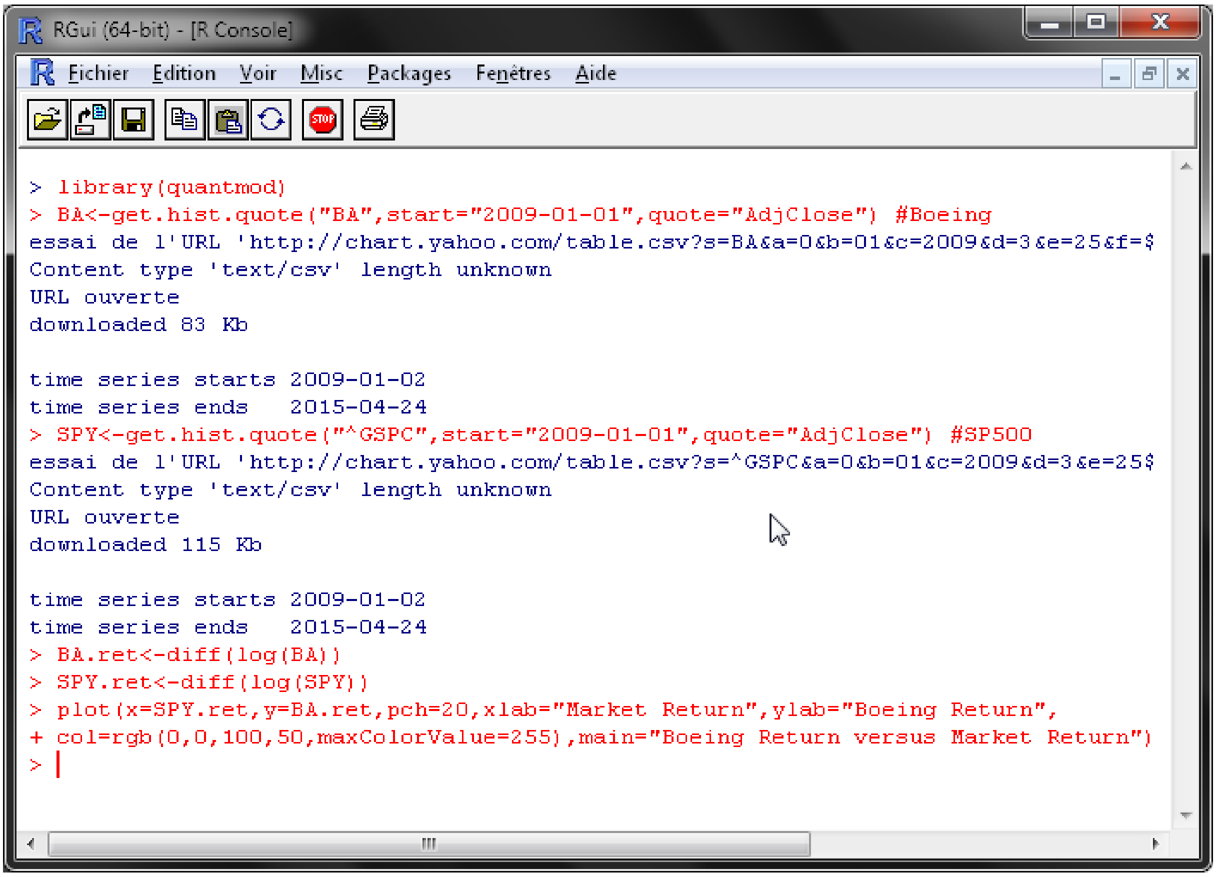
On peut comparer plusieurs tickets:

```
> library("PerformanceAnalytics")
> library("tseries")
> pricesIBM<-get.hist.quote("IBM",start="2009-01-01",quote="AdjClose",
+ compression="d")
time series starts 2009-01-02
time series ends   2019-11-08
> pricesGOOG<-get.hist.quote("GOOG",start="2009-01-01",quote="AdjClose",
+ compression="d")
time series starts 2009-01-02
time series ends   2019-11-08
> ReturnsIBM<-Return.calculate(pricesIBM,method="simple")
> ReturnsGOOG<-Return.calculate(pricesGOOG,method="simple")
> merged<- merge(ReturnsIBM,ReturnsGOOG)
> chart.CumReturns(merged,main="Total Returns IBM vs Google",
+     legend.loc="topleft")
> 
```

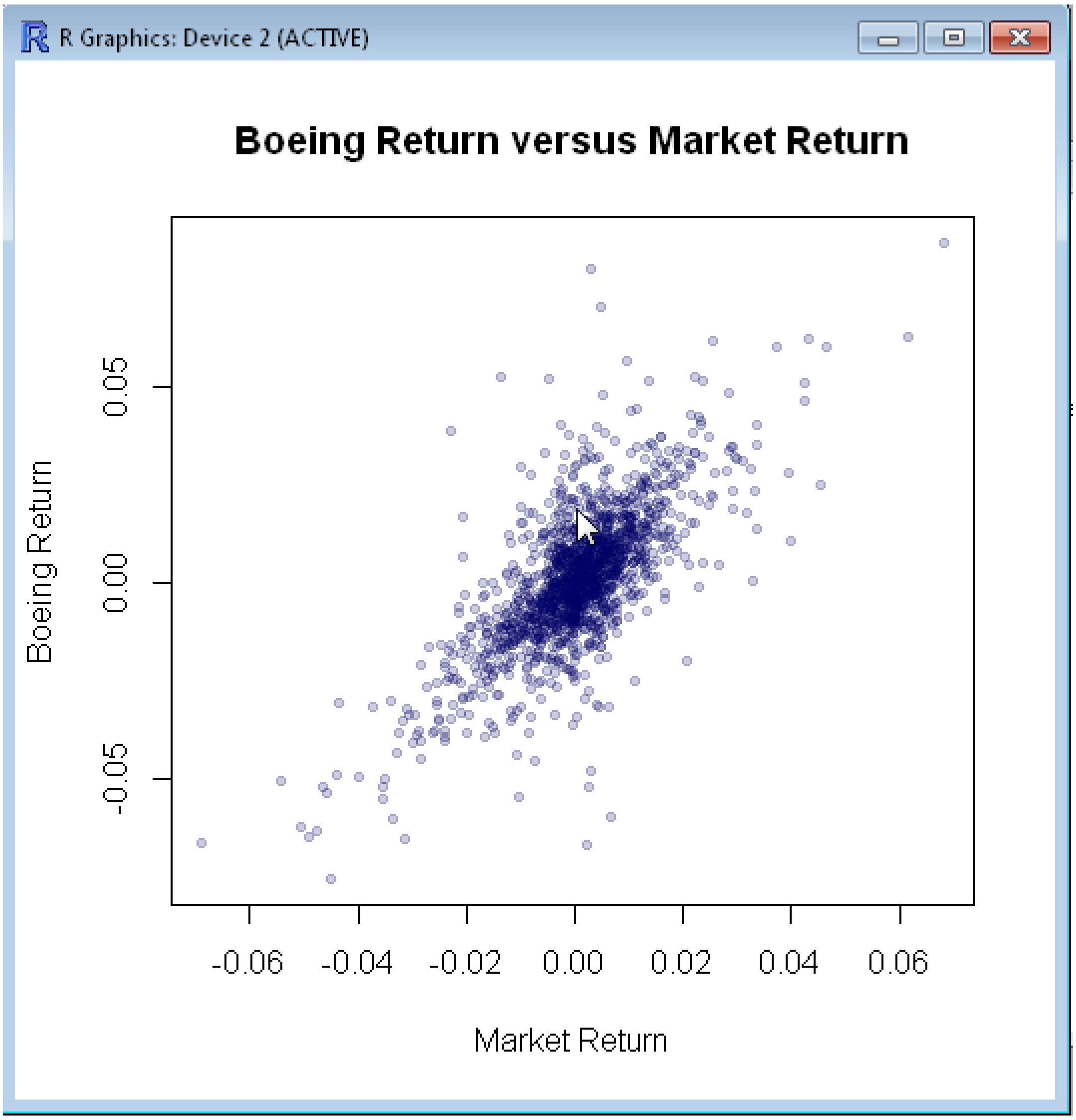
Ce qui donne:



Et nous pouvons faire un plot des retours vis-à-vis d'un indice du marché:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(quantmod)
> BA<-get.hist.quote("BA",start="2009-01-01",quote="AdjClose") #Boeing
essai de l'URL 'http://chart.yahoo.com/table.csv?s=BA&a=0&b=01&c=2009&d=3&e=25&f=$
Content type 'text/csv' length unknown
URL ouverte
downloaded 83 Kb
time series starts 2009-01-02
time series ends   2015-04-24
> SPY<-get.hist.quote("^GSPC",start="2009-01-01",quote="AdjClose") #SP500
essai de l'URL 'http://chart.yahoo.com/table.csv?s=^GSPC&a=0&b=01&c=2009&d=3&e=25$
Content type 'text/csv' length unknown
URL ouverte
downloaded 115 Kb
time series starts 2009-01-02
time series ends   2015-04-24
> BA.ret<-diff(log(BA))
> SPY.ret<-diff(log(SPY))
> plot(x=SPY.ret,y=BA.ret,pch=20,xlab="Market Return",ylab="Boeing Return",
+ col=rgb(0,0,100,50,maxColorValue=255),main="Boeing Return versus Market Return")
>

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Boeing Return versus Market Return
Boeing Return
0.05
0.00
-0.05
-0.06
-0.04
-0.02
0.00
0.02
0.04
0.06
Market Return

Plotter des ratios des performances cumulées

Bon c'est vraiment le genre de fonctions qui ne sert qu'à gagner du temps sinon quoi nous pourrions le programmer nous-même....

Considérons le jeu de données suivant:

```
> data(managers)
> managers
              HAM1    HAM2    HAM3    HAM4    HAM5    HAM6 EDHEC LS EQ  SP500 TR US 10Y TR US 3m TR
1996-01-31  0.0074      NA  0.0349  0.0222      NA      NA          NA  0.034000   0.00380  0.00456
1996-02-29  0.0193      NA  0.0351  0.0195      NA      NA          NA  0.009300  -0.03532  0.00398
1996-03-31  0.0155      NA  0.0258 -0.0098      NA      NA          NA  0.009600  -0.01057  0.00371
1996-04-30 -0.0091      NA  0.0449  0.0236      NA      NA          NA  0.014700  -0.01739  0.00428
1996-05-31  0.0076      NA  0.0353  0.0028      NA      NA          NA  0.025800  -0.00543  0.00443
1996-06-30 -0.0039      NA -0.0303 -0.0019      NA      NA          NA  0.003800   0.01507  0.00412
1996-07-31 -0.0231      NA -0.0337 -0.0446      NA      NA          NA -0.044200  -0.00100  0.00454
1996-08-31  0.0395 -0.0001  0.0461  0.0351      NA      NA          NA  0.021100  -0.00448  0.00451
1996-09-30  0.0147  0.1002  0.0653  0.0757      NA      NA          NA  0.056300   0.02229  0.00470
1996-10-31  0.0288  0.0338  0.0395 -0.0180      NA      NA          NA  0.027600   0.02869  0.00428
1996-11-30  0.0156  0.0737  0.0666  0.0458      NA      NA          NA  0.075600   0.02797  0.00427
1996-12-31  0.0176  0.0298  0.0214  0.0439      NA      NA          NA -0.019800  -0.02094  0.00442
1997-01-31  0.0212  0.0794  0.0771  0.0437      NA      NA      0.0281  0.062500  -0.00055  0.00457
1997-02-28  0.0022 -0.0082 -0.0374  0.0312      NA      NA     -0.0006  0.007800  -0.00167  0.00390
1997-03-31  0.0094 -0.0269 -0.0336  0.0113      NA      NA     -0.0084 -0.041100  -0.01958  0.00422
1997-04-30  0.0126 -0.0061  0.0286  0.0354      NA      NA      0.0084  0.059700   0.01954  0.00477
1997-05-31  0.0438  0.0539  0.0759  0.0789      NA      NA      0.0394  0.060900   0.01033  0.00513
1997-06-30  0.0231  0.0552  0.0054  0.0412      NA      NA      0.0223  0.044800   0.01665  0.00365
1997-07-31  0.0154  0.1150  0.1081  0.0794      NA      NA      0.0454  0.079600   0.04161  0.00450
1997-08-31  0.0237 -0.0197 -0.0028  0.0143      NA      NA      0.0107 -0.056000  -0.02148  0.00428
1997-09-30  0.0219  0.0576  0.0549  0.0217      NA      NA      0.0429  0.054800   0.02153  0.00458
```

Nous souhaitons comparer rapidement la performance des managers relatifs au rendement du SP500:

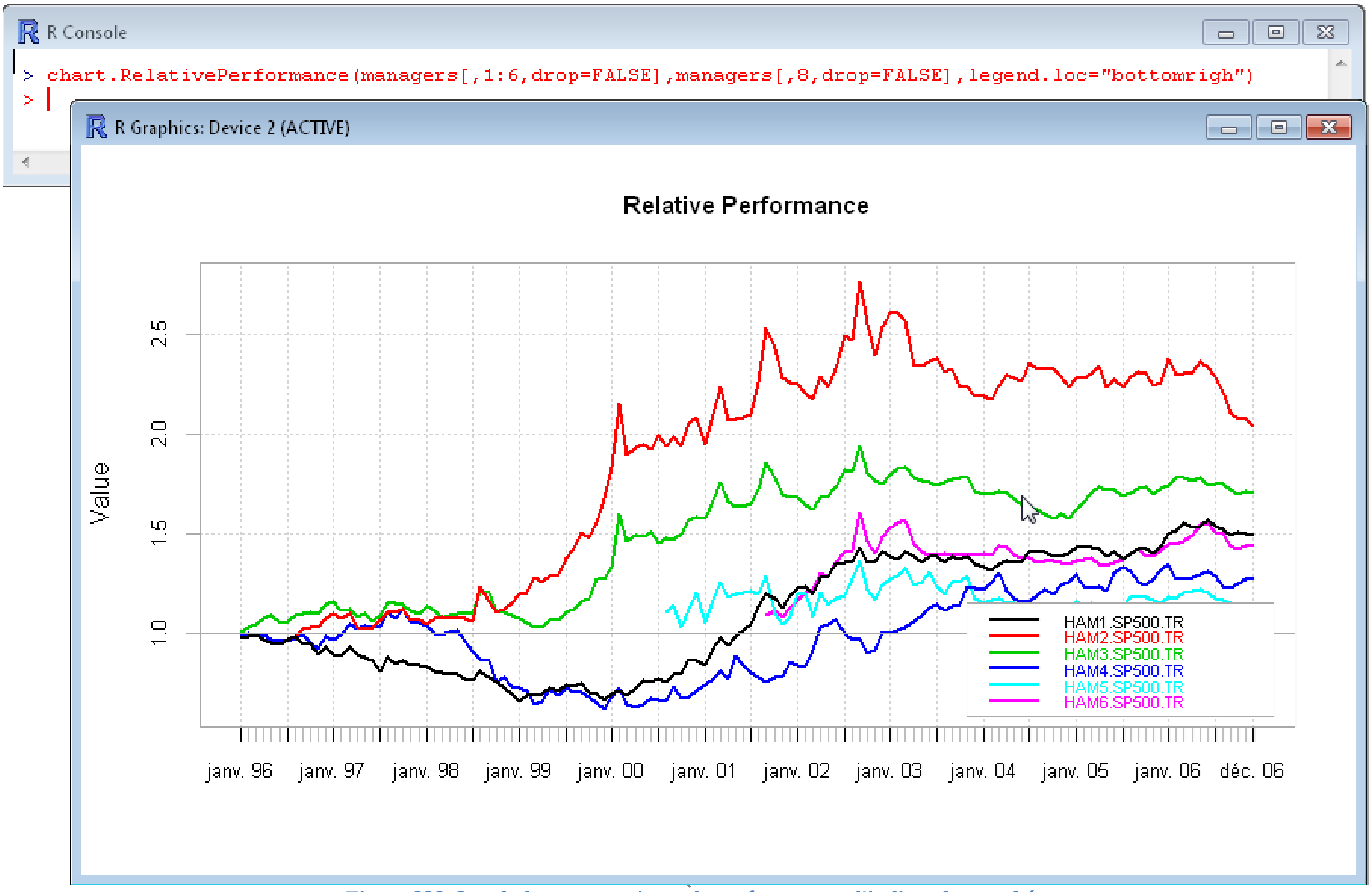
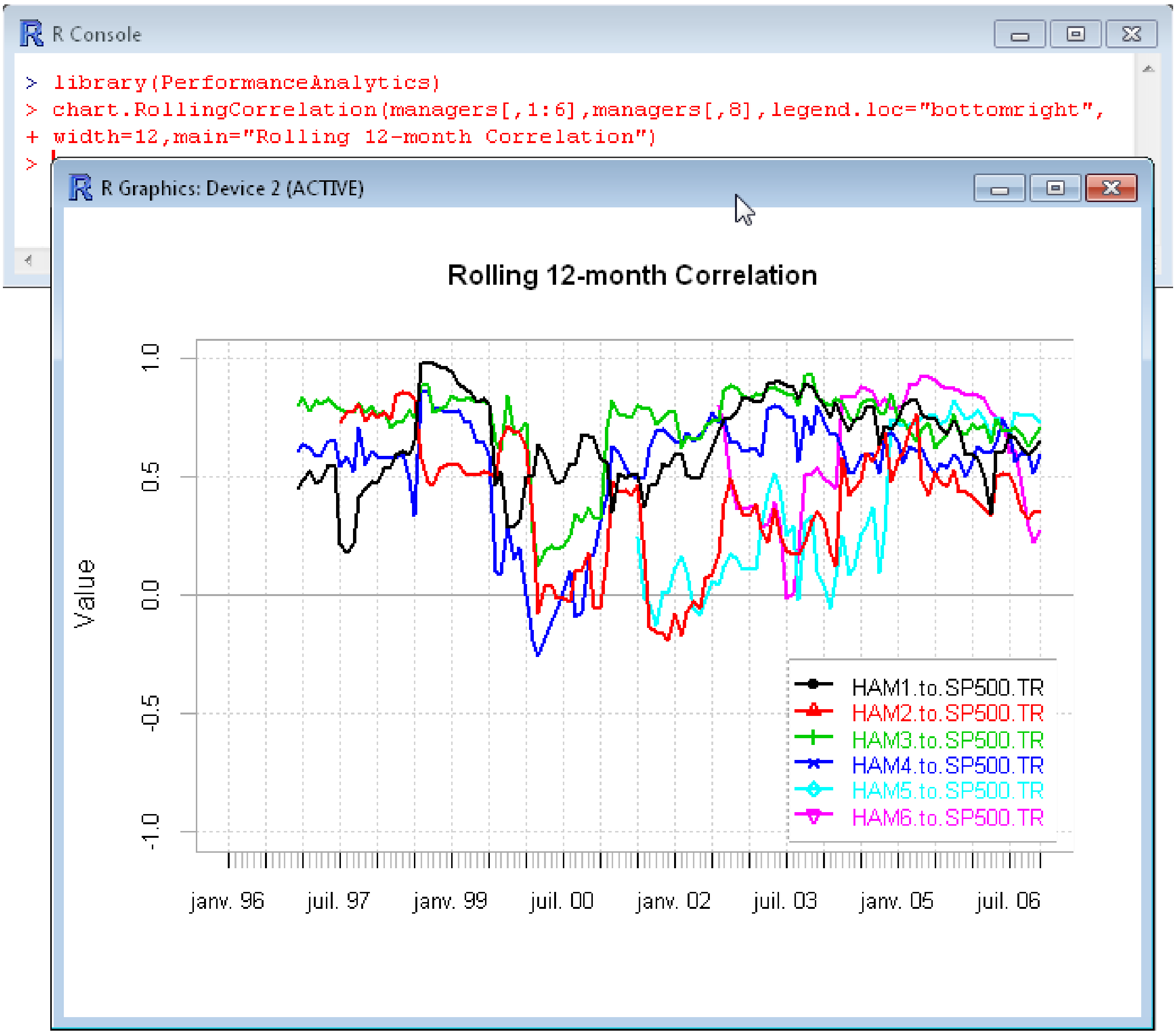


Figure 333 Graph des comparaisons de performances d'indices de marché

Plotter des corrélations par rapport à un indice sur une fenêtre

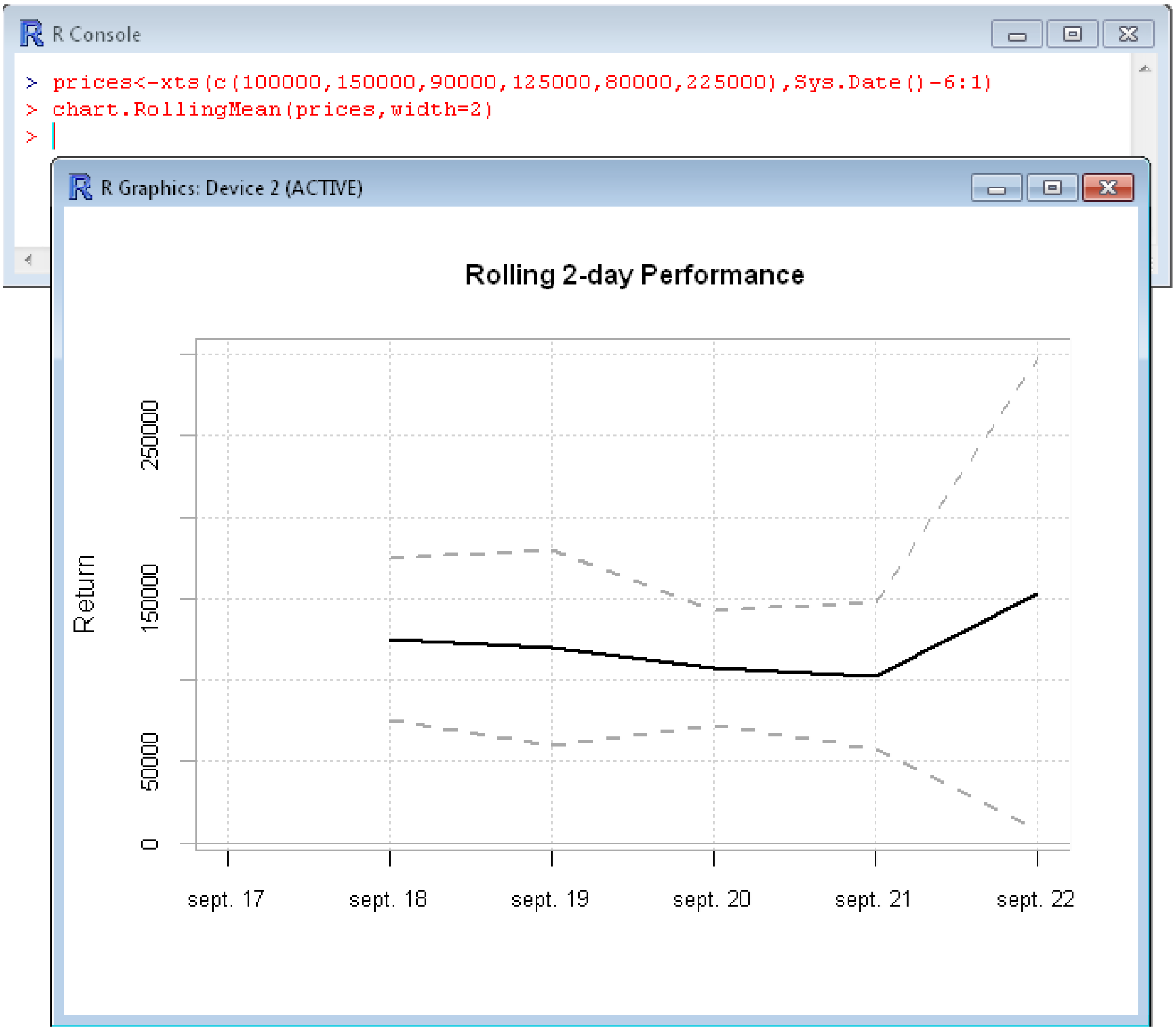
Nous allons dans la même idée de confort qu'avant comparer la corrélation du rendement des managers par rapport à un indice donné sur une fenêtre par exemple de 12 mois. Nous utilisons alors la fonction **chart.RollingCorrelation( )** :



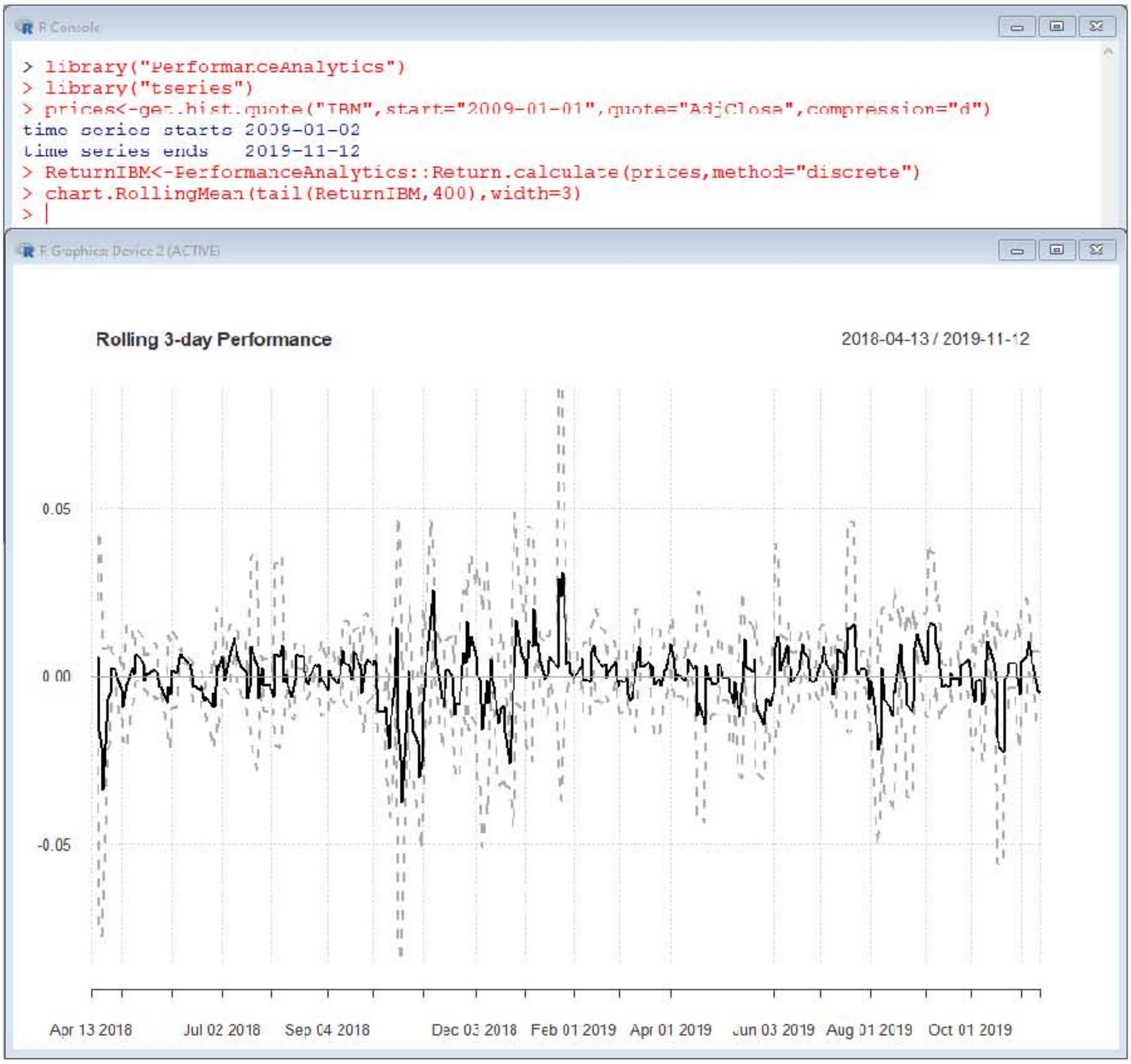
Bon voilà quoi rien d'extraordinaire....

Plotter la moyenne et son intervalle de confiance sur une fenêtre temporelle

L'idée est la même qu'avant mais avec une moyenne et son intervalle de confiance en utilisant la fonction **chart.RollingMean( )** (attention comme à l'habitude de s'assurer de la conformité avec les hypothèses normalement à respecter pour utiliser ce type d'analyse!):

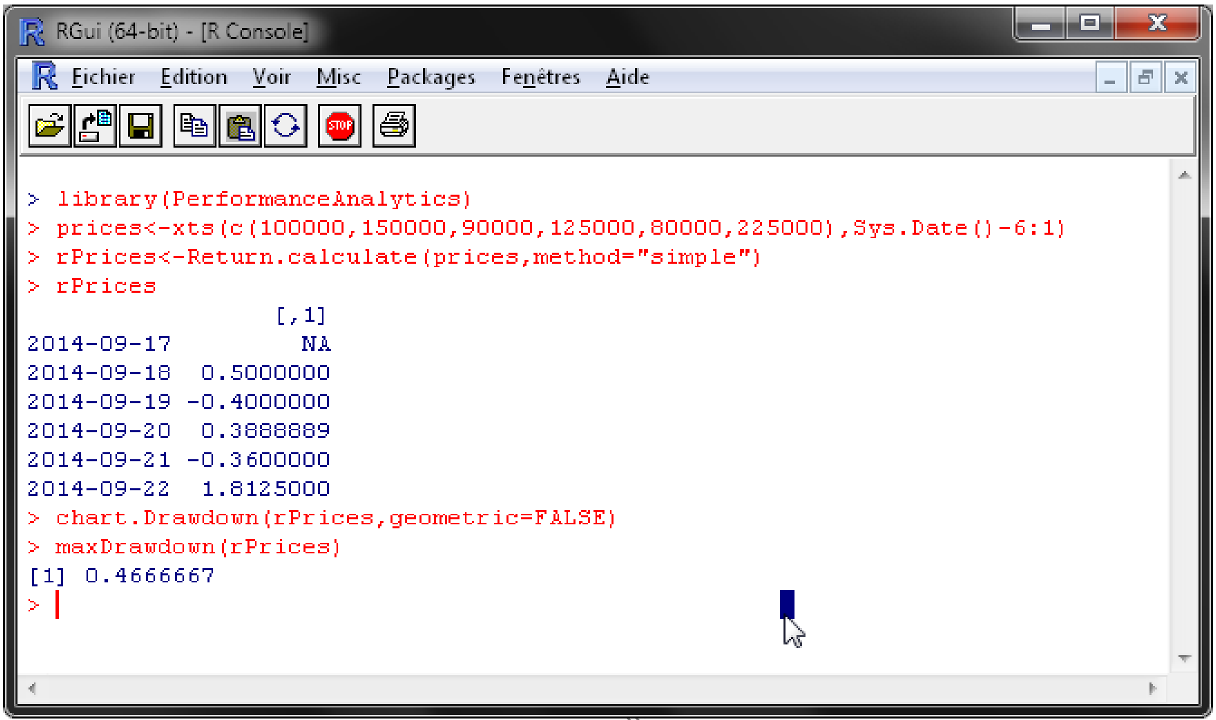


Avec un vrai jeu de données, cela donne quelque chose de plus intéressant:

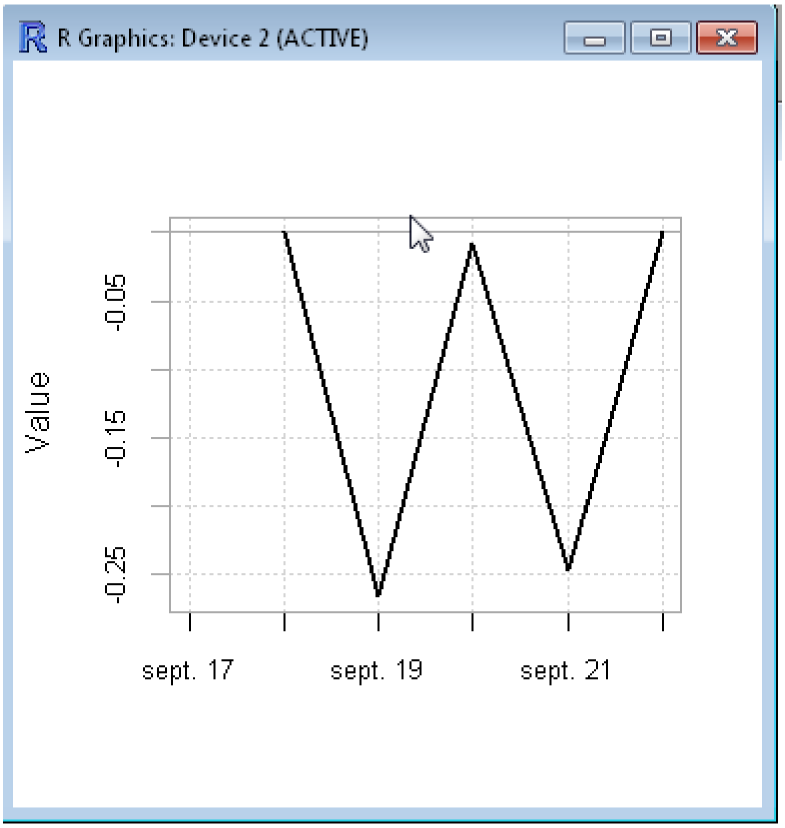

R Console
> library("PerformanceAnalytics")
> library("tseries")
> prices<-get.hist.quote("IBM",start="2009-01-01",quote="AdjClose",compression="d")
time series starts 2009-01-02
time series ends 2019-11-12
> ReturnIBM<-PerformanceAnalytics::Return.calculate(prices,method="discrete")
> chart.RollingMean(tail(ReturnIBM,400),width=3)
R Graphics: Device 2 (ACTIVE)
Rolling 3-day Performance
2018-04-13 / 2019-11-12
0.05
0.00
-0.05
Apr 13 2018
Jul 02 2018
Sep 04 2018
Dec 03 2018
Feb 01 2019
Apr 01 2019
Jun 03 2019
Aug 01 2019
Oct 01 2019

Calculer le maximum Draw Down et faire un graph

Nous avons vu dans le cours théorique une des définitions existantes du maximum Draw Down, nous allons voici ici si nous retrouvons le même résultat qu'à la main et que dans le cours MATLAB:



Nous obtenons bien le même résultat qu'avec MATLAB. Le graphique associé nous donne:



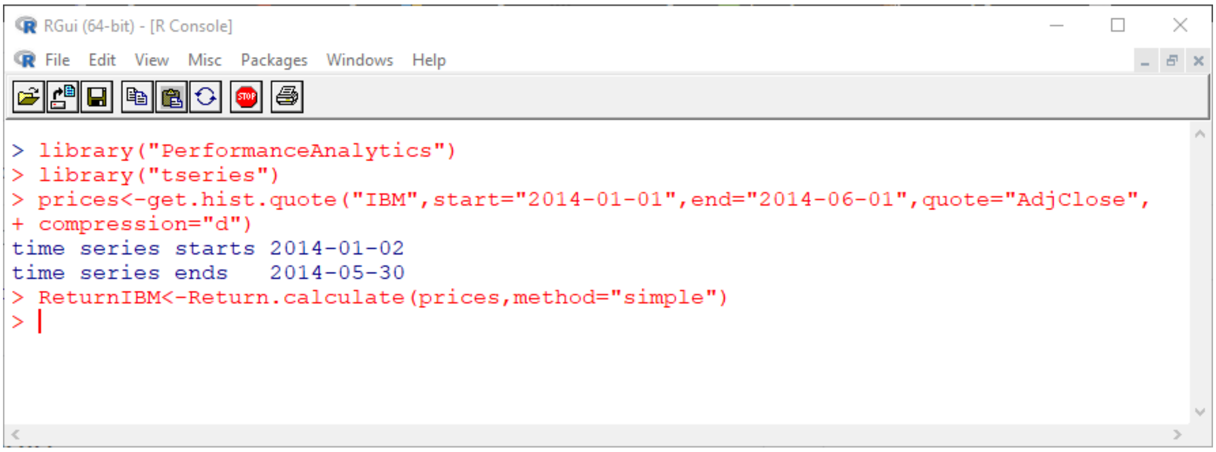
Pour ceux qui veulent comprendre ce graph, voici le code source de la fonction (c'est empirique et bête):

```
2   function (R, geometric = TRUE, ...)
3   { # @author Peter Carl
4
5       # DESCRIPTION:
6       # Calculate the drawdown levels in a timeseries
7
8       # FUNCTION:
9
10      x = checkData(R)
11
12      # Get dimensions and labels
13      columns = ncol(x)
14      columnnames = colnames(x)
15
16      colDrawdown <- function(x, geometric) {
17          if(geometric)
18              Return.cumulative = cumprod(1+x)
19          else
20              Return.cumulative = 1+cumsum(x)
21          maxCumulativeReturn = cummax(c(1,Return.cumulative))[-1]
22          column.drawdown = Return.cumulative/maxCumulativeReturn - 1
23      }
```

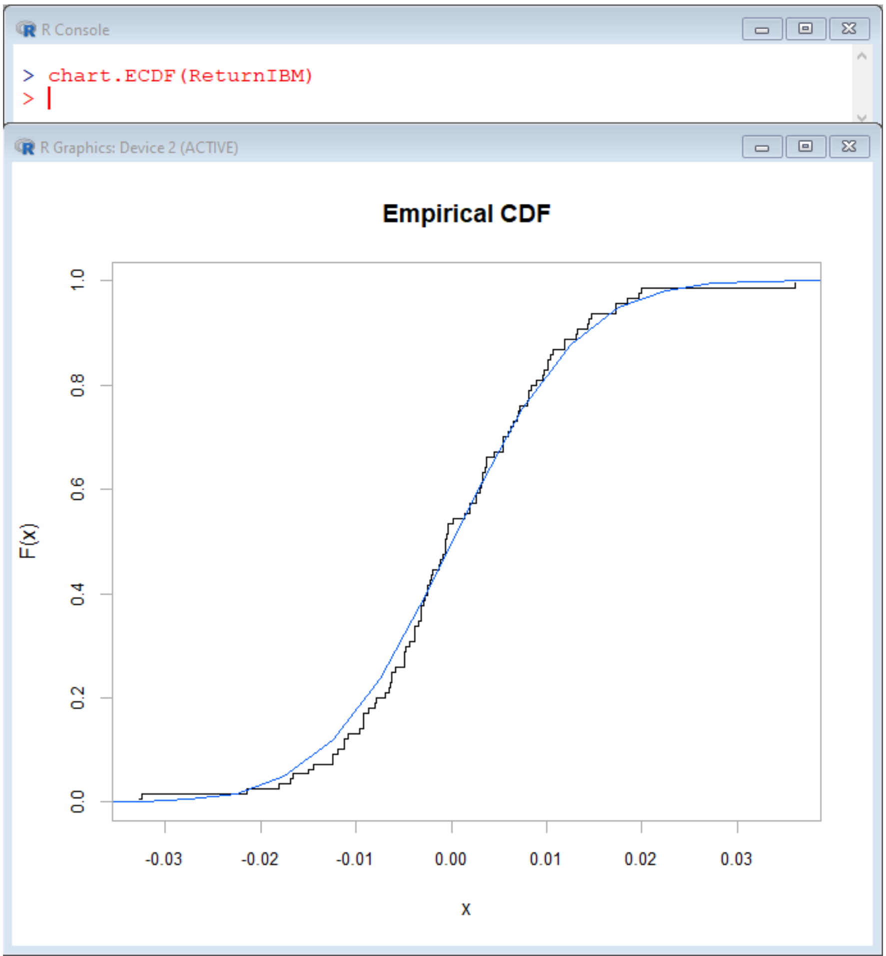
Plotter l'ECDF et l'histogramme des retours/rendements (return)

Bon c'est un ultra-classique qui ne nécessite même pas d'explications. Il s'agit juste de voir que le package **PerformanceAnalytics** a aussi une commande ECDE propre qui compare à une loi Normale Centrée Réduite.

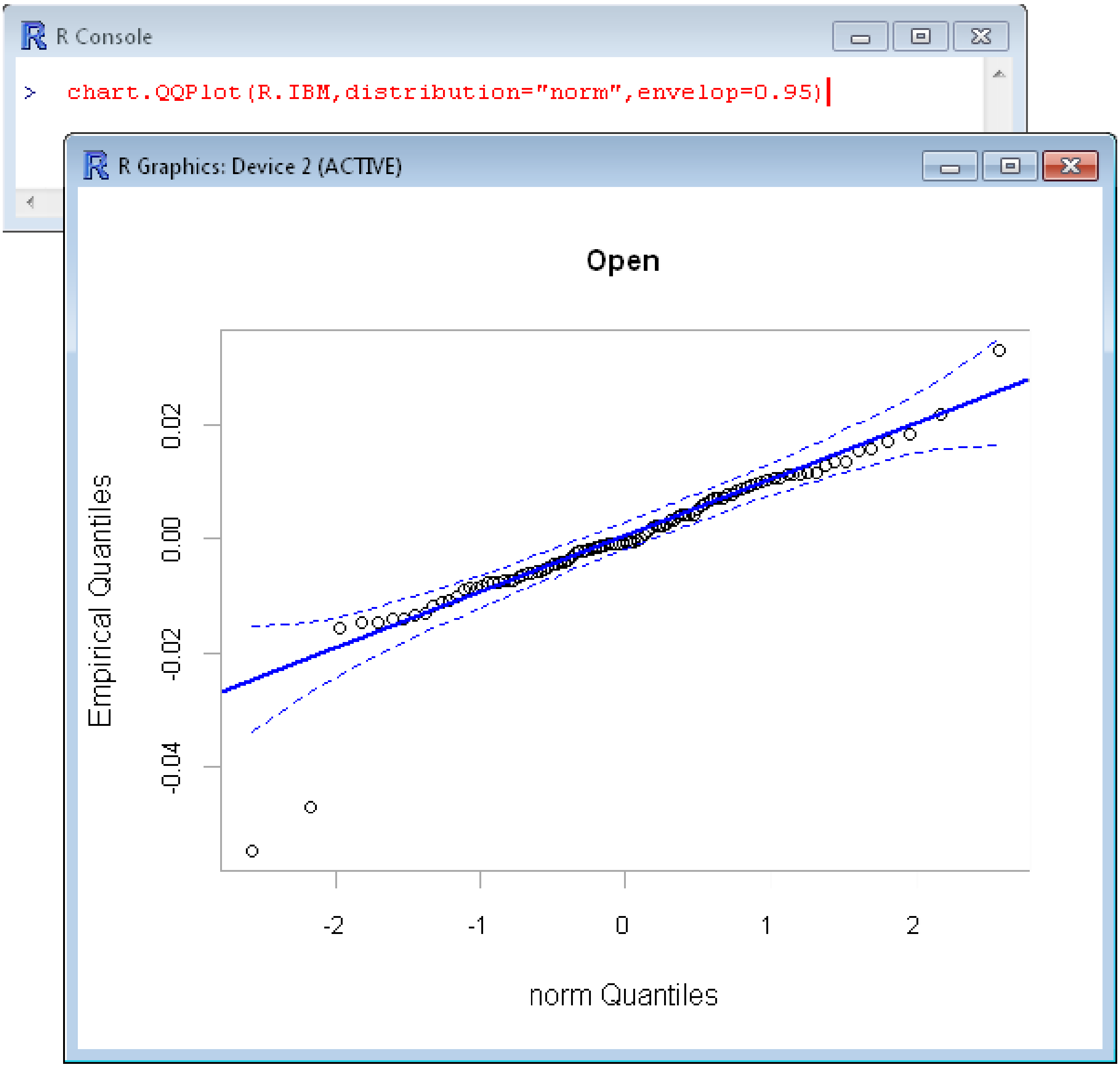
D'abord nous importons les données:



Ensuite, nous comparons de façon très qualitative avec la commande **chart.ECDF( )** :



ou dans sa version QQplot avec la commande **chart.QQPlot( )** toujours du même package:



Dans le même genre ultra-classique, nous avons avec la commande **chart.Histogram( )** :

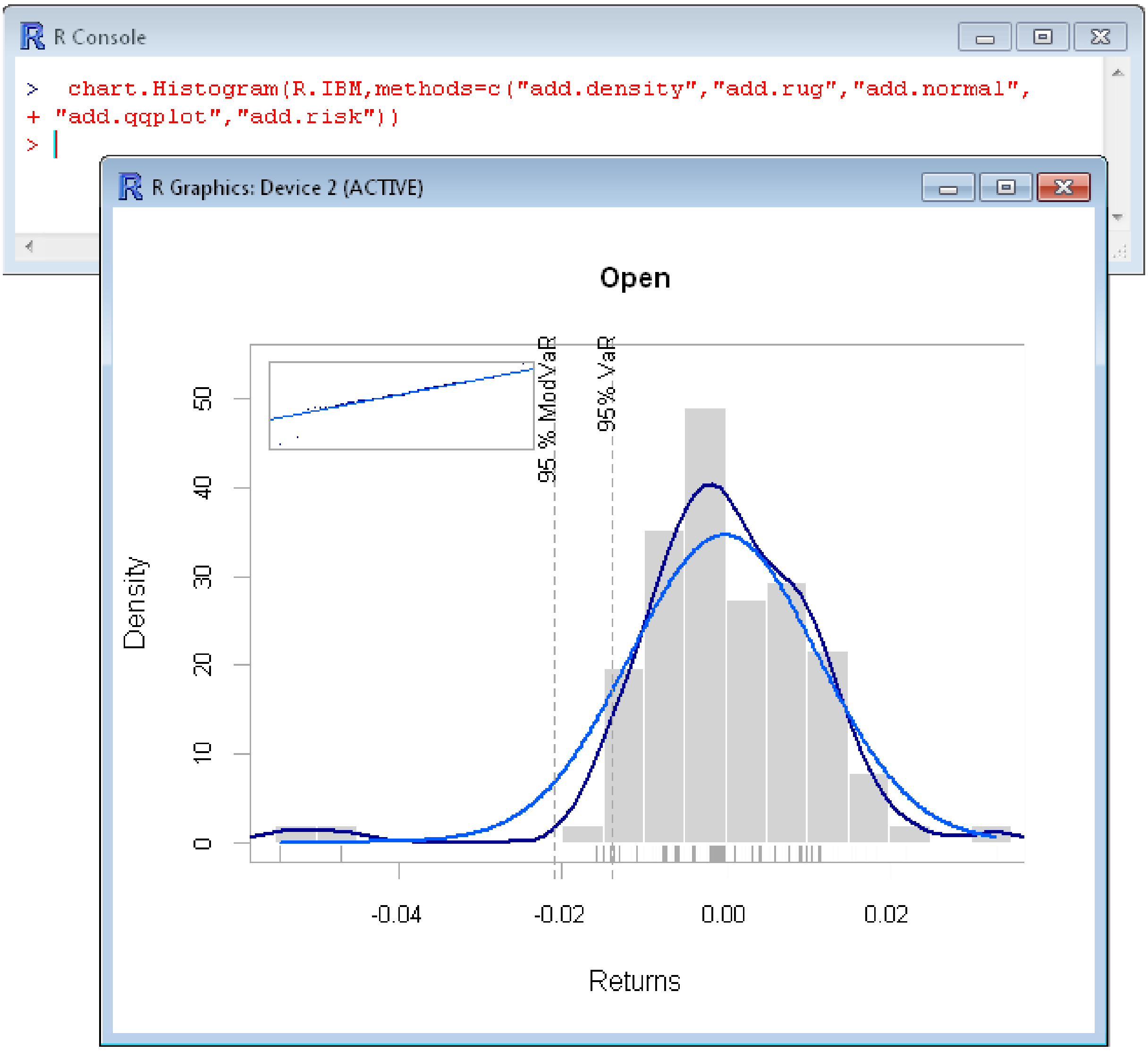


Figure 334 Graph de la VaR avec Q-Q plot, histogramme, densité et jitter

Notez le petit Q-Q plot dans le coin supérieur gauche…

MEDAF ou CAPM

Les données utilisées ici seront les mêmes que celles utilisées dans le cours théorique lors de notre étude du modèle de diversification efficient de Sharpe.

Nous allons voir comment retrouver les paramètres du MEDAF comme beta, la prime de risque… que nous avions calculé à la main.

D'abord calculons le rendement moyen de chacun des titres avec des commandes classiques de R:

```
R Console
> require(PerformanceAnalytics)
> # les données converties en Time series (series temporelles)
> ra <- ts(c(-0.15,0.05,-0.43,0.79,0.32))
> rb <- ts(c(0.29,0.18,0.24,0.25,0.17))
> rc <- ts(c(0.38,0.63,0.46,0.36,-0.57))
> rm <- ts(c(0.45,0.61,0.54,0.29,0.12))
> data <- data.frame(ra,rb,rc,rm)
> data
     ra   rb    rc   rm
1 -0.15 0.29  0.38 0.45
2  0.05 0.18  0.63 0.61
3 -0.43 0.24  0.46 0.54
4  0.79 0.25  0.36 0.29
5  0.32 0.17 -0.57 0.12
> #l'espérance de chaque titre
> (mu <-lapply(data,mean))
$ra
[1] 0.116

$rb
[1] 0.226

$rc
[1] 0.252

$rm
[1] 0.402

> #l'espérance d'un titre
> mu$rb
[1] 0.226
```

Nous faisons ensuite d'autres petits calculs mais déjà pour certains qui sont plus pertinents et qui utilisent les commandes **CAPM.beta( )** ou **BetaCoVariance( )** du package **PerformanceAnalytics**:

```
R Console
> #Obtenir la variance ou le risque des titre
> (sigma <-lapply(data,var))
$ra
[1] 0.21728

$rb
[1] 0.00253

$rc
[1] 0.22247

$rm
[1] 0.03917

> sigma$rb
[1] 0.00253
> #Obtenir le beta d'un titre
> (beta <- cov(ra,rm)/var(rm))
[1] -1.498979
> # Obtenir le beta d'un titre avec une fonction déjà existente
> CAPM.beta(ra, rm,Rf=0)
[1] -1.498979
> # une autre fonction
> BetaCoVariance(ra,rm)
[1] -1.498979
> #Obtenir le beta de plusieurs titres
> (beta<- lapply(data,function(x)CAPM.beta(x,rm)))
$ra
[1] -1.498979

$rb
[1] 0.054506

$rc
[1] 2.1265
```

Donc jusque là les résultats sont parfaitement conformes à ce que nous avions calculé à la main dans le cours théorique.

Maintenant calculons la prime de risque avec un actif sans risque de 5% (CML signifie pour rappel: droite de marché ou "Capital Market Line" en anglai) avec la commande **CAPM.RiskPremium( )** du package **PerformanceAnalytics** ainsi que la rentabilité espérée avec la commande **CAPM.CML( )** :

```
> #Déterminer la prime de risque de marché
> (prime <- CAPM.RiskPremium(rm, Rf = 0.05))
                         [,1]
Risk Premium (Rf=5%) 0.352
> # la rentabilité espérée d'un titre sur la droite de marché
> CAPM.CML(ra,rm,Rf=0.05)
[1] 0.2563117
> #Pour plusieurs titres
> (cml <- lapply(data[,1:3,drop =F],function(x)CAPM.CML(x,rm,Rf=0.05)))
$ra
[1] 0.2563117

$rb
[1] 0.4519521

$rc
[1] 0.4981944
```

Comme nous l'avons démontré dans le cours c'est de l'algèbre élémentaire et donc nous n'avions pas fait d'applications numériques. Faisons donc cela ici même exceptionnellement. Rappelons que:

$$\lambda = E(R_m) - R_f$$

Nous avons alors dans le cas présent pour le premier exemple du **CAPM.RiskPremium( )** :

$$\lambda = 0.402 - 0.05 \cong 0.352$$

et ainsi de suite pour les autres...

Ensuite comme la rentabilité espérée était aussi triviale dans le cours théorique, nous n'avions pas fait d'application numérique. Donc allons-y:

$$E(R_p) = \frac{E(R_m) - R_f}{\sigma(R_m)} \sigma(R_p) + R_f$$

Nous avons alors dans le cas présent pour le premier exemple:

$$E(R_A) = \frac{0.402 - 0.05}{\sqrt{0.03917}} \sqrt{0.2178} + 0.05 = 0.87904051$$

et donc nous n'obtenons pas la même chose. Cela semblerait à priori être une erreur du package. Effectivement dans le code source, nous avons (version 1.1.0):

```
CAPM.CML <- function (Ra, Rb, Rf = 0)
{ #@author Brian G. Peterson

    Ra = checkData(Ra)
    Rb = checkData(Rb)
    if(!is.null(dim(Rf)))
        Rf = checkData(Rf)

    Ra.ncols = NCOL(Ra)
    Rb.ncols = NCOL(Rb)

    pairs = expand.grid(1:Ra.ncols, 1:Rb.ncols)

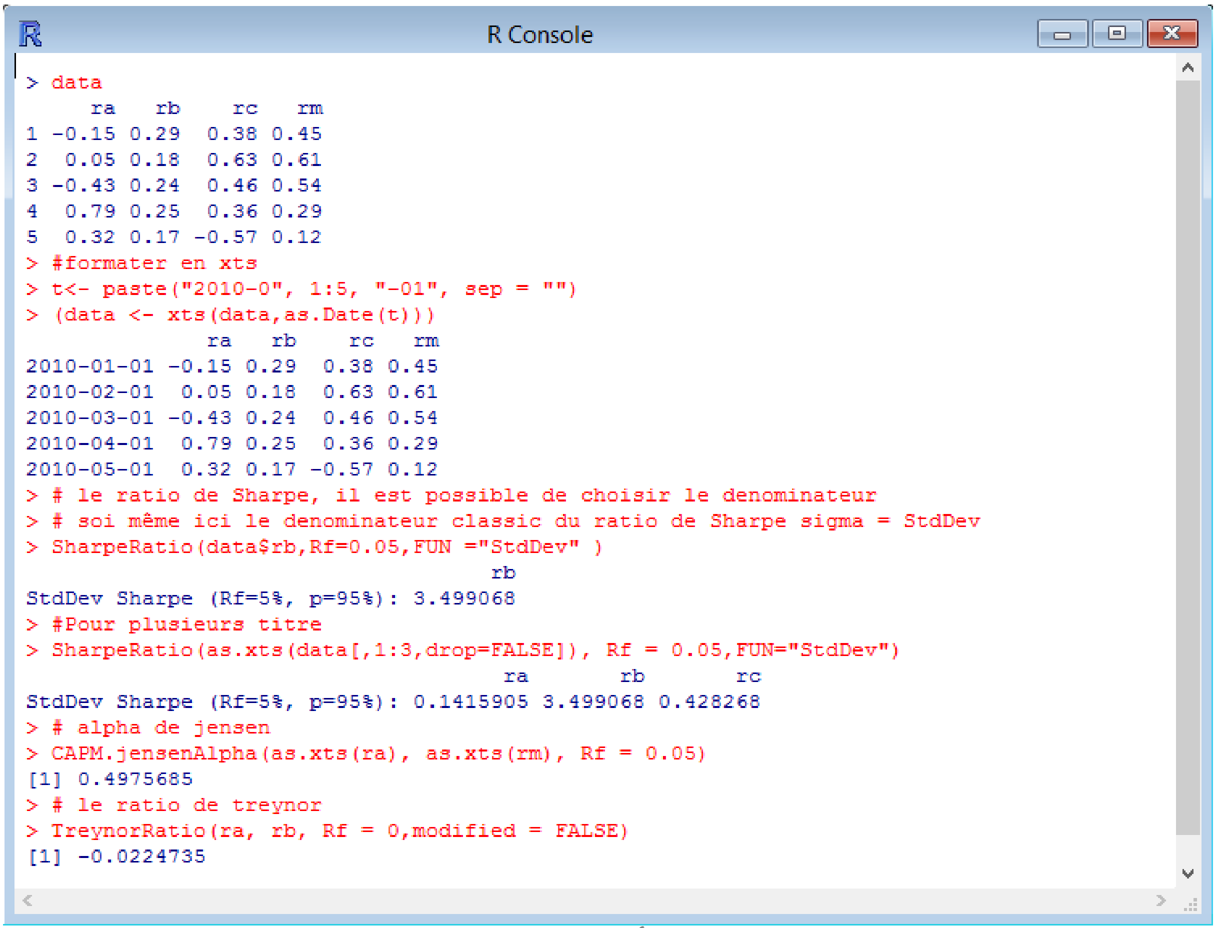
    cml <-function (Ra, Rb, Rf)
    {
        CML = mean(Rf, na.rm=TRUE) + CAPM.CML.slope(Rb, Rf)*mean(Ra, na.rm=TRUE)
        return(CML)
    }

    result = apply(pairs, 1, FUN = function(n, Ra, Rb, Rf) cml(Ra[,n[1]], Rb[,n[2]], Rf), Ra = Ra, Rb = Rb, Rf = Rf)

    if(length(result) ==1)
        return(result)
    else {
        dim(result) = c(Ra.ncols, Rb.ncols)
        colnames(result) = paste("Capital Market Line:", colnames(Rb))
        rownames(result) = colnames(Ra)
        return(t(result))
    }
}
```

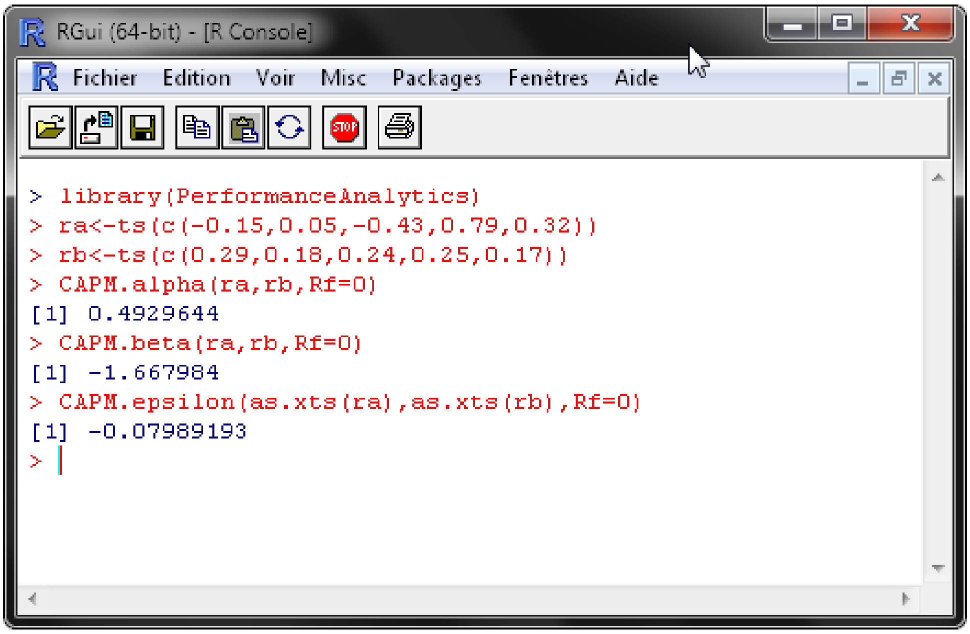
Donc il y a bien une erreur car à ma connaissance on ne doit pas multiplier par la moyenne mais par l'écart-type!

Nous pouvons avoir des indicateurs de ratios de performances avec le ratio de Sharpe utilisant la commande **SharpeRatio( )** , l'alpha de Jensen avec la commande **CAPM.jensenAlpha( )** et enfin le ratio de Treynor avec la commande **TreynorRatio( )** :



Bref rien d'extraordinaire ici... la finance est avec la statistique le domaine expert dans la création à l'infini d'indicateurs empiriques...

Il existe aussi des fonctions pour calculer le résidu epsilon de la droite de régression et alpha qui est l'ordonnée à l'origine en utilisant les commandes **CAPM.alpha( )** et **CAPM.epsilon( )** toujours du même package:



Là aussi rien d'extraordinaire. Le paramètre alpha étant simplement l'ordonnée à l'origine de la fonction de régression linéaire:

$$R_a = f\left(R_b\right) = \beta R_b + \alpha + \varepsilon$$

Nous n'avions pas calculé dans le cours théorique l'epsilon pour la simple raison que c'est une abstraction... L'auteur du package a fait le choix empirique et tout à fait discutable de déterminer celui sur la base du calcul suivant:

$$\varepsilon = \left[\left(\prod_{i=1}^{n}\left(1+R_a\right)\right)-1\right]-\beta\left[\left(\prod_{i=1}^{n}\left(1+R_b\right)\right)-1\right]-\alpha$$

Donc il s'agit de calculer la différence entre les rendements composé des deux portefeuilles. Choix tout à fait discutable...

Risque total, systématique et spécifique

Maintenant revenons à la relation démontrée dans le cours théorique:

$$\underbrace{\sigma_i}_{\text{Risque total}} = \sqrt{\underbrace{\beta_i^2\sigma_m^2}_{(\text{Risque systématique})^2} + \underbrace{\sigma_\varepsilon^2}_{(\text{Risque spcécifique})^2}}$$

Pour laquelle nous n'avons pas fait d'application numérique pratique (état donné les approximations implicites pour écrire cette relation qui par voie de conséquence est alors un peu trop scolaire).

Observons que le package **PerformanceAnalytics** a les fonctions **SystematicRisk( )** , **SpecificRisk( )** et **TotalRisk( )** pour cela (comme il existe plusieurs manières dans la pratique de calculer le risque spécifique... nous ne souhaitons pas ici valider comme nous le faisons d'habitude la sortie du logiciel).

```
R Console
> ra<-ts(c(-0.15,0.05,-0.43,0.79,0.32))
> rm<-ts(c(0.45,0.61,0.54,0.29,0.12))
> data<-data.frame(ra,rm)
> #on formate en xts car une des fonctions n'accepte que cela
> t<-paste("2010-0",1:5,"-01",sep="")
> data<-xts(data,as.Date(t))
> data
              ra   rm
2010-01-01 -0.15 0.45
2010-02-01  0.05 0.61
2010-03-01 -0.43 0.54
2010-04-01  0.79 0.29
2010-05-01  0.32 0.12
>
> #ce serait à vérifier...
> SyR<-SystematicRisk(data[,1,drop=F],data[,2,drop=F],Rf=0)
> SyR
[1] -1.027691826
> SpR<-SpecificRisk(data[,1,drop=F],data[,2,drop=F],Rf=0)
> SpR
[1] 1.113987257
> TR<-TotalRisk(data[,1,drop=F],data[,2,drop=F],Rf=0)
> TR
[1] 1.515624656
>
> #check
> sqrt(SyR^2+SpR^2)
[1] 1.515624656
>
```

Modèle de Fama-French (CAPM multi-facteurs) et régression croisée de Fama-MacBeth

Bon je ne veux pas débattre et m'étendre sur ce modèle empirique mais juste présenter brièvement sa mise en application car:

1. Il est censé faire partie de la culture générale du praticien

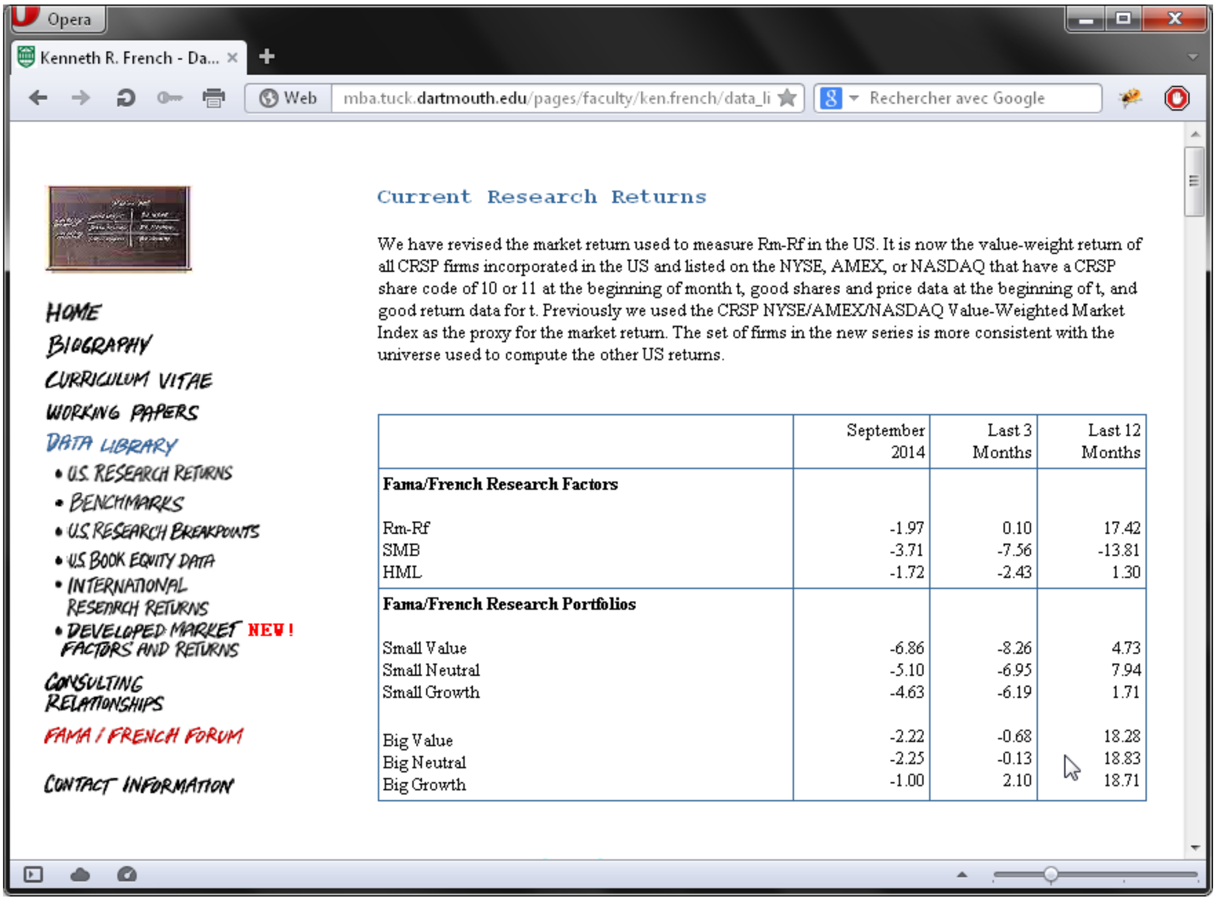
2. Il m'est régulièrement demandé dans les formations

Bon étant donné ma passion dévorante ([ironie off] pour ce modèle nous allons nous en tenir au minimum syndical.

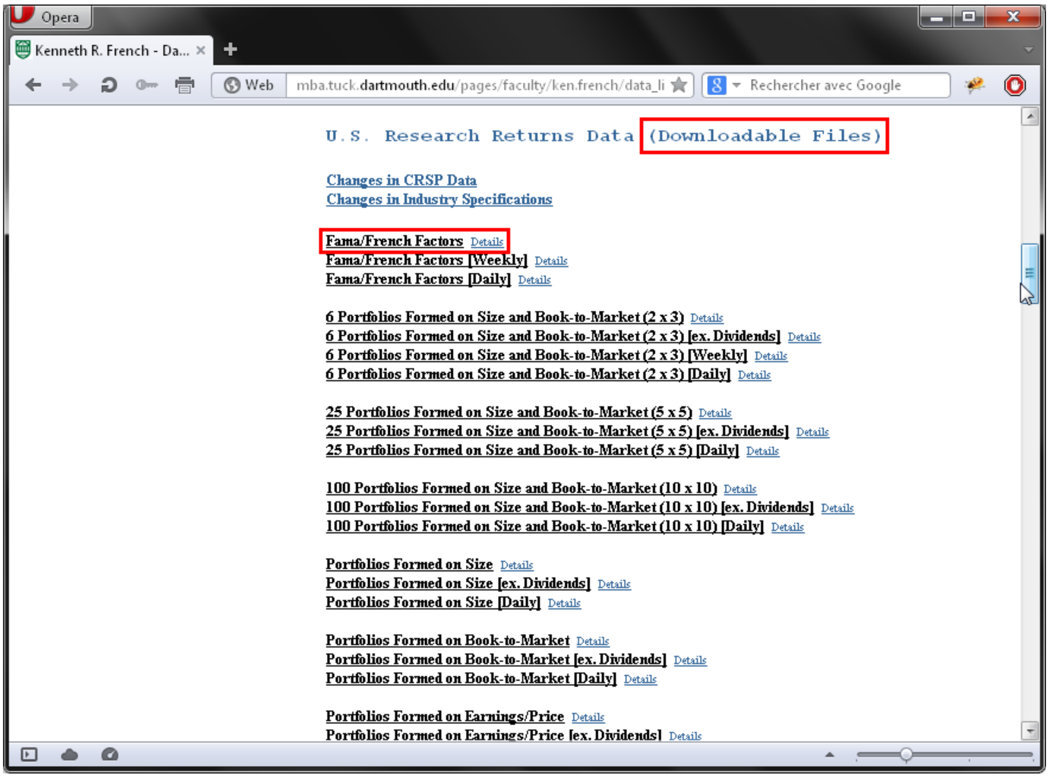
D'abord, nous allons sur le site d'un des auteurs du modèle

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

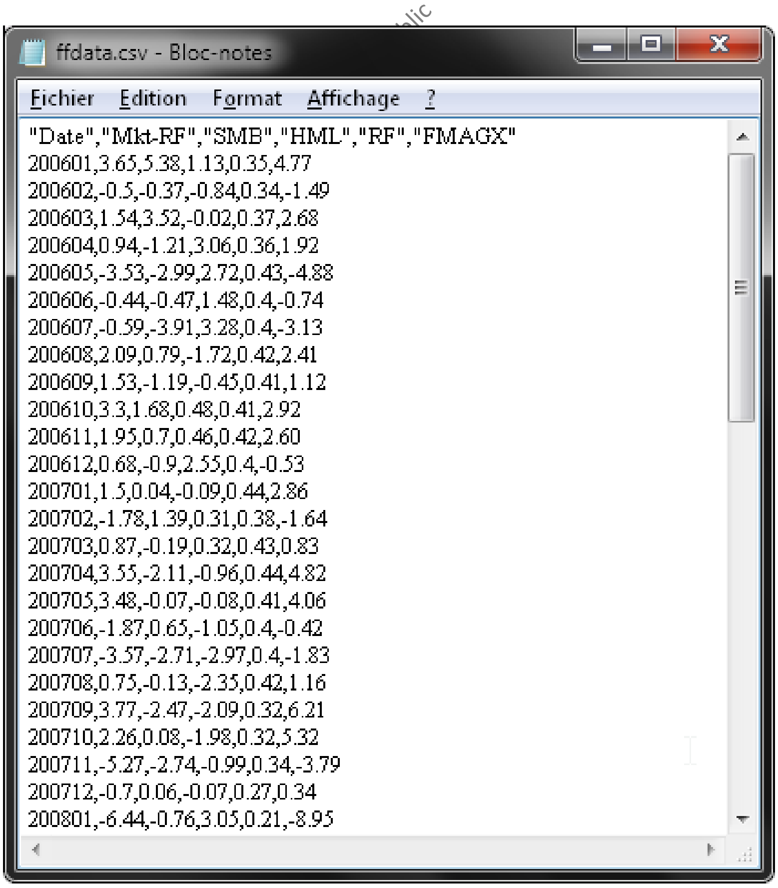
Pour télécharger les valeurs coefficients SMB (Small Minus Big: Facteur de taille) et HML (High Minus Low: Facteur de prix) pour les U.S. (dans le cas de l'exemple qui va suivre):



Plus bas dans la page se trouve le lien qui nous intéresse:



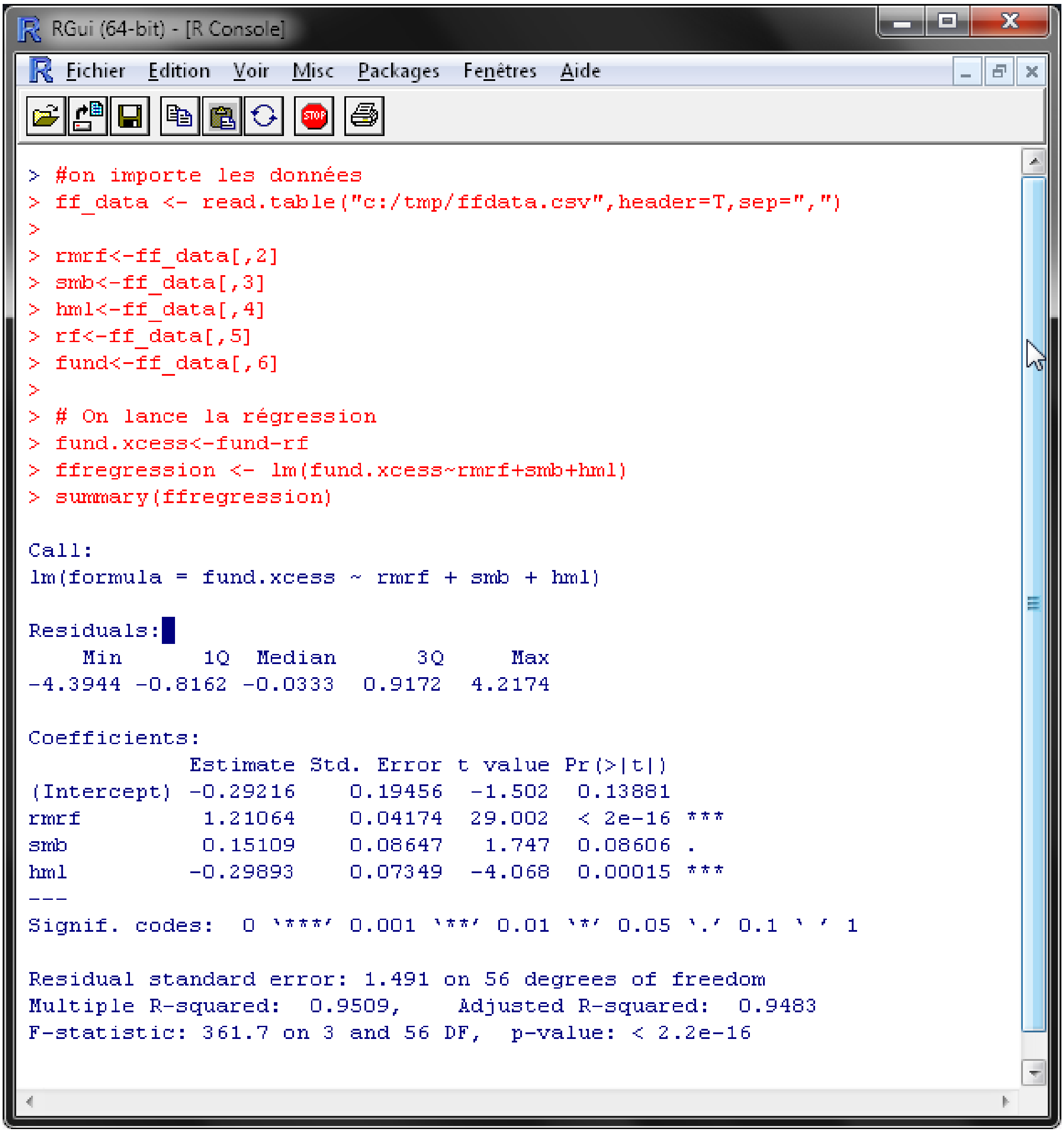
S'y trouve alors un fichier avec les facteurs qui nous intéressent qui une fois un peu remanié et auquel on rajoute dans la dernière colonne notre investissement d'intérêt (il s'agit d'un Fond comme un autre):


ffdata.csv - Bloc-notes

Fichier Edition Format Affichage ?

```
"Date","Mkt-RF","SMB","HML","RF","FMAGX"
200601,3.65,5.38,1.13,0.35,4.77
200602,-0.5,-0.37,-0.84,0.34,-1.49
200603,1.54,3.52,-0.02,0.37,2.68
200604,0.94,-1.21,3.06,0.36,1.92
200605,-3.53,-2.99,2.72,0.43,-4.88
200606,-0.44,-0.47,1.48,0.4,-0.74
200607,-0.59,-3.91,3.28,0.4,-3.13
200608,2.09,0.79,-1.72,0.42,2.41
200609,1.53,-1.19,-0.45,0.41,1.12
200610,3.3,1.68,0.48,0.41,2.92
200611,1.95,0.7,0.46,0.42,2.60
200612,0.68,-0.9,2.55,0.4,-0.53
200701,1.5,0.04,-0.09,0.44,2.86
200702,-1.78,1.39,0.31,0.38,-1.64
200703,0.87,-0.19,0.32,0.43,0.83
200704,3.55,-2.11,-0.96,0.44,4.82
200705,3.48,-0.07,-0.08,0.41,4.06
200706,-1.87,0.65,-1.05,0.4,-0.42
200707,-3.57,-2.71,-2.97,0.4,-1.83
200708,0.75,-0.13,-2.35,0.42,1.16
200709,3.77,-2.47,-2.09,0.32,6.21
200710,2.26,0.08,-1.98,0.32,5.32
200711,-5.27,-2.74,-0.99,0.34,-3.79
200712,-0.7,0.06,-0.07,0.27,0.34
200801,-6.44,-0.76,3.05,0.21,-8.95
```

Et maintenant vient la partie R pour déterminer les coefficient de la régression croisée de Fama-MacBeth:



Après libre à vous d'utiliser le modèle de la même manière que pour le CAPM (voir page 2642). Bref la même chose peut être fait avec n'importe quel tableur d'entrée de gamme.

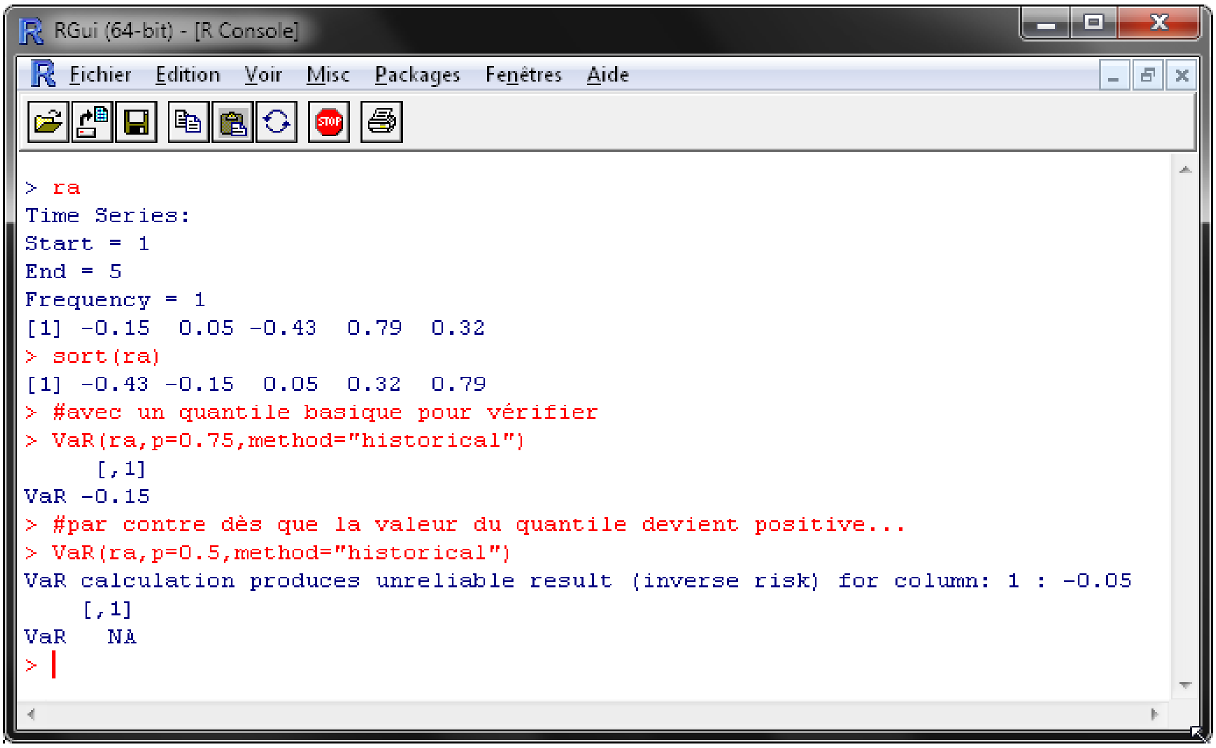
Exemples 403.: Values at Risk

R 3.0.2

Value at Risk Historique et Gaussienne

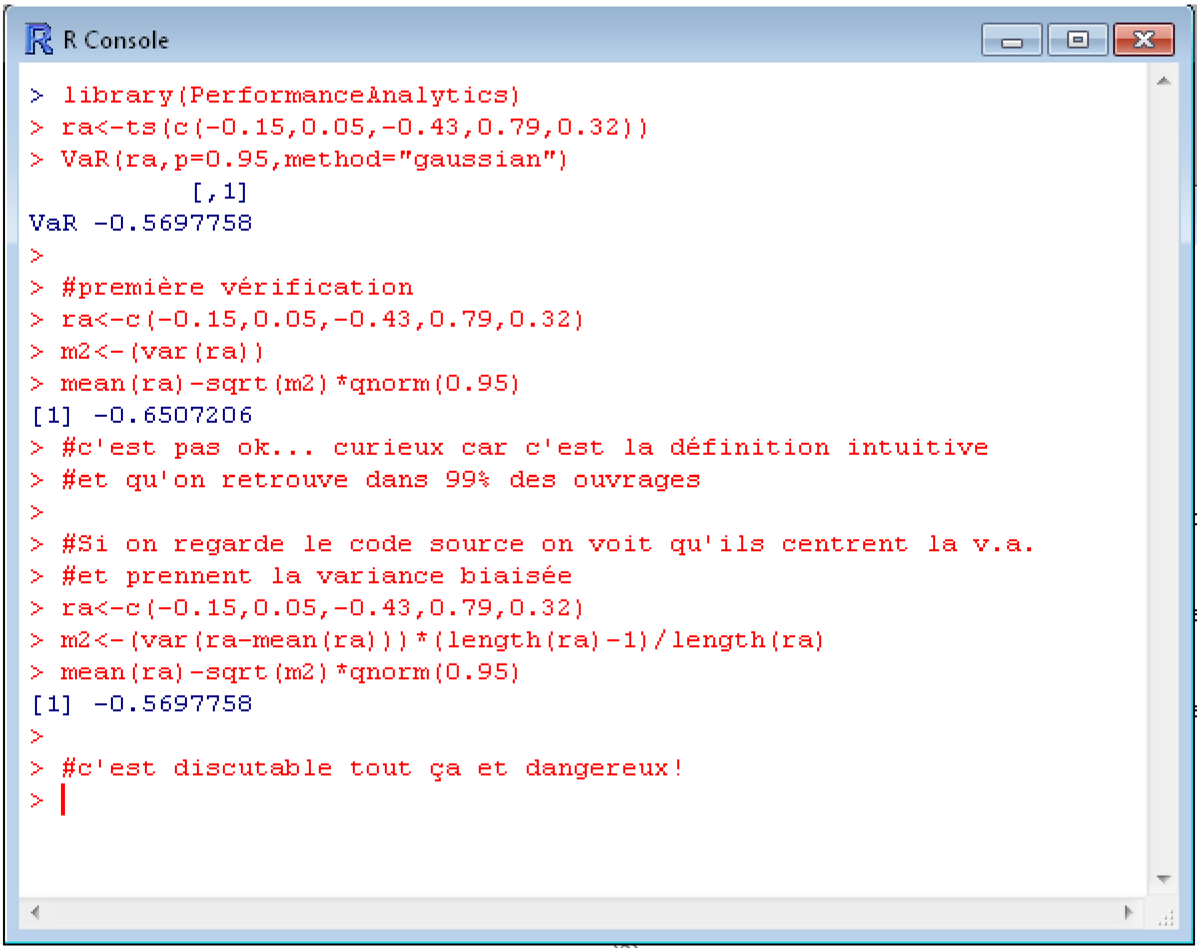
Le but ici va être de vérifier si les calculs coïncident avec ce que nous avons vu dans le cours théorique lors de notre étude des nombreux modèles de VaR.

Commençons par la VaR historique en utilisant la commande **VaR( )** :



Donc a priori rien à signaler de particulier. C'est logique et c'est trivial.

Voyons donc maintenant la VaR gaussienne, toujours avec la commande **VaR( )** :



Faisons maintenant un plot (malheureusement l'abscisse est imposée dans le code source du package mais on comme on a accès au code source...) en utilisant la commande **chart.VaRSensitivity( )** et vérifions la correspondance avec les calculs faits précédemment (raison pour laquelle j'ai rajouté une ligne verticale):

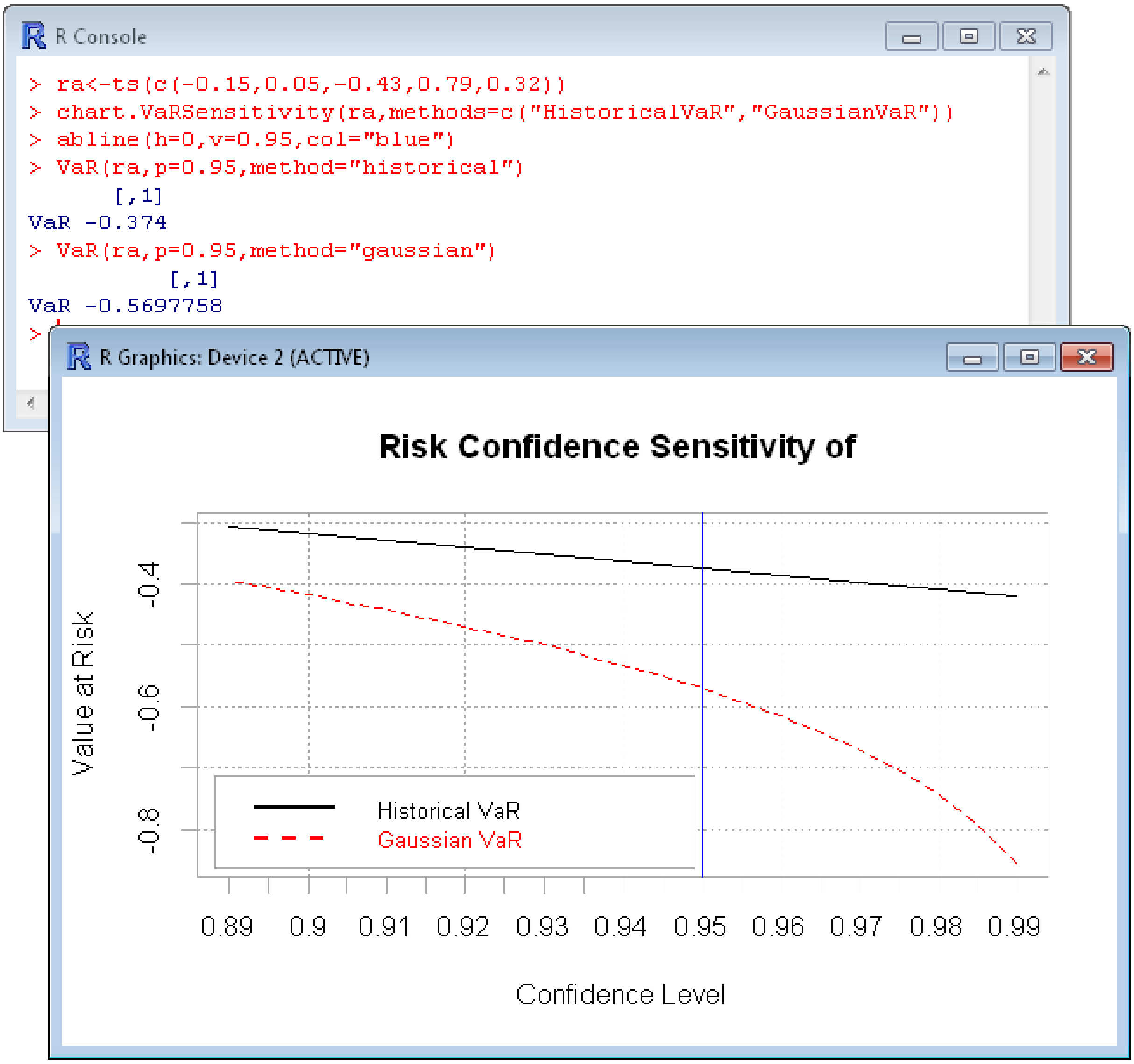


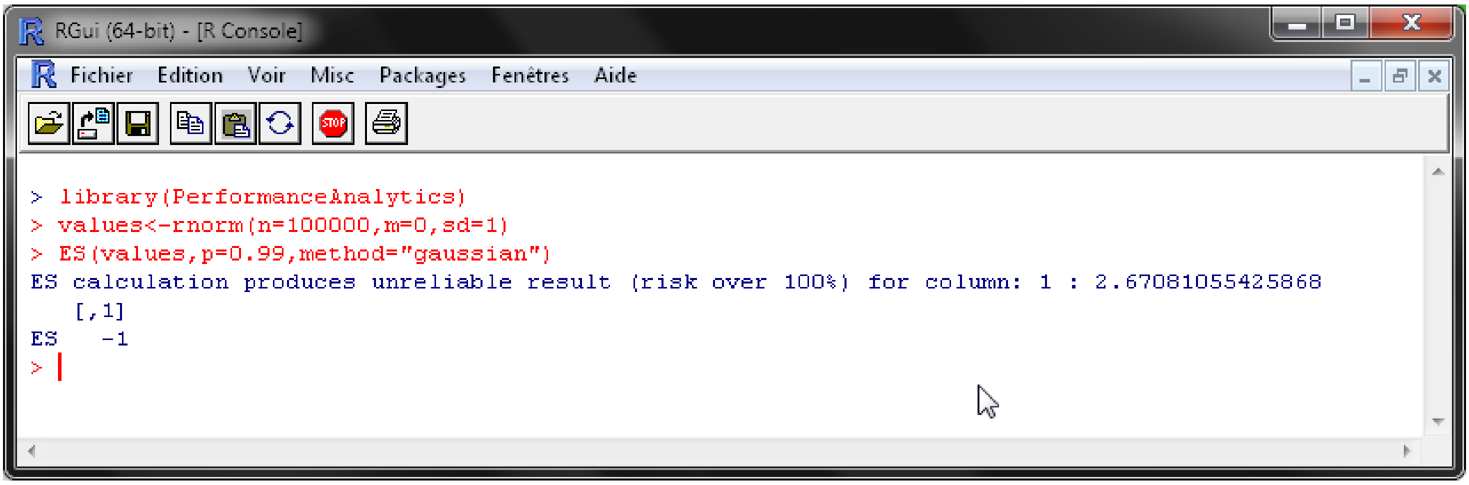
Figure 335 Graph de l'intervalle de confiance de la VaR Gaussienne et Historique

Donc nous voyons que tout est conforme!

Conditional VaR (Expected Shortfall)

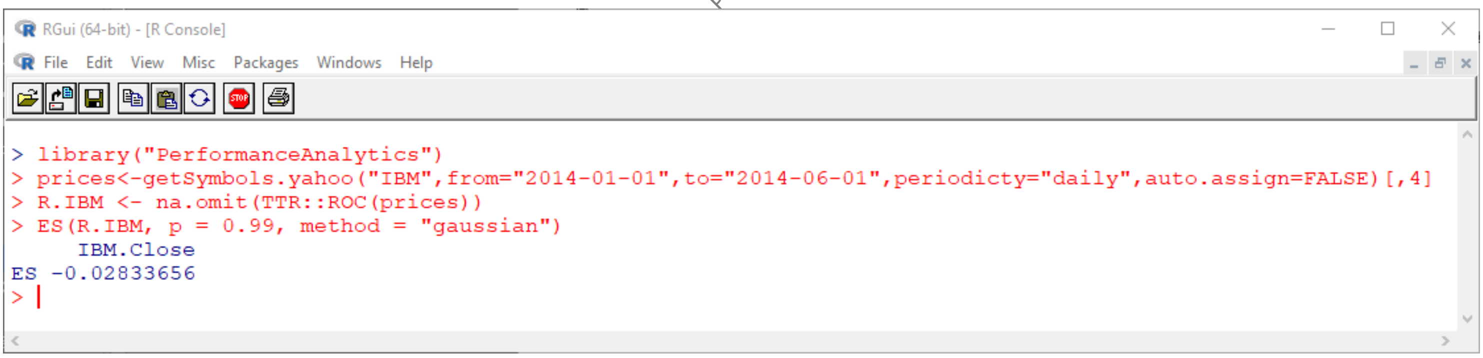
Nous avons démontré dans le cours théorique comment calculer la conditional VaR (Expected Shortfall) dans le cas Gaussien. Nous allons vérifier ici que nous retrouvons le même résultat que celui calculé à la main.

Pour ce faire, nous allons utiliser la commande **ES( )** avec des données provenant aussi d'une loi Normale:



Donc excepté les messages d'erreurs le résultat est bien le bon!

Avec des données un peu plus réalistes, cela donne:

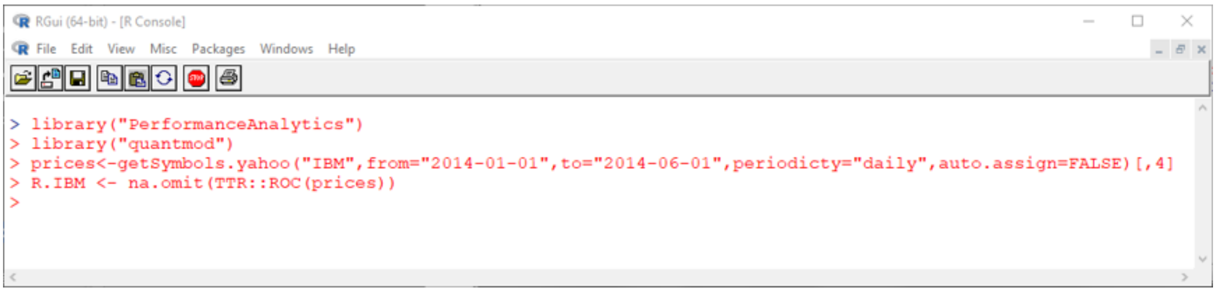


ToDo: Trouver la raison de ce message d'erreur!

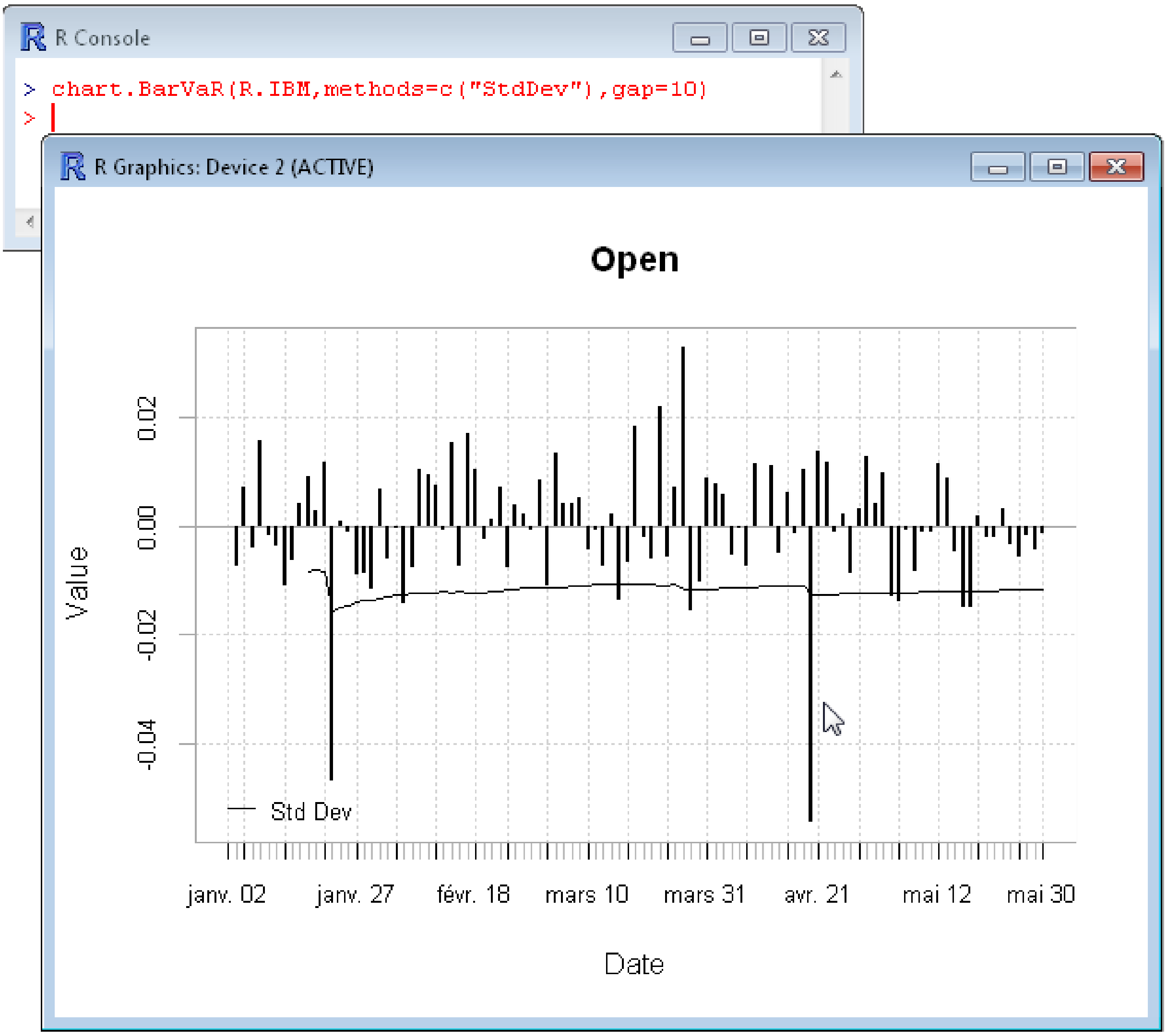
Plotter Écart-type ou la VaR historique/Gaussienne/Conditionnelle

Sans vérifier si les données correspondent aux calculs effectués à la main, observons que nous pouvons avec le package **PerformanceAnalytics** obtenir simplement les 4 indicateurs de risque cités dans le titre.

D'abord nous importons les données:



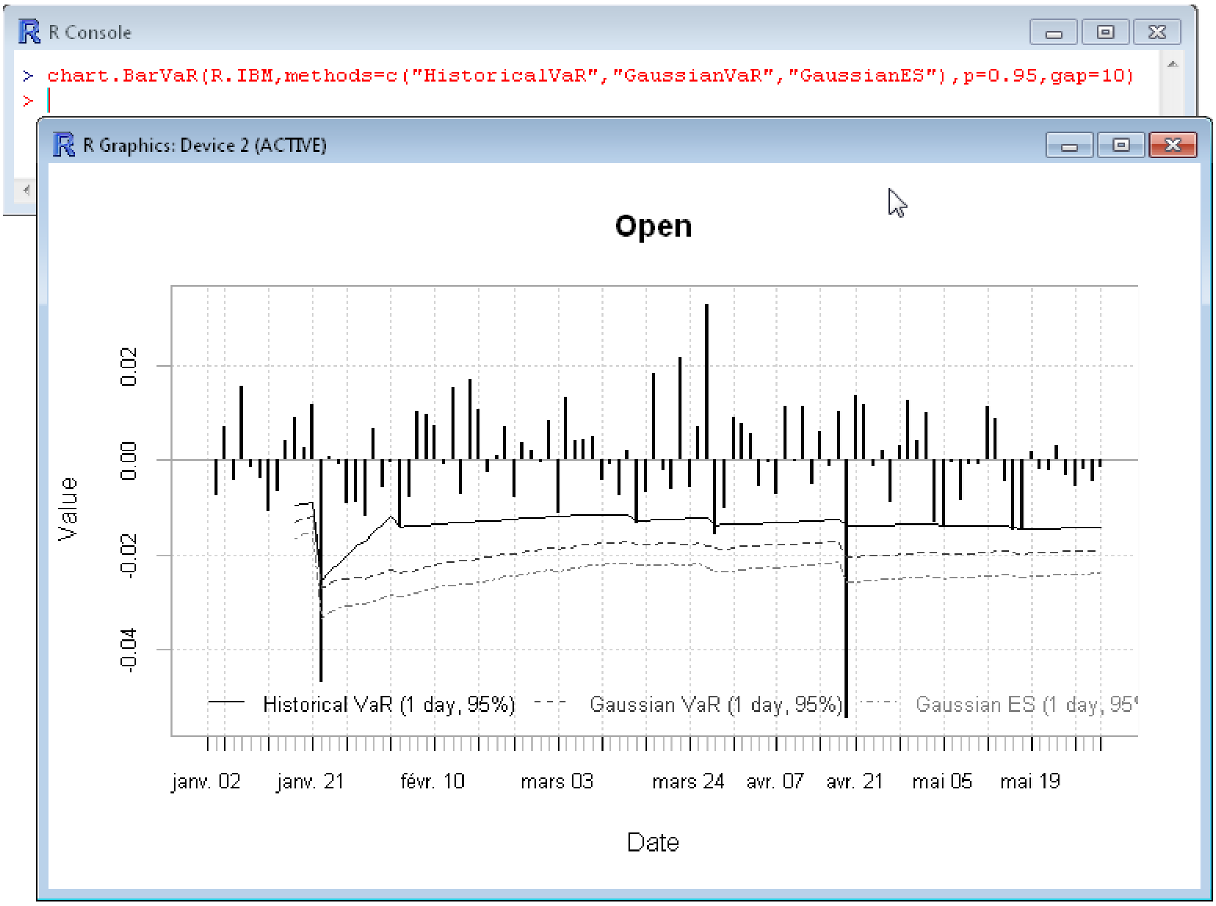
Commençons par l'indicateur un peu hors catégorie (car indépendant de toute probabilité dans ce package) qui est l'écart-type seul avec un départ du calcul depuis le 10ème jour avec la fonction **chart.BarVar( )**:



Avec un départ du calcul à 20 jours:



et ainsi de suite... Voyons maintenant la même chose avec les 3 VaR à un niveau de 95% que nous avons vues dans le cours théorique avec aussi un départ des calculs à 10 et 20 jours:



Avec un départ du calcul à 20 jours:

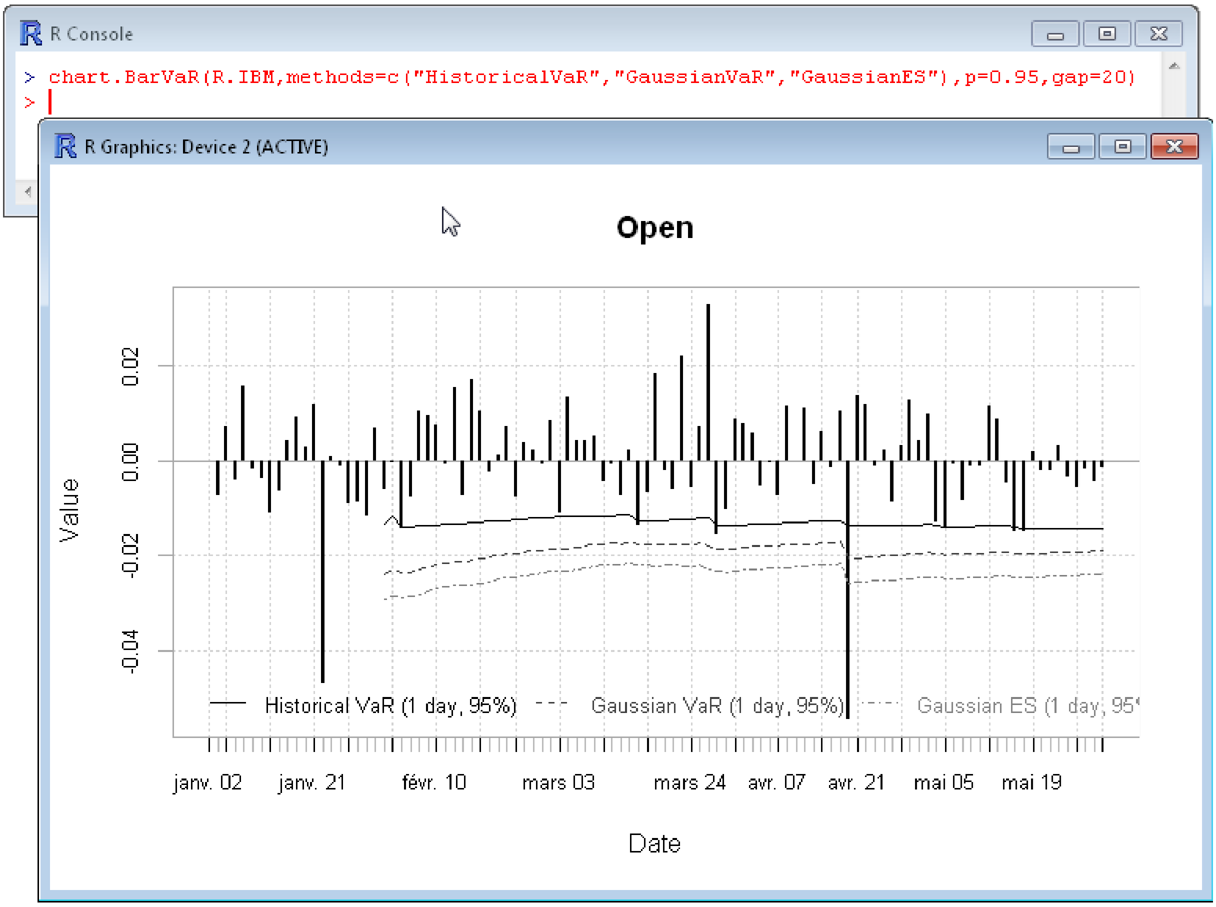
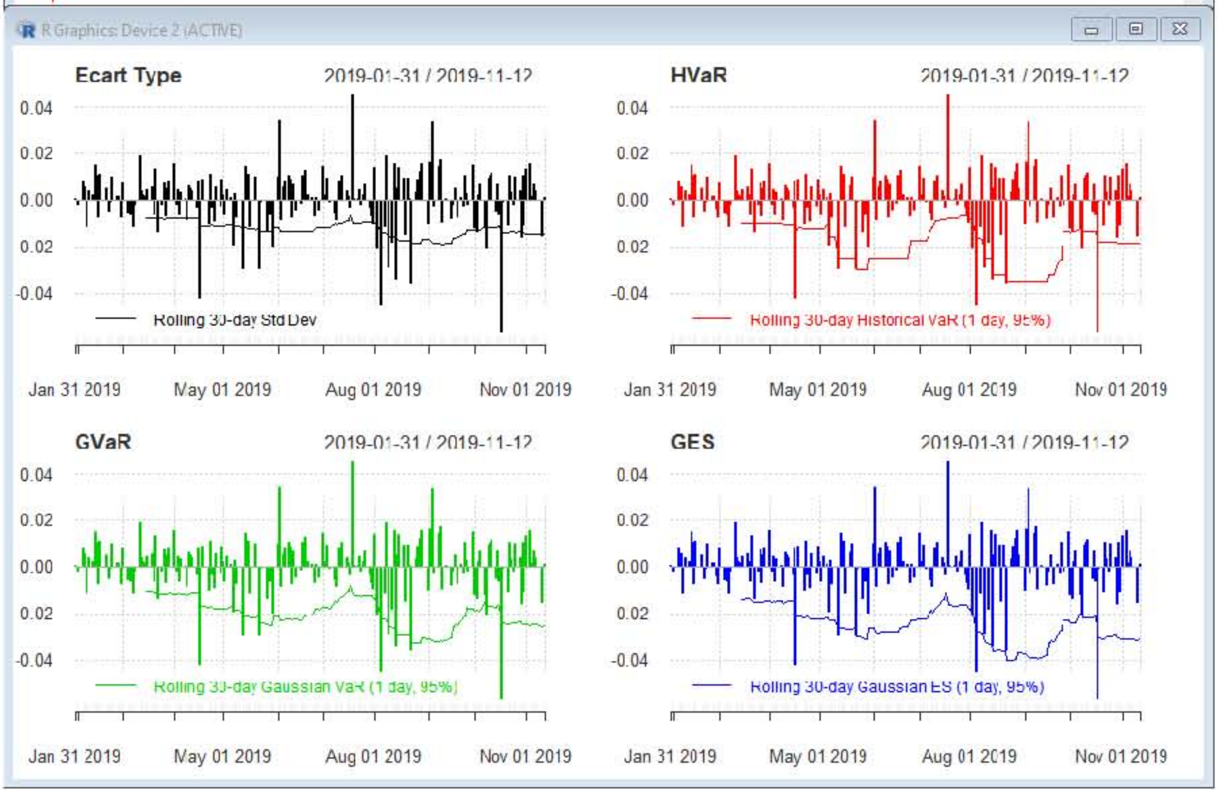


Figure 336 Graph VaR Historique, Gaussienne et Conditionnelle

Faisons maintenant une compilation et un cas plus esthétique avec un départ du calcul au début et un fenêtre (rolling) de 30 jours:

```
> library("PerformanceAnalytics")
> library("tseries")
>
> prices<-get.hist.quote("IBM",start="2009-01-01",quote="AdjClose",compression="d")
time series starts 2009-01-02
time series ends   2019-11-12
> ReturnIBM<-Return.calculate(prices,method="discrete")
>
> #on prend que les 200 dernières valeurs
> ROCIBM<-na.omit(TTR::ROC(tail(prices,200)))
> par(mfrow=c(2,2))
> chart.BarVaR(ROCIBM,methods=c("StdDev"),gap=0,width=30,main="Ecart Type",colorset=1)
> chart.BarVaR(ROCIBM,methods=c("HistoricalVaR"),gap=0,width=30,main="HVaR",colorset=2)
> chart.BarVaR(ROCIBM,methods=c("GaussianVaR"),gap=0,width=30,main="GVaR",colorset=3)
> chart.BarVaR(ROCIBM,methods=c("GaussianES"),gap=0,width=30,main="GES",colorset=4)
>
```



Monte Carlo Value at Risk

Voyons donc un cas particulier de la VaR de Monte Carlo mais l'idée est relativement facilement généralisable:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

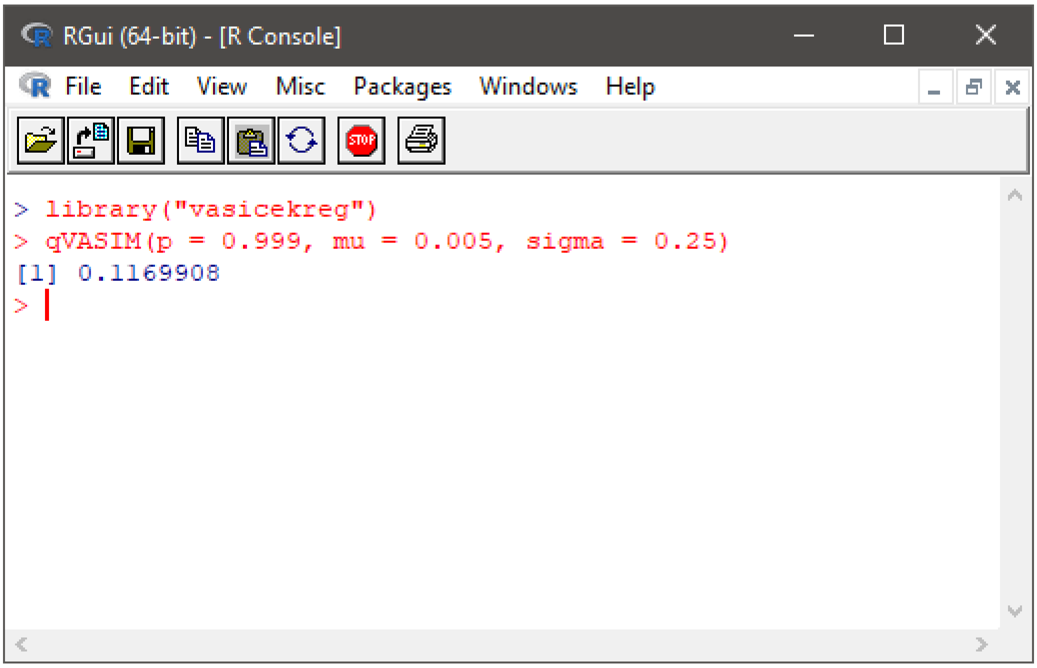
> #On génère aléatoirement deux séries de prix
> p<-matrix(c(rnorm(1000,mean=50,sd=4),rnorm(1000,mean=5,sd=0.5)),ncol=2);
> colnames(p)<-c("A1","A2")
> head(p)
           A1       A2
[1,] 47.20033 4.383123
[2,] 43.43404 4.976469
[3,] 46.74710 4.676421
[4,] 52.50418 4.665694
[5,] 46.65974 5.131612
[6,] 48.99237 4.138979
>
> poids<-c(0.4,0.6)
>
> #on calcule les moyennes et écarts-types
> (mu<-apply(p,2,mean))
       A1        A2
50.085771  4.970927
> (sigma<-cov(p))
            A1         A2
A1 15.78649116 0.06390502
A2  0.06390502 0.24301737
>
> library("MASS")
> #on génère à partir du vecteur des moyennes
> #et de la matrice de covariance une loi normale bivariée
> MC<-mvrnorm(10000,mu,sigma)
> head(MC)
           A1       A2
[1,] 50.36180 4.495778
[2,] 51.02892 5.105456
[3,] 54.50432 3.901817
[4,] 53.58052 5.015394
[5,] 51.40691 5.356199
[6,] 56.05881 5.200256
>
>
> MCportfolio <- MC%*%as.matrix(poids)
> head(MCportfolio)
         [,1]
[1,] 22.84219
[2,] 23.47484
[3,] 24.14282
[4,] 24.44144
[5,] 23.77648
[6,] 25.54368
>
> #et finalement la VaR Monte Carlo à 95%
> quantile(MCportfolio,p=0.05)
      5%
20.29566
```

Distribution de perte de portefeuille de Vasicek (distribution de Vasicek)

Nous allons simplement ici voir si nous pouvons reproduire (et faire mieux) le petit calcul à la main relatif au modèle de grand portefeuille homogène, appelé aussi "Portefeuille homogène de Vasicek" sur lequel se sont basées les Exigences de fonds propres de Bâle (RWA: Basel's capital requirements).

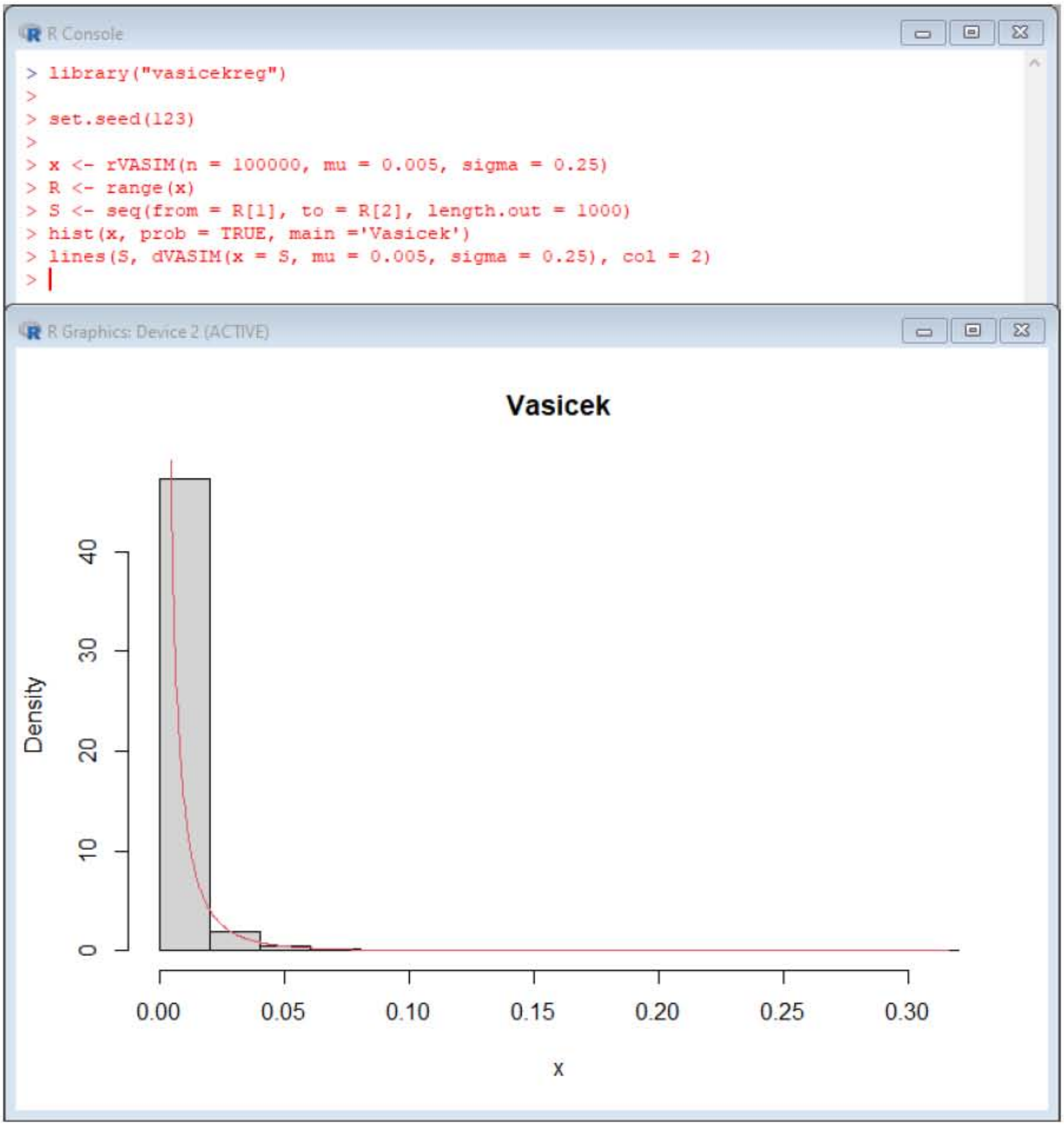
Donc refaisons le même calcul que dans le cours théorique: Quel est le 99.9ème centile associé à la probabilité de défaut de 0.05% d'un gros portefeuille homogène associée à une corrélation avec le marché de 0.25?

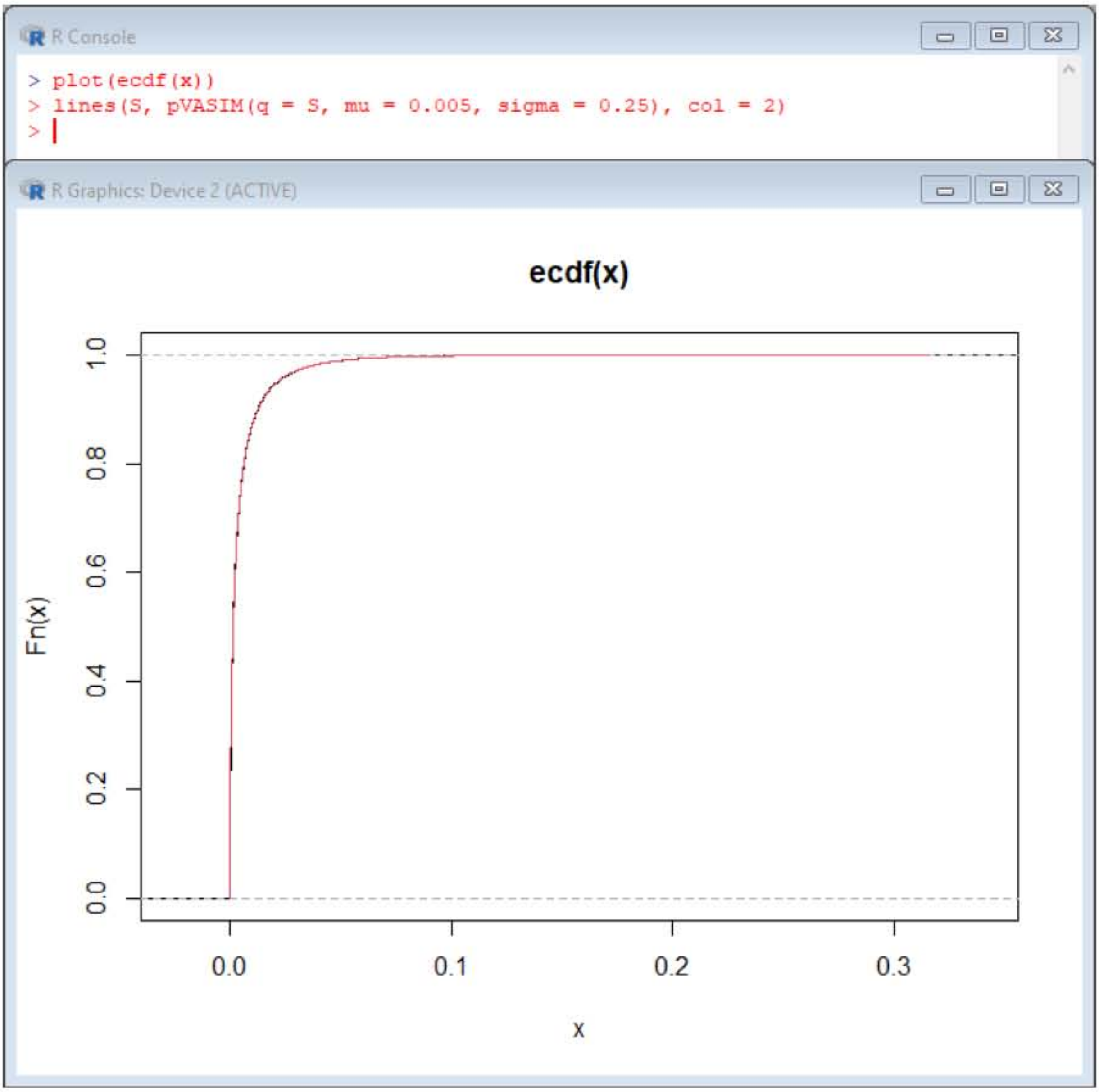
Le calcul est simplement donné par la fonction qVASIM( ) du package **vasicekreg**:

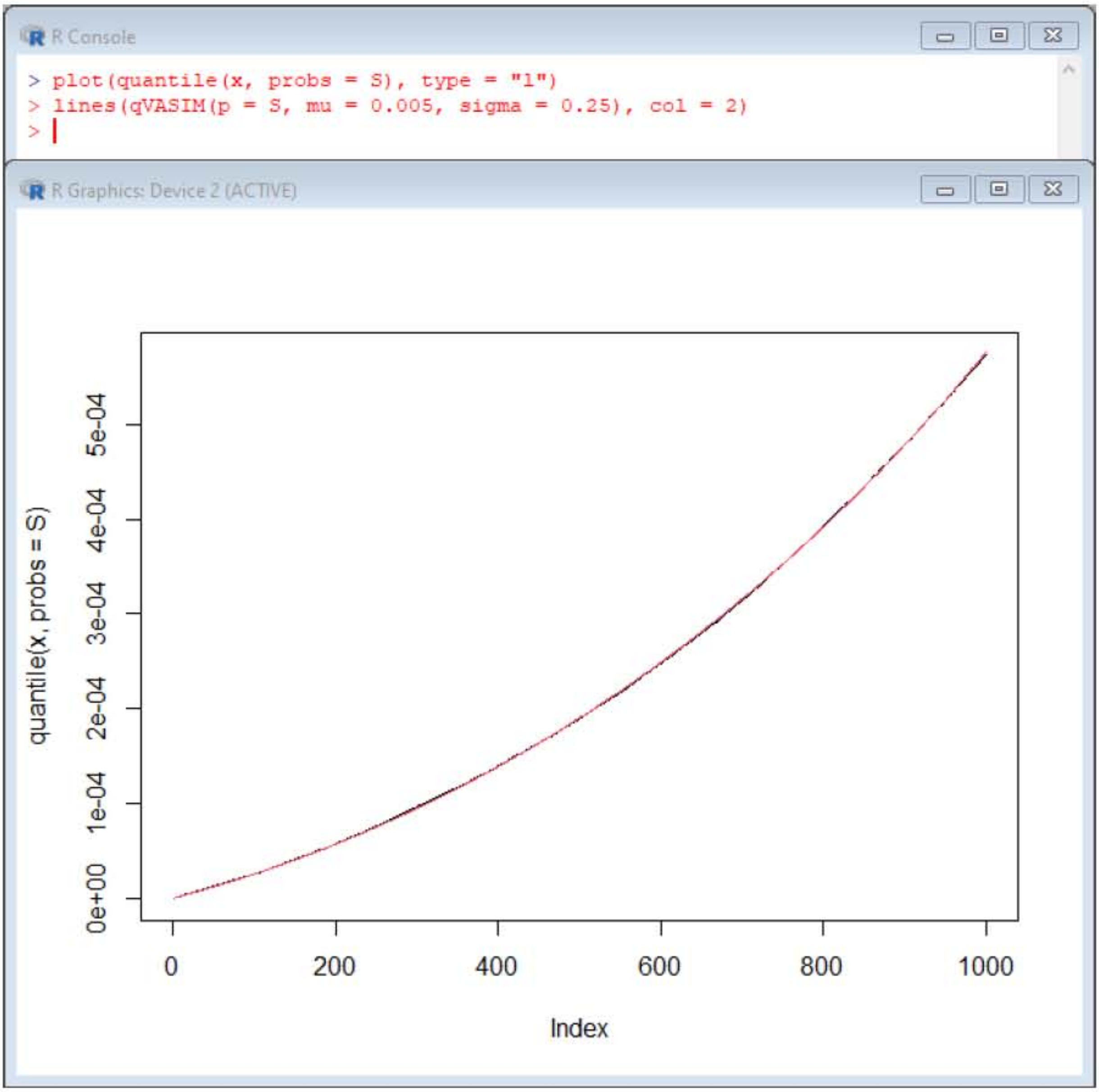


On retombe bien sur exactement la même valeur que dans le cours théorique!

Contrairementa au cours théorique on peut rapidement faire certains graphiques qui peuvent être plus ou moins intéressants pour certains:


R Console
> library("vasicekreg")
>
> set.seed(123)
>
> x <- rVASIM(n = 100000, mu = 0.005, sigma = 0.25)
> R <- range(x)
> S <- seq(from = R[1], to = R[2], length.out = 1000)
> hist(x, prob = TRUE, main ='Vasicek')
> lines(S, dVASIM(x = S, mu = 0.005, sigma = 0.25), col = 2)
R Graphics: Device 2 (ACTIVE)
Vasicek
Density
0
10
20
30
40
0.00
0.05
0.10
0.15
0.20
0.25
0.30
x


R Console
> plot(ecdf(x))
> lines(S, pVASIM(q = S, mu = 0.005, sigma = 0.25), col = 2)
R Graphics: Device 2 (ACTIVE)
ecdf(x)
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.1
0.2
0.3
x

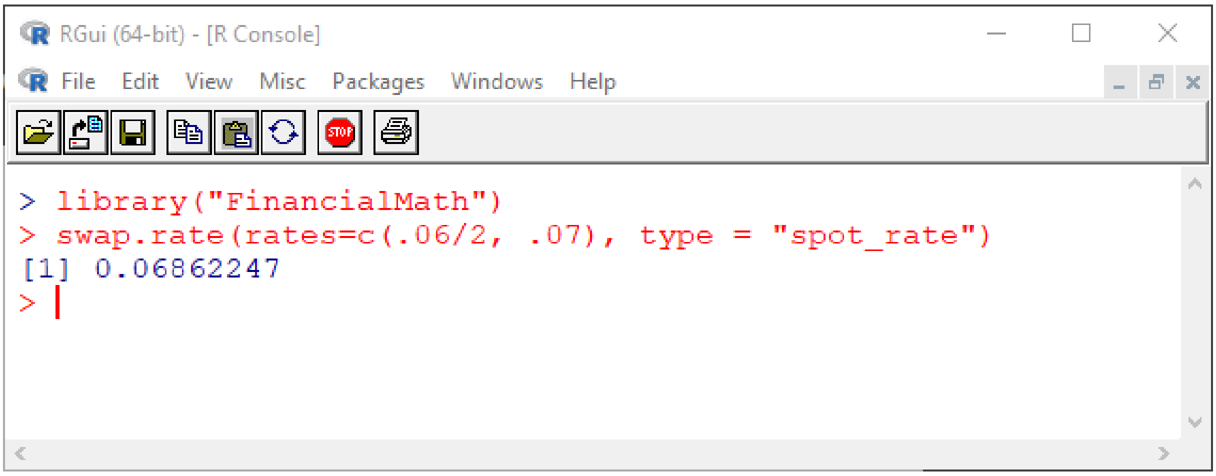

R Console
> plot(quantile(x, probs = S), type = "l")
> lines(qVASIM(p = S, mu = 0.005, sigma = 0.25), col = 2)
R Graphics: Device 2 (ACTIVE)
quantile(x, probs = S)
0e+00
1e-04
2e-04
3e-04
4e-04
5e-04
0
200
400
600
800
1000
Index

Exemple 404.: Pricing de SWAP de taux d'intérêts (IRS)

R 3.5.3

Bon nous avons mentionné dans le cours théorique que le pricing de SWAP est une vraie jungle étant donné qu'il y a plusieurs manières de calculer ces derniers et différents types de SWAP, que cela dépend de l'instrument financier, du type de méthode du calcul des taux ainsi que si on prend la partie fixe ouf flottant.

Mais voyons déjà le cas simple d'un IRS (Interest Rate Swap) à l'aide de la fonction **swap.rate( )** du package **FinancialMath**:



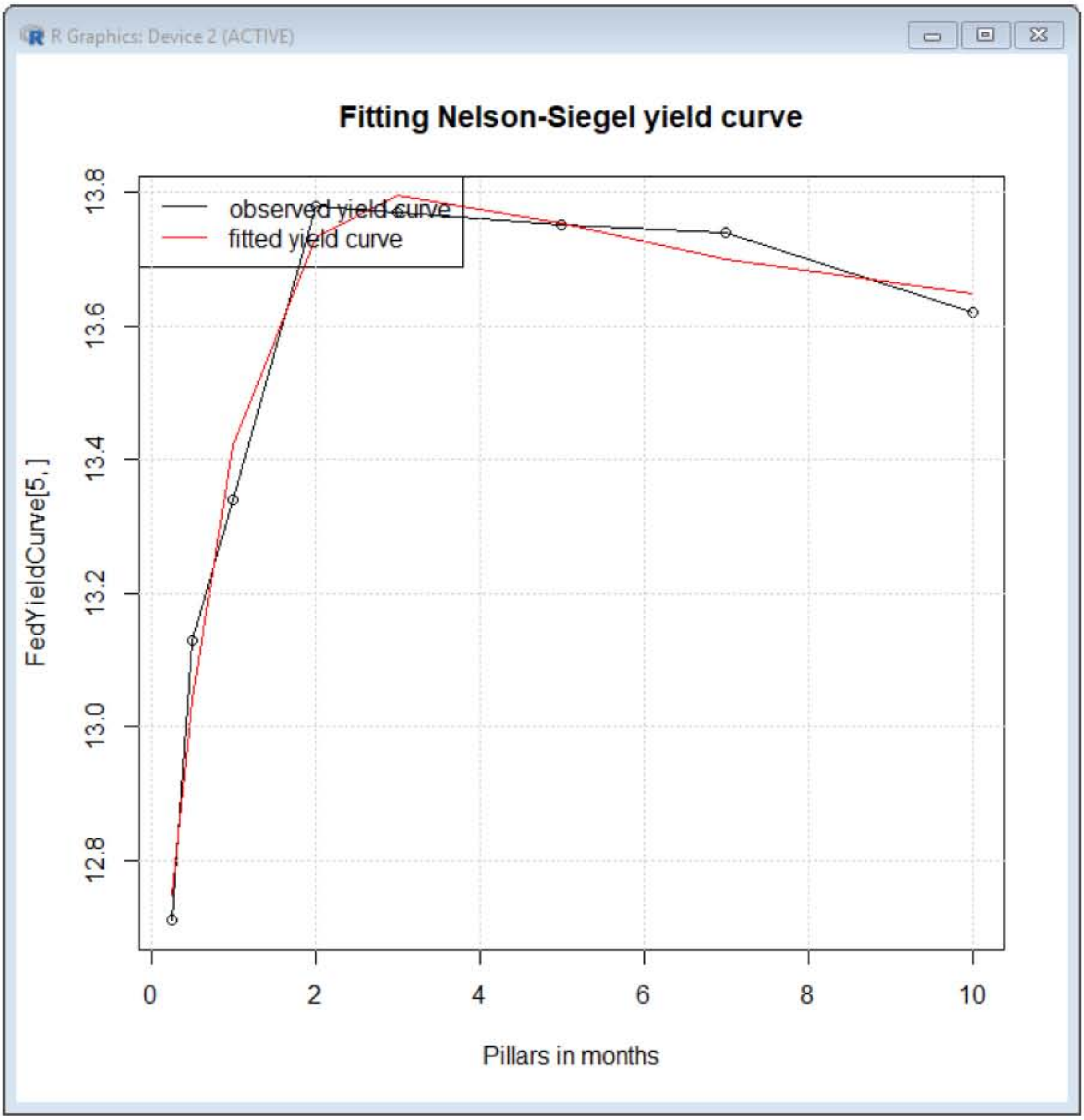
Nous retombons donc bien sur la valeur du cours théorique calculée à la main!

Courbe des taux zéro-coupons

Nous avons pour les SWAPs (et pas que!) un élément central est l'estimation de la courbe des taux zéro-coupon. Il existe de nombreux modèles, mais un modèle semi-empirique relativement simple est celui de Nelson-Siegel. Voyons comment utiliser ce dernier à l'aide du package **YieldCurve**, en reprenant à l'identique l'exemple fourni dans la documentation du package, faisant appel à son jeu de de données *FedYieldCurve* et ses fonctions **Nelson.Siegel( )** ainsi que NSRate( ):

```
> library("YieldCurve")
> data(FedYieldCurve)
> class(FedYieldCurve)
[1] "xts" "zoo"
> dim(FedYieldCurve)
[1] 372   8
> head(FedYieldCurve,10)
           R_3M  R_6M  R_1Y  R_2Y  R_3Y  R_5Y  R_7Y R_10Y
1981-12-31 12.92 13.90 14.32 14.57 14.64 14.65 14.67 14.59
1982-01-31 14.28 14.81 14.73 14.82 14.73 14.54 14.46 14.43
1982-02-28 13.31 13.83 13.95 14.19 14.13 13.98 13.93 13.86
1982-03-31 13.34 13.87 13.98 14.20 14.18 14.00 13.94 13.87
1982-04-30 12.71 13.13 13.34 13.78 13.77 13.75 13.74 13.62
1982-05-31 13.08 13.76 14.07 14.47 14.48 14.43 14.47 14.30
1982-06-30 11.86 12.80 13.24 13.80 14.00 14.07 14.07 13.95
1982-07-31  9.00 10.51 11.43 12.32 12.62 13.00 13.14 13.06
1982-08-31  8.19  9.83 10.85 11.78 12.03 12.25 12.36 12.34
1982-09-30  7.97  8.63  9.32 10.19 10.62 10.80 10.88 10.91
Warning message:
index class is Date, which does not support timezones.
Expected 'UTC' timezone, but indexTZ is ''
> maturity.Fed <- c(1/4, 0.5, 1,2,3,5,7,10)
> (NSParameters<-Nelson.Siegel(rate=first(FedYieldCurve,'10 month'),maturity=maturity.Fed))
             beta_0      beta_1   beta_2    lambda
1981-12-31 14.34594 -1.76249751 3.650061 0.9999507
1982-01-31 14.14681  0.05426534 2.219142 0.9999507
1982-02-28 13.61065 -0.54316951 2.708078 0.9999507
1982-03-31 13.61517 -0.51818755 2.757486 0.9999507
1982-04-30 13.52630 -1.16619236 2.392793 0.9999340
1982-05-31 14.13378 -1.43747932 3.237697 0.9999507
1982-06-30 13.84696 -2.49133196 3.694406 0.9999507
1982-07-31 13.02162 -4.77732118 4.909178 0.9999507
1982-08-31 12.10749 -4.77732737 6.112491 0.9999507
1982-09-30 11.02616 -3.75388964 2.730966 0.9999340
Warning message:
index class is Date, which does not support timezones.
Expected 'UTC' timezone, but indexTZ is ''
> (y <- NSrates(NSParameters[5,], maturity.Fed))
              X0.25     X0.5       X1       X2       X3       X5      X7      X10
1982-04-30 12.74806 13.04023 13.42137 13.73275 13.79568 13.75386 13.6992 13.64886
> plot(maturity.Fed,FedYieldCurve[5,],main="Fitting Nelson-Siegel yield curve",
+ xlab=c("Pillars in months"), type="o")
> lines(maturity.Fed,y, col=2)
> legend("topleft",legend=c("observed yield curve","fitted yield curve"),col=c(1,2),lty=1)
> grid()
>
```

Ce qui donne graphiquement:


R Graphics: Device 2 (ACTIVE)
Fitting Nelson-Siegel yield curve
observed yield curve
fitted yield curve
FedYieldCurve[5,]
13.8
13.6
13.4
13.2
13.0
12.8
0
2
4
6
8
10
Pillars in months

Exemple 405.: Classification ascendante hiérarchique de séries chronologiques

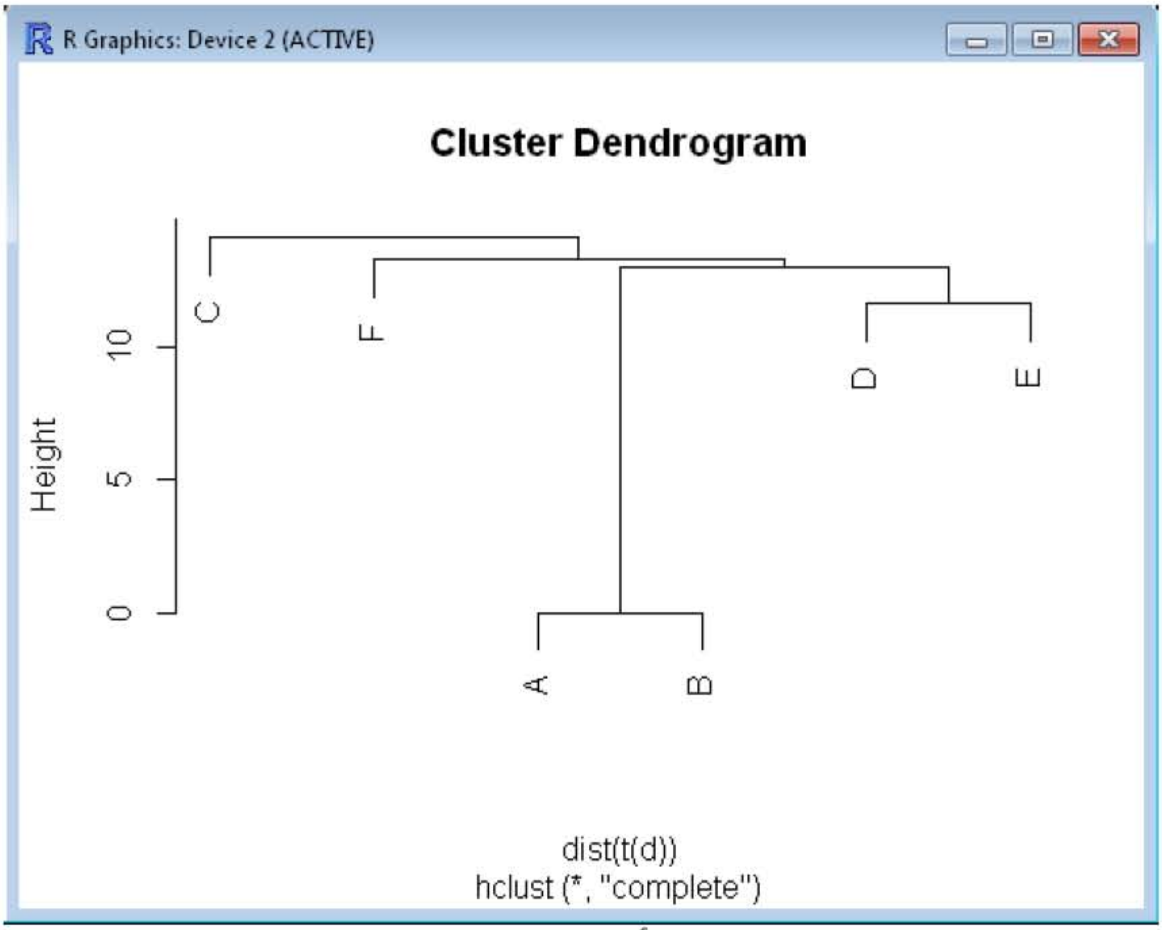
R 3.0.2

Nous allons voir ici l'utilisation de la technique de classification ascendante hiérarchique (étudiée dans le cours théorique) mais avec de séries chronologiques et dont le code a été mis à disposition sur un forum par Vincent ZOONEKYND et qui hasard faisant est d'usage courant dans certains instituts financiers en Suisse mais à ce jour plutôt avec Matlab (et donc avec un visuel beaucoup plus élémentaire).

Commençons:

```
> # On crée six séries fictives de variables N(0,1)
> n<-100
> k<-6
> d<-matrix(rnorm(k*n),nc=k)
> #on fait exprès que 2 d'entre elles soient identiques pour tester le résultat final
> d[,2]<-d[,1]
> #on donne des intitulés simples à chacune des séries
> colnames(d)<-LETTERS[1:k]
> #on somme les valeurs pour que le rendu de l'exemple soit moins fluctuant
> x<-apply(d,2,cumsum)
> #On fait un hclust des séries
> r<-hclust(dist(t(d)))
> #on crée un vecteur qui contiendra l'ordre des feuilles de l'arbre
> #r$order donner les permutations qui faut faire à l'ordre d'origine
> #pour construire l'arbre sans que les branches se croisent
> o<-1:k;o[r$order]<-1:k;o;colnames(d)[r$order]
[1] 3 4 1 5 6 2
[1] "C" "F" "A" "B" "D" "E"
> plot(r)
> windows() #pour garder l'arbre d'origine sur une fenêtre à part
>
```

Ce qui donne pour l'instant:



Ensuite:


```
R Console
> #On crée un plot avec l'art des commandes R...
> #d'abord on met les marges entre chaque future entité à zéro
> #et on met une marge à gauche de 2 pour pouvoir y mettre plus tard
> #une structure de légendes particulières
> op <- par(mar=c(0,0,0,0),oma=c(0,2,0,0))
> #on prepare le plot avec un xlim de 5 pour pouvoir y mettre les 5
> #structures de graphe. Si l'arbre venait à avoir à niveau de plus
> #il faudrait ajouter une unité (on peut rendre ce script plus adaptatif...)
> plot(NA,ylim=c(.5,k+.5),xlim=c(0,4),axes=FALSE)
> # Dendrogram. See ?hclust for details.
> xc<-yc<-rep(NA,k)
> #on crée un vecteur qui a autant d'éléments que de séries pour écraser les légendes des axes
> #pour Y on mettra le nom des séries et pour X on y mettra les séries et les niveaux de l'arbre
> xc<-yc<-rep(NA,k)
> #on dessine maintenant subtilement l'arbre de façon horizontale
> #on va utiliser la propriété merge de hclust qui renvoie une matrice
> r$merge
     [,1] [,2]
[1,]   -1   -2
[2,]   -4   -5
[3,]    1    2
[4,]   -6    3
[5,]   -3    4
>
```

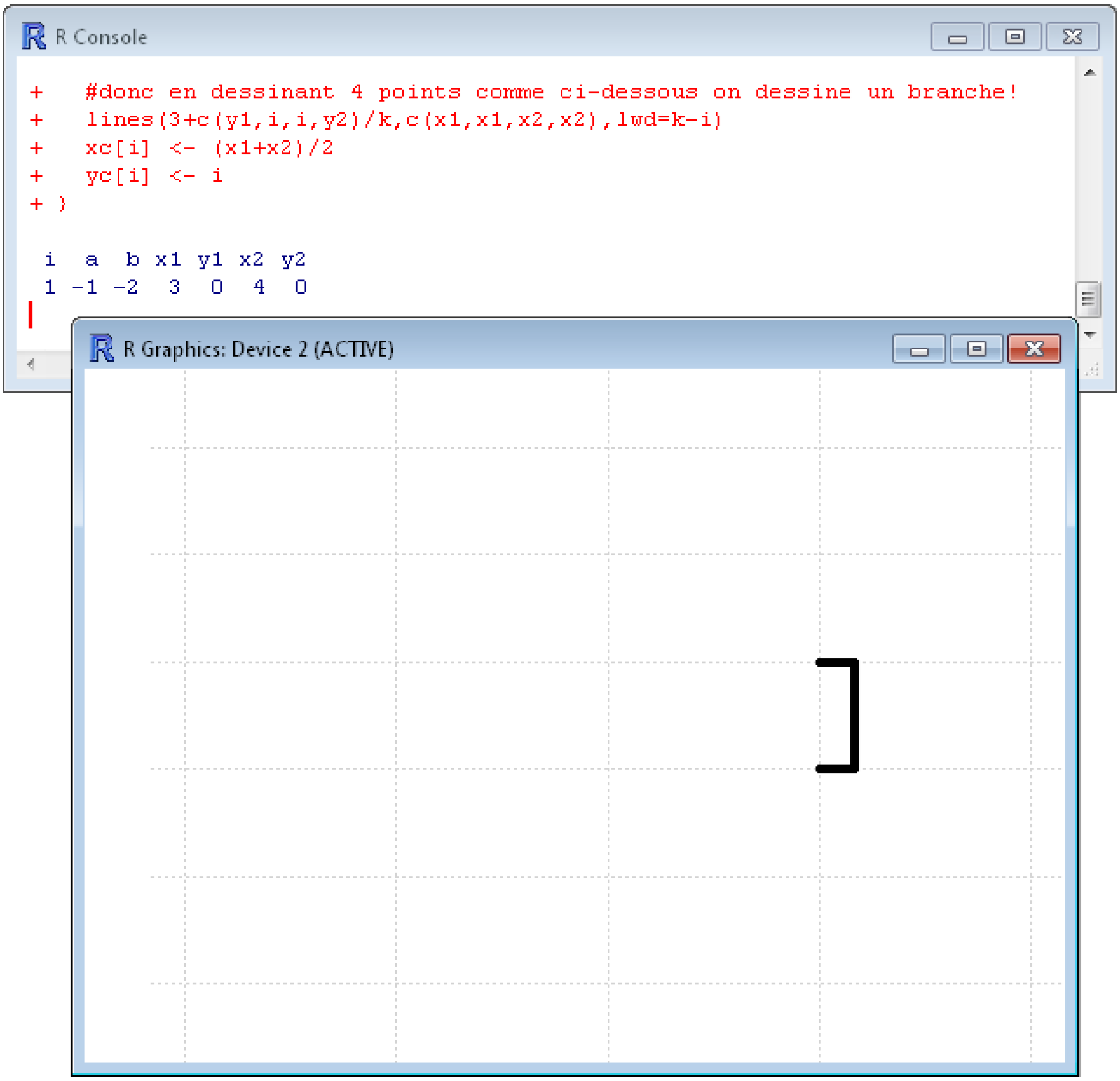
Avant de poursuivre, indiquons comment lire cette matrice... (aidez vous du graphe précédent pour vous aider!).

1. La première ligne signifie que les éléments (séries chronologiques) 1 (A) et 2 (B) ont été fusionnés et comme ils sont des singletons, le signe est négatif.
2. À la deuxième étape (deuxième ligne), nous avons les éléments (séries chronologiques) 4 (D) et 5 (E) ont été fusionnés et comme ils sont des singletons, le signe est négatif.
3. À la troisième ligne, comme les valeurs sont positives, cela signifie que nous avons associé les deux premiers associations (donc celles des deux premières étapes) ensemble.
4. À la quatrième ligne, nous associons le 6ème (F) élément (6ème série chronologie) avec l'association de la ligne 3 de cette même matrice {1,2}.
5. et ainsi de suite...

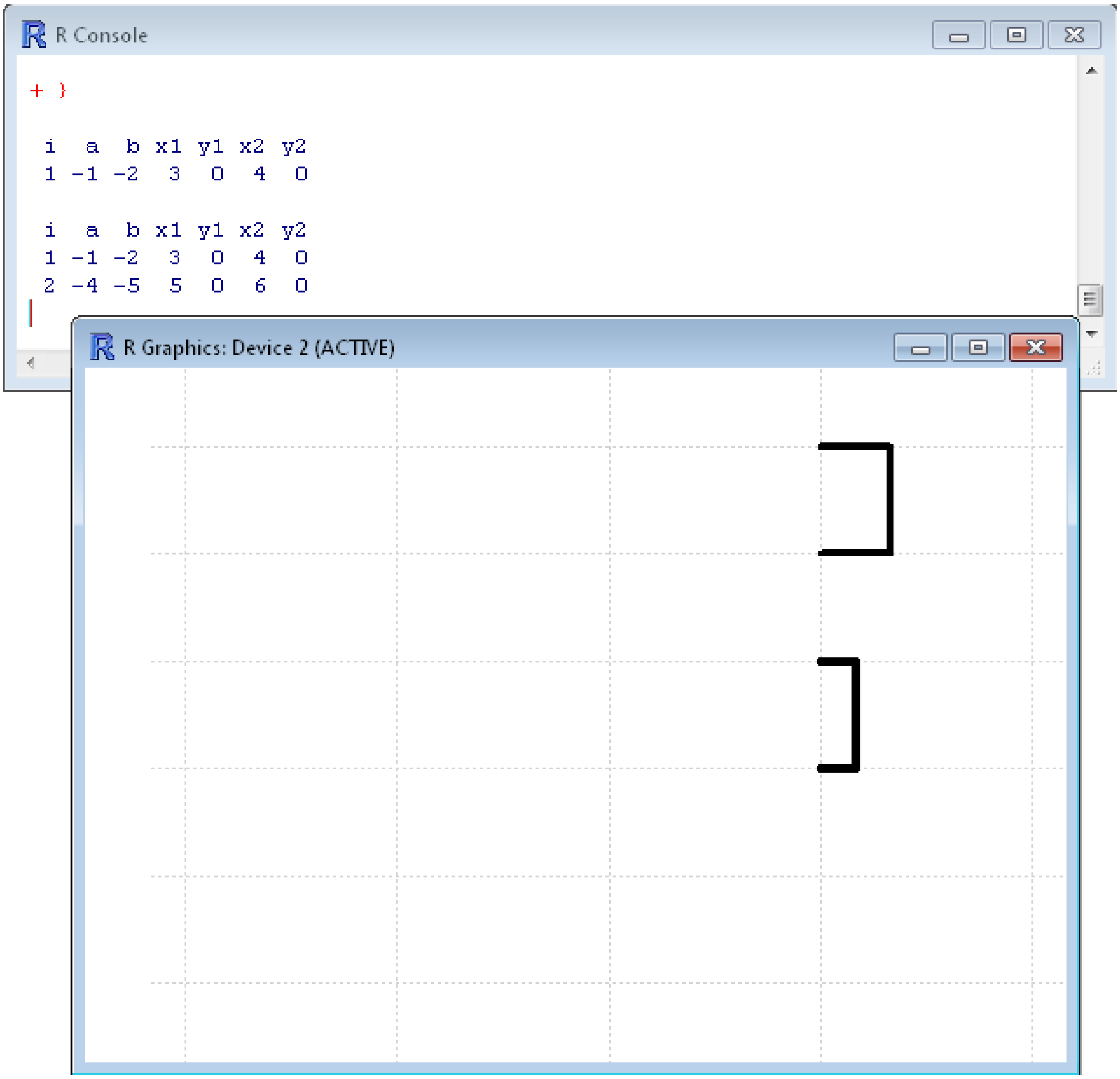
Nous poursuivons:

```
R Console
> #on construit subtilement l'arbre à l'horizontale
> o; #pour rappel
[1] 3 4 1 5 6 2
> grid() #la grille comme aide visuelle pour comprendre ce qui se passe
> v<-0 #pour comprendre ce qui est fait dans la boucle
> for(i in 1:(k-1)) {
+   a <- r$merge[i,1]
+   x1 <- if( a<0 ) o[-a] else xc[a]
+   y1 <- if( a<0 ) 0 else yc[a]
+   b <- r$merge[i,2]
+   x2 <- if( b<0 ) o[-b] else xc[b]
+   y2 <- if( b<0 ) 0 else yc[b]
+   #juste pour comprendre la construction
+   v<-rbind(v,c(i=i,a=a,b=b,x1=x1,y1=y1,x2=x2,y2=y2))
+   #on construit deux lignes parallèles à chaque itération
+   #on divise par k pour que l'arbre soit pas trop en longuer horizontalement
+   #et ne déborde ainsi pas de la zone du graphique (mais c'est un choix empirique)
+   #plus on est loin des feuilles moins les branches sont épaisses aussi
+   readline() #pour voir la construction étape par étape
+   print(v[-1,]) #pour voir le détails des calculs étape par étape
+   #ne pas oublier que la commande line utilise les coordonnées du point précédent
+   #donc en dessinant 4 points comme ci-dessous on dessine un branche!
+   lines(3+c(y1,i,i,y2)/k,c(x1,x1,x2,x2),lwd=k-i)
+   xc[i] <- (x1+x2)/2
+   yc[i] <- i
+ }
```

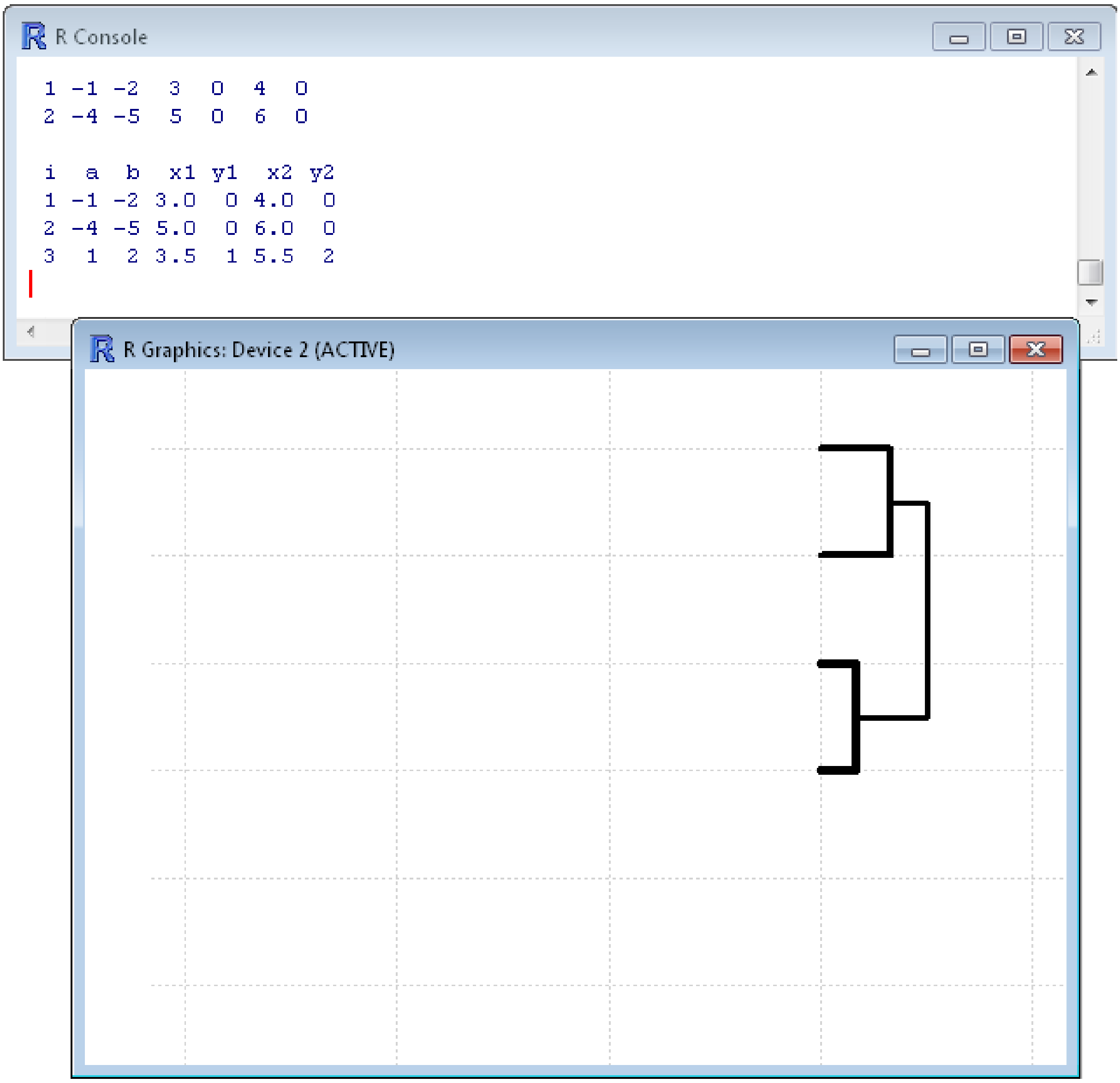
Ce qui donne à la première itération (puisque nous avons mis une pause avec la commande **readline( )**):



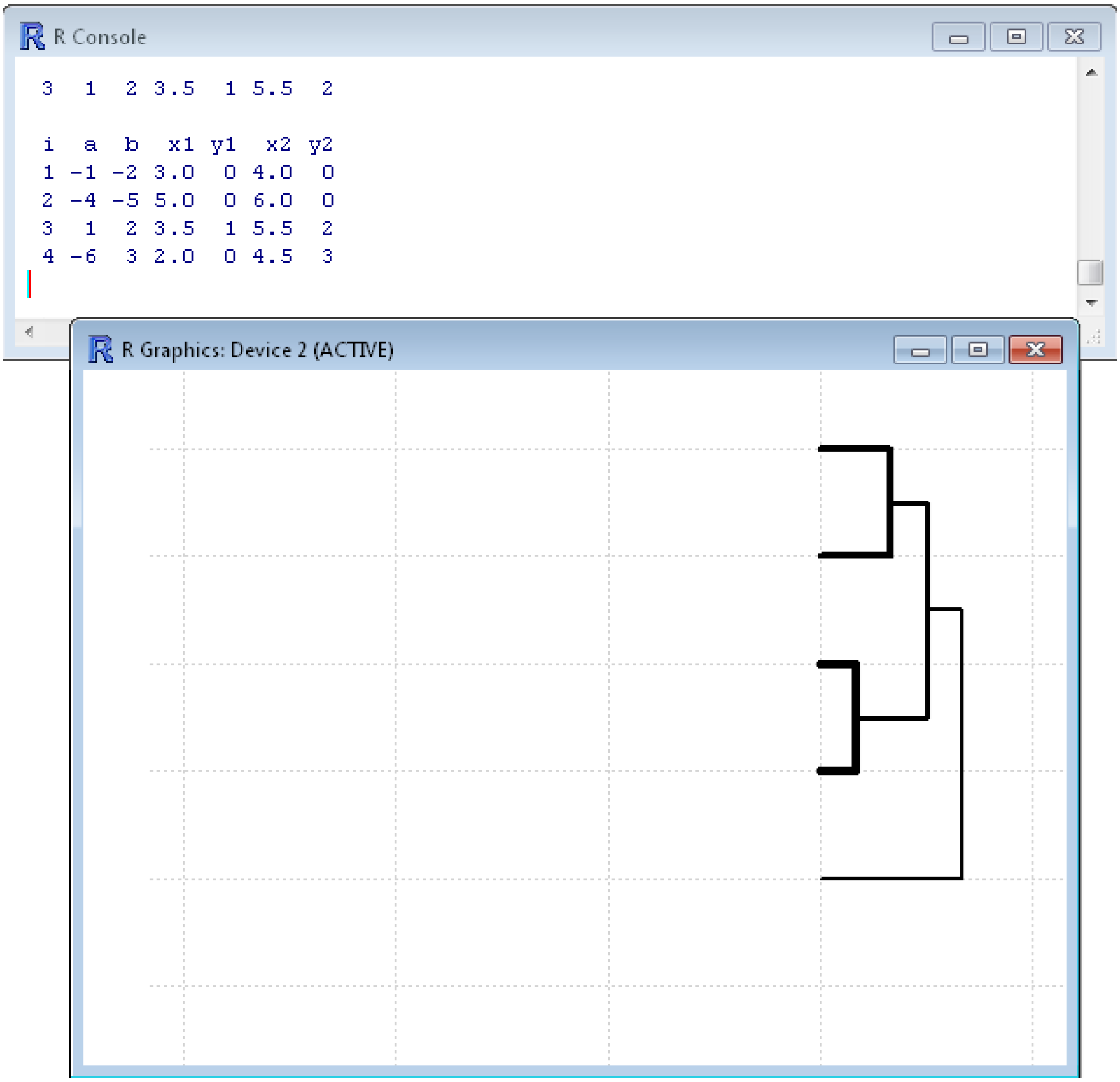
à la deuxième itération:



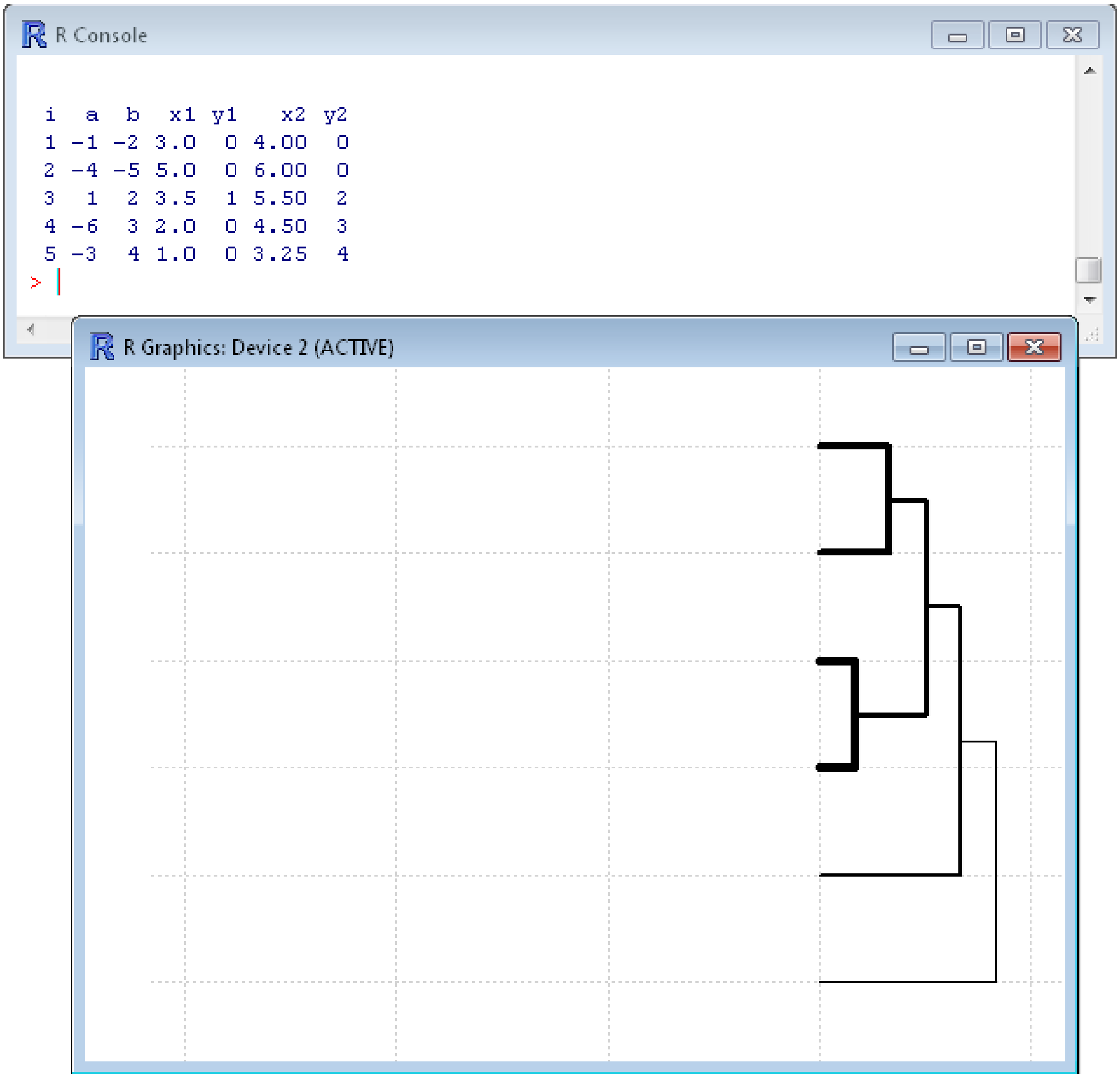
à la troisième itération:



à la quatrième itération:



et enfin à la cinquième et dernière itération:



Pour la suite enlevez le readline( ) sinon quoi cela va poser des problèmes!!!

Maintenant, nous posons les séries chronologiques en face des feuilles ainsi que l'axe des légendes:

```
> #Section où on "dessine" les courbes chronologiques
> #d'abord on met sur l'axe Y les lettre des colonnes de l'ordre du clust
> axis(2,1:k,colnames(d)[r$order],las=1)
> u<-par()$usr;u #les marges du graphiques par défaut
[1] -0.16  4.16  0.26  6.74
> r$order #pour rappel...
[1] 3 6 1 2 4 5
> for(i in 1:k) {
+   f <- c(0,3,i-.5,i+.5)
+   f <- c(
+     (f[1]-u[1])/(u[2]-u[1]),
+     (f[2]-u[1])/(u[2]-u[1]),
+     (f[3]-u[3])/(u[4]-u[3]),
+     (f[4]-u[3])/(u[4]-u[3])
+   )
+   par(new=TRUE,fig=f)
+   #on ajoute simplement la courbe i qui nous intéresse dans la zone délimite par f
+   plot(x[,r$order[i]],axes=FALSE,xlab="",ylab="",main="",type="l",col="navy",lwd=2)
+   box()
+ }
> par(op)
>
```

Ce qui donne au final:

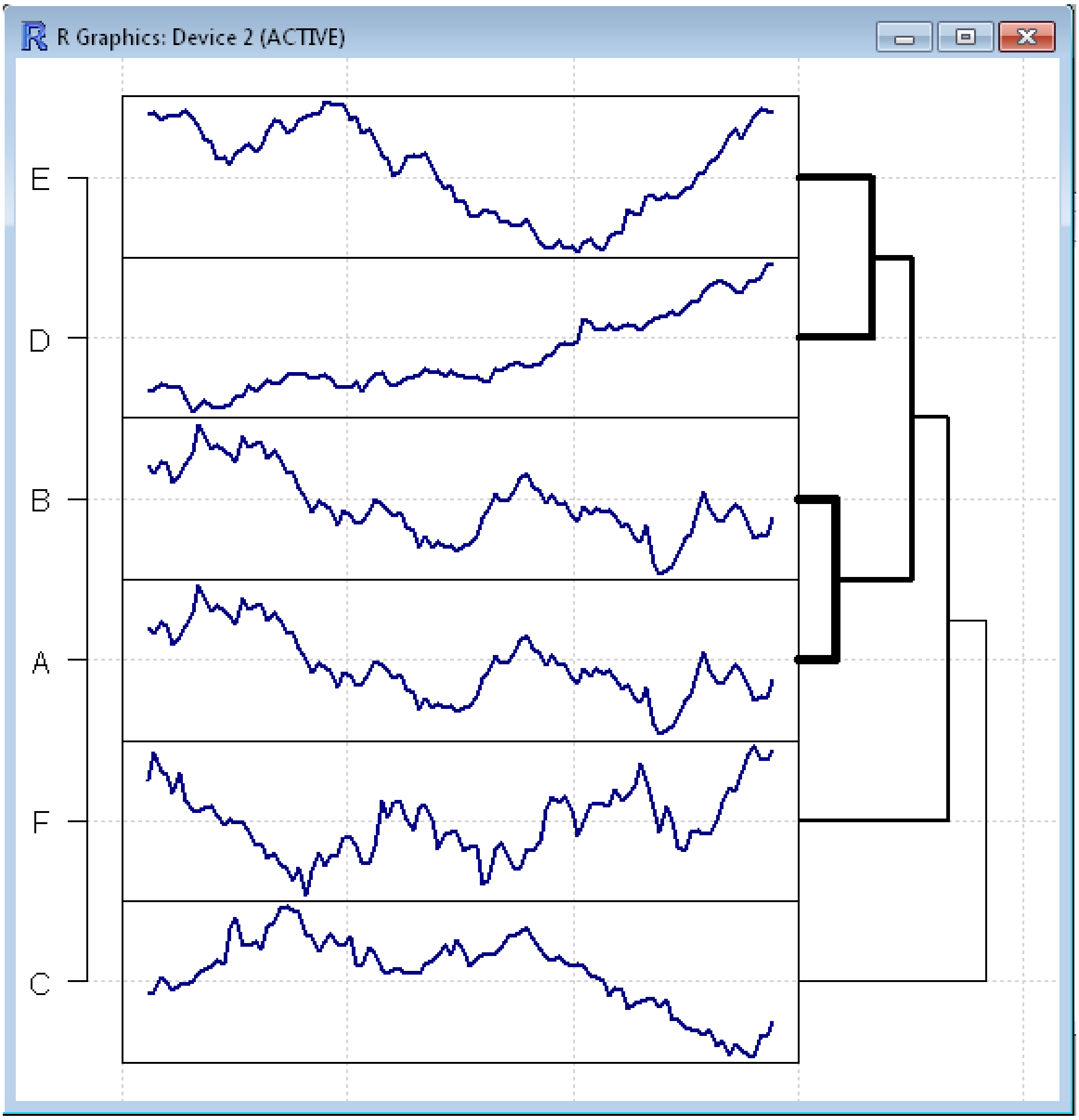


Figure 337 Graph de classification ascendante hiérarchique de séries temporelles

Voici le script sans les commentaires pour ceux qui veulent copier/coller (c'est un excellent exercice à décortiquer pour le cerveau!):

```
n <- 100
k <- 6
d <- matrix(rnorm(k*n),nc=k)
d[,2] <- d[,1]colnames(d) <- LETTERS[1:k]
x <- apply(d,2,cumsum)
r <- hclust(dist(t(d)))
op <- par(mar=c(0,0,0,0),oma=c(0,2,0,0))
plot(NA,ylim=c(.5,k+.5), xlim=c(0,4),axes=FALSE)
xc <- yc <- rep(NA,k)
o <- 1:k
o[r$order] <- 1:k
for(i in 1:(k-1)) {
  a <- r$merge[i,1]
  x1 <- if( a<0 ) o[-a] else xc[a]
  y1 <- if( a<0 ) 0 else yc[a]
  b <- r$merge[i,2]
  x2 <- if( b<0 ) o[-b] else xc[b]
  y2 <- if( b<0 ) 0 else yc[b]
  lines(
    3+c(y1,i,i,y2)/k,
    c(x1,x1,x2,x2),
    lwd=k-i
  )
  xc[i] <- (x1+x2)/2
  yc[i] <- i
}
axis(2,1:k,colnames(d)[r$order],las=1)
u <- par()$usr
for(i in 1:k) {
  f <- c(0,3,i-.5,i+.5)
  f <- c(
    (f[1]-u[1])/(u[2]-u[1]),
    (f[2]-u[1])/(u[2]-u[1]),
    (f[3]-u[3])/(u[4]-u[3]),
    (f[4]-u[3])/(u[4]-u[3])
  )
  par(new=TRUE,fig=f)

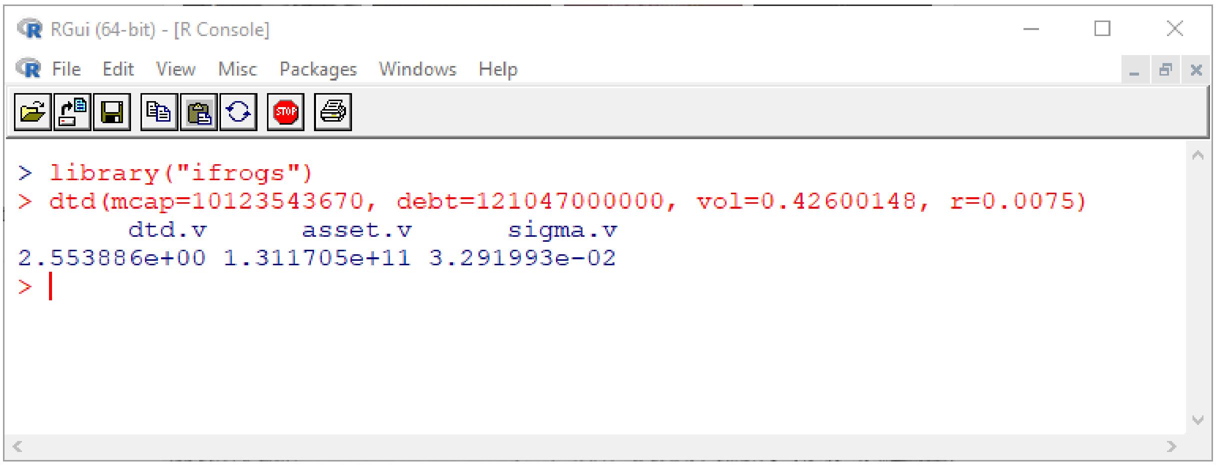
plot(x[,r$order[i]],axes=FALSE,xlab="",ylab="",main="",type="l",col="navy",
lwd=2)
  box()
}
par(op)
```

Exemple 406.: Modèle Merton-KMV de risque de défaut

R 3.6.1

Nous allons ici faire un cas pratique de calcul du modèle Merton-KMV basées sur les mêmes valeurs que celles utilisées dans la démonstration faire avec Microsoft Excel.

Nous utilisons pour cela le package **ifrogs** et la fonction **dtd( )** (qui donne combien d'écarts-types est une entreprise loin du point par défaut):



Nous ne somme pas trop loin du résultat obtenu avec Microsoft Excel (à quelques pourcents) et MATLAB. À nouveau ce genre de différences explique pourquoi il faut tester plusieurs solutions informatiques pour comparer les résultats.

Exemple 407.: Évaluation immobilière (real estate)

R 3.6.1

Influence des investissements

Une analyse de l'influence supposée des investissements sur le tourisme:

```
> mydata<-read.csv("c:/tmp/Tourism_Romanian_1995_2007.csv",header=T,sep=",",skip=1)
> colnames(mydata)<-c("Year","GDP_in_Million_RON","Hotel_Restaurants_Investments","Tourists_In_Thousands","Overnight_Stays_In_Thousands")
> mydata
   Year GDP_in_Million_RON Hotel_Restaurants_Investments Tourists_In_Thousands Overnight_Stays_In_Thousands
1  1995             8461.5                        1389.0                  7070                        24111
2  1996            11924.2                        1040.7                  6595                        21838
3  1997             9375.7                         627.3                  5727                        19611
4  1998             7758.8                         581.4                  5552                        19183
5  1999             6072.9                         526.3                  5109                        17670
6  2000             5491.7                         316.7                  4920                        17647
7  2001             5237.2                         584.8                  4875                        18122
8  2002             4586.9                         557.0                  4847                        17277
9  2003             4556.1                         704.4                  5057                        17845
10 2004             4850.4                         956.9                  5639                        18501
11 2005             5872.2                        1080.6                  5805                        18373
12 2006             7060.5                        1332.0                  6216                        18992
13 2007             8402.4                        1600.9                  6972                        20593
> summary(aov(GDP_in_Million_RON~.,data=mydata[,-1]))
                              Df   Sum Sq  Mean Sq F value   Pr(>F)
Hotel_Restaurants_Investments  1  9638396  9638396   9.282 0.013869
Tourists_In_Thousands          1 38464246 38464246  37.041 0.000182
Overnight_Stays_In_Thousands   1   915578   915578   0.882 0.372253
Residuals                      9  9345893  1038433
> summary(lm(GDP_in_Million_RON~.,data=mydata[,-1]))

Call:
lm(formula = GDP_in_Million_RON ~ ., data = mydata[, -1])

Residuals:
    Min      1Q  Median      3Q     Max
-1248.3  -381.9  -169.0   608.5  1702.2

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                  -1.621e+04  3.623e+03  -4.474  0.00155
Hotel_Restaurants_Investments -8.174e+00  2.167e+00  -3.771  0.00441
Tourists_In_Thousands          6.585e+00  1.749e+00   3.765  0.00445
Overnight_Stays_In_Thousands  -3.888e-01  4.141e-01  -0.939  0.37225

Residual standard error: 1019 on 9 degrees of freedom
Multiple R-squared:  0.8399,    Adjusted R-squared:  0.7865
F-statistic: 15.73 on 3 and 9 DF,  p-value: 0.0006344

> 
```

27. Finance Actuarielle

Le but de ce chapitre va être de vérifier les résultats renvoyés par le package **lifecontingencies** correspond aux résultats obtenus suites aux quelques démonstrations mathématiques triviales étudiées dans le cours théorique d'introduction élémentaire à la finance actuarielle.

Pour des données à télécharger, vous pouvez visiter:

http://www.mortality.org

Pour cela, nous utiliserons le fichier suivant:

Survie.csv - Bloc-notes

Fichier Edition Format Affichage ?

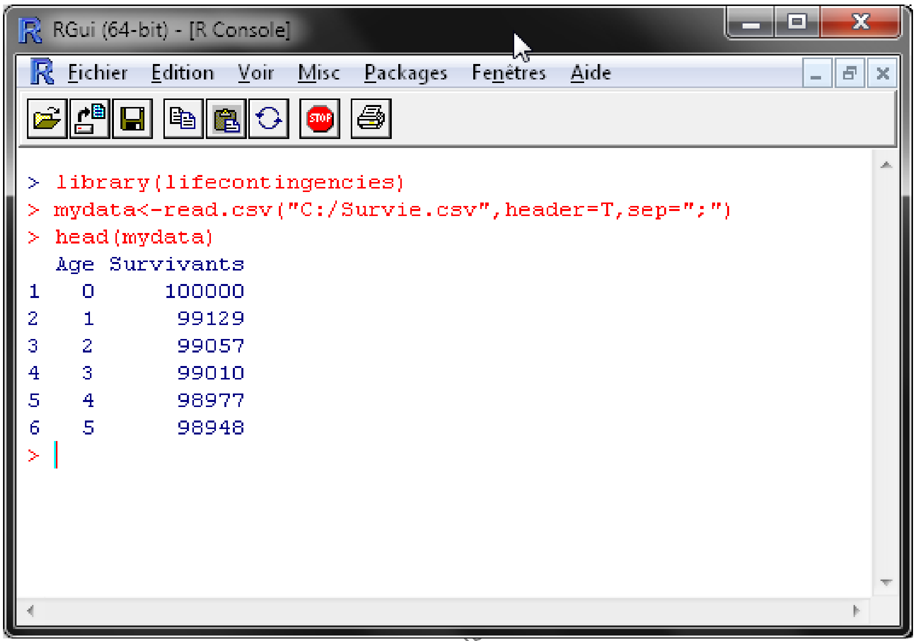
```
Age;Survivants
0;100000
1;99129
2;99057
3;99010
4;98977
5;98948
6;98921
7;98897
8;98876
9;98855
10;98835
11;98814
12;98793
13;98771
14;98745
15;98712
16;98667
17;98606
18;98520
19;98406
20;98277
21;98137
22;97987
23;97830
24;97677
25;97524
26;97373
27;97222
28;97070
29;96916
30;96759
31;96597
32;96429
33;96255
```

qui se poursuit bien évidemment au-delà de 33 ans...

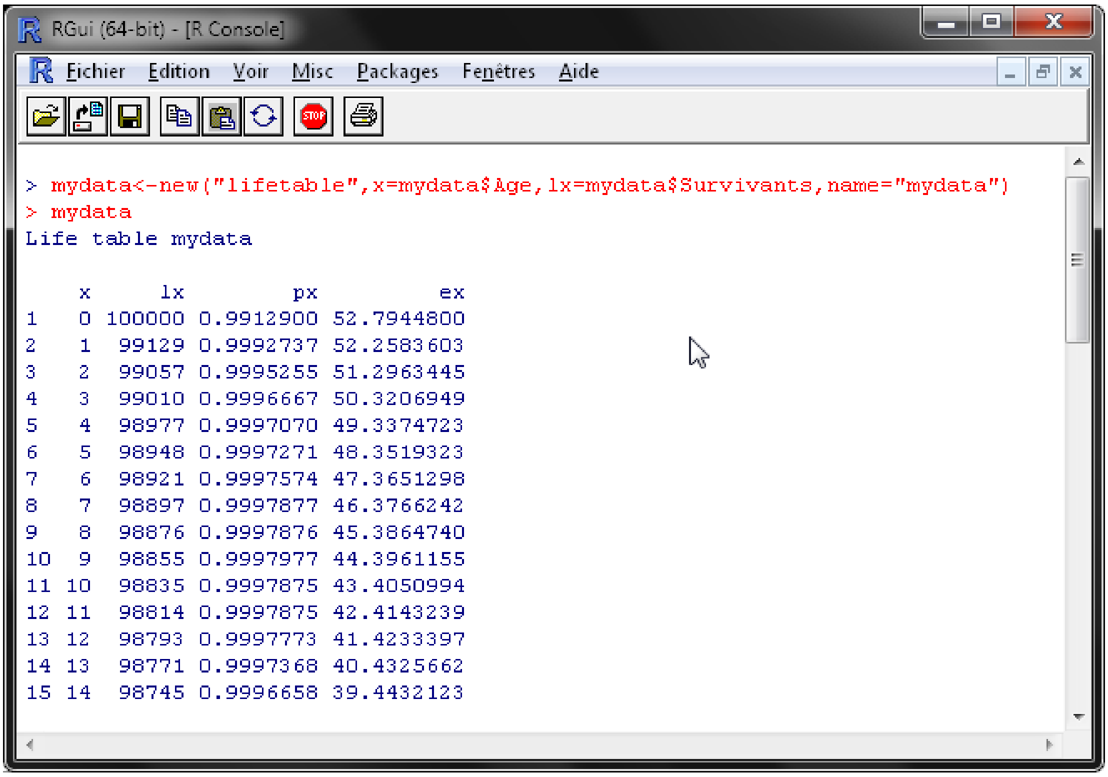
Exemple 408.: Générer une table des survivants simple et indicateurs de survies

R 3.0.2

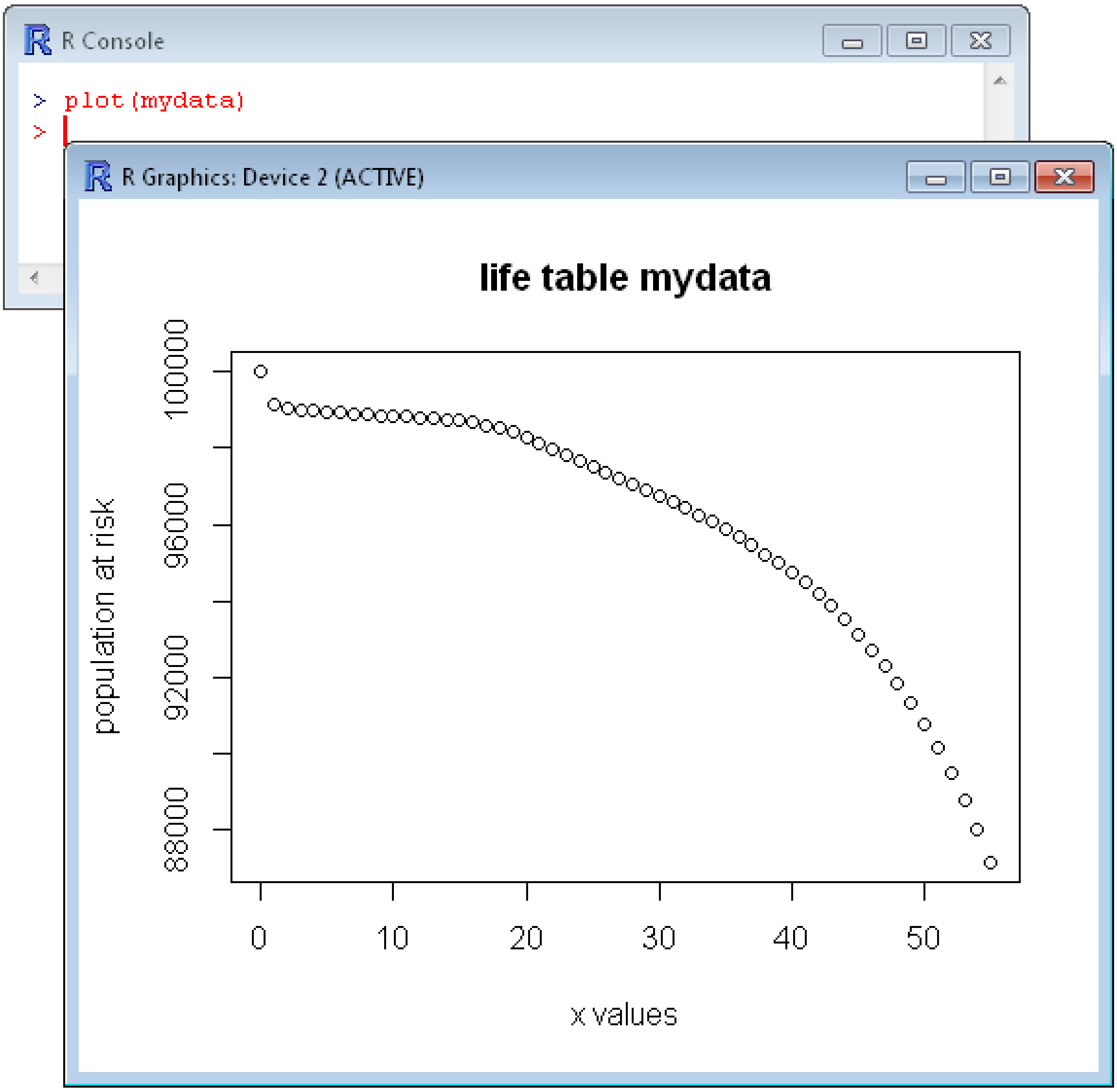
Nous chargeons donc d'abord le package **lifecontingencies** et le fichier *.csv qui nous intéresse:



Nous avons alors en utilisant la commande **new("lifetable",...)** du package **lifecontingencies**:

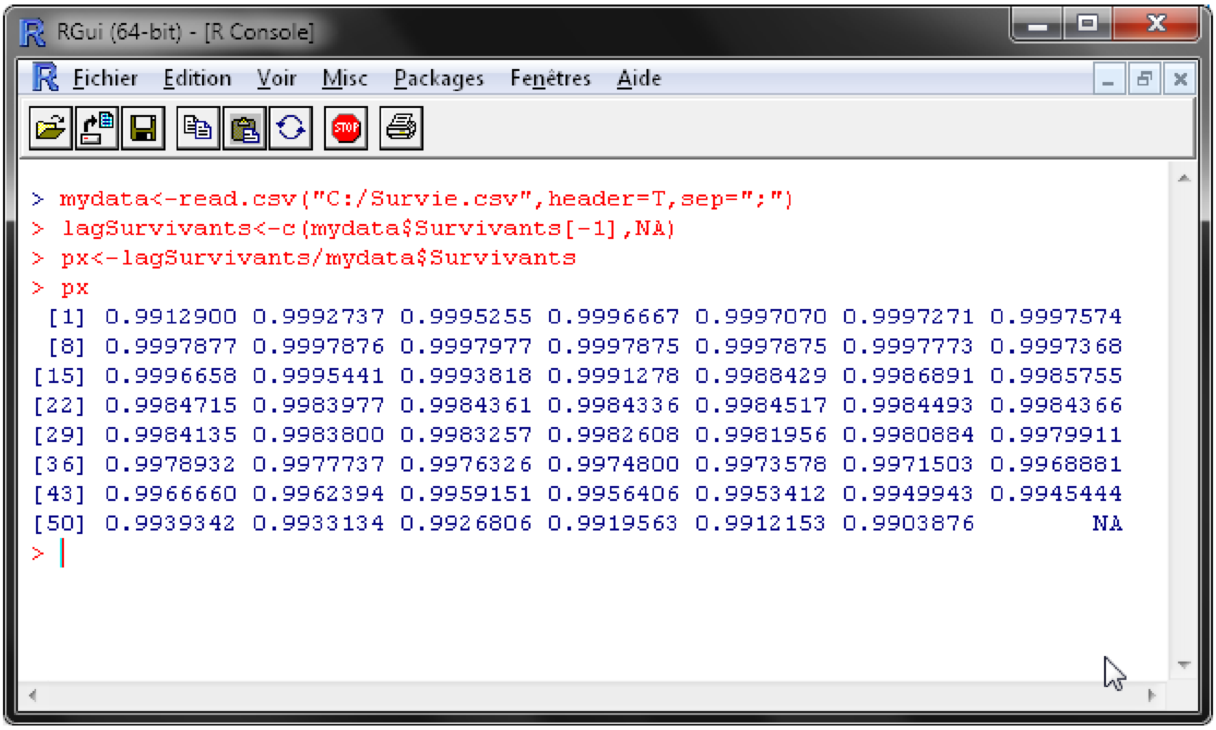


avec au passage un petit plot:



Pour les deux premières colonnes *x* et *lx* il n'y a rien de particulier à dire.

La colonne *px* est très simple à calculer à la main:



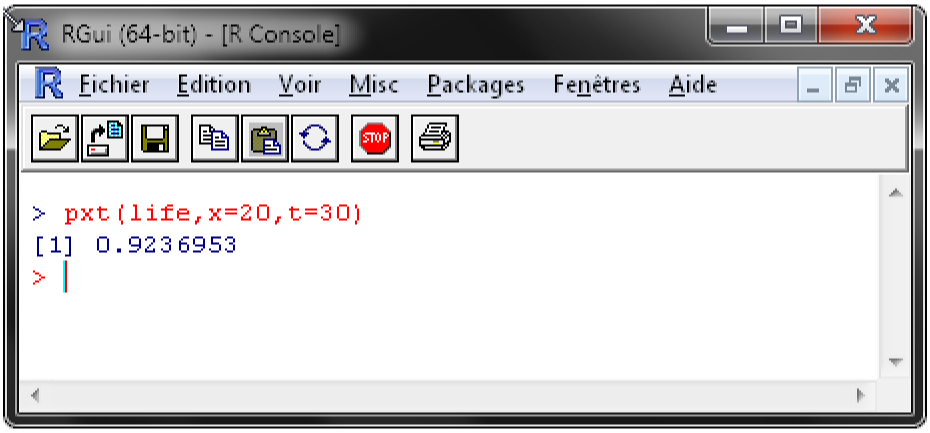
La colonne *ex* qui donne l'espérance de vie à la naissance est déjà plus intéressante à vérifier pour voir si elle correspond aux notions vues dans le cours théorique. Pour obtenir l'espérance de vie à la naissance relativement à cette table, nous savons qu'il suffit de faire la somme des probabilités de survie de la fin jusqu'au temps d'intérêt divisé par la probabilité de survie jusqu'à ce même temps d'intérêt. Comme je n'ai pas trouvé dans R une commande unique pour vérifier cela à la main nous pouvons dans le cas présent nous reporter à la feuille Microsoft Excel suivante:

	A	B	C	D	E
1	Age	Survivants (l_x)	Proportion	p_x	Espérance de vie e_x
2	0	100000	1.000000	0.991290	52.794480
3	1	99129	0.991290	0.999274	52.258360
4	2	99057	0.990570	0.999526	51.296345
5	3	99010	0.990100	0.999667	50.320695
6	4	98977	0.989770	0.999707	49.337472
7	5	98948	0.989480	0.999727	48.351932
8	6	98921	0.989210	0.999757	47.365130
9	7	98897	0.988970	0.999788	46.376624
10	8	98876	0.988760	0.999788	45.386474
11	9	98855	0.988550	0.999798	44.396116
12	10	98835	0.988350	0.999788	43.405099
13	11	98814	0.988140	0.999787	42.414324
14	12	98793	0.987930	0.999777	41.423340
15	13	98771	0.987710	0.999737	40.432566
16	14	98745	0.987450	0.999666	39.443212
17	15	98712	0.987120	0.999544	38.456398
18	16	98667	0.986670	0.999382	37.473938
19	17	98606	0.986060	0.999128	36.497120
20	18	98520	0.985200	0.998843	35.528979
21	19	98406	0.984060	0.998689	34.570138

Avec toute simplement:

	A	B	C	D	E
1	Age	Survivants (l_x)	Proportion	p_x	Espérance de vie e_x
2	0	100000	=B2/B2	=C3/C2	=SOMME(C3:$C$57)/C2
3	1	99129	=B3/$B$2	=C4/C3	=SOMME(C4:$C$57)/C3
4	2	99057	=B4/$B$2	=C5/C4	=SOMME(C5:$C$57)/C4
5	3	99010	=B5/$B$2	=C6/C5	=SOMME(C6:$C$57)/C5
6	4	98977	=B6/$B$2	=C7/C6	=SOMME(C7:$C$57)/C6
7	5	98948	=B7/$B$2	=C8/C7	=SOMME(C8:$C$57)/C7
8	6	98921	=B8/$B$2	=C9/C8	=SOMME(C9:$C$57)/C8
9	7	98897	=B9/$B$2	=C10/C9	=SOMME(C10:$C$57)/C9
10	8	98876	=B10/$B$2	=C11/C10	=SOMME(C11:$C$57)/C10
11	9	98855	=B11/$B$2	=C12/C11	=SOMME(C12:$C$57)/C11
12	10	98835	=B12/$B$2	=C13/C12	=SOMME(C13:$C$57)/C12
13	11	98814	=B13/$B$2	=C14/C13	=SOMME(C14:$C$57)/C13
14	12	98793	=B14/$B$2	=C15/C14	=SOMME(C15:$C$57)/C14
15	13	98771	=B15/$B$2	=C16/C15	=SOMME(C16:$C$57)/C15
16	14	98745	=B16/$B$2	=C17/C16	=SOMME(C17:$C$57)/C16
17	15	98712	=B17/$B$2	=C18/C17	=SOMME(C18:$C$57)/C17
18	16	98667	=B18/$B$2	=C19/C18	=SOMME(C19:$C$57)/C18
19	17	98606	=B19/$B$2	=C20/C19	=SOMME(C20:$C$57)/C19
20	18	98520	=B20/$B$2	=C21/C20	=SOMME(C21:$C$57)/C20
21	19	98406	=B21/$B$2	=C22/C21	=SOMME(C22:$C$57)/C21

Nous pouvons ensuite calculer la probabilité de la durée encore à vivre, c'est-à-dire dans le cas de l'exemple ci-dessous la probabilité que si un individu est encore vivant à 20 ans qu'il atteigne les 20+30=50 ans en utilisant la commande **pxt( )** :



Ce qui correspond bien à au résultat vu dans le cours théorique:

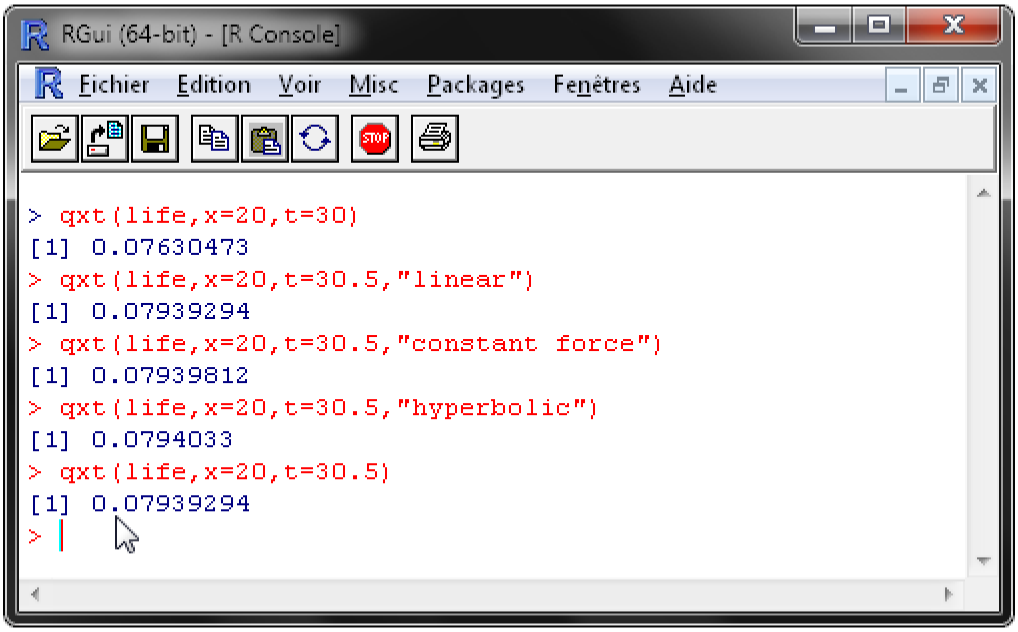
	A	B	C	D	E	F	G	H	I	J	K
1	Age	Survivants (l_x)	Proportion	p_x	Espérance de vie e_x						
2	0	100000	1.000000	0.991290	52.794480		Probabilité d'atteindre 50 ans lorsqu'on est encore vivant à 20 ans:				
3	1	99129	0.991290	0.999274	52.258360		$_{30}p_{20}$:	0.923695			
4	2	99057	0.990570	0.999526	51.296345						

Soit explicitement:

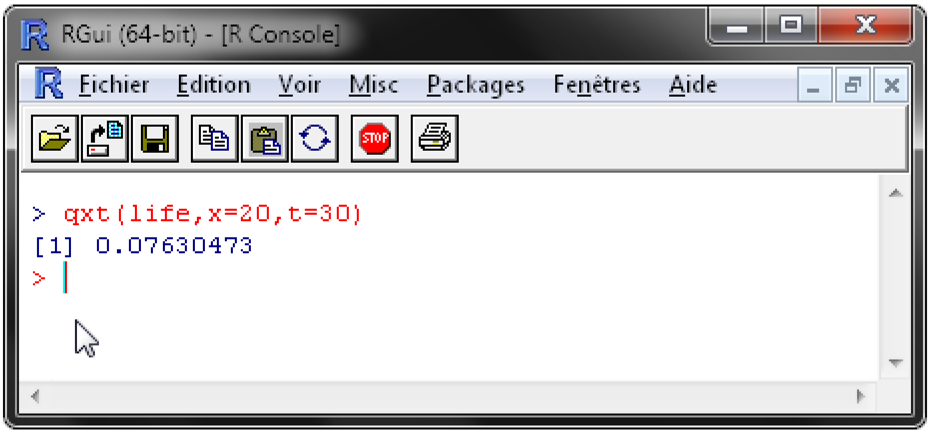
	G	H
1		
2	Probabilité d'atteindre 50 ans lorsqu'on est encore vivant à 20 ans:	
3	${}_{30}p_{20}$:	=C52/C22

Soit la proportion de survivants à 50 ans sur la proportion de survivants à 20 ans.

Le package intègre plusieurs techniques empiriques d'interpolation pour des âges non entiers:



La probabilité de décéder pendant la même période sera donnée par la commande **qxt( )** :



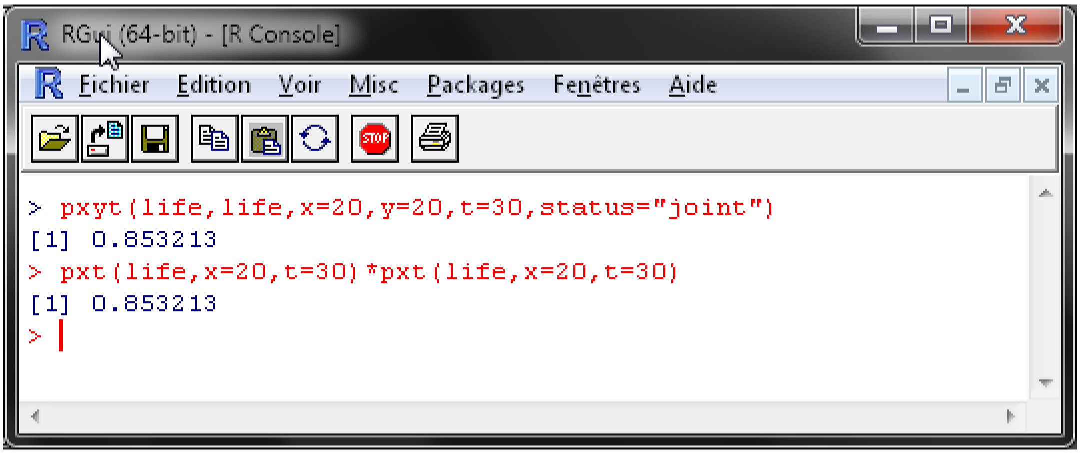
Ce qui correspond bien à au résultat vu dans le cours théorique:

	G	H	I
1			
2	Probabilité d'atteindre 50 ans lorsqu'on est encore vivant à 20 ans:		
3	$_{30}p_{20}$:	0.923695	
4			
5	Probabilité de décéder entre la 20ème et 50ème année:		
6	$_{30}q_{20}$:	0.07630473	

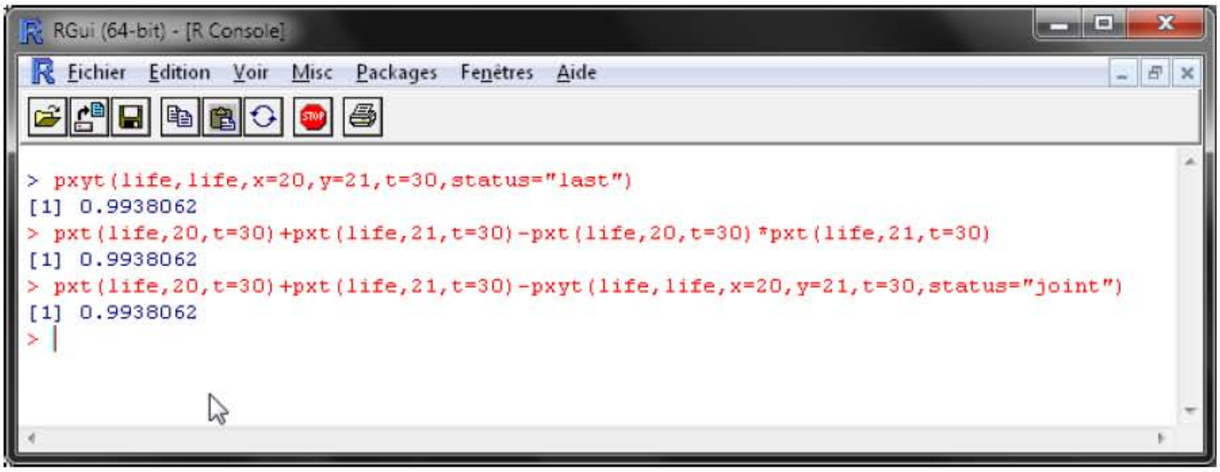
Soit explicitement:

	G	H
1		
2	Probabilité d'atteindre 50 ans lorsqu'on est encore vivant à 20 ans:	
3	$_{30}p_{20}$:	=C52/C22
4		
5	Probabilité de décéder entre la 20ème et 50ème année:	
6	$_{30}q_{20}$:	=(C22-C52)/C22

Il est aussi possible de travailler sur plusieurs têtes de plusieurs tables différentes de vie. Par exemple si dans notre cas nous souhaitons savoir quelle est la probabilité jointe (pour rappel c'est donc par définition le produit des probabilités) de survie que deux individus d'âge de départ identiques ou différentes ont de survivre pendant la même période de temps, nous utilisons alors la commande **pxyt( )** qui correspond à $_{h}p_{xy}$:



Ou pour avoir la probabilité qu'une des personnes soit encore en vie (2 événements incompatibles non nécessairement disjoints) $_{h}p_{\overline{xy}}$:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> pxyt(life,life,x=20,y=21,t=30,status="last")
[1] 0.9938062
> pxt(life,20,t=30)+pxt(life,21,t=30)-pxt(life,20,t=30)*pxt(life,21,t=30)
[1] 0.9938062
> pxt(life,20,t=30)+pxt(life,21,t=30)-pxyt(life,life,x=20,y=21,t=30,status="joint")
[1] 0.9938062
>

Exemple 409.: Capitals différés (annuités)

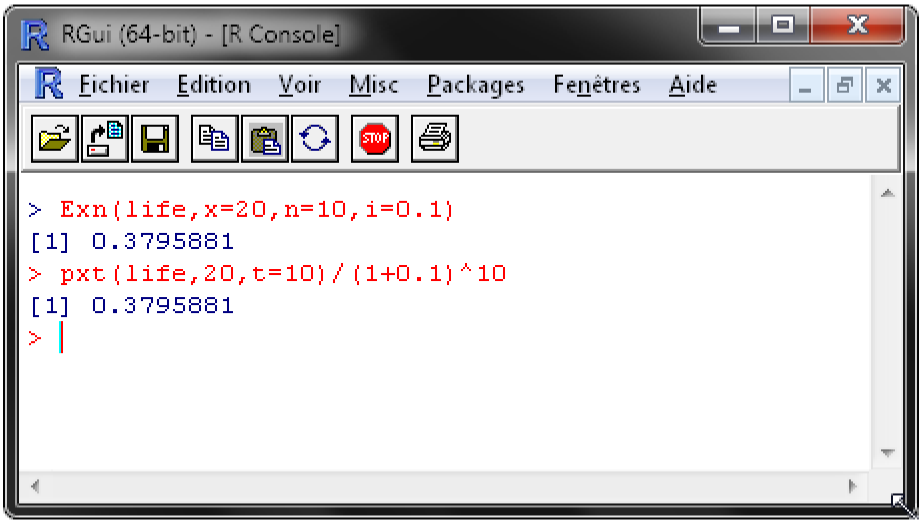
R 3.0.2

Nous pouvons aussi contrôler que les calculs de capitaux différés dans le cours théorique de techniques de gestion sont accessibles avec ce même package **lifecontingencies**.

Calculons la valeur probable d'un capital qui pour rappel est donné par:

$${}_{m|}E_x = f_e^m \frac{S_{x+m}}{S_x} = f_e^m \; {}_{m|}p_x$$

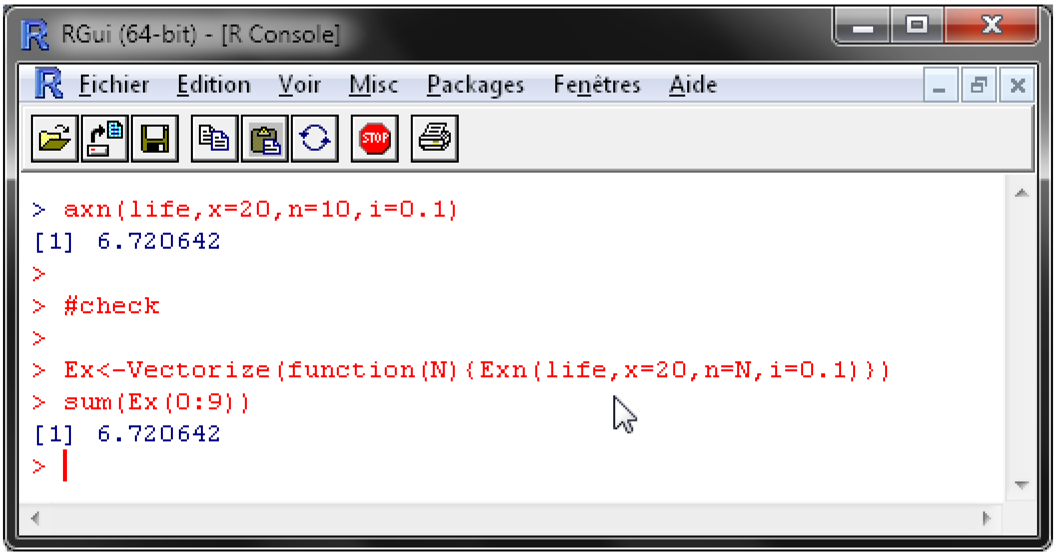
Donc en utilisant la commande **Exn( )** ou **pxt( )** , nous avons:



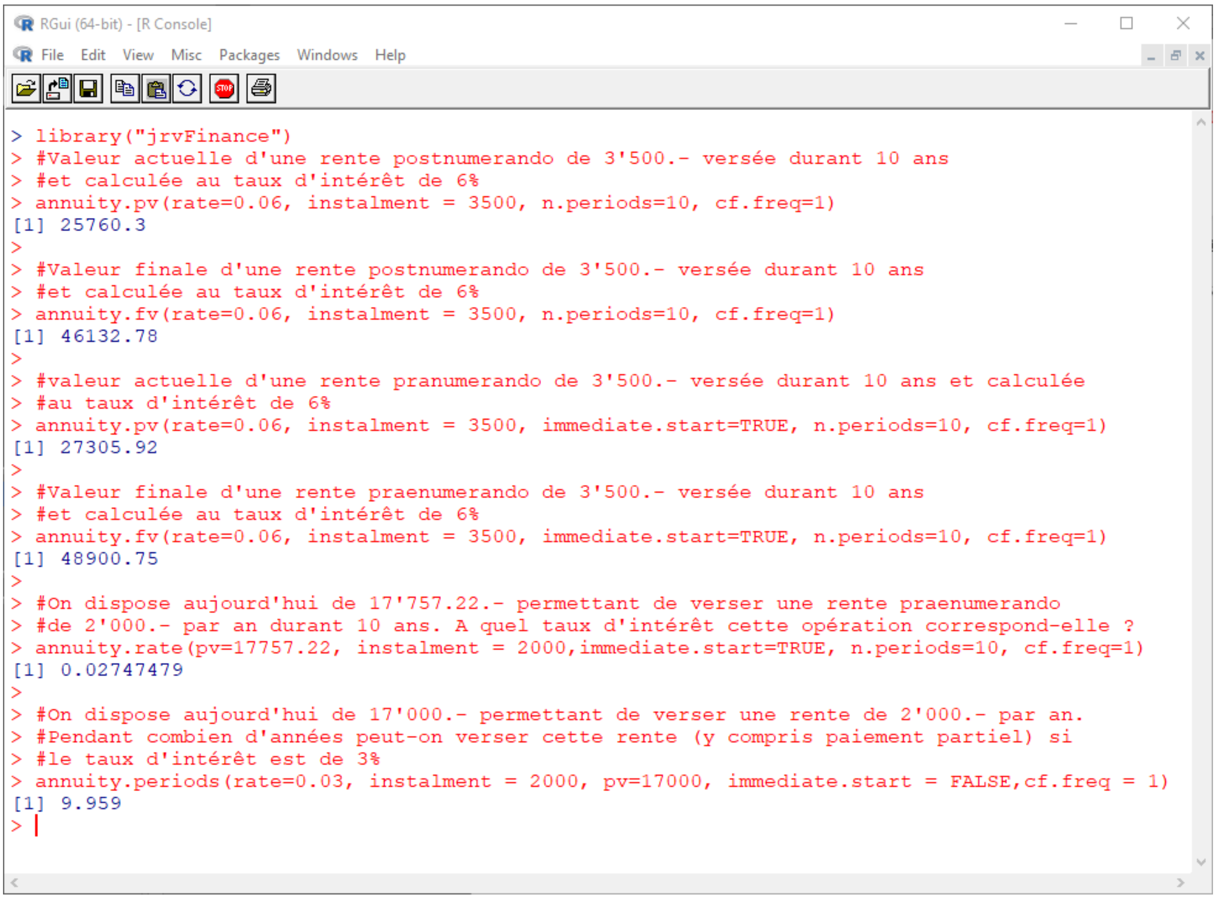
Les calculs d'annuités sont alors probablement in extenso faciles à obtenir. Rappelons la relation de la rente viagère temporaire praenumerando démontrée dans le cours théorique:

$$\ddot{a}_{x:\overline{n}|} = 1 + f_e \frac{S_{x+1}}{S_x} + f_e^2 \frac{S_{x+2}}{S_x} + \ldots + f_e^{n-1} \frac{S_{x+n-1}}{S_x} = \sum_{t=0}^{n-1} f_e^t \frac{S_{x+t}}{S_x}$$

Ce calcul se fait avec la commande **axn( )** :



Voyons maintenant comment reproduire les calculs que nous avions fait dans Microsoft Excel à l'aide des fonctions **annuity.pv( )**, **annuity.fv( )**, **annuity.rate( )** et **annuity.periods( )** du package **jrvFinance**:



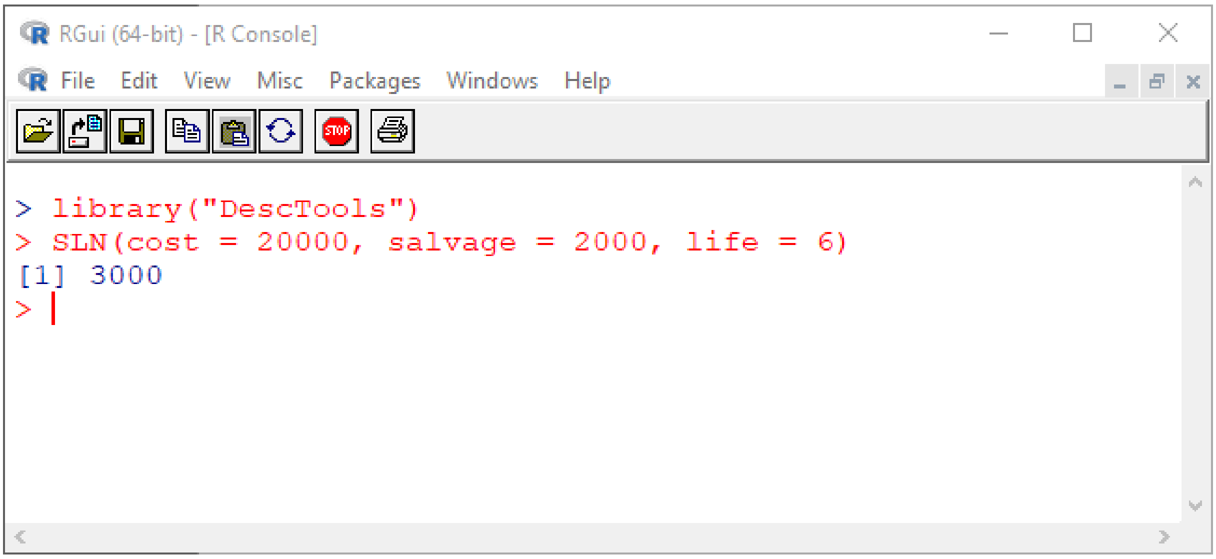
On retrouve bien les mêmes résultats que les calculs faits à la main et que dans Microsoft Excel!

Exemple 410.: Amortissements

R 3.6.1

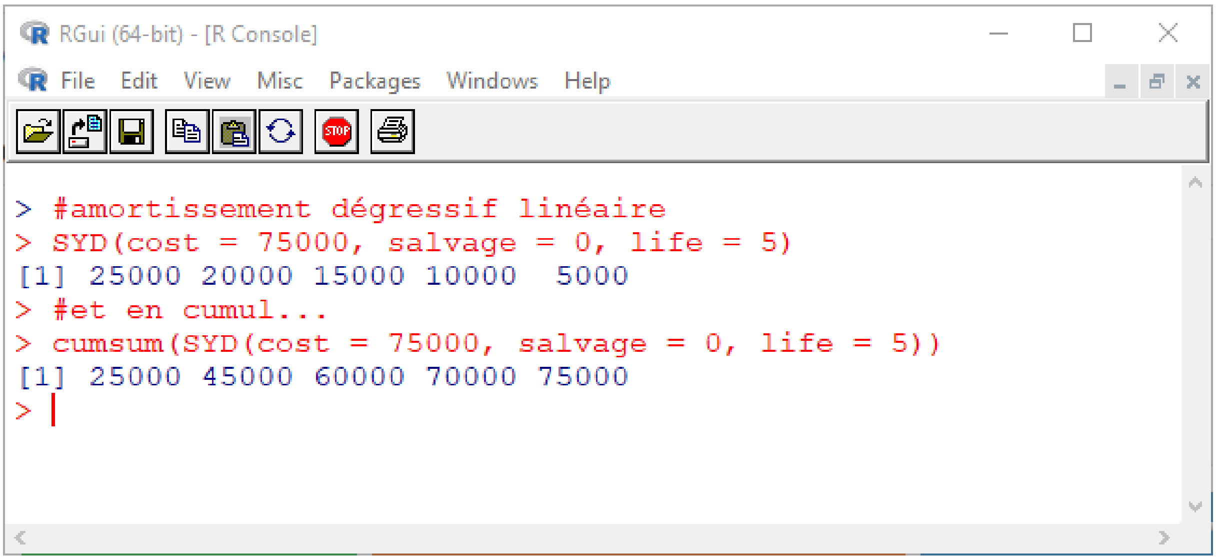
Le but va être ici de reproduire les calculs faits à la main dans le cours théorique et dans Microsoft Excel.

On commence par l'amortissement linéaire avec la fonction **SLN( )** du package **DescTools**:



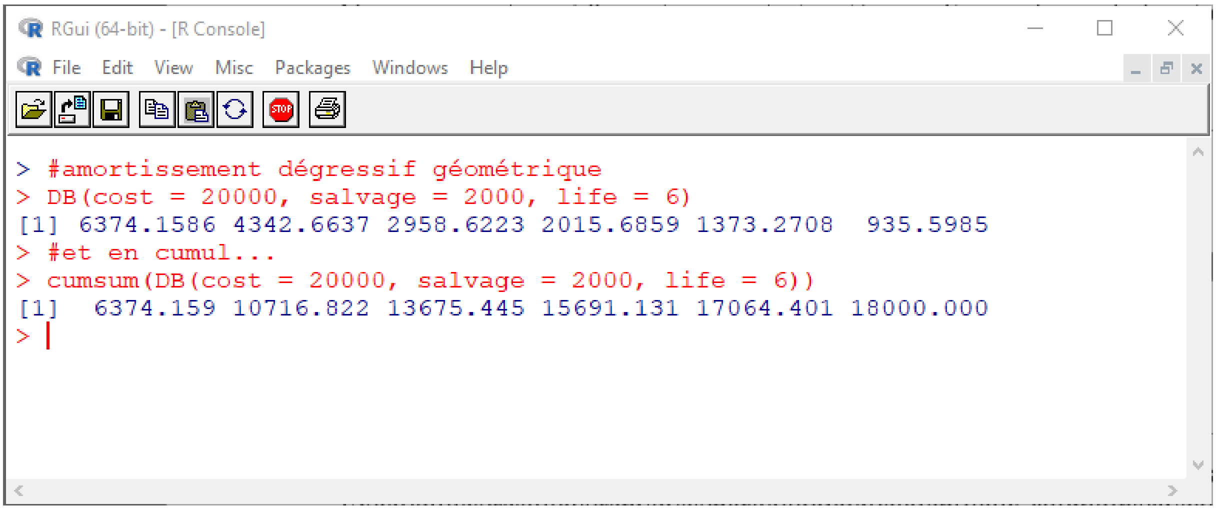
On retrouve bien la même chose que les calculs faits à la main et que dans Microsoft Excel!

Voiyons maintenant l'amortissement arithmétique dégressif avec la fonction **SYD( )** toujours du même package:



On retrouve bien la même chose que les calculs faits à la main et que dans Microsoft Excel!

Voyons maintenant l'amortissement géométrique dégressif avec la fonction **DB( )** toujours du même package:



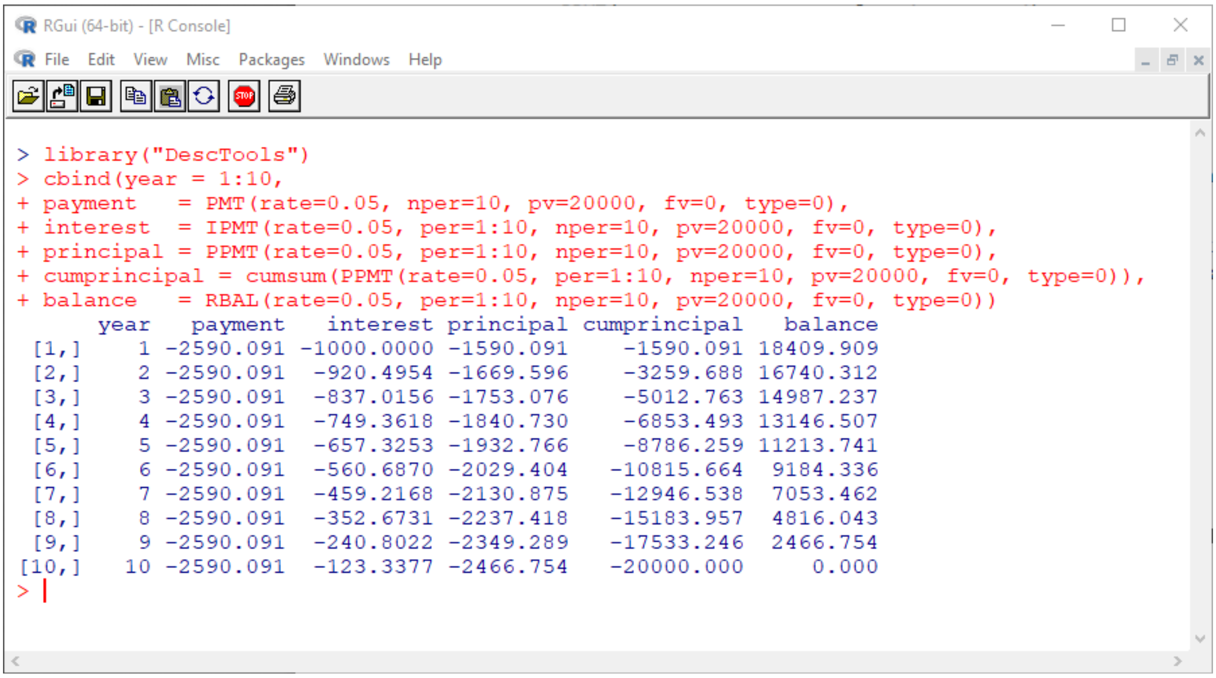
On retrouve bien la même chose que les calculs faits à la main mais contrairement à Microsoft Excel où les résultats renvoyés par le logiciel étaient faux, au moins avec R ces derniers sont justes…!

Exemple 411.: Emprunts

R 3.6.1

Le but va être ici de reproduire les calculs faits à la main dans le cours théorique et dans Microsoft Excel.

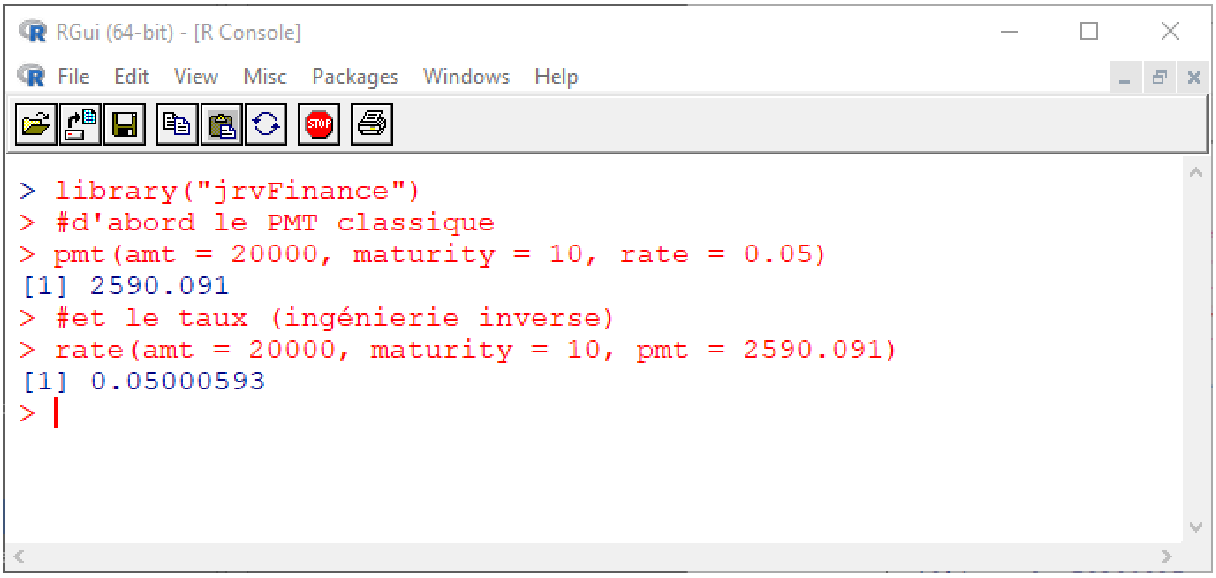
Pour cela, allons utiliser les fonctions **PMT( )**, **IPMT( )**, **PPMT( )** et **RBAL( )** du package DescTools:



```
> library("DescTools")
> cbind(year = 1:10,
+ payment    = PMT(rate=0.05, nper=10, pv=20000, fv=0, type=0),
+ interest   = IPMT(rate=0.05, per=1:10, nper=10, pv=20000, fv=0, type=0),
+ principal = PPMT(rate=0.05, per=1:10, nper=10, pv=20000, fv=0, type=0),
+ cumprincipal = cumsum(PPMT(rate=0.05, per=1:10, nper=10, pv=20000, fv=0, type=0)),
+ balance    = RBAL(rate=0.05, per=1:10, nper=10, pv=20000, fv=0, type=0))
      year   payment   interest principal cumprincipal   balance
 [1,]    1 -2590.091 -1000.0000 -1590.091    -1590.091 18409.909
 [2,]    2 -2590.091  -920.4954 -1669.596    -3259.688 16740.312
 [3,]    3 -2590.091  -837.0156 -1753.076    -5012.763 14987.237
 [4,]    4 -2590.091  -749.3618 -1840.730    -6853.493 13146.507
 [5,]    5 -2590.091  -657.3253 -1932.766    -8786.259 11213.741
 [6,]    6 -2590.091  -560.6870 -2029.404   -10815.664  9184.336
 [7,]    7 -2590.091  -459.2168 -2130.875   -12946.538  7053.462
 [8,]    8 -2590.091  -352.6731 -2237.418   -15183.957  4816.043
 [9,]    9 -2590.091  -240.8022 -2349.289   -17533.246  2466.754
[10,]   10 -2590.091  -123.3377 -2466.754   -20000.000     0.000
>
```

Nous obtenons donc bien le même résultat que dans le cours théorique et qu'avec un tableur.

Sinon, concernant le fait de trouver le taux à partir des paiements, de la valeur initiale et finale, je n'ai rien trouvé dans le package **DescTools** (mais j'ai peut-être mal cherché!). Par contre le package **jrvFinance** a la fonction **rate( )** adéquate!:

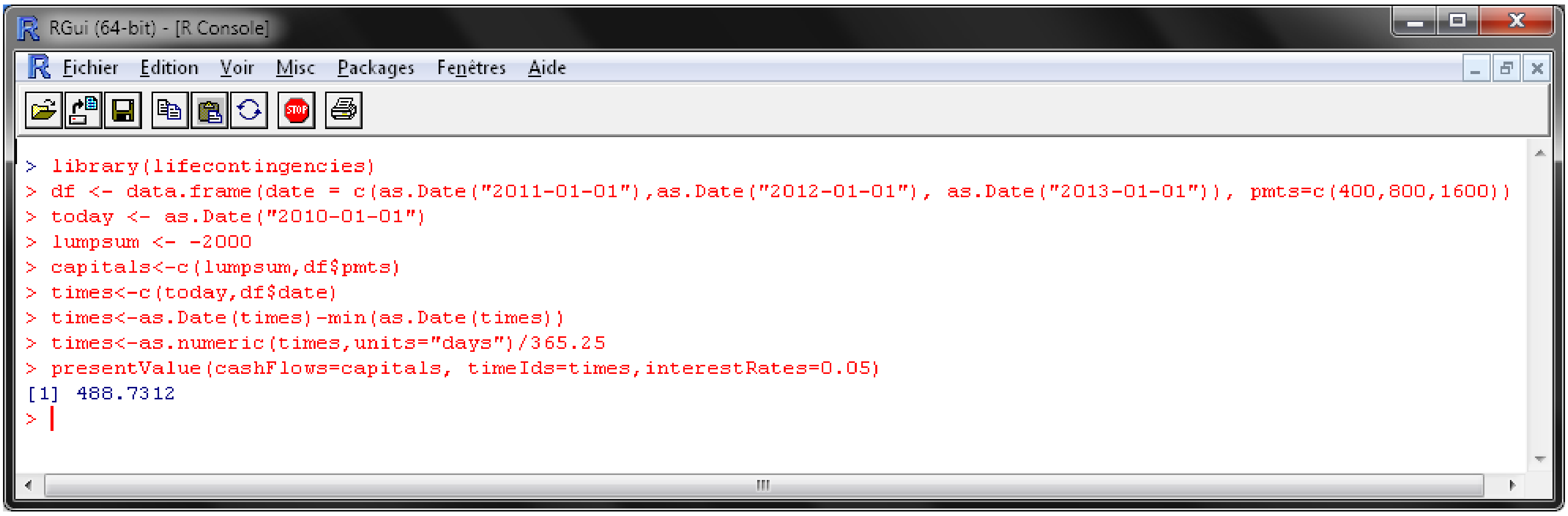


```
> library("jrvFinance")
> #d'abord le PMT classique
> pmt(amt = 20000, maturity = 10, rate = 0.05)
[1] 2590.091
> #et le taux (ingénierie inverse)
> rate(amt = 20000, maturity = 10, pmt = 2590.091)
[1] 0.05000593
>
```

Exemple 412.: Valeur actuelle nette (VAN/NPV) et Taux de rendement interne (IRR) périodiques

R 3.1.2

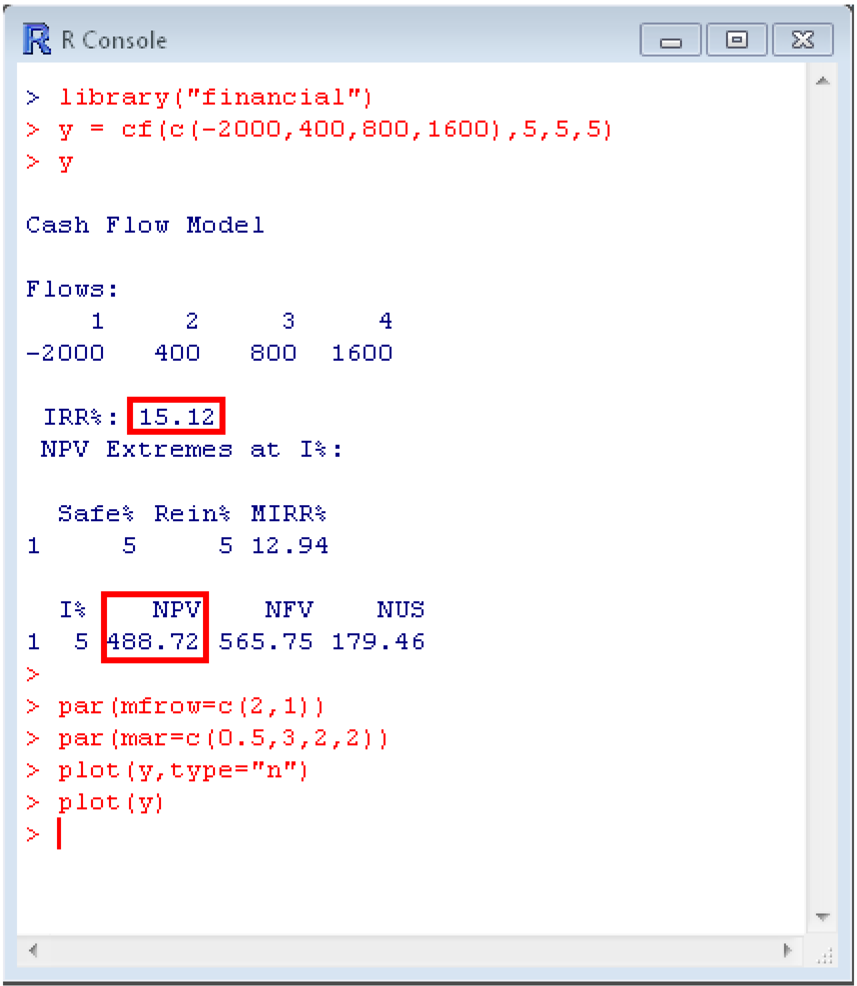
Bon je préfère utiliser le tableur Microsoft Excel pour cela mais suite à la demande d'un participant à une de mes formations voici comment calculer le même VAN que dans le cours théorique et qu'avec un tableur en utilisant la fonction **presentValue( )** :



Nous obtenons donc bien le même résultat que dans le cours théorique et qu'avec un tableur.

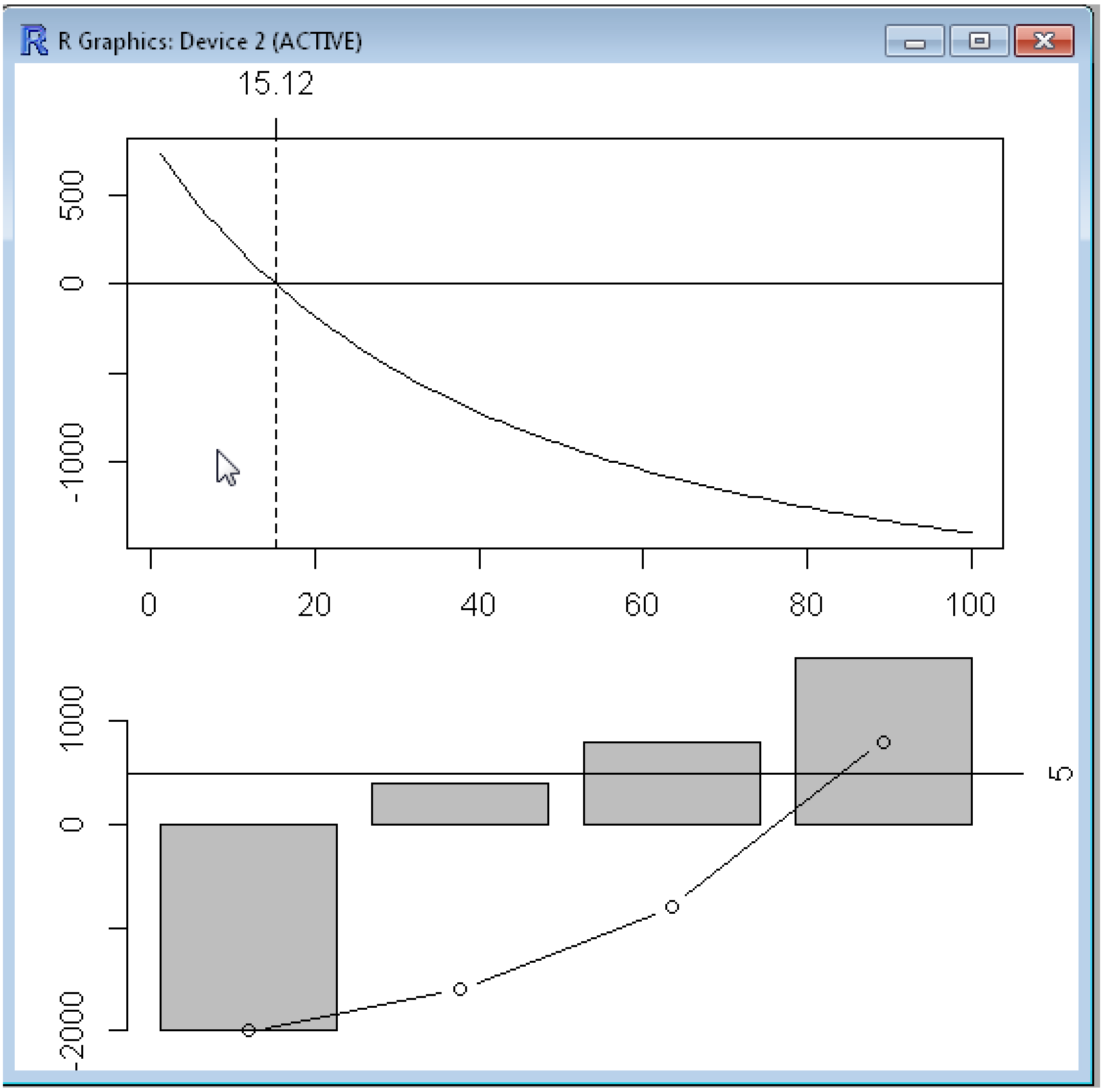
Le lecteur pourra se référer au package tvm présenté plus bas pour une version plus simple d'implémentation!

Malheureusement ce package ne contient pas de fonction pour le calculer de retour sur investissement. Donc voici une autre approche avec le package **financial** et la fonction **cf( )** où cf est l'abréviation de "cash-flows" mais qui malheureusement ne gère que les cas parfaitement périodiques:



Où nous retrouvons bien la valeur de NFV (Net Future Value) vue calculée dans le cours théorique!

Et avec les graphiques:



Nous pouvons réobtenir les résultats ci-dessus avec le package **tvm** et ses fonctions **npv( )** et **irr( )**:

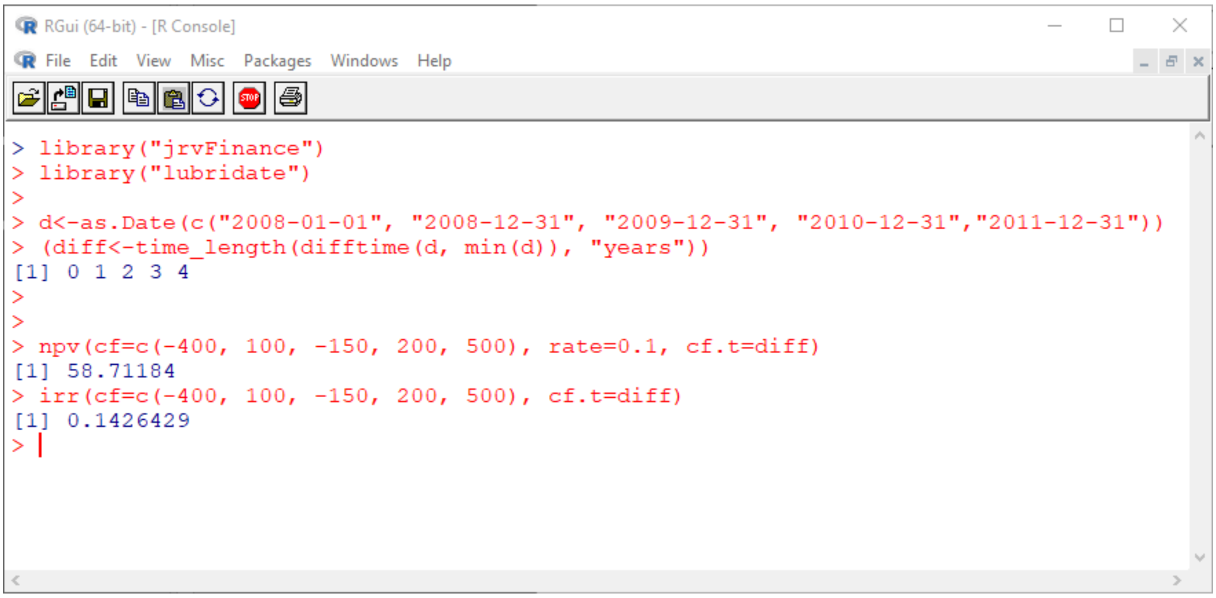
```
> library("tvm")
> npv(i = 0.05, cf = c(-2000, 400, 800, 1600), ts = c(0, 1, 2, 3))
[1] 488.7161
> irr(cf = c(-2000, 400, 800, 1600), ts = c(0, 1, 2, 3))
[1] 0.1511656
>
```

Exemple 413.: Valeur actuelle nette (VAN/NPV) et Taux de rendement interne (IRR) apériodiques

R 3.6.1

Le but va être ici de reproduire les calculs faits à la main dans le cours théorique et dans Microsoft Excel.

On utilise les fonctions **npv( )** et **irr( )** du package jrvFinance qui vont faire l'équivalent des fonctions XNPV( ) et XIRR( ) de Microsoft Excel:

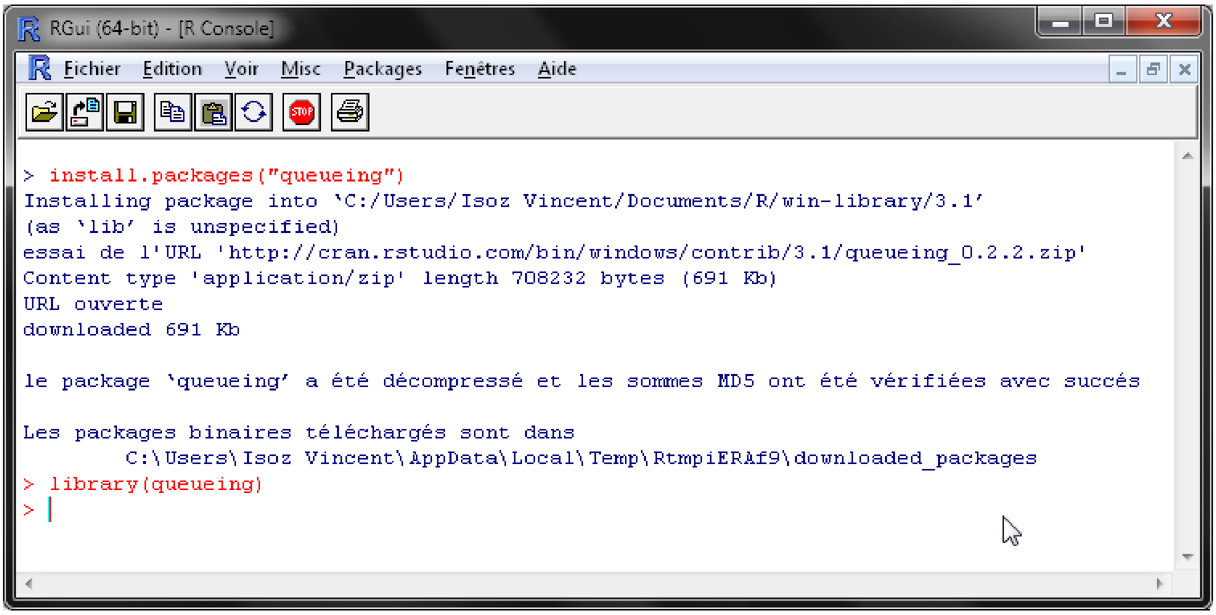


On retrouve bien la même chose que les calculs faits à la main et que dans Microsoft Excel!

28. Théorie des files d'attentes et simulations d'événements discrets

Le but ici va être de vérifier ici les quelques résultats importants obtenus dans le cours théorique vis à vis de la théorie des files d'attentes utilisée de façon importante dans la production, la logistique, la circulation ou autres domaines où l'on trouve des files d'attentes que l'on doit optimiser et qui satisfont les hypothèses des théorèmes démontrés dans le cours théorique.

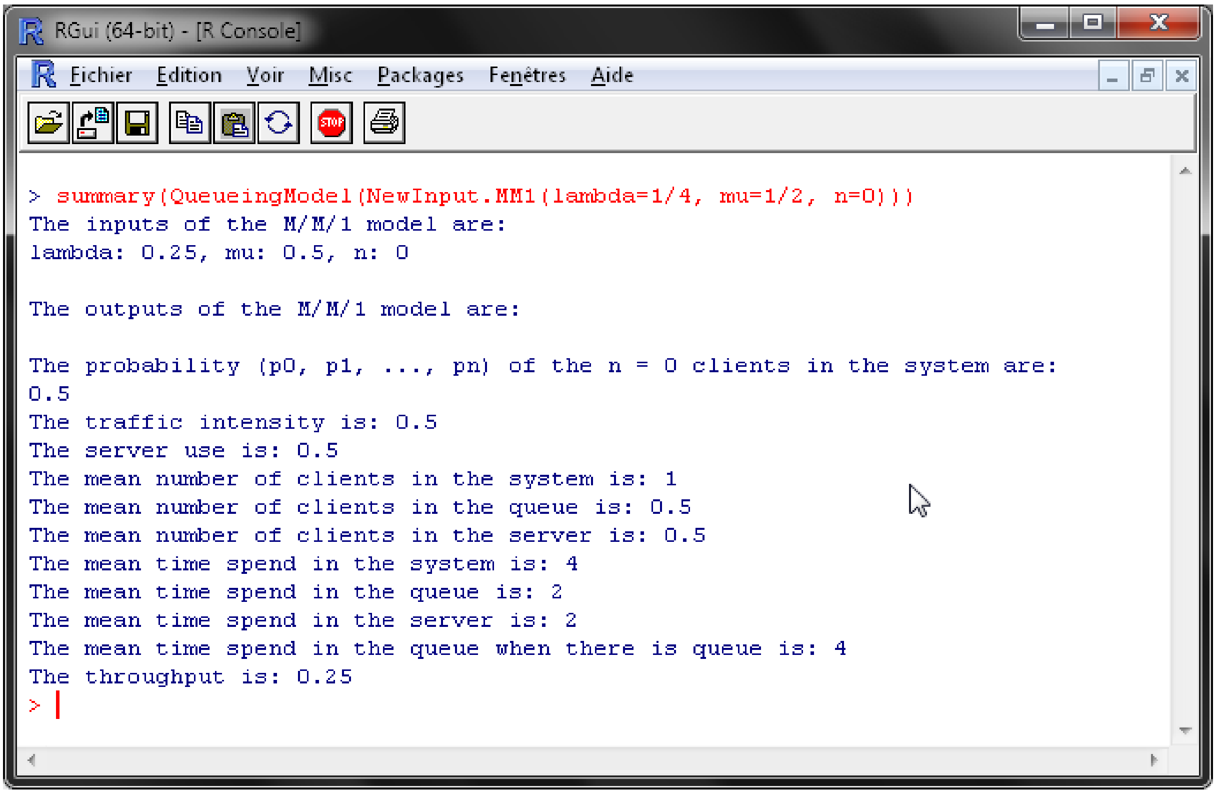
Nous allons pour cela installer le package **queueing**:



Exemple 414.: Résumé d'une file d'attente M/M/1

R 3.0.2

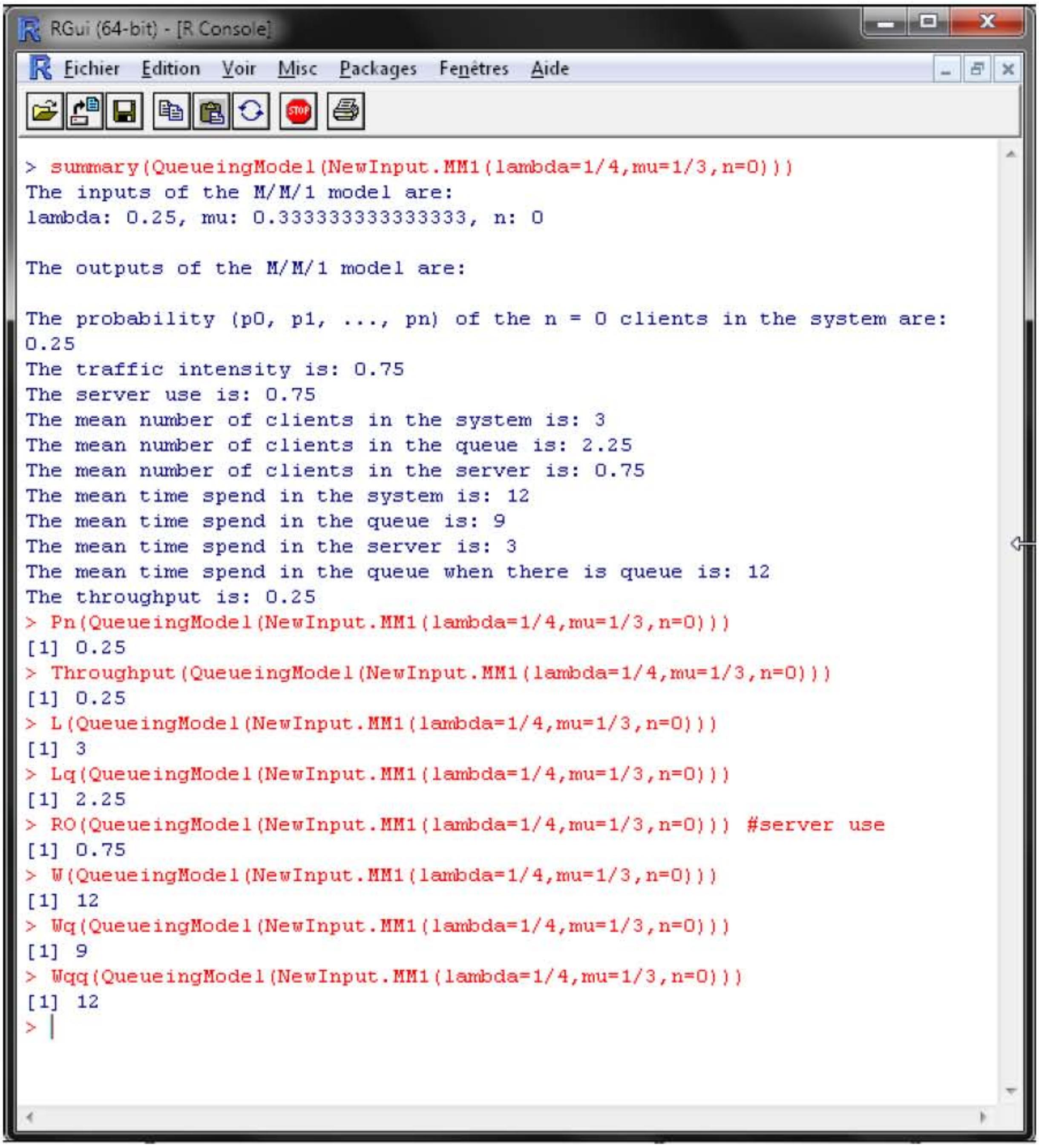
Le package **queuing** a une commande résumant toutes les paramètres d'une file d'attente M/M/1 tel que nous l'avons démontré dans le cours théorique en utilisant les commandes **QueueingModel( )** et **NewInput.MM1( )** :



Nous retrouvons bien les valeurs du cours théorique.

Après nous pouvons faire cela pour de nombreuses autres files d'attentes avec ce package mais nous nous limiterons pour l'instant à la M/M/1 car c'est la seule dont nous avons démontré les propriétés mathématiques en détails.

Nous pouvons extraire les valeurs indépendamment les unes des autres (si nous avons besoin de faire des graphiques par exemple):


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> summary(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
The inputs of the M/M/1 model are:
lambda: 0.25, mu: 0.333333333333333, n: 0
The outputs of the M/M/1 model are:
The probability (p0, p1, ..., pn) of the n = 0 clients in the system are:
0.25
The traffic intensity is: 0.75
The server use is: 0.75
The mean number of clients in the system is: 3
The mean number of clients in the queue is: 2.25
The mean number of clients in the server is: 0.75
The mean time spend in the system is: 12
The mean time spend in the queue is: 9
The mean time spend in the server is: 3
The mean time spend in the queue when there is queue is: 12
The throughput is: 0.25
> Pn(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 0.25
> Throughput(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 0.25
> L(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 3
> Lq(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 2.25
> RO(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0))) #server use
[1] 0.75
> W(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 12
> Wq(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 9
> Wqq(QueueingModel(NewInput.MM1(lambda=1/4,mu=1/3,n=0)))
[1] 12
>

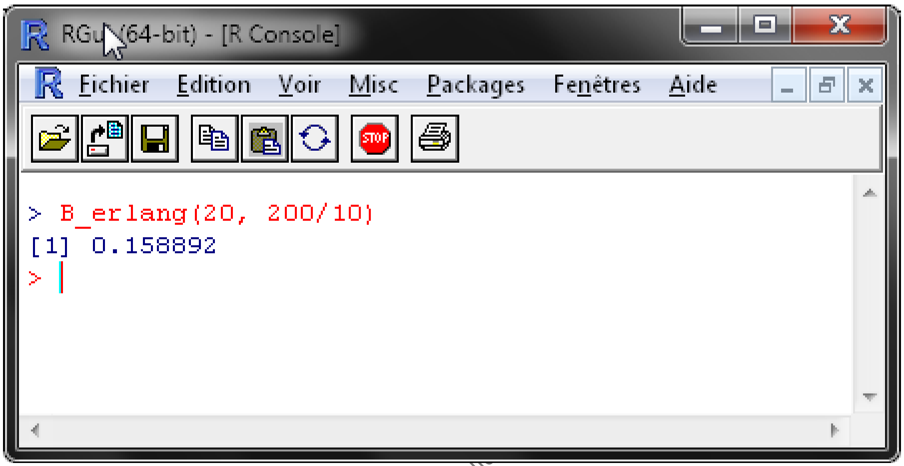
Exemple 415.: Probabilité d'Erlang-B

R 3.0.2

Rappelons l'exercice du cours après avoir démontré la relation mathématique d'Erlang-B:

Dans une entreprise, nous avons dénombré aux heures de pointe 200 appels d'une durée moyenne de 6 minutes à l'heure (temps de service moyen). Quelle est la probabilité de saturation avec 20 opérateurs (sachant que la durée de service suit une loi exponentielle et la distribution des arrivées une loi de Poisson)?

Nous utilisons alors la commande **B_erlang( )** du package **queueing**:



Nous retrouvons exactement le même résultat!

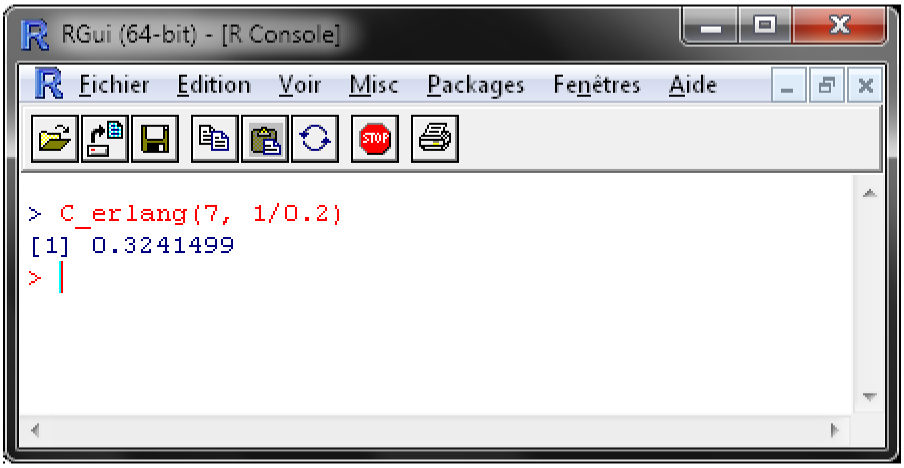
Exemple 416.: Probabilité d'Erlang-C

R 3.0.2

Rappelons l'exercice du cours après avoir démontré la relation mathématique d'Erlang-C:

Quelle est la probabilité d'être mis en attente si nous prenons un nombre N de 7 serveurs s'il y a un taux d'arrivée de 1 appel par minute et une durée moyenne de service de 5 minutes.

Nous utilisons alors la commande **C_erlang( )** du package **queueing**:



Exemple 417.: Simualtions d'événements discrets (D.E.S.)

R 4.0.5

L'exemple que nous allons voir ici est un simple copier/coller et retranscription en français de l'exemple crée par Mohamed Kamal:

https://www.linkedin.com/in/mohamedkamalpharm/

qui a simplement repris l'exemple de la vignette du package:

https://cran.r-project.org/web/packages/simmer/vignettes/simmer-01-introduction.html

et sa page web correspondante:

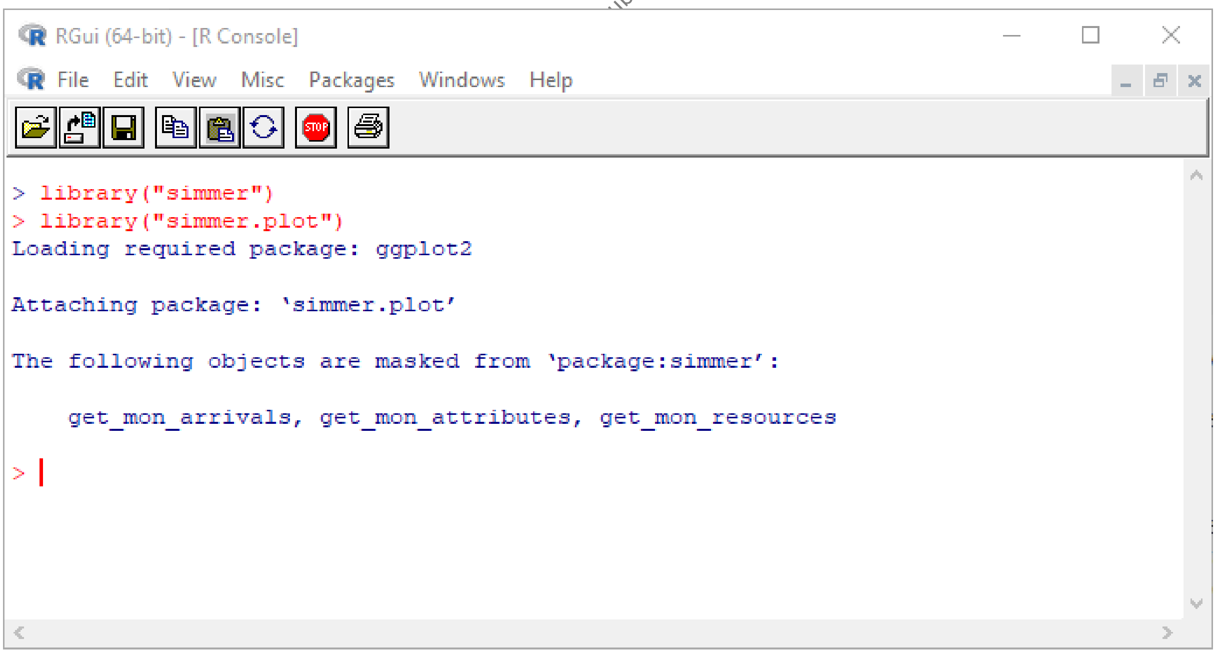
https://www.linkedin.com/pulse/discrete-event-simulation-using-r-oncology-hospital-example-kamal/

et sa chaîne Youtube:

https://www.youtube.com/c/StatPharm/videos

Le D.E.S. est donc un sytème physique qui subit des changements probabilistes à certain points précis dans le temps.

Chargeons les packages:

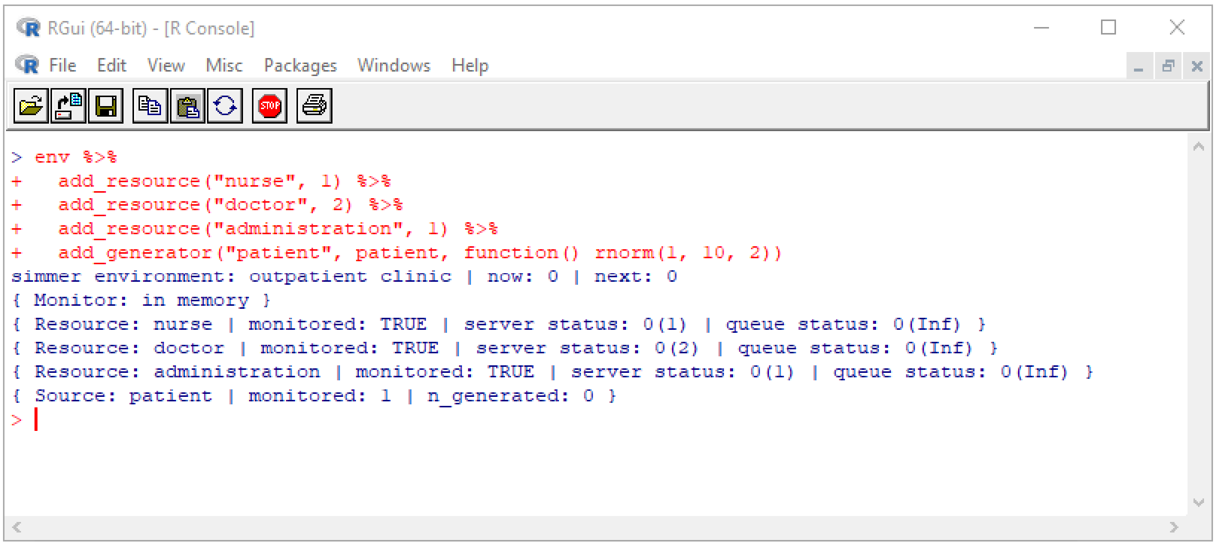


Mettons en place une trajectoire simple. Disons que nous voulons simuler une consultation ambulatoire où un patient est d'abord vu par une infirmière pour un accueil, ensuite par un médecin pour la consultation et enfin par le personnel administratif pour fixer un rendez-vous de suivi.

```
> patient <- trajectory("patients' path") %>%
+   ## add an intake activity
+   seize("nurse", 1) %>% #création d'un objet infirmière
+   #moyenne de traitement de 1 minute avec écart-type de 15 minutes (loi normale)
+   timeout(function() rnorm(1, 15)) %>%
+   release("nurse", 1) %>%
+   ## ajout de l'activité de consultation ou de traitement
+   seize("doctor", 1) %>%
+   #moyenne de traitement de 1 minute avec écart-type de 10 minutes (loi normale)
+   timeout(function() rnorm(1, 20)) %>%
+   release("doctor", 1) %>%
+   ## activité de planification
+   seize("administration", 1) %>%
+   #moyenne de traitement de 1 minute avec écart-type de 5 minutes (loi normale)
+   timeout(function() rnorm(1, 5)) %>%
+   release("administration", 1)
> patient
trajectory: patients' path, 9 activities
{ Activity: Seize        | resource: nurse, amount: 1 }
{ Activity: Timeout      | delay: function() }
{ Activity: Release      | resource: nurse, amount: 1 }
{ Activity: Seize        | resource: doctor, amount: 1 }
{ Activity: Timeout      | delay: function() }
{ Activity: Release      | resource: doctor, amount: 1 }
{ Activity: Seize        | resource: administration, amount: 1 }
{ Activity: Timeout      | delay: function() }
{ Activity: Release      | resource: administration, amount: 1 }
> str(patient)
Class 'trajectory' <environment: 0x000000003c2f24a8>
>
```

Dans ce cas, l'argument de l'activité *timeout* est une fonction, qui est évaluée dynamiquement pour produire un temps d'attente stochastique, mais cela pourrait aussi être une constante. En dehors de cela, cette fonction peut être aussi complexe que vous le souhaitez et peut faire ce que vous voulez : interagir avec les entités de votre modèle de simulation, obtenir l'état des ressources, prendre des décisions en fonction de ces dernières…

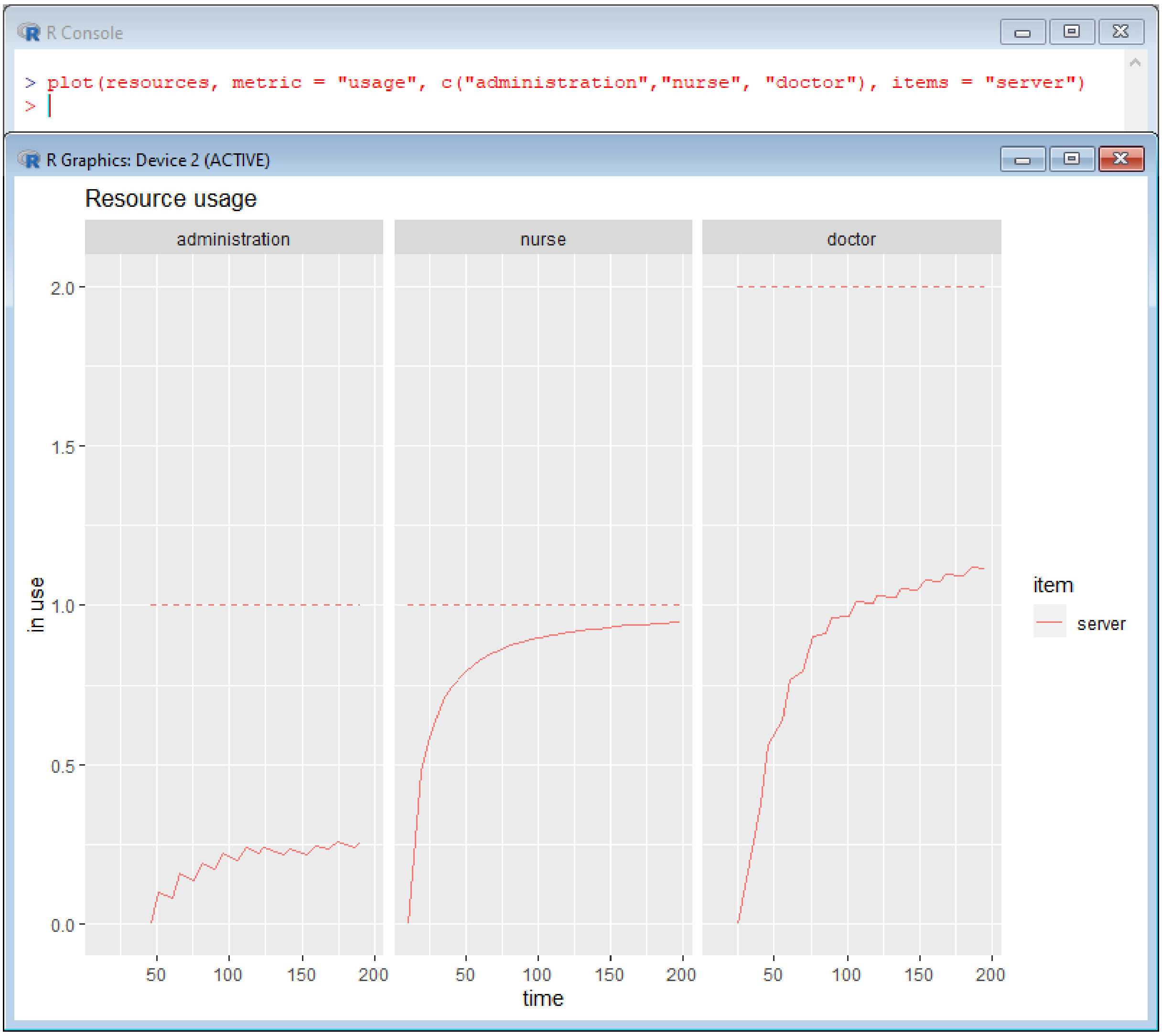
Une fois la trajectoire connue, vous pouvez y attacher des arrivées et définir les ressources nécessaires. Dans l'exemple ci-dessous, trois types de ressources sont ajoutées : les ressources infirmières et administratives, chacune d'une capacité de 1, et la ressource médecin, d'une capacité de 2. La dernière méthode ajoute un générateur d'arrivées (patients) suivant les patient de trajectoire. Le temps entre les patients est d'environ 10 minutes (une gaussienne de moyenne 10 minutes et écart-type de 2). (remarque : le renvoi d'un temps d'inter-arrivée négatif à un moment donné arrêterait le générateur):



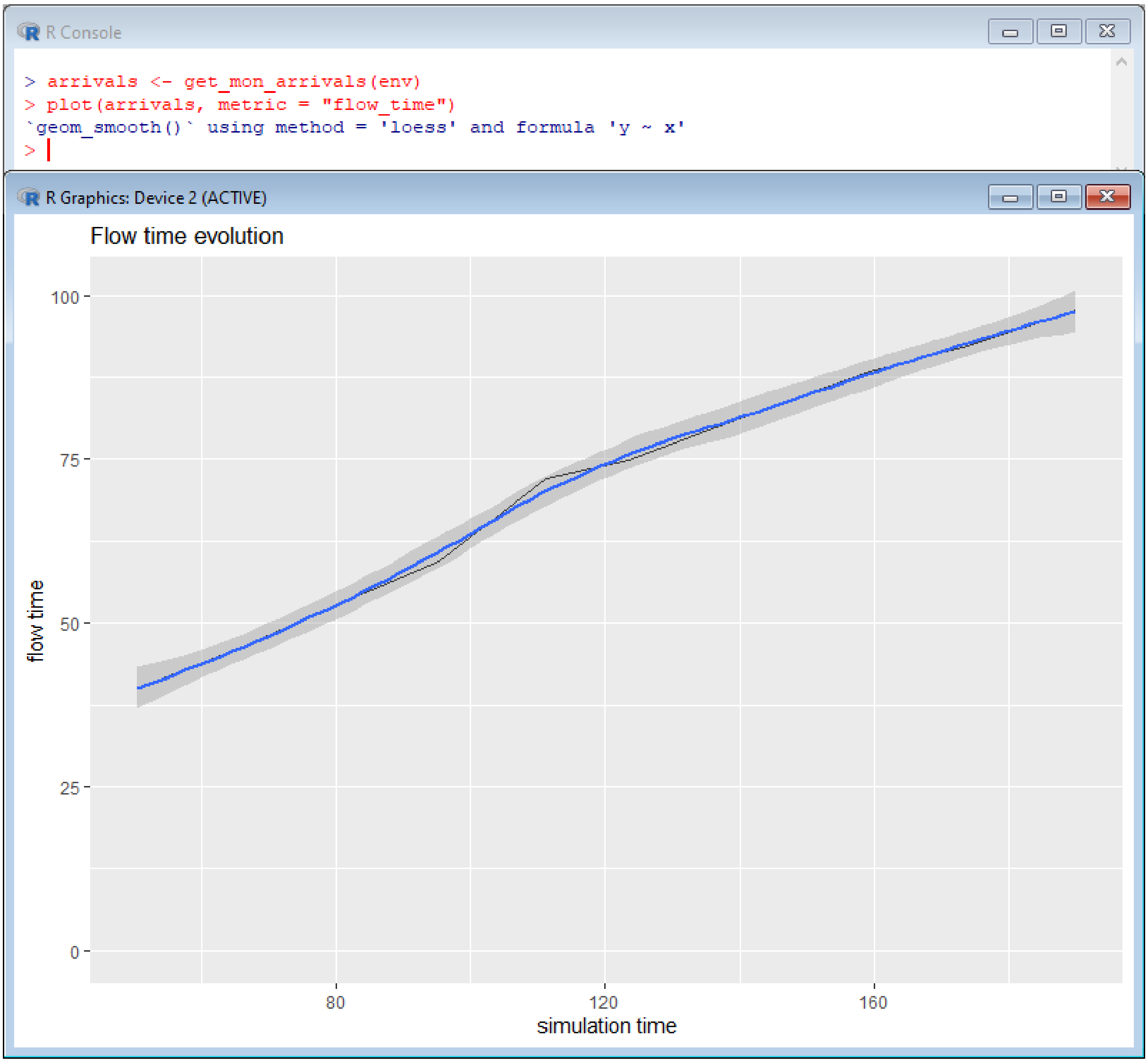
On peut lancer la simulation et voir le taux d'utilisation des ressources:



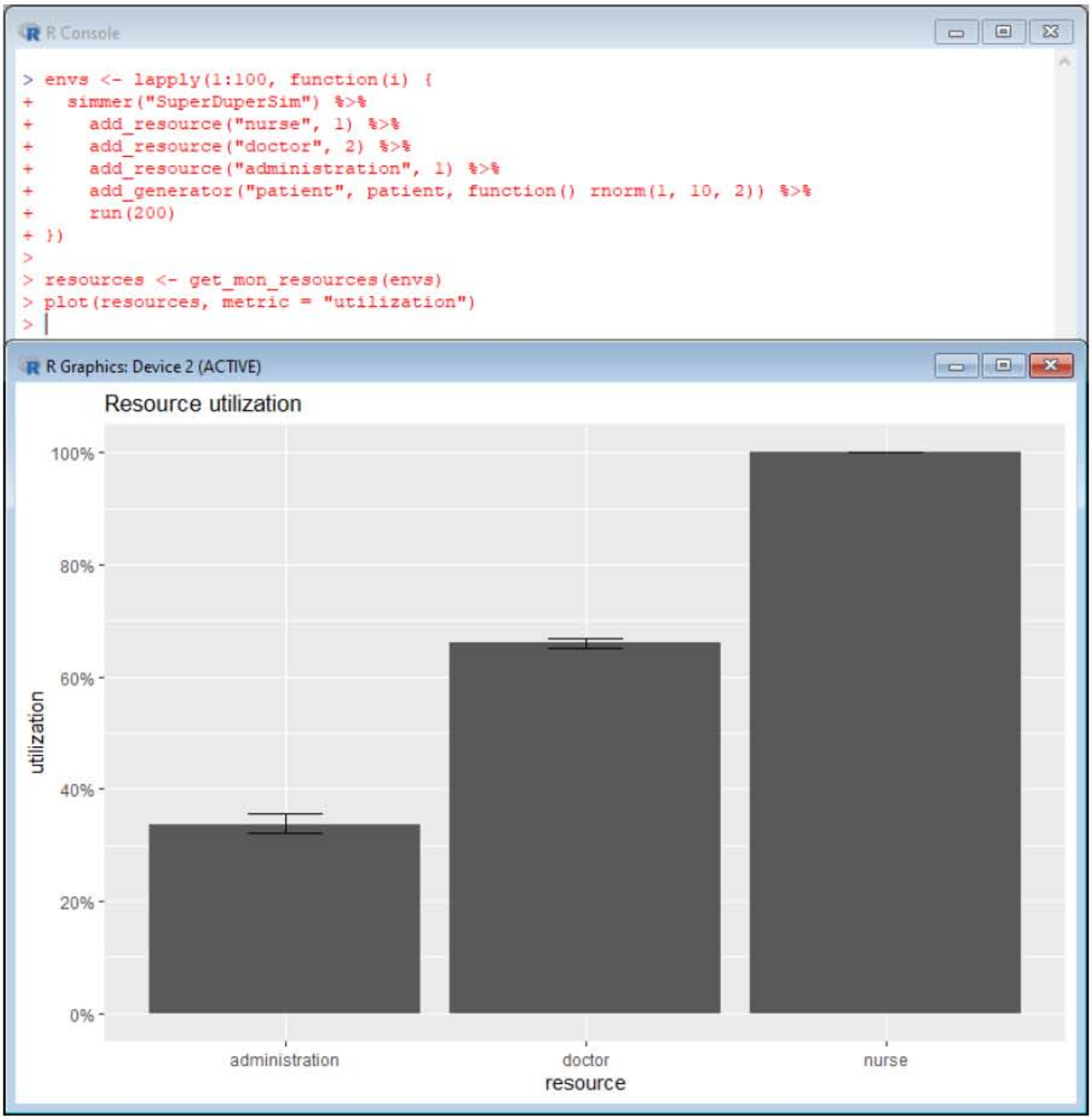
Il est également possible d'avoir un aperçu de l'activité des ressources pendant la simulation:

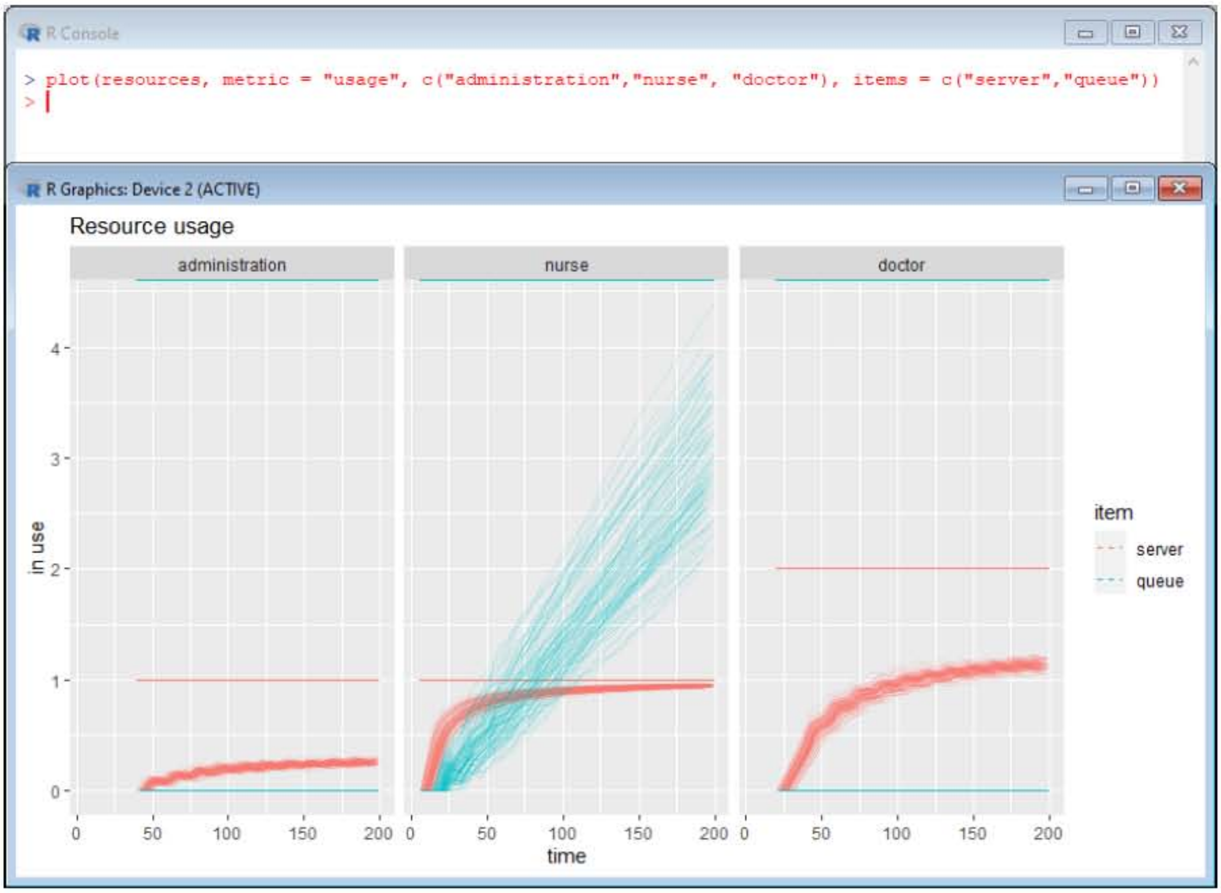


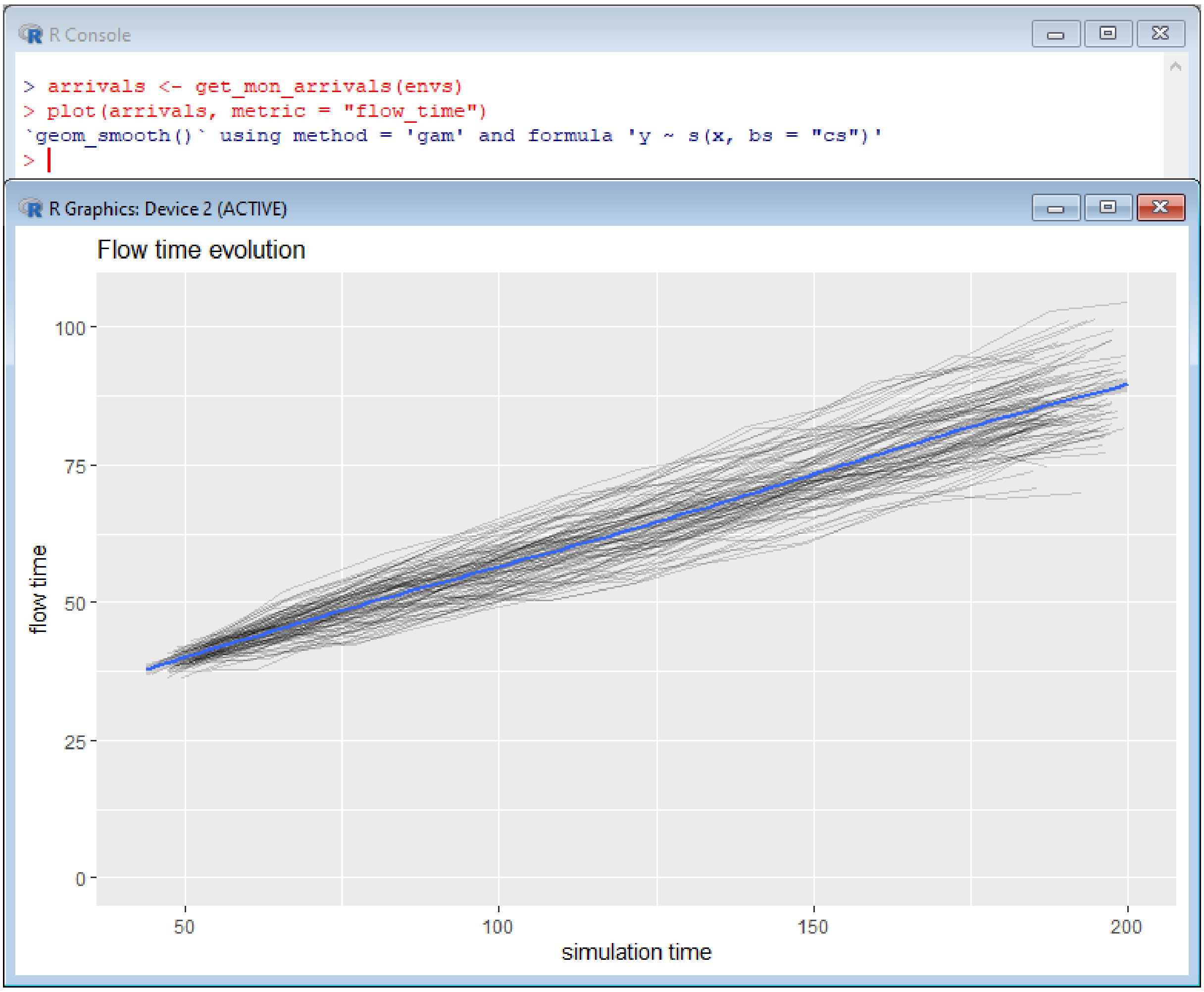
Ensuite, nous pouvons regarder l'évolution du temps d'écoulement des arrivées au cours de la simulation. Dans le graphique ci-dessous, chaque ligne individuelle représente une réplication. Une ligne lisse est tracée sur eux. Toutes les arrivées qui n'ont pas terminé toute leur trajectoire sont exclues du tracé.



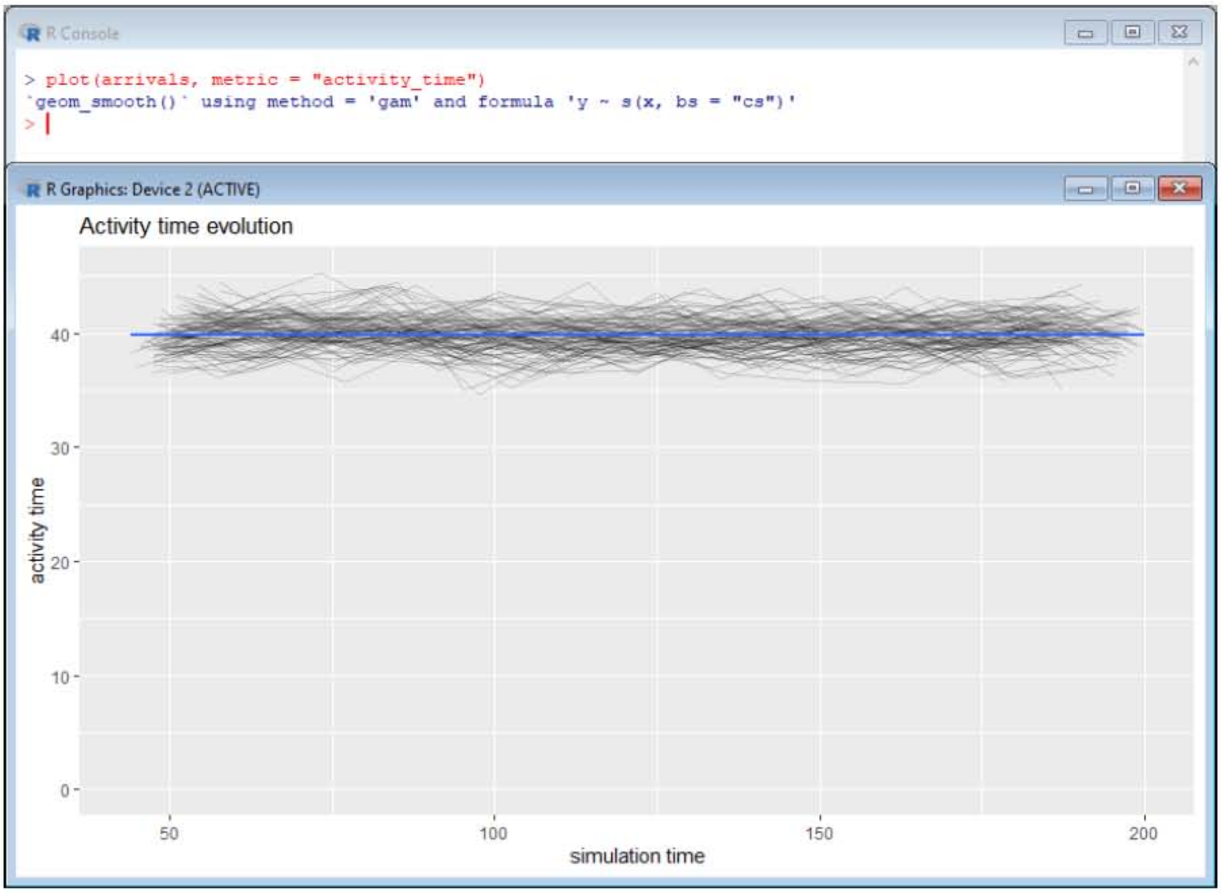
On peut faire les mêmes analyses sur un paquet de simulation (ce qui est scientifiquement plus objectif bien évidemment!):


R Console
> envs <- lapply(1:100, function(i) {
+   simmer("SuperDuperSim") %>%
+   add_resource("nurse", 1) %>%
+   add_resource("doctor", 2) %>%
+   add_resource("administration", 1) %>%
+   add_generator("patient", patient, function() rnorm(1, 10, 2)) %>%
+   run(200)
+ })
>
> resources <- get_mon_resources(envs)
> plot(resources, metric = "utilization")
>
R Graphics: Device 2 (ACTIVE)
Resource utilization
100%
80%
60%
40%
20%
0%
utilization
administration
doctor
nurse
resource

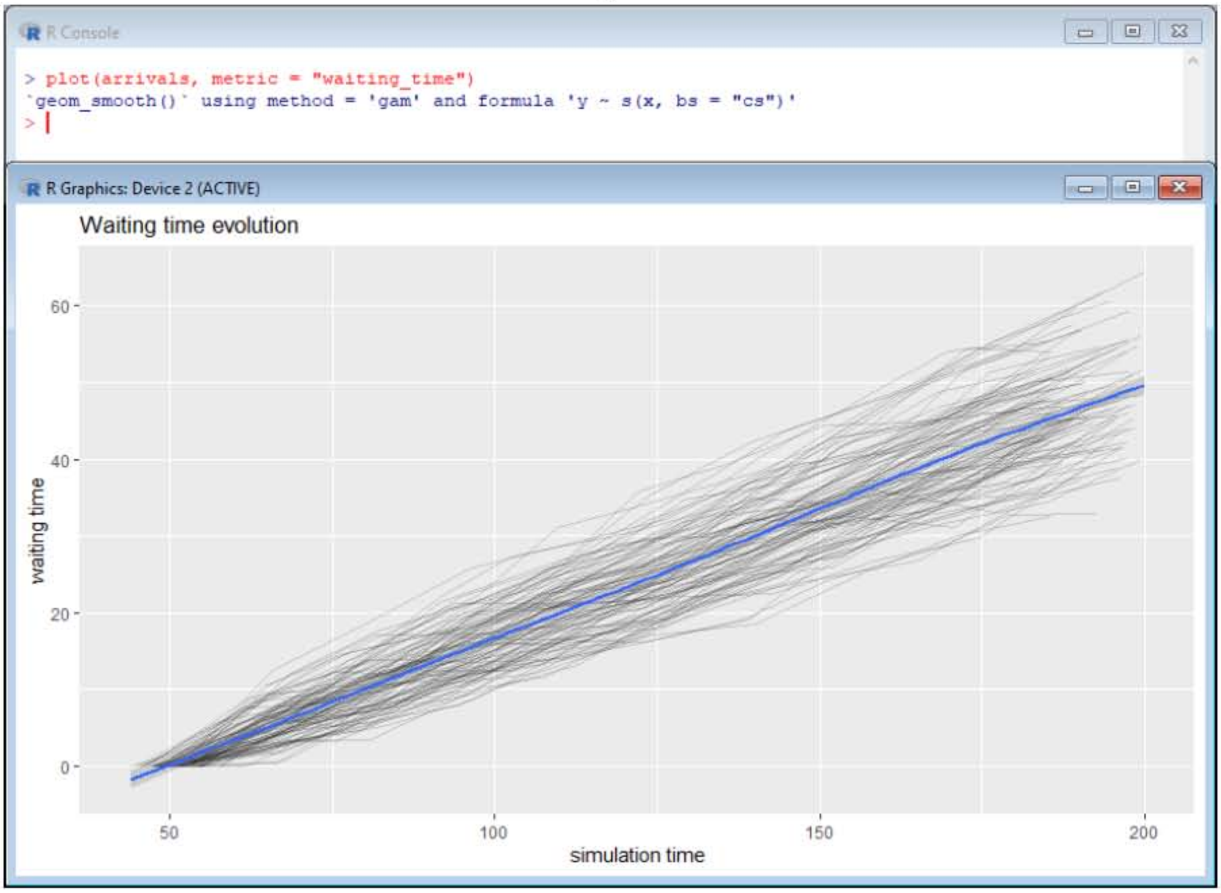

R Console
> plot(resources, metric = "usage", c("administration","nurse", "doctor"), items = c("server","queue"))
R Graphics: Device 2 (ACTIVE)
Resource usage
administration
nurse
doctor
in use
time
item
server
queue



Donc le temps de passage ne cesse d'augmenter de manière a peu près linéaire. Et:


R Console
> plot(arrivals, metric = "activity_time")
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
R Graphics: Device 2 (ACTIVE)
Activity time evolution
activity time
simulation time
0
10
20
30
40
50
100
150
200

Et:


R Console
> plot(arrivals, metric = "waiting_time")
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
R Graphics: Device 2 (ACTIVE)
Waiting time evolution
waiting time
simulation time
0
20
40
60
50
100
150
200

29. Statistiques bayésiennes

Le but ici n'est pas de redévelopper la discussion relativement au débat qui concerne les statisticiens fréquentistes contre les statistiques bayésiens mais uniquement de vérifier que nous retrouvons bien les quelques rares calculs bayésiens fait à la main dans la cours théorique ou ceux qui ont été simplement mentionnés avec suffisamment de détails mathématiques.

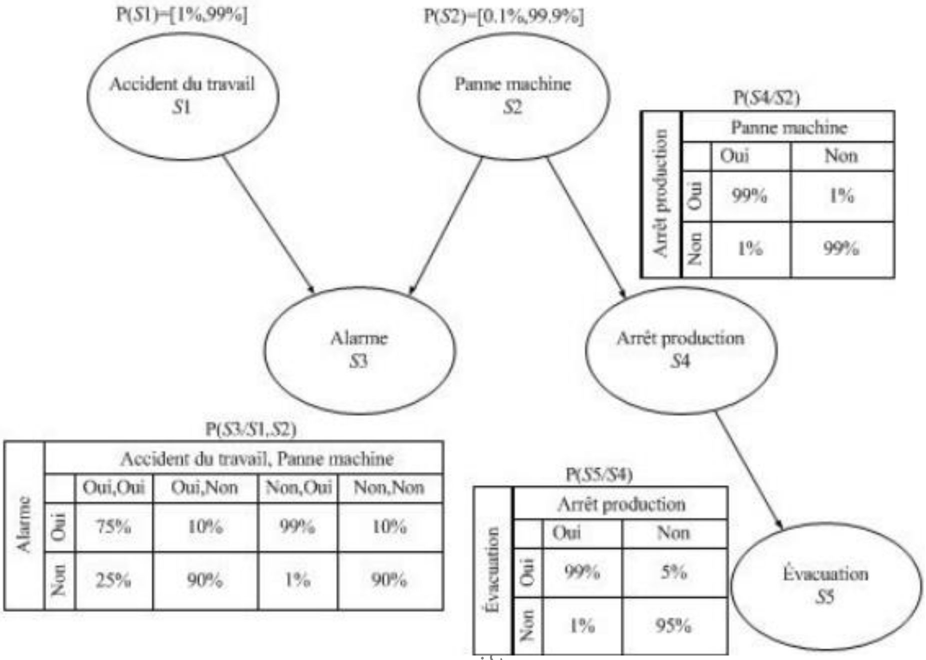
Exemple 418.: Classification bayésienne naive

Nous n'allons pas revenir sur la technique de classification bayésienne naïve vue en détails à la page 2142.

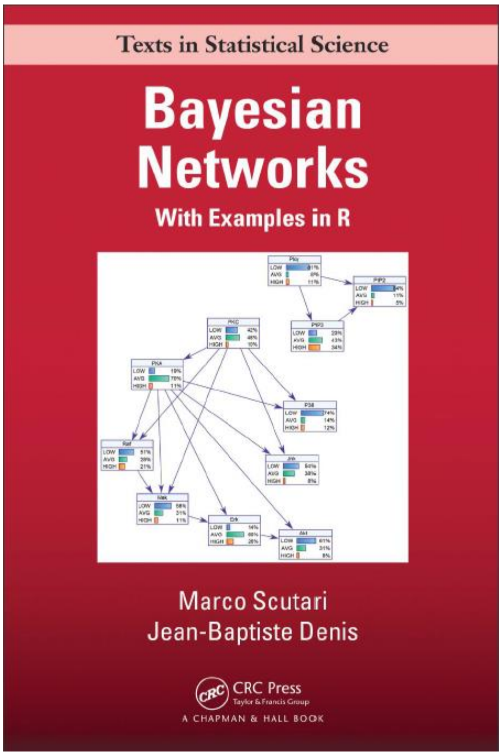
Exemple 419.: Réseaux Bayésiens

R 3.6.2

Nous allons ici reproduire l'exemple vu dans le cours théorique et comparer aux résultats obtenu avec le logiciel MSBN 1.4. Pour rappel le cas d'étude est résumé par:



Pour faire l'analyse avec R nous allons suivre la procédure de l'excellent livre et package associé (**bnlearn**) suivant:



Pour construire le réseau bayésian visible ci-dessus nous devons d'abord écire le script ci-dessous:

```
> library("bnlearn")
> #W est "Accident du Travail"
> #M est "Panne Machine"
> #A est "Alarme"
> #P est "Arrêt production"
> #E est "Évacuation"
>
> #On construit le graphe bayésien
> dag <- empty.graph(nodes = c("W", "M", "A", "P", "E"))
> dag

  Random/Generated Bayesian network

  model:
   [W][M][A][P][E]
  nodes:                                 5
  arcs:                                  0
    undirected arcs:                     0
    directed arcs:                       0
  average markov blanket size:           0.00
  average neighbourhood size:            0.00
  average branching factor:              0.00

  generation algorithm:                  Empty

> dag <- set.arc(dag, from = "W", to = "A")
> dag <- set.arc(dag, from = "M", to = "A")
> dag <- set.arc(dag, from = "M", to = "P")
> dag <- set.arc(dag, from = "P", to = "E")
> #on affichage la structure selon le package bnlearn
> dag

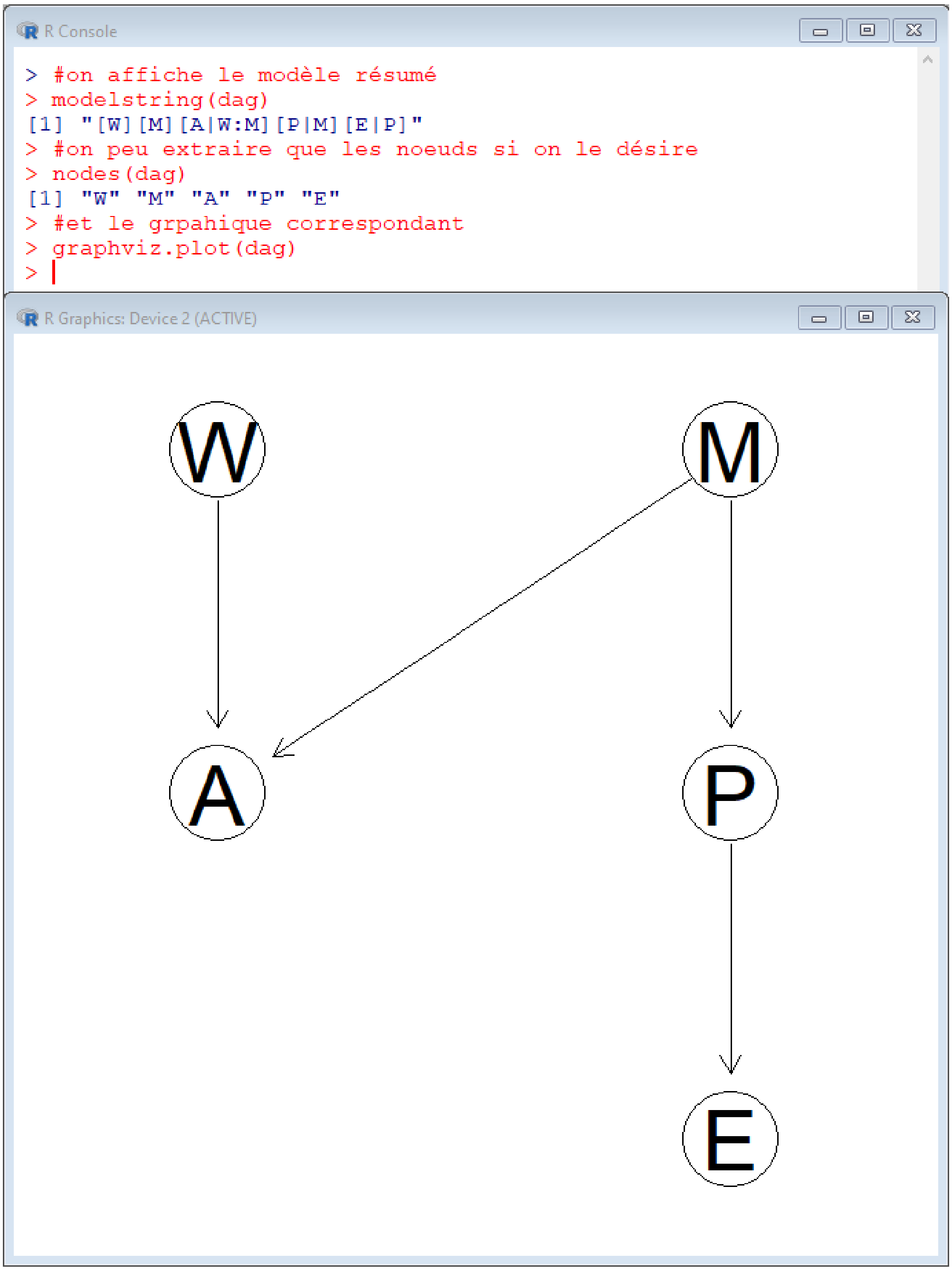
  Random/Generated Bayesian network

  model:
   [W][M][A|W:M][P|M][E|P]
  nodes:                                 5
  arcs:                                  4
    undirected arcs:                     0
    directed arcs:                       4
  average markov blanket size:           2.00
  average neighbourhood size:            1.60
  average branching factor:              0.80

  generation algorithm:                  Empty

>
```

On peut afficher le modèle sous forme de chaîne de caractère ainsi qu'un piètre visuel graphique du réseau bayésien correspondant:



Maintenant construisons les tables de probabilité (avec une subtilité difficile à devenir pour la dernière table sinon quoi cela ne marche pas[43]):

[43] Merci à 'l'auteur du package pour nous l'avoir indiqué

```
> #on construit les légendes des tables de probabilité
> W.lv <- c("Oui", "Non")
> M.lv <- c("Oui", "Non")
> A.lv <- c("Oui", "Non")
> P.lv <- c("Oui", "Non")
> E.lv <- c("Oui", "Non")
>
> #on construit les tables en y mettant les probabilités
> (W.prob <- array(c(0.01, 0.99), dim = 2, dimnames = list(W = W.lv)))
W
 Oui  Non
0.01 0.99
> (M.prob <- array(c(0.001, 0.999), dim = 2, dimnames = list(M = M.lv)))
M
  Oui   Non
0.001 0.999
> (P.prob <- array(c(0.99, 0.01, 0.01, 0.99), dim = c(2, 2),dimnames = list(P= P.lv, M = M.lv)))
     M
P      Oui  Non
  Oui 0.99 0.01
  Non 0.01 0.99
> (E.prob <- array(c(0.99, 0.01, 0.05, 0.95), dim = c(2, 2),
+ dimnames = list(E= E.lv, P = P.lv)))
     P
E      Oui  Non
  Oui 0.99 0.05
  Non 0.01 0.95
> (A.prob <- array(c(0.75, 0.99, 0.1, 0.1, 0.25, 0.01,0.9, 0.9), dim = c(2, 2, 2),
+ dimnames = list(W = W.lv, M = M.lv, A = A.lv)))
, , A = Oui

     M
W      Oui Non
  Oui 0.75 0.1
  Non 0.99 0.1

, , A = Non

     M
W      Oui Non
  Oui 0.25 0.9
  Non 0.01 0.9

> #on permute la matrice pour la rendre compatible avec les exigence du package bnlearn
> A.prob = aperm(A.prob, c(3, 2, 1))
>
```

Enfin on peut construire l'objet de réseau bayésien comme le nécessite la package **bnlearn**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> #on construit le réseau bayésien
> cpt <- list(W = W.prob, M = M.prob, P = P.prob, E = E.prob,A = A.prob)
> bn <- custom.fit(dag, cpt)
> #et on affichage de toutes les tables
> bn

  Bayesian network parameters

  Parameters of node W (multinomial distribution)

Conditional probability table:
 W
 Oui  Non
0.01 0.99

  Parameters of node M (multinomial distribution)

Conditional probability table:
 M
  Oui   Non
0.001 0.999

  Parameters of node A (multinomial distribution)

Conditional probability table:

, , M = Oui

     W
A      Oui  Non
  Oui 0.75 0.99
  Non 0.25 0.01

, , M = Non

     W
A      Oui  Non
  Oui 0.10 0.10
  Non 0.90 0.90


  Parameters of node P (multinomial distribution)

Conditional probability table:

     M
```

Et maintenant on calcule toutes les probabilités conditionnelles a posteriori (les mêmes que dans le cours théorique) mais en remarquant que l'auteur, dans un souci de généralisation, traite le calcul vie des simulation de Monte Carlo (on sait bien évidemment pourquoi!):

```
RGui (64-bit) - [R Console]
File  Edit  View  Misc  Packages  Windows  Help

> #on fait tous les calculs avec 1 milliard de simulations de Monte Carlo
> #Probabilité a posteriori d'un arrêt production à cause d'une panne machine
> #Calcul à la main: P(S2/S4)=9.02%
> cpquery(bn, event = (M == "Oui"), evidence = (P == "Oui"),n = 10^9)*100
[1] 9.039312
>
> #Probabilité a posteriori d'une évacuation due à un arrêt de production
> #Calcul à la main: P(S4/S5)=18.02%
> cpquery(bn, event = (P == "Oui"), evidence = (E == "Oui"),n = 10^9)*100
[1] 18.01475
>
> #Probabilité a posteriori d'une alarme sachant qu'il y a eu une panne machine
> #Calcul à la main: P(S3/S2)=98.76%
> cpquery(bn, event = (A == "Oui"), evidence = (M == "Oui"),n = 10^9)*100
[1] 98.75191
>
> #Probabilitéa posteriori d'une panne machine sachant qu'il y a eu une alarme
> #Calcul à la main: P(S2/S3)=0.9789%
> cpquery(bn, event = (M == "Oui"), evidence = (A == "Oui"),n = 10^9)*100
[1] 0.9786511
>
> #Probabilitéa posteriori d'un accident du travail sachant qu'il y a eu une alarme
> #Calcul à la main: P(S1/S3)=0.9976%
> cpquery(bn, event = (W == "Oui"), evidence = (A == "Oui"),n = 10^9)*100
[1] 0.9986268
>
```

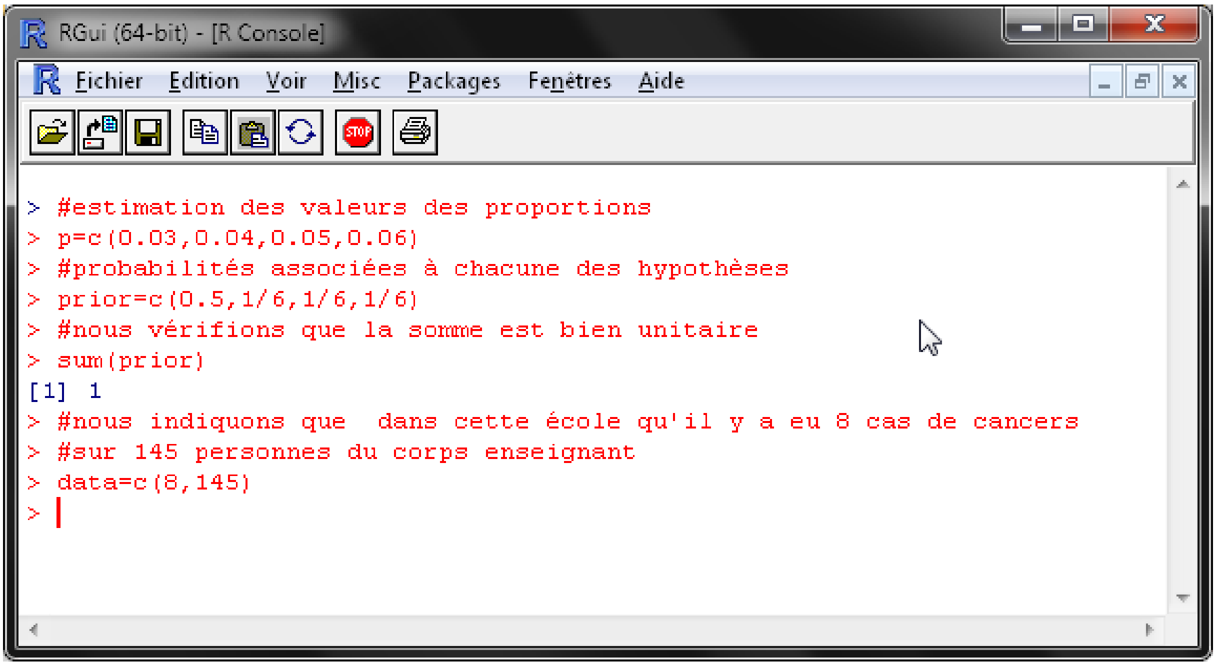
Nous retrouvons donc bien les valeurs calculées à la main pour les probabilités conditionnelles a posteriori. Par contre nous n'avons pas trouvé dans le package **bnlearn** de méthode pour calculer les constantes de normalisation.

Exemple 420.: Statistiques bayésiennes discrètes (cas d'une proportion)

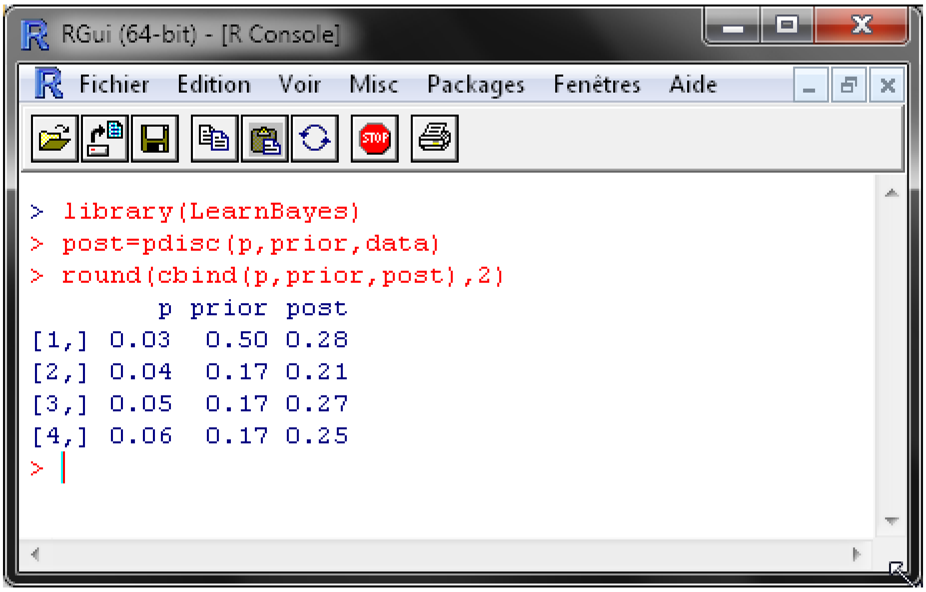
R 3.0.2

Nous allons donc reprendre ici le même calcul dans le cours théorique basée sur l'étude des cancers dans une école américaine a proximité de lignes haute tension.

Nous commençons par définir ce que nous savons et les hypothèses a priori:



Nous utilisons ensuite la commande **pdisc( )** du package **LearnBayes** pour calculer les probabilités a posteriori:



Nous retrouvons donc bien les probabilités discrètes a posteriori calculées à la main dans le cours théorique.

Exemple 421.: Statistiques bayésiennes continues (cas d'estimation de la moyenne avec a priori et posteriori Normales)

R 3.0.2

Nous avons démontré dans le cours théorique que si la distribution a priori est une distribution Normale d'espérance μ_0 et de variance τ_0^2 et que la distribution du maximum de vraisemblance est Normale, alors la fonction de distribution a posteriori aussi Normale d'espérance a posteriori:

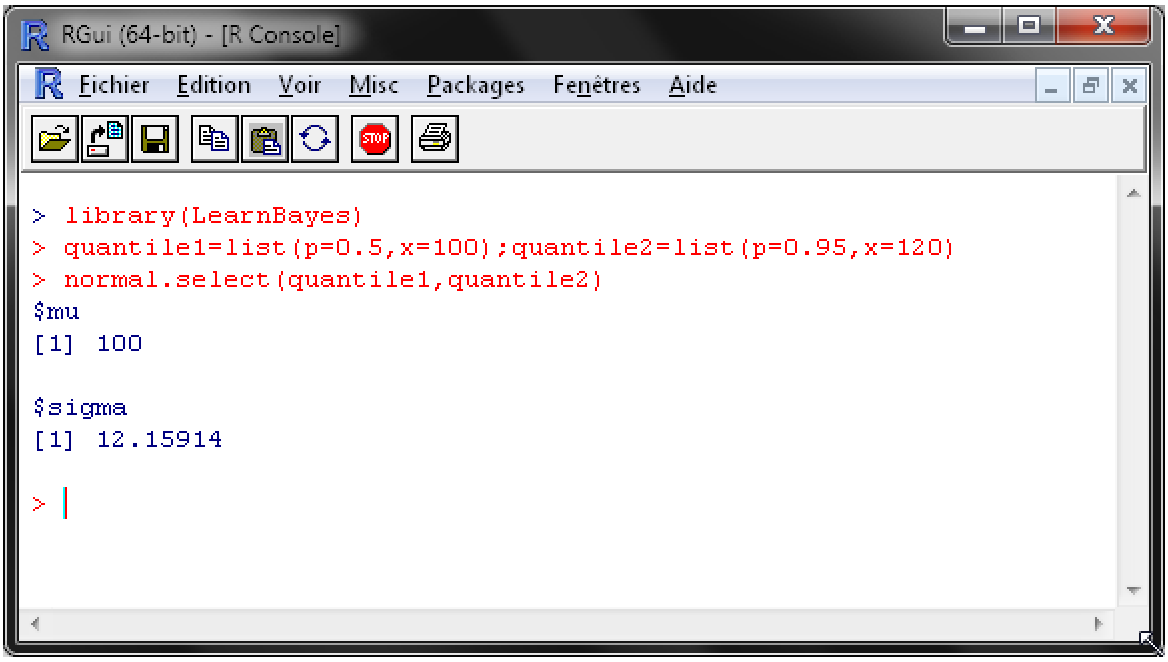
$$\mu_1 = \frac{\dfrac{\mu_0}{\tau_0^2} + \dfrac{n\bar{y}}{\sigma^2}}{\dfrac{1}{\tau_0^2} + \dfrac{n}{\sigma^2}}$$

et de variance a posteriori:

$$\tau_1^2 = \left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2} \right)^{-1}$$

Supposons que nous souhaitons mesurer le quotient intellectuel d'un group d'individus. Nous avons une fort prémonition que a priori il y a 50% de probablité cumulée (médiane) que son QI soit de 100 et qu'il y a 9/10 que celui-ci soit compris entre 80 et 120 (ces informations peuvent aussi être basées sur des données historiques passées).

Donc à l'aide du package **LearnBayes** et de la commande **normal.select( )** nous pouvons déjà déterminer la distribution Normal la mieux ajustée à ses données a priori:



Notre groupe test composé de 4 personnes passe donc 4 tests de QI qui nous donnent les informations suivantes:

$$n = 4, \bar{y} = 110$$

et que nous savons qu'au niveau mondial la variance vraie est donnée par $\sigma^2 = 15$.

Nous avons au final comme informations:

$$\begin{aligned} &n = 4 \\ &\bar{y} = 110 \\ &\sigma^2 = 15 \\ &\mu_0 = 100 \\ &\tau_0^2 = 12.159 \end{aligned}$$

Le calcul de l'espérance a posteriori est l'écart-type a posteriori est alors immédiate en transcrivant les relations suivantes dans R:

$$\mu_1 = \frac{\dfrac{\mu_0}{\tau_0^2} + \dfrac{n\bar{y}}{\sigma^2}}{\dfrac{1}{\tau_0^2} + \dfrac{n}{\sigma^2}} = 107.2442, \tau_1 = \sqrt{\left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)^{-1}} = 6.383469$$

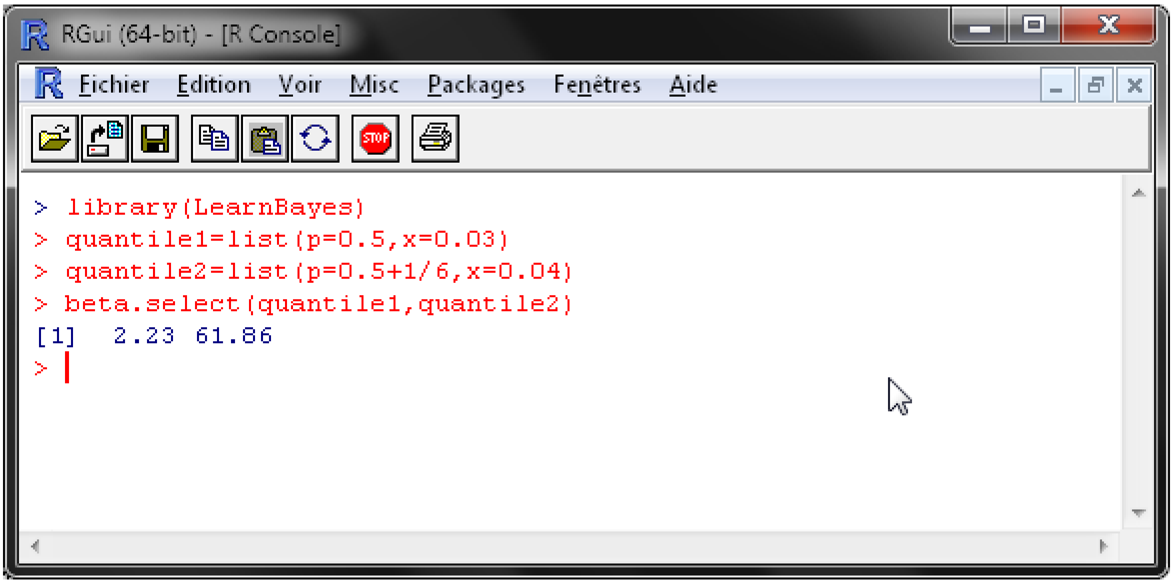
Nous avons donc les paramètres de la loi Normale a posteriori!

Exemple 422.: Statistiques bayésiennes continues (cas d'une proportion avec a priori beta estimée et a posteriori binomiale)

R 3.0.2

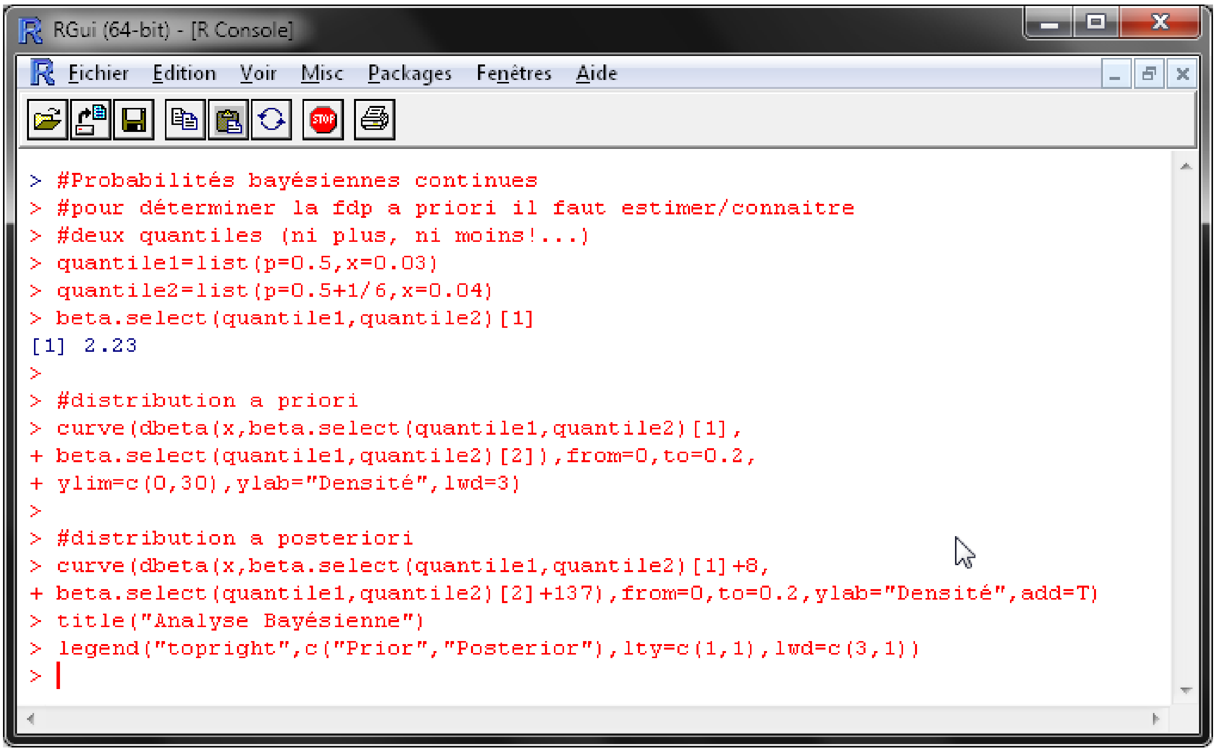
L'idée ici va être d'estimer a priori la fonction de distribution de la proportion du cas discret mais de façon continue. Comme il n'y a pas des milliers de fonctions de distributions dont le support est compris entre]0,1[nous allons nous rabattre sur la seule que nous avons rencontrée dans le cours théorique: la loi bêta.

Nous avons alors en faisant l'hypothèse que la médiane de la proportion de cancers est de 3% et que le 66 centile est de 4% les paramètres de la fonction beta qui seront donnés par la commande **beta.select( )** du package **LearnBayes**:

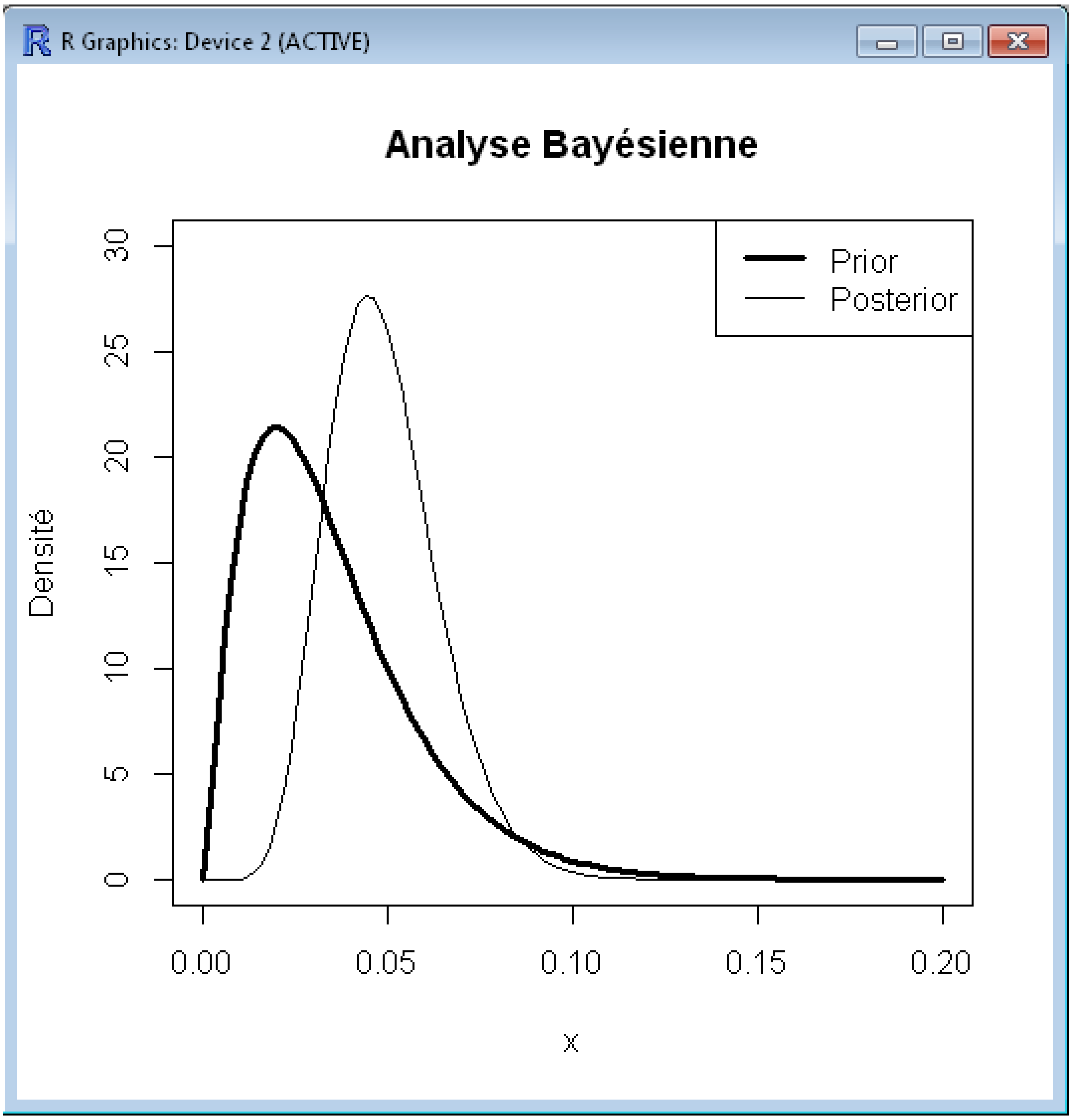


Nous avons démontré dans le cours théorique que si la distribution a priori est une distribution beta et que la distribution du maximum de vraisemblance est binomiale, alors la fonction de distribution a posteriori est une distribution binomiale aussi avec les paramètres qui changent un tout petit peu.

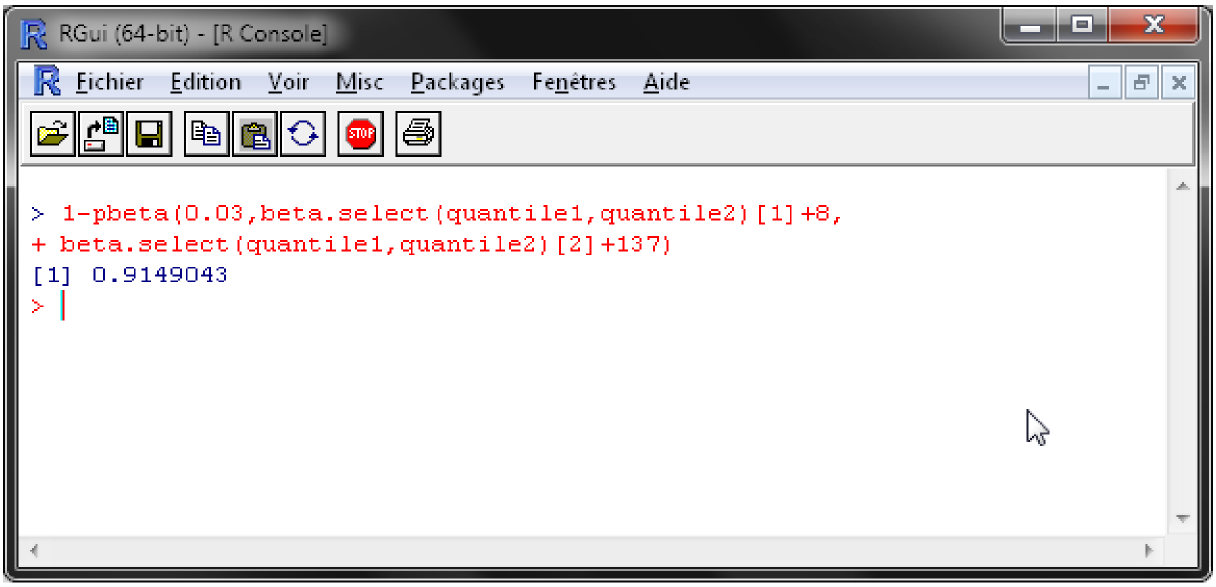
Nous avons alors:



Ce qui donne graphiquement:

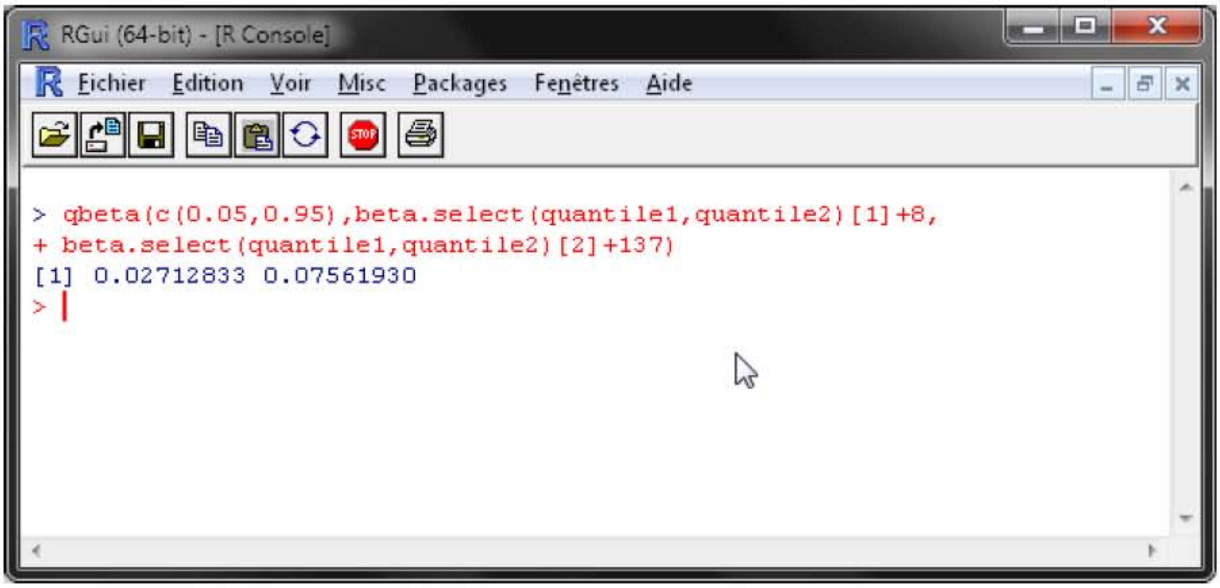


Ainsi, la probabilité a posteriori que la proportion soit plus grande ou égale à 3% est alors donnée par:



Donc il n'y a rien d'extraordinaire a posteriori d'avoir une proportion de cancers au-delà de 3%.

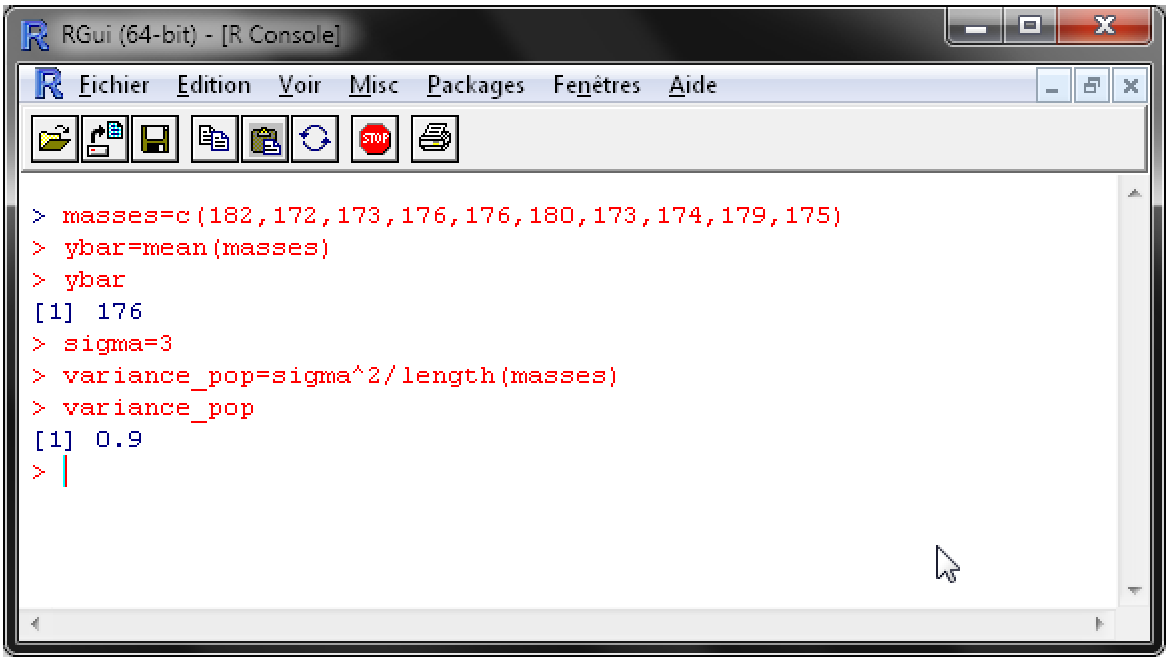
Nous pouvons même calculer l'intervalle à 90% en bilatéral de la proportion a posteriori:



Exemple 423.: Test d'hypothèse bayésien unilatéral (cas d'estimation de la moyenne avec a priori et posteriori Normales)

R 3.0.2

Nous pesons un échantillon étalon de laboratoire sur une balance de précision d'écart-type connu 3 et obtenons les masses suivantes:



Nous avons maintenant un nouvel échantillon et nous savons que a priori que sa masse devrait être de moyenne 170 avec un écart-type aussi de 3 et distribué selon une loi Normale.

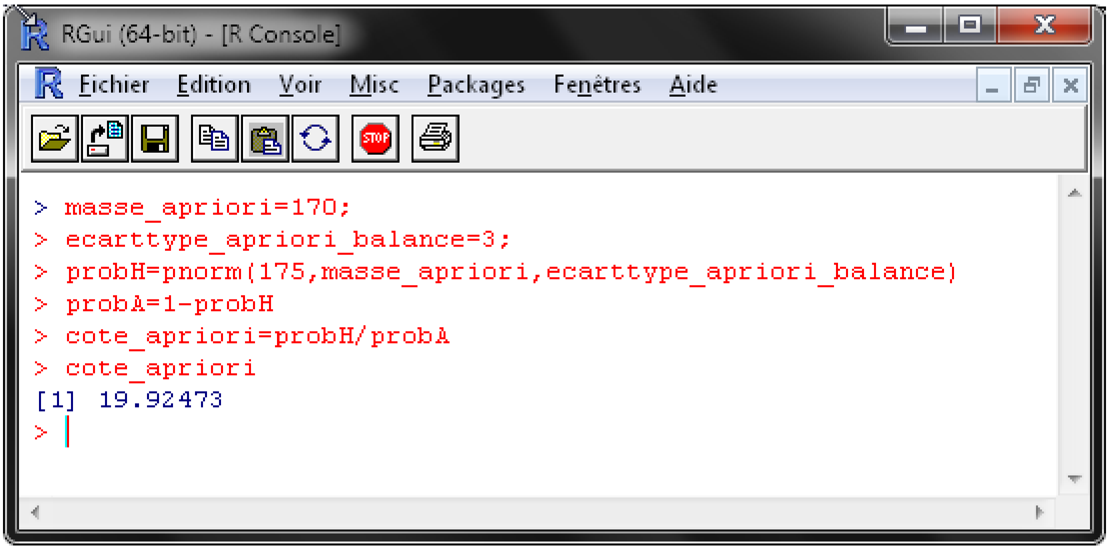
Nous émettons les hypothèses suivantes:

$$H_0 : \mu \leq 175, H_1 : \mu > 175$$

La cote de l'hypothèse nulle est alors donnée a priori par:

$$\frac{H_0}{H_1} = \frac{P(\mu \leq 175)}{P(\mu > 175)}$$

Ce qui se calcule très aisément avec R ou même n'importe quel tableur:



Donc l'hypothèse nulle a une cote a priori de 20 contre 1 d'être plus plausible que l'hypothèse alternative. Et quid du a posteriori sachant que nous avons des mesures passées d'un échantillon étalon?

Là encore, sachant que nous avons les informations suivantes:

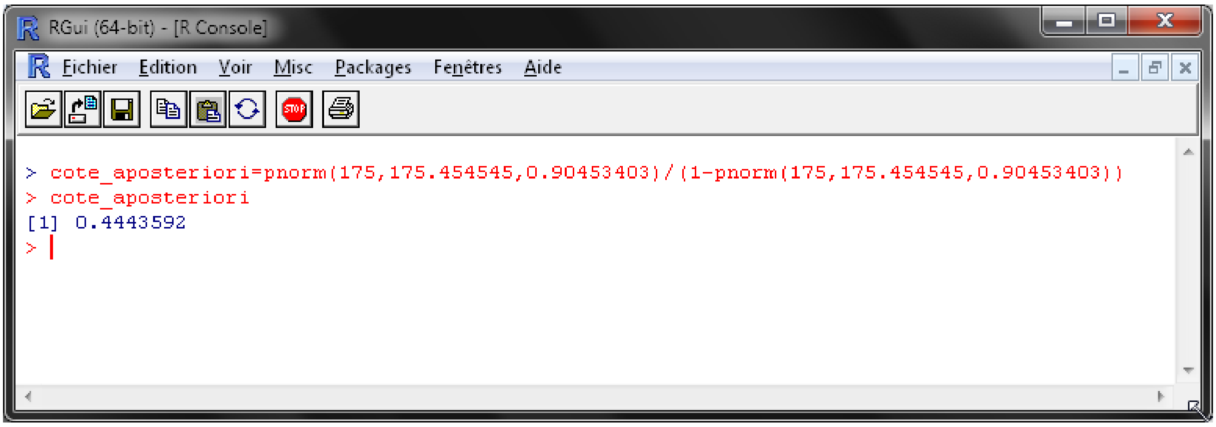
$$\begin{aligned} n &= 10 \\ \bar{y} &= 176 \\ \sigma^2 &= 9 \\ \mu_0 &= 170 \\ \tau_0^2 &= 9 \end{aligned}$$

Le calcul de l'espérance a posteriori est l'écart-type a posteriori est alors immédiate en transcrivant les relations suivantes dans R:

$$\mu_1 = \frac{\dfrac{\mu_0}{\tau_0^2} + \dfrac{n\bar{y}}{\sigma^2}}{\dfrac{1}{\tau_0^2} + \dfrac{n}{\sigma^2}} = 175.454545, \tau_1 = \sqrt{\left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)^{-1}} = 0.90453403$$

Ce qui nous donne les paramètres de la distribution a posteriori de la masse selon l'hypothèse de Normalité.

Nous pouvons donc calculs la cote de l'hypothèse nulle a posteriori:



Donc l'hypothèse nulle a une cote a posteriori de 1 contre 2 d'être plus plausible que l'hypothèse alternative.

Bon et alors quoi? A priori l'hypothèse nulle est plus plausible mais a posteriori elle n'est le pas! Eh bien nous allons devoir associer une probabilité à l'hypothèse nulle.

Suite à venir...

Exemple 424.: Intervalle de crédibilité

R 3.6.3

Voyons aussi un cas ludique d'application copié du blog d'Arthur CHARPENTIER[44] et qui est instructif aussi bien au niveau de l'apprentissage de R (manipulationd de matrice et vecteurs et certains subtilités graphiques) que de la compréhension de statistiques:

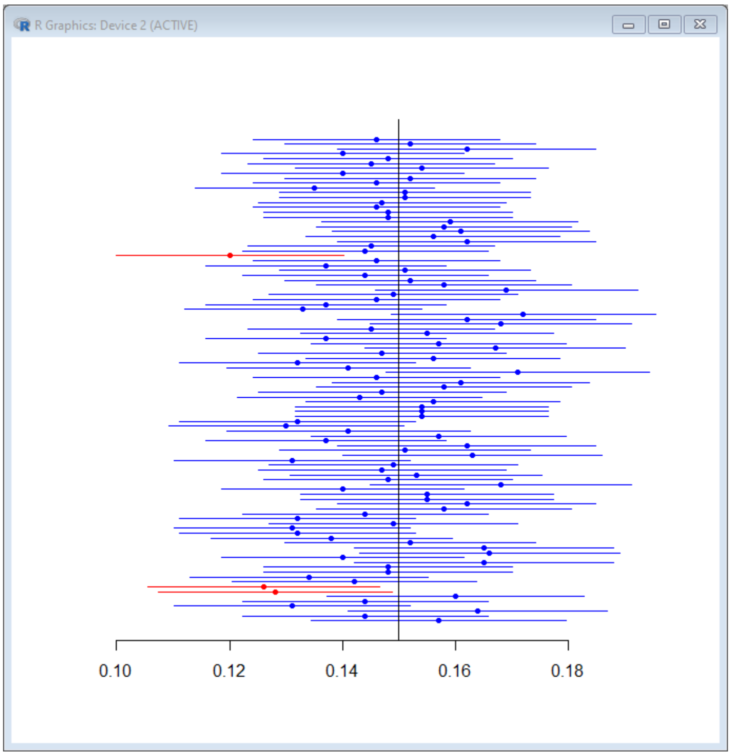
```
> #exemple pris de: https://freakonometrics.hypotheses.org/18117
> ############################################
> #partie intervalle de confiance de la proprtion (binomiale)
> n <- 1000 #nombre d'événements (contrats d'assurance))
> xbar <- 150 #nombre d'événements d'intérêt (accidents déclarés)
> ns <- 100 #nombre de simulations
> #on produit un vecteur de 100*1000 éléments binaires de probabilité 150/1000
> v<-rbinom(n*ns,size=1,prob=xbar/n)
> #on les range en 1000 lignes dans une matrice
> #(l'idée étant que chaque colonne contienne une simulation)
> M<-matrix(v,nrow=n)
>
> #on crée une fonction pour calculer l'intervalle de confiance de le proportion à 95%
> fIC<-function(x) mean(x)+c(-1,1)*1.96*sqrt(mean(x)*(1-mean(x)))/sqrt(n)
> #on l'applique sur chacune des ns colonne de la matrice M
> #ce qui donnera donc 100 (ns) intervalles de confiance
> head(IC<-t(apply(M,2,fIC)),10)
           [,1]      [,2]
 [1,] 0.1344514 0.1795486
 [2,] 0.1222392 0.1657608
 [3,] 0.1410501 0.1869499
 [4,] 0.1100877 0.1519123
 [5,] 0.1222392 0.1657608
 [6,] 0.1372775 0.1827225
 [7,] 0.1072929 0.1487071
 [8,] 0.1054318 0.1465682
 [9,] 0.1203656 0.1636344
[10,] 0.1128862 0.1551138
> #on calcule aussi les 100 (ns) moyennes empiriques
> MN<-apply(M,2,mean)
> #on crée une matrice contenant de façon binaire les cas ou la moyenne attendu est en
> #dehors des intervalles simulés
> k<-(xbar/n<IC[,1])|(xbar/n>IC[,2])
> #on fait un plot pour mettre cela en évidence de manière sympa (de façon astucieuse)
> plot(MN,1:ns,xlim=range(IC),axes=FALSE,xlab="",ylab="",pch=19,cex=.7,col=c("blue","red")[1+k])
> axis(1)
> segments(IC[,1],1:ns,IC[,2],1:ns,col=c("blue","red")[1+k])
> abline(v=xbar/n)
>
```

où on utilisé pour l'intervalle l'expression habituelle démontrée dans le cours théorique:

$$\overline{x} \pm \sqrt{\frac{\overline{x}(1-\overline{x})}{n}}$$

Ce qui donne graphiquement:

[44] https://web.archive.org/web/20190107115108/https://freakonometrics.hypotheses.org/18117



Nous avons ci-dessus une belle illustration du concept d'intervalle de confiance de la proportion a posteriori les données étant connues. On voit bien - sans surprises - que certains statistiquement ne contiennent pas la valeur vraie de la proportion.

Nous avons vu au cours que, puisque la distribution Beta est le conjugué de celle de Bernoulli, la distribution postérieure sera également Beta. Plus précisément selon la notation d'Arthur Charpentier:

$$\mathcal{B}\left(\alpha + \sum x_i, \beta + n - \sum x_i\right)$$

et selon celleu cours théorique:

$$P_{a,b}\left(\sum_i^N y_i + a, nN - \sum_i^N y_i + b\right)$$

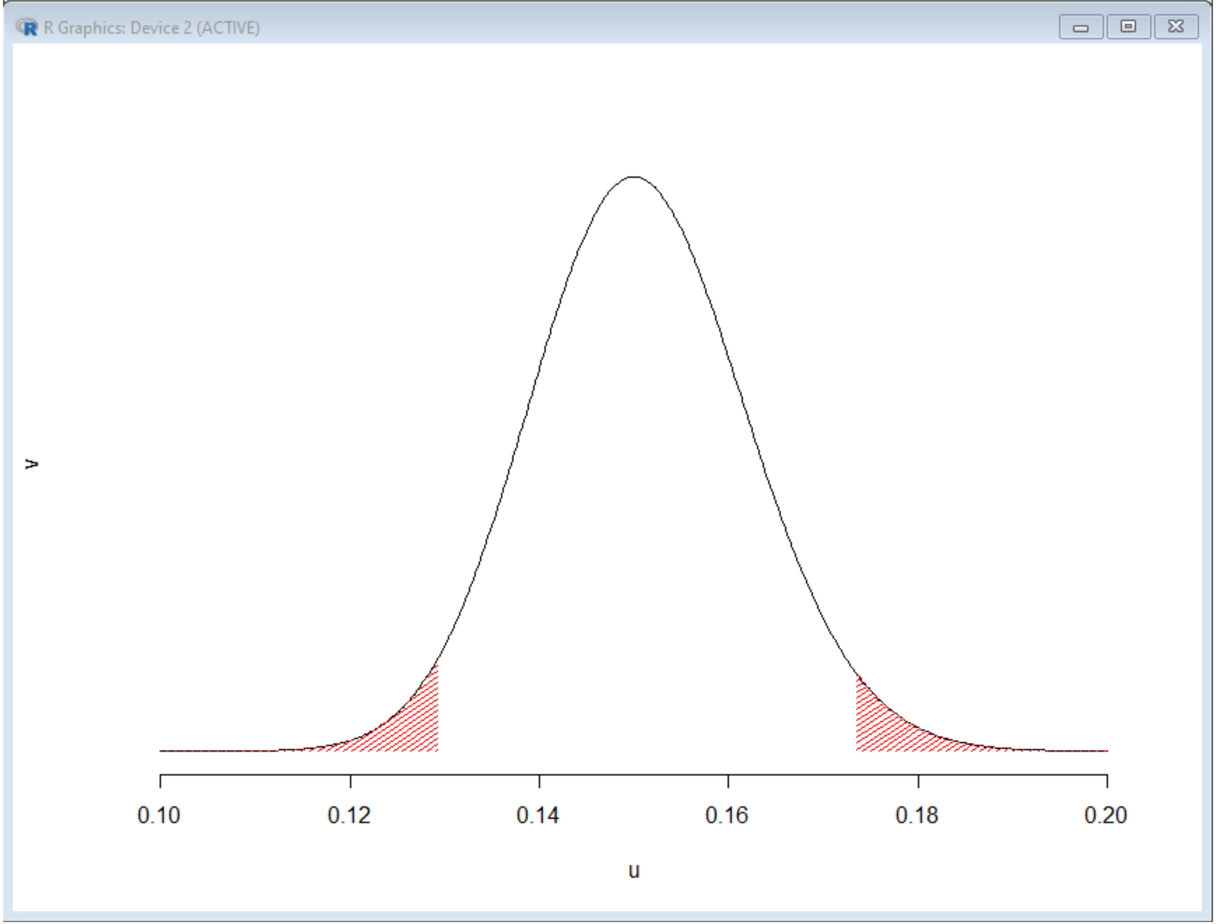
Dans le cours théorique nous avons traité le cas où ne prenions (cas où la loi beta se réduit a un a priori uniforme):

$$P_{a,b} = P_{1,1} = \frac{\Gamma(2)}{\Gamma(1)\Gamma(1)} p^{1-1}(1-p)^{1-1} 1_{]0,1[} = 1 \cdot 1_{]0,1[}$$

Et dans l'exemple que nous prenons ici, nous prenons évidemment N = 1.

Sur la base de cette propriété, l'intervalle de confiance de la moyenne est basé sur les quantiles de cette distribution (postérieure):

```
> n <- 1000 #nombre d'événements (contrats d'assurance))
> xbar <- 150 #nombre d'événements d'intérêt (accidents déclarés)
> #intervalle de crédibilité avec a priori beta uniforme
> u<-seq(.1,.2,length=501)
> v<-dbeta(u,1+xbar,1+n-xbar)
> plot(u,v,axes=FALSE,type="l")
> I<-u<qbeta(.025,1+xbar,1+n-xbar)
> polygon(c(u[I],rev(u[I])),c(v[I],rep(0,sum(I))),col="red",density=30,border=NA)
> I<-u>qbeta(.975,1+xbar,1+n-xbar)
> polygon(c(u[I],rev(u[I])),c(v[I],rep(0,sum(I))),col="red",density=30,border=NA)
> axis(1)
>
```



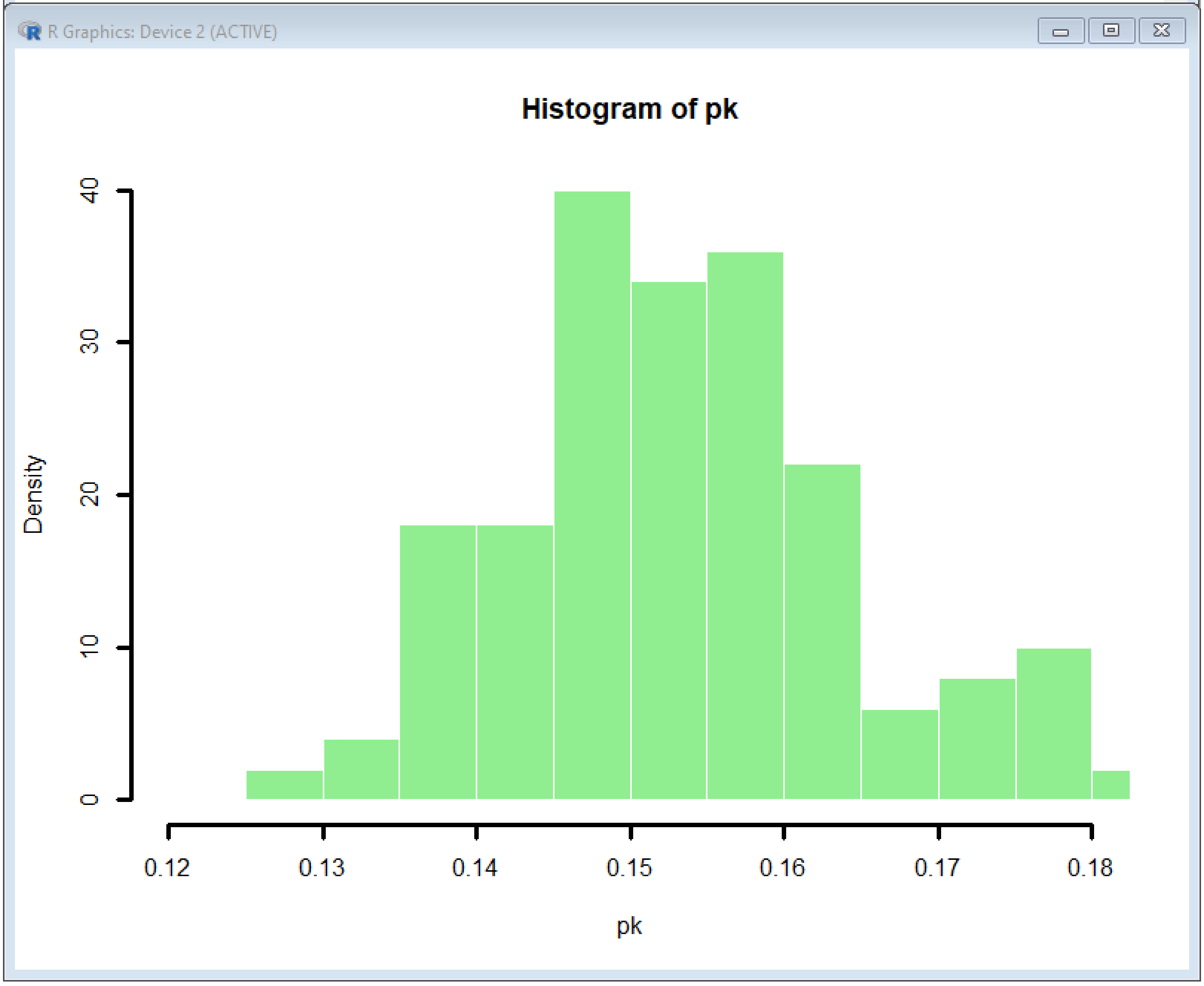
Les bornes de l'intervalle de crédibilit sont donnés alors par:

```
> qbeta(.025,1+xbar,1+n-xbar)
[1] 0.1292306
> qbeta(.975,1+xbar,1+n-xbar)
[1] 0.1734871
>
```

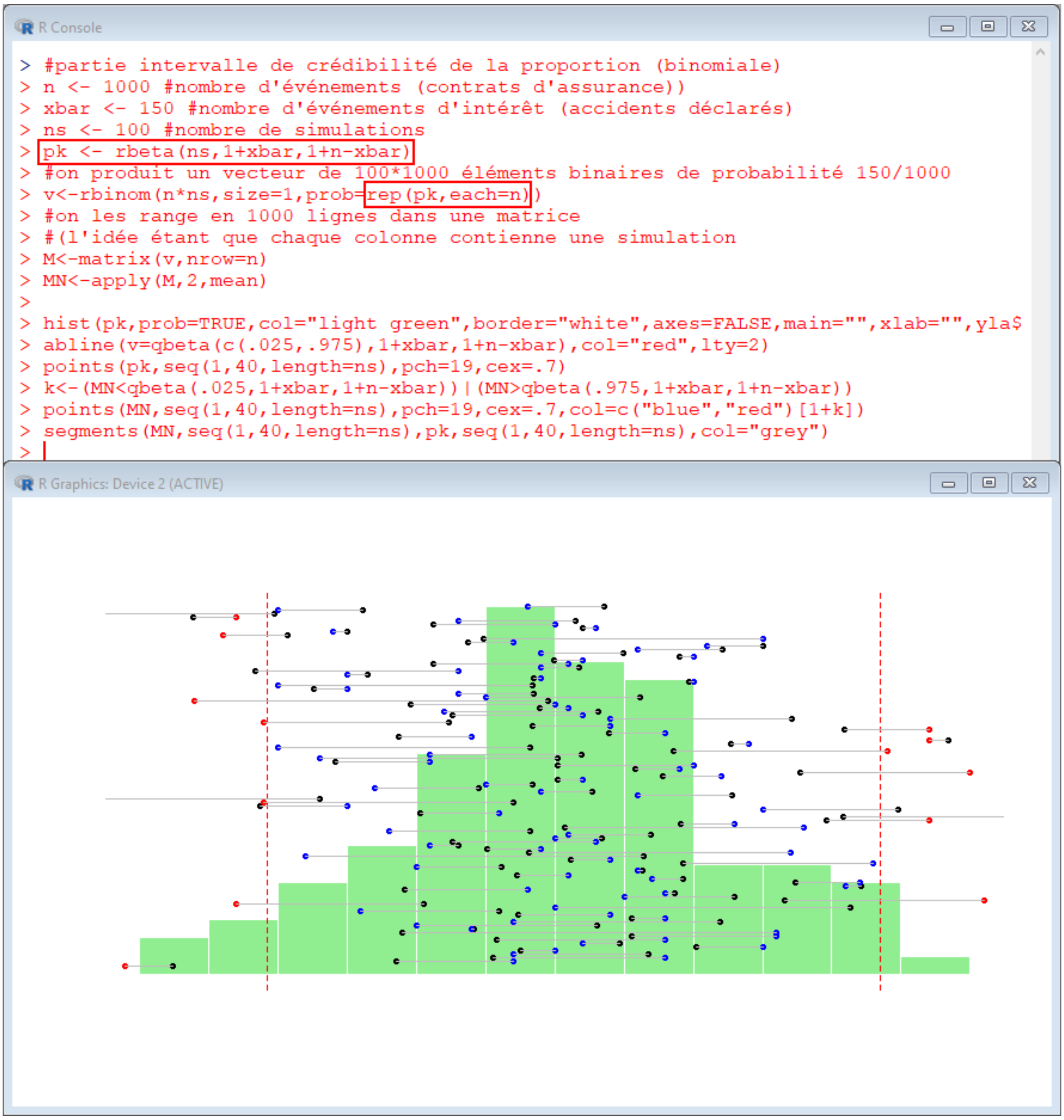
Maintenant, posons-nous la question de ce que cela signifie d'avoir un intervalle de crédibilité de 95%? Eh bien, cette fois, nous ne calculons pas un intervalle de confiance en utilisant une moyenne empirique ponctuelle, mais avec plusieurs moyennes empiriques obtenues par la distribution la distribution postérieure ci-dessus (compte tenu des observations).

On peut bien évidemment afficher un histogramme des moyennes de cette distribution a posteriori:

```
> n <- 1000 #nombre d'événements (contrats d'assurance))
> xbar <- 150 #nombre d'événements d'intérêt (accidents déclarés)
> ns <- 100 #nombre de simulations
> pk <- rbeta(ns,1+xbar,1+n-xbar)
> hist(pk,prob=TRUE,col="light green",border="white",lwd=3,xlim=c(.12,.18))
>
```



Donc maintenant que fait-on? Eh bien on reprend le même type de code qu'au début qu'on change juste un petit peu (on a mis dans un rectangle rouge la modification majeure):



Ici, il y a 95% de probabilité que ces moyenmes empiriques se situent dans l'intervalle de crédibilité, défini à l'aide des quantiles de la distribution postérieure. On peut en fait visualiser toutes ces moyennes: en noir la moyenne utilisée pour générer l'échantillon, puis, en bleu ou rouge, les moyennes obtenues sur ces échantillons simulés.

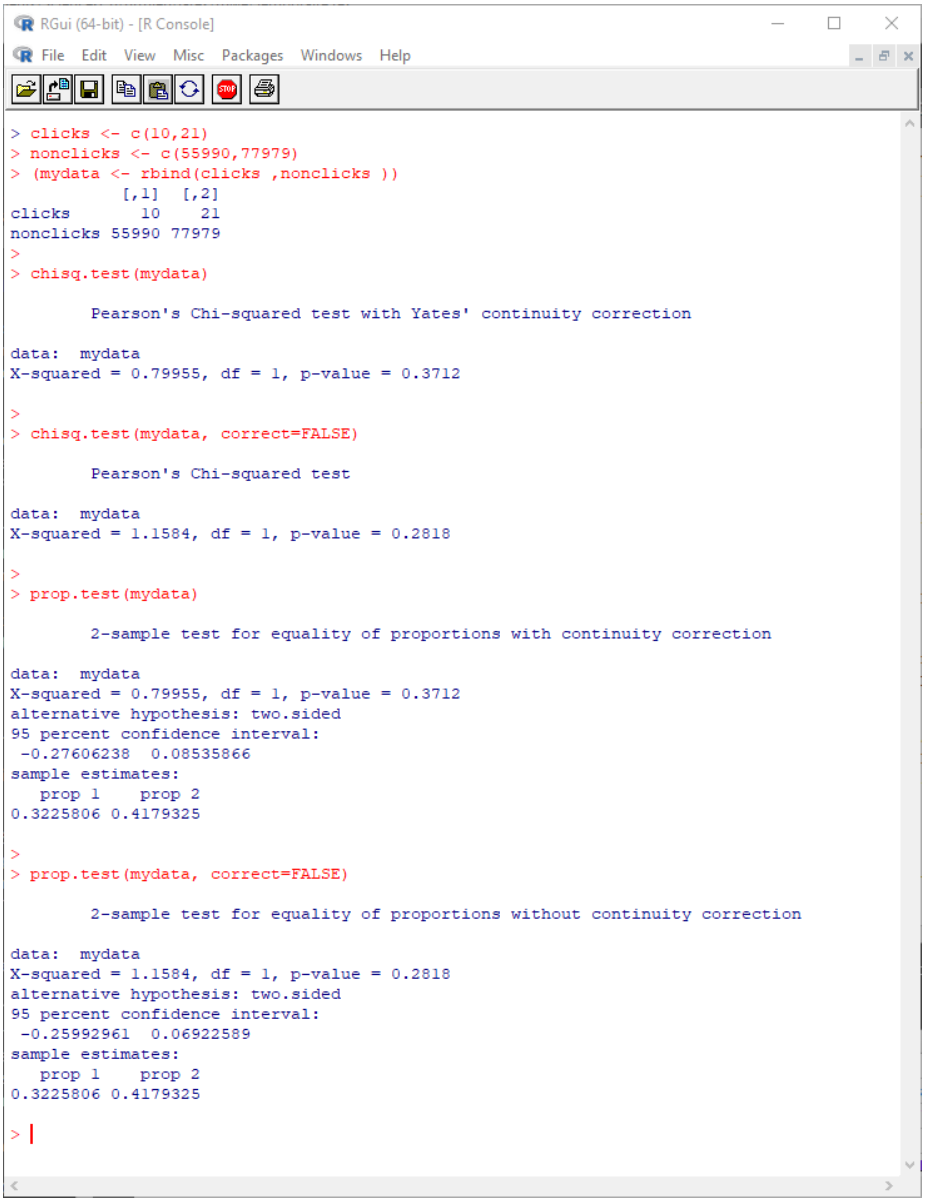
Exemple 425.: Test A/B Bayésian avec probabilités a priori beta

R 3.5.1

Non sans mal, nous avons démontré dans le cours théorique comment calculer dans le cours théorique la probabilité a posteriori que deux taux (compris entre 0% et 100%) suivant a priori une loi beta avaient d'être plus grand l'un que l'autre.

Nous avons ensuite fait de même avec des données de comptage avec comme distribution a priori, une distribution Gamma.

Attention à ne pas confondre le test bayésian A/B avec les tests A/B des proportions et du chi-deux dont nous avons démontré dans le cours théorique l'équivalence mathématique. Nous pouvons rapidement le vérifier ici en observant que **R utilise toujours le chi-deux même pour la fonction prop.test( )**:



```
> clicks <- c(10,21)
> nonclicks <- c(55990,77979)
> (mydata <- rbind(clicks ,nonclicks ))
           [,1]  [,2]
clicks       10    21
nonclicks 55990 77979
>
> chisq.test(mydata)

        Pearson's Chi-squared test with Yates' continuity correction

data:  mydata
X-squared = 0.79955, df = 1, p-value = 0.3712

>
> chisq.test(mydata, correct=FALSE)

        Pearson's Chi-squared test

data:  mydata
X-squared = 1.1584, df = 1, p-value = 0.2818

>
> prop.test(mydata)

        2-sample test for equality of proportions with continuity correction

data:  mydata
X-squared = 0.79955, df = 1, p-value = 0.3712
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.27606238  0.08535866
sample estimates:
   prop 1    prop 2
0.3225806 0.4179325

>
> prop.test(mydata, correct=FALSE)

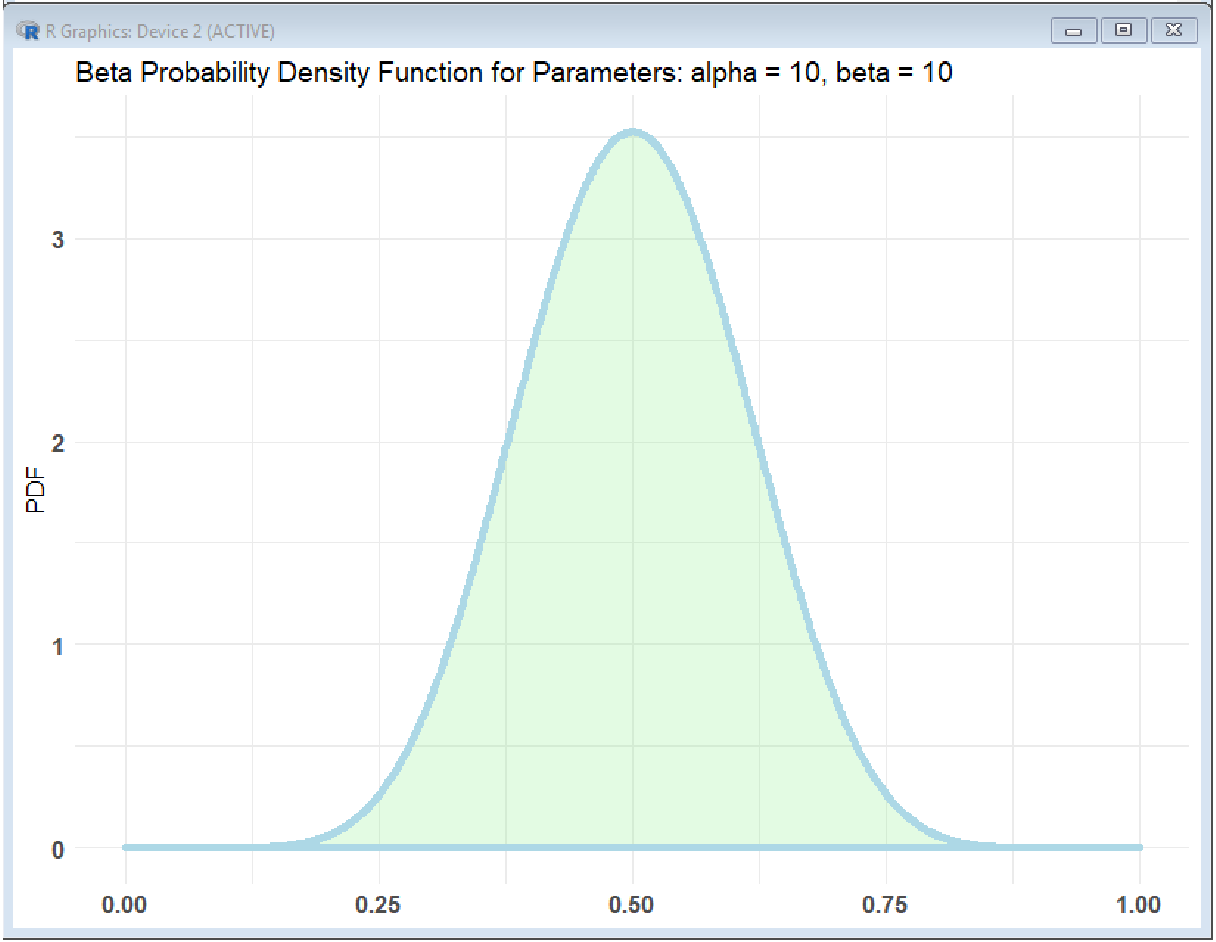
        2-sample test for equality of proportions without continuity correction

data:  mydata
X-squared = 1.1584, df = 1, p-value = 0.2818
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.25992961  0.06922589
sample estimates:
   prop 1    prop 2
0.3225806 0.4179325

>
```

Voyons comme cas d'application, comment le package **bayesAB** et sa fonction **bayesTest( )** implémente cela:

```
> library("bayesAB")
>
> #on crée les deux groupes
> control <- rbinom(20, 1, 0.5)
> treatment <- rbinom(20, 1, 0.3)
>
> #choisissons la distribution beta comme a priori
> #(parmi les nombreux choix du package!)
> #affichibs une des distributions a priori
> plotBeta(10, 10)
>
```



Lançons d'abord les calculs du test:

```
> test <- bayesTest(treatment, control, distribution = "bernoulli", priors = list("alpha" = c(10,15), "beta" = c(10,20)))
> print(test)
--------------------------------------------
Distribution used: bernoulli
--------------------------------------------
Using data with the following properties:
          A    B
Min.     0.0 0.00
1st Qu.  0.0 0.00
Median   0.0 0.00
Mean     0.3 0.35
3rd Qu.  1.0 1.00
Max.     1.0 1.00
--------------------------------------------
Conjugate Prior Distribution: Beta
Conjugate Prior Parameters:
$`alpha`
[1] 10 15

$beta
[1] 10 20

--------------------------------------------
Calculated posteriors for the following parameters:
Probability
--------------------------------------------
Monte Carlo samples generated per posterior:
[1] 1e+05
>
```

Nous obtenons alors:

```
> summary(test)
Quantiles of posteriors for A and B:

$`Probability`
$`Probability`$`A`
       0%       25%       50%       75%      100%
0.1330820 0.3413049 0.3890985 0.4380707 0.7525513

$`Probability`$B
       0%       25%       50%       75%      100%
0.1581962 0.3619542 0.4101828 0.4600918 0.7733047


--------------------------------------------

P(A > B) by (0)%:

$`Probability`
[1] 0.41687

--------------------------------------------

Credible Interval on (A - B) / B for interval length(s) (0.9) :

$`Probability`
        5%        95%
-0.3829929  0.4394876

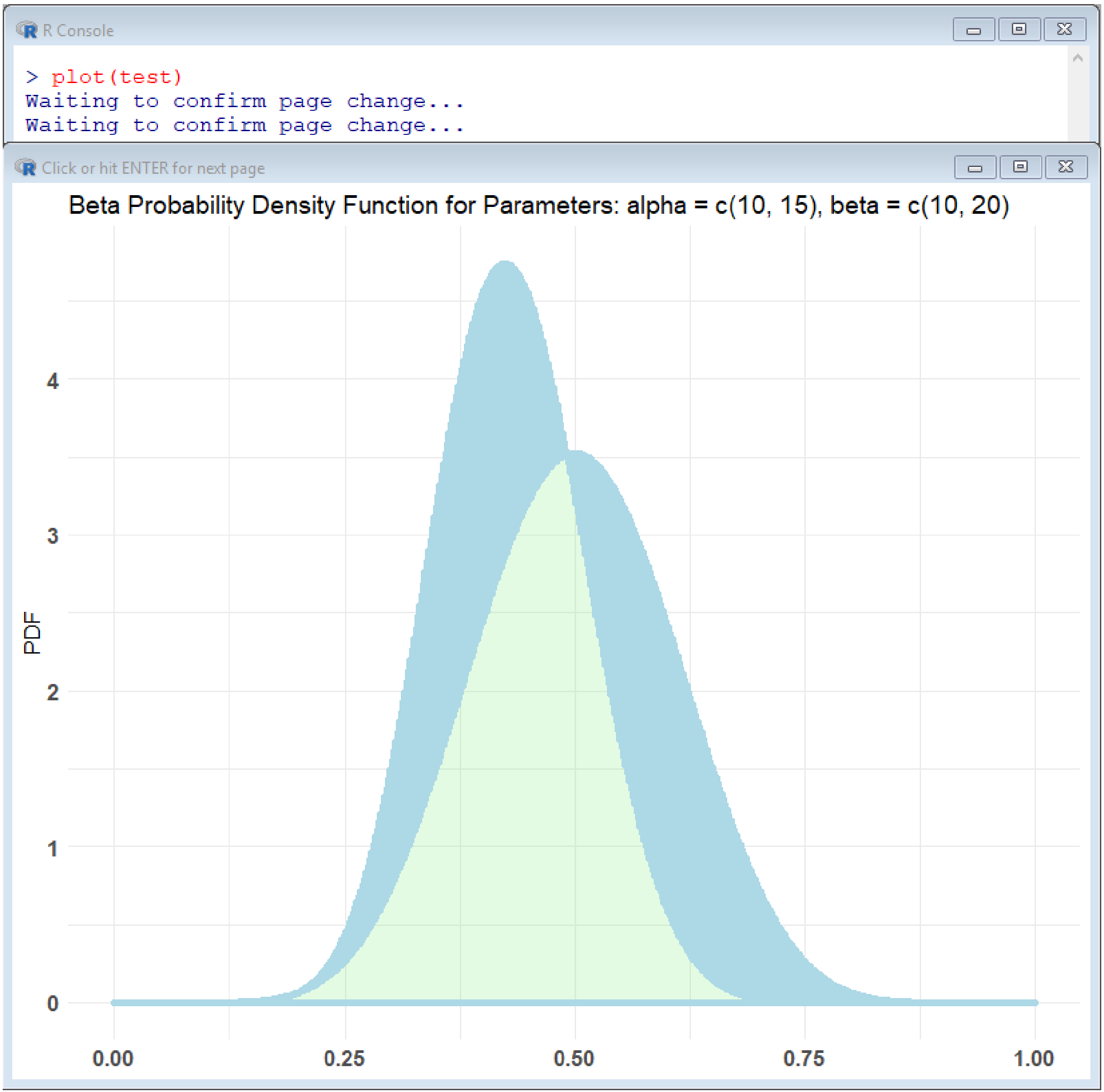
--------------------------------------------

Posterior Expected Loss for choosing B over A:

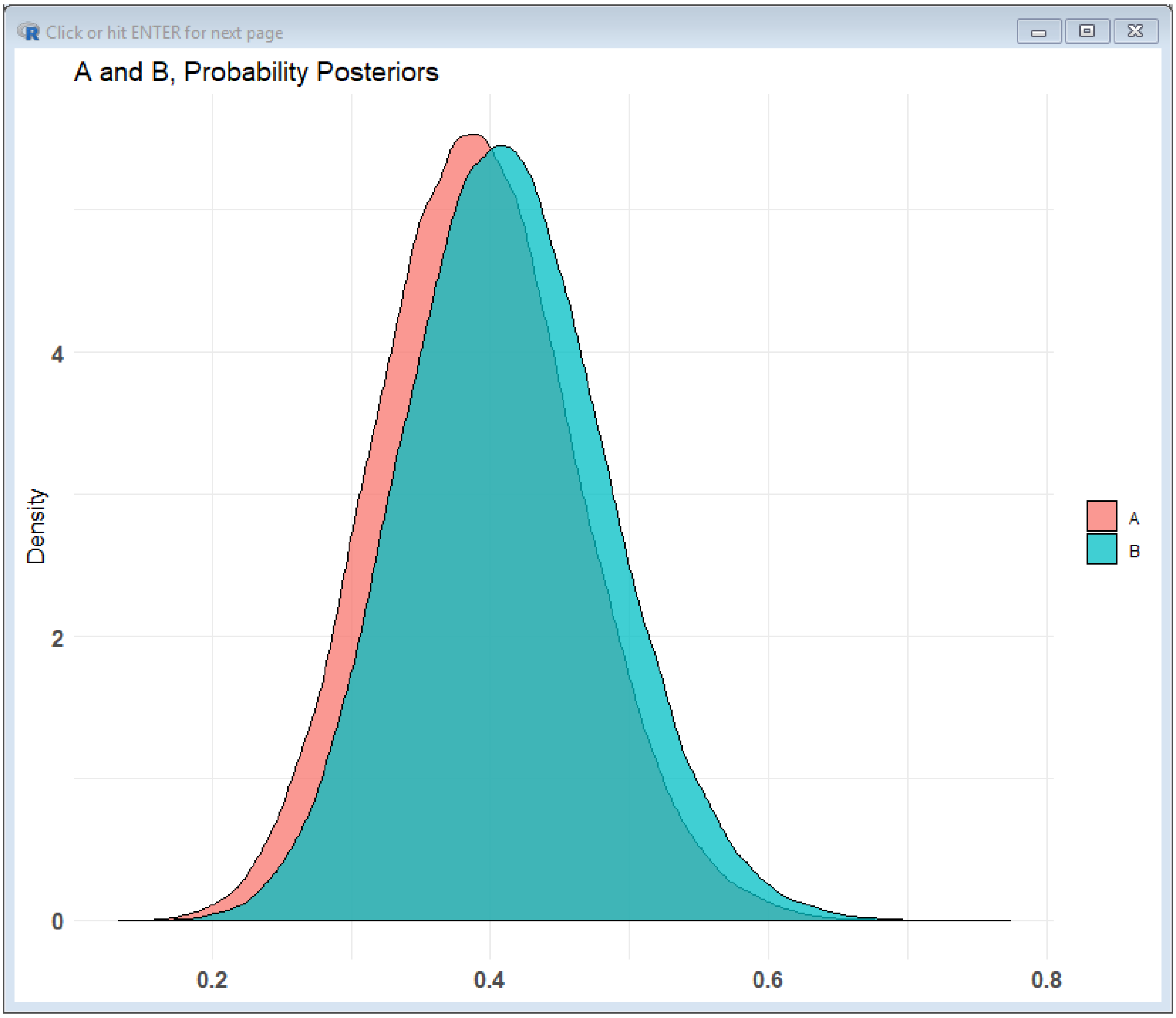
$`Probability`
[1] 0.159195

>
```

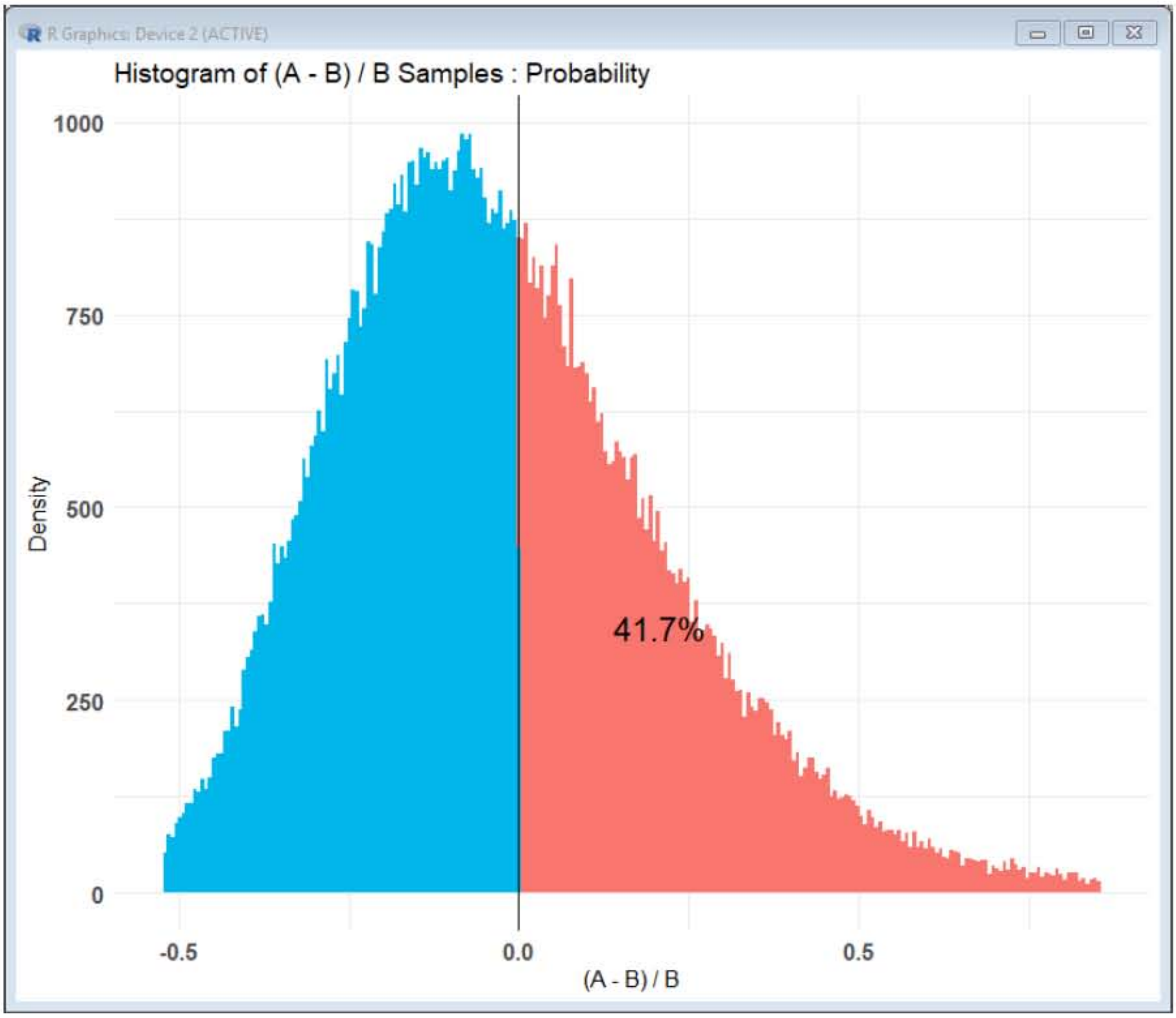
Et avec la fonction plot( ) native de ce package, nous obtenons deux graphs à la suite. Le premier étant:



Et le deuxième graphe:



Et enfin le dernier et troisième graph:


R Graphics: Device 2 (ACTIVE)
Histogram of (A - B) / B Samples : Probability
1000
750
500
250
0
Density
41.7%
-0.5
0.0
0.5
(A - B) / B

Exemple 426.: BIC (Bayesian Information Criterion) et BF (Bayes Factor)

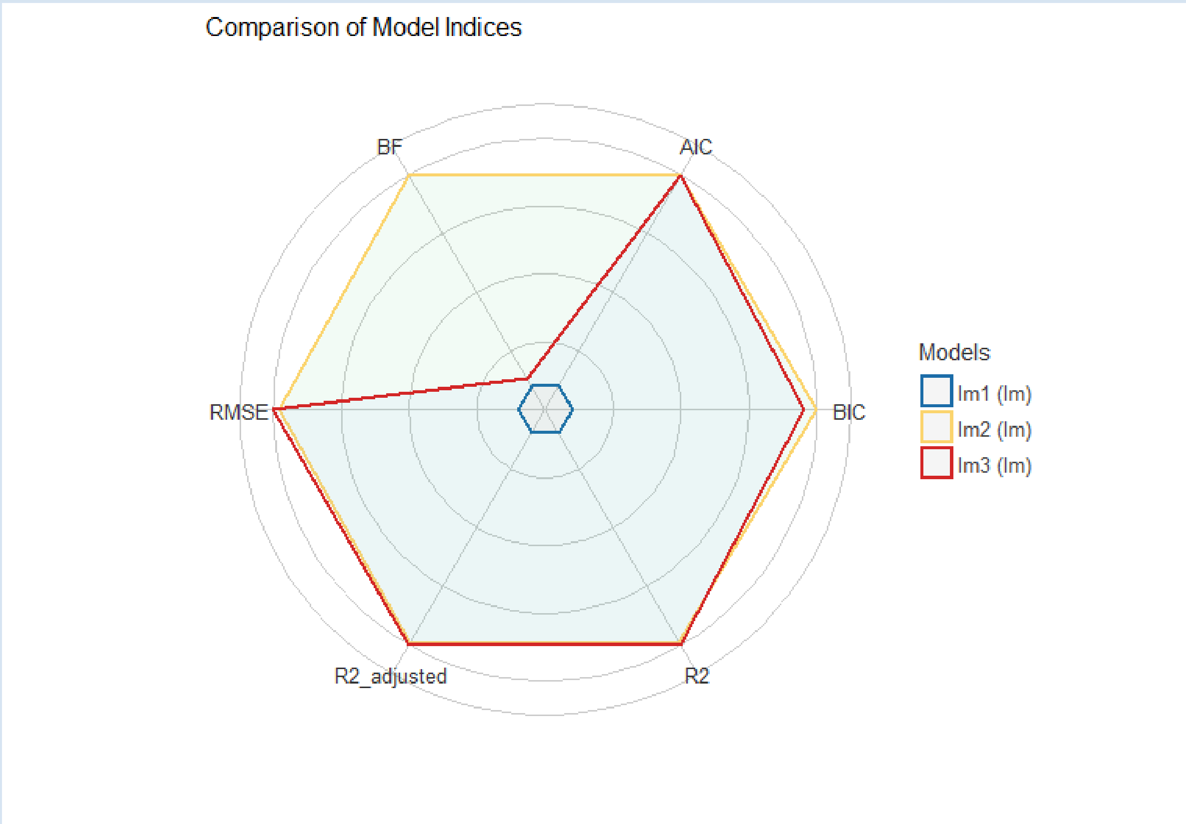
R 3.5.1

Nous avons déjà rencontré le BIC et le BF de multiples fois à travers cet ouvrage et pour différents modèles statistiques ou de machines learning.

Revenons sur le premier exemple que nous avons vu dans ce livre et qui est le plus simple et probablement parmi les plus courants dans les statistiques de base:

```
> library("performance")
> library("see")
> data(iris)
> lm1<-lm(Sepal.Length~Species,data=iris)
> lm2<-lm(Sepal.Length~Species+Petal.Length,data=iris)
> lm3<-lm(Sepal.Length~Species*Petal.Length,data=iris)
> (result<-compare_performance(lm1,lm2,lm3))
# Comparison of Model Performance Indices

Model | Type |    AIC |    BIC |   R2 | R2_adjusted | RMSE |       BF
-----------------------------------------------------------------------
lm1   |   lm | 231.45 | 243.49 | 0.62 |        0.61 | 0.51 |
lm2   |   lm | 106.23 | 121.29 | 0.84 |        0.83 | 0.33 | 3.45e+26
lm3   |   lm | 106.77 | 127.84 | 0.84 |        0.83 | 0.33 | 1.30e+25
> plot(result)
>
```



30. Statistiques spatiales

Une grande partie des statistiques spatiales sont relatives au fait de représenter graphiquement des courbes de niveaux (isoclines) ou des densités surfaciques de probabilités ou simplement des diagrammes à barre ou des diagrammes à secteurs représentés sur des cartes géographiques. Nous ne reviendrons pas sur tous ces cas déjà traités dans le chapitre sur les Graphiques à partir de la page 1063.

Distribution circulaire

Avec le package **circular** et ses fonctions **circular( )** et **rose.diag( )** déjà on peut représenter des fréquences d'observations et un histogramme circulaire sur 360° degrés:

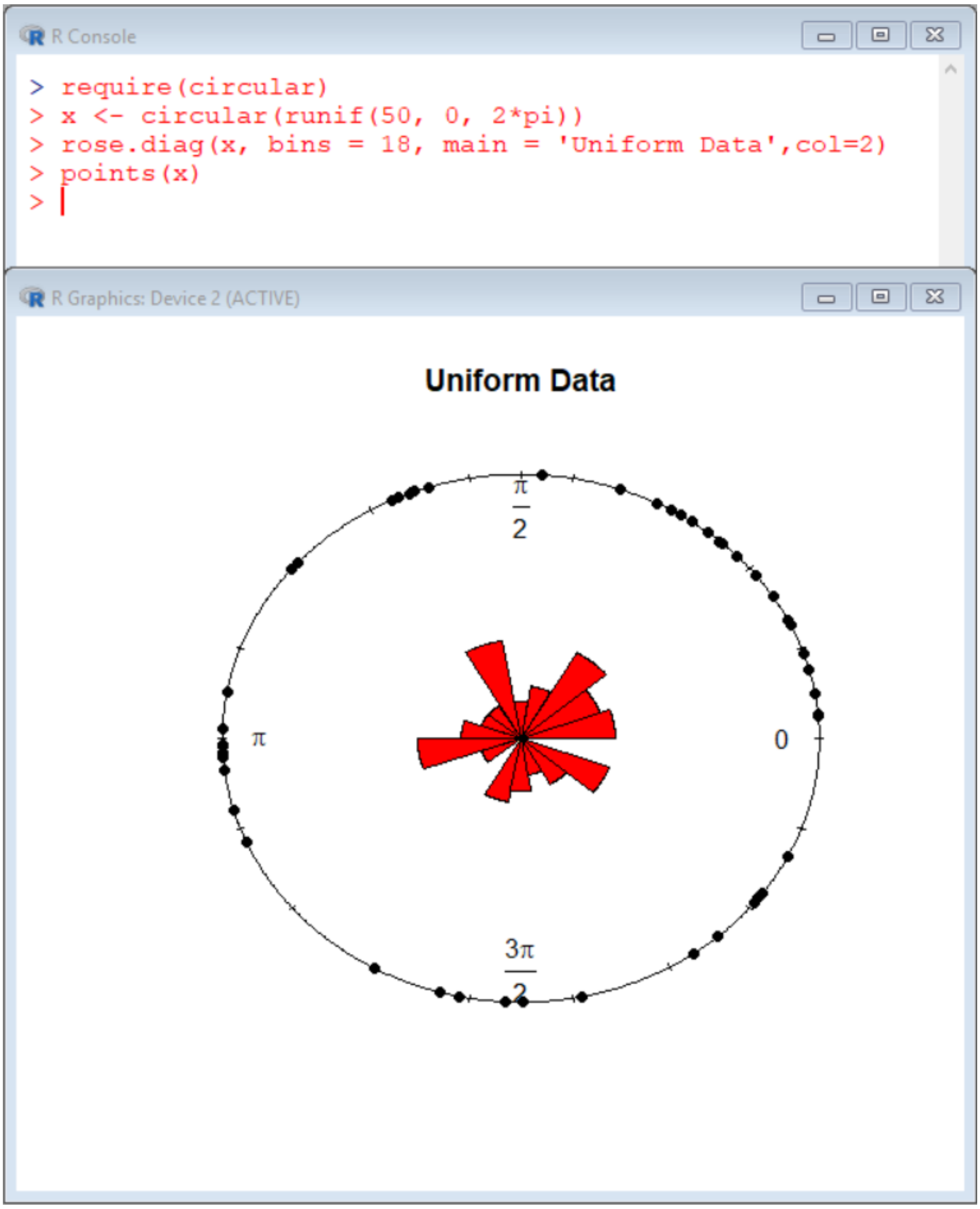
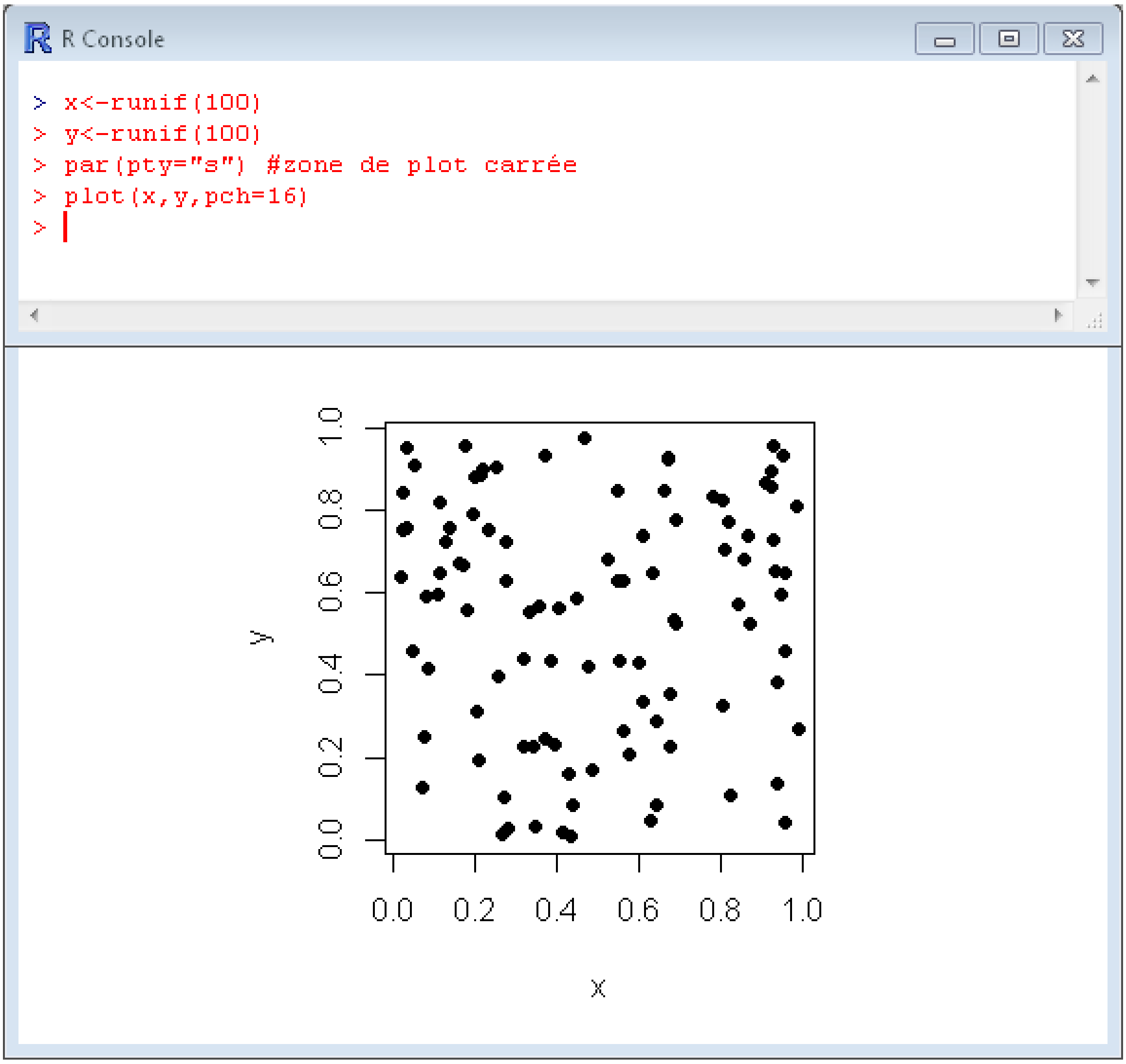


Figure 338 Graph de distribution statistique spatiale angulaire

Plus proche voisin et distance moyenne

Considérons le cas le plus simples de plus proches voisins.

Pour cela nous générons des points aléatoirement qui nous serviront d'exemple:



Pour chaque point nous calculons la distance euclidienne aux autres points et nous traçons une ligne pour chaque point avec son unique plus proche voisin (nn=nearest neighbour) ce qui donne:

```
> distance<-function(x1,y1,x2,y2) sqrt((x2-x1)^2+(y2-y1)^2)
>
> nn<-numeric(100)
> d<-numeric(100)
> for (i in 1:100){
+ for (k in 1:100) d[k]<-distance(x[i],y[i],x[k],y[k])
+ nn[i]<-which(d==min(d[-i]))
+ }
>
> for (i in 1:100) lines(c(x[i],x[nn[i]]),c(y[i],y[nn[i]]))
>
```

où le *min(d[-i])* permet d'éliminer la distance nulle entre le point traité dans la boucle et lui-même.

Ce qui donne:

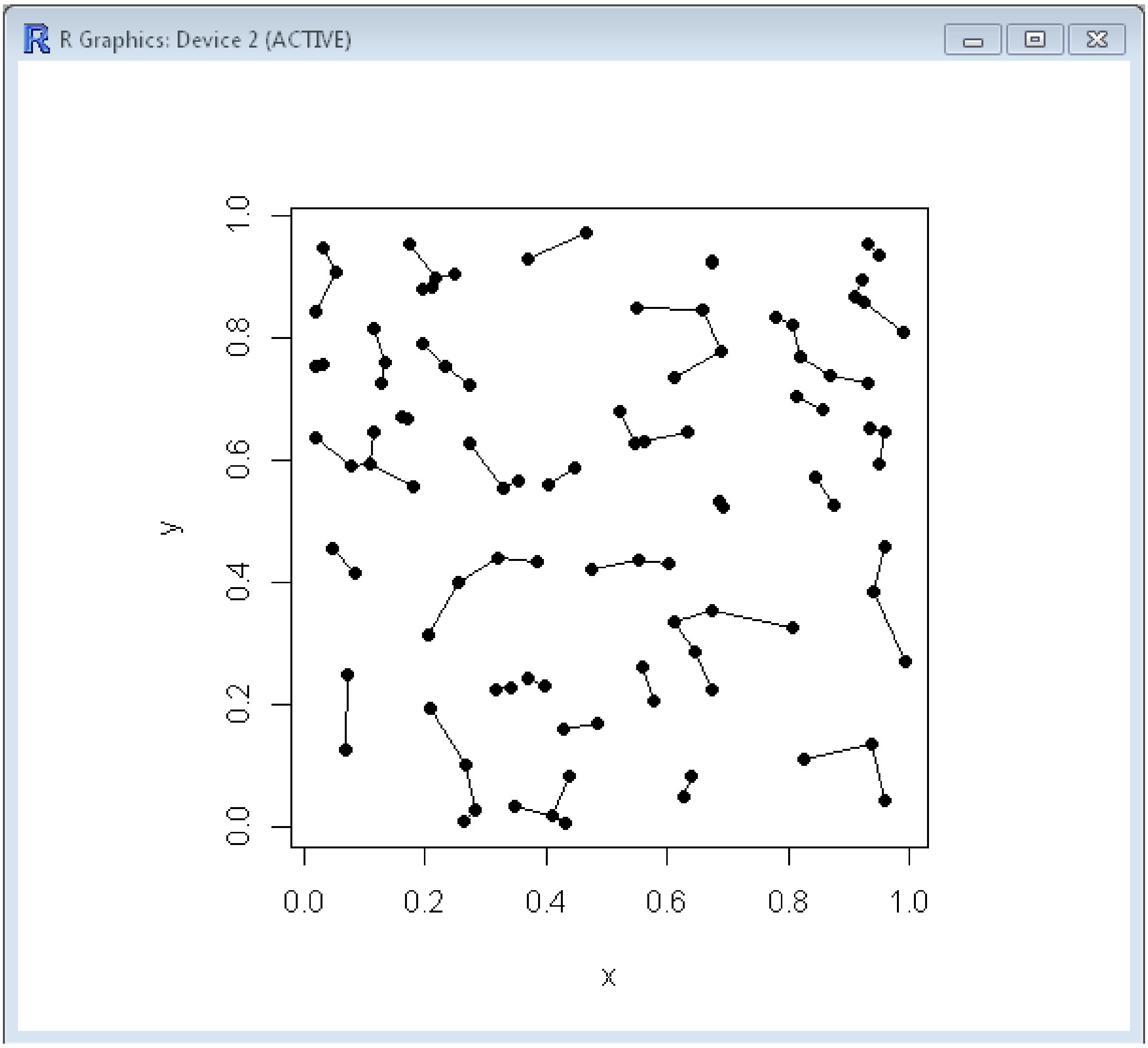
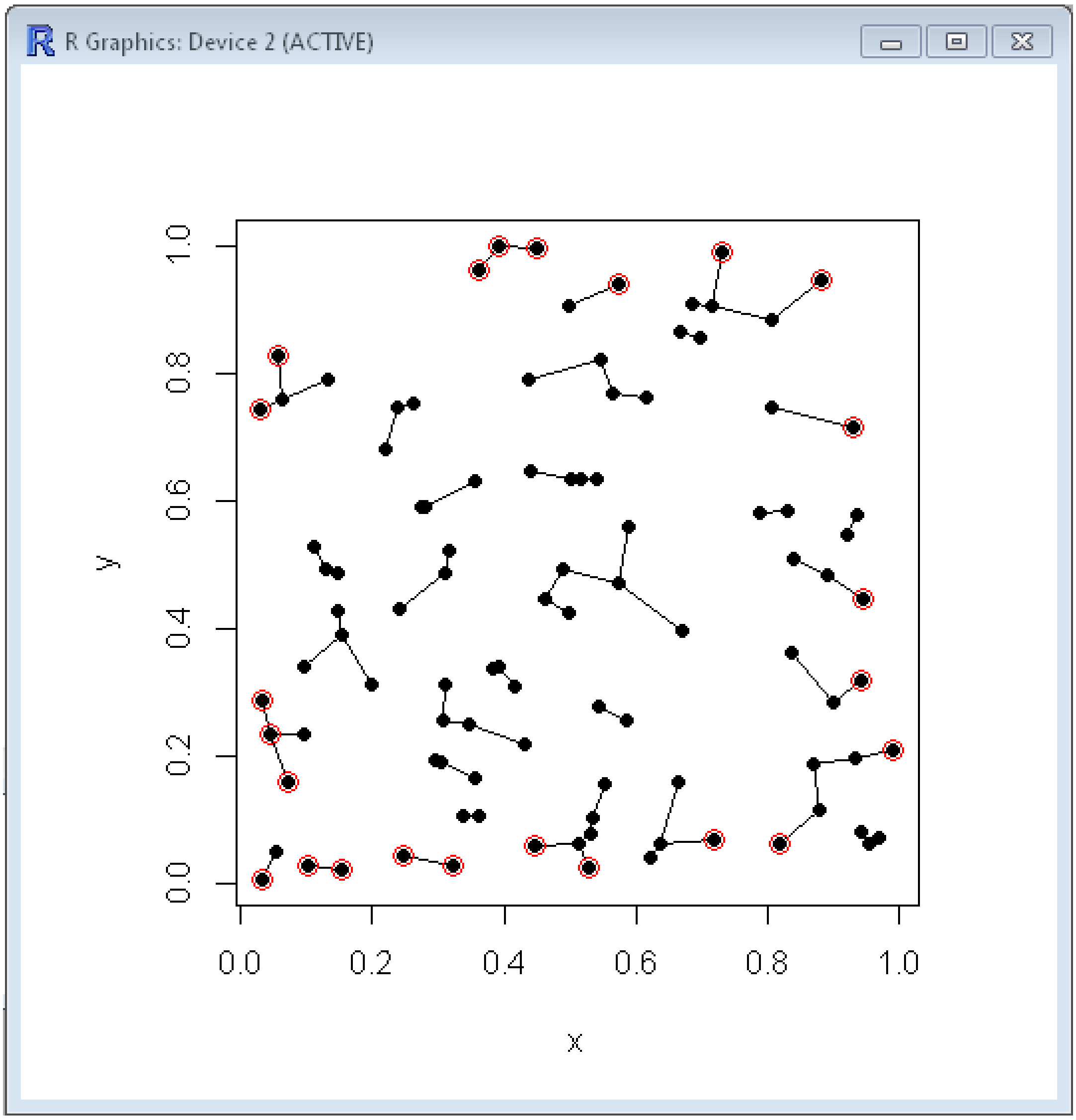


Figure 339 Graph plus proches voisins

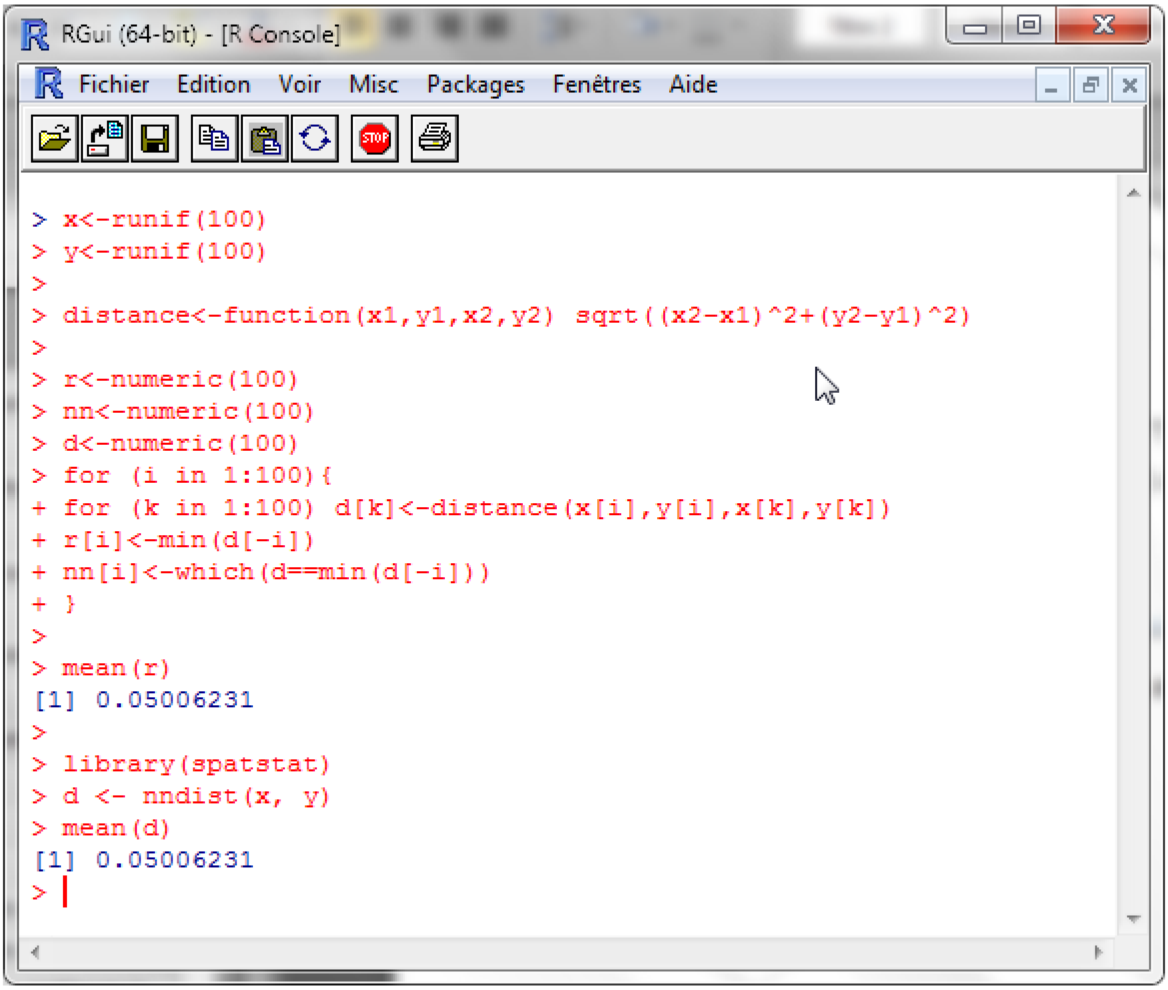
Ensuite, il peut être intéressant de savoir quels sont les points qui sont plus proches du bord que de leur plus proche voisin, pour cela nous modifions un peu le code précédent:

```
> distance<-function(x1,y1,x2,y2) sqrt((x2-x1)^2+(y2-y1)^2)
>
> r<-numeric(100)
> nn<-numeric(100)
> d<-numeric(100)
> for (i in 1:100){
+ for (k in 1:100) d[k]<-distance(x[i],y[i],x[k],y[k])
+ r[i]<-min(d[-i])
+ nn[i]<-which(d==min(d[-i]))
+ }
>
> for (i in 1:100) lines(c(x[i],x[nn[i]]),c(y[i],y[nn[i]]))
>
> topd<-1-y
> rightd<-1-x
> edge<-pmin(x,y,topd,rightd)
> id<-which(edge<r)
> points(x[id],y[id],col="red",cex=1.5)
> sum(edge<r)#nombre de points plus proches des bords que de leur nn
[1] 24
>
```


Ce qui donne:



Nous pouvons aussi calculer "à la main" ou avec le package de statistiques spatiales **spatstat** et sa fonction **nndist( )** la distance moyenne:



Nous avons démontré dans le cours théorique que l'espérance de la distance du plus proche voisin dans le cas d'une distribution aléatoire était donnée dans un disque par:

$$r = E(R) = \frac{1}{2\sqrt{\rho}}$$

Or nous avons dans notre exemple 100 points dans un carré de côté unitaire. En première approximation avec un disque inscrit au carré, nous avons alors:

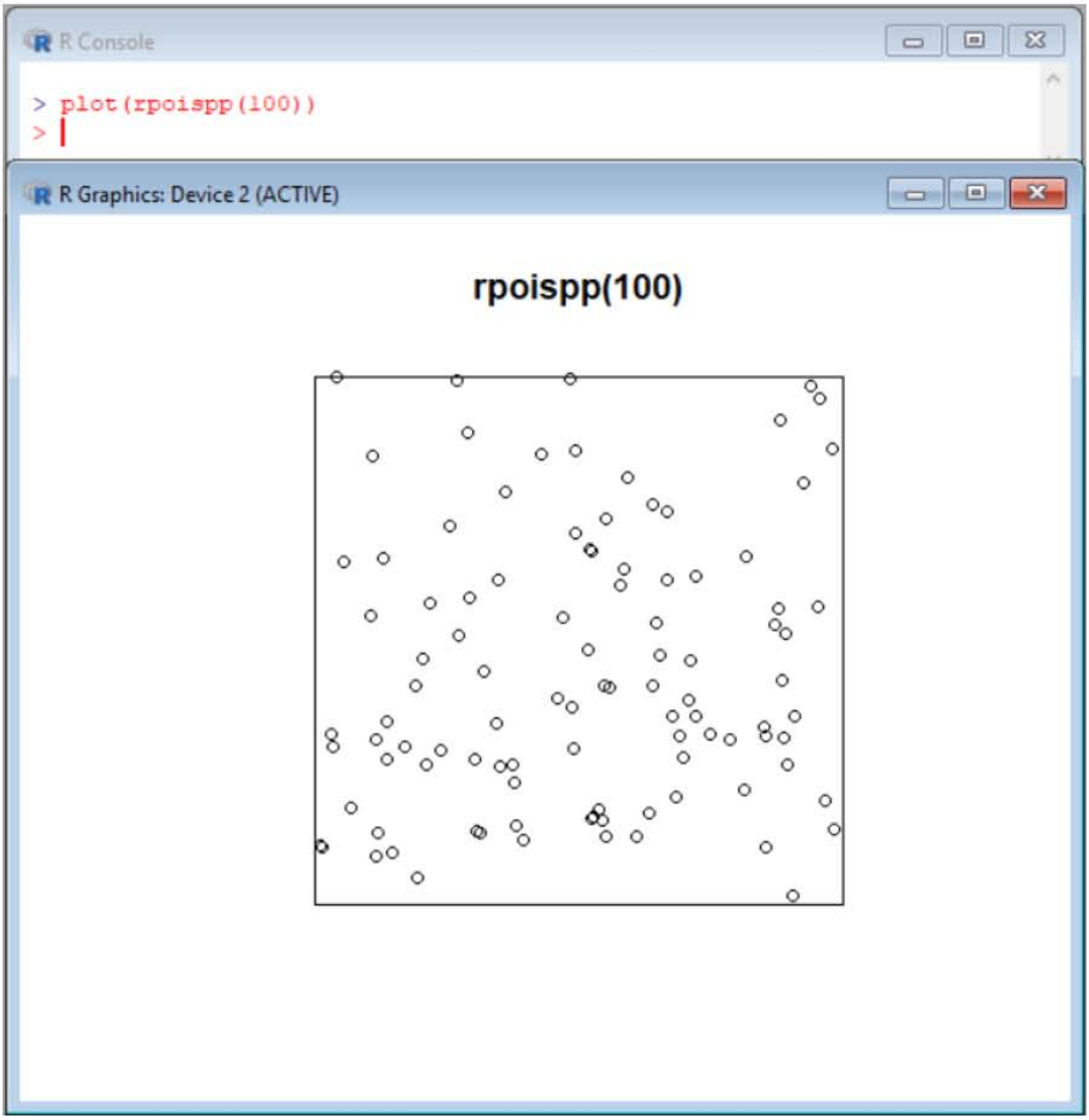
$$\rho = \frac{100}{\pi R^2} \cong 127$$

Dès lors:

$$r = E(R) = \frac{1}{2\sqrt{\rho}} = \frac{1}{2 \cdot \sqrt{127}} = 0.044$$

Comme 0.044 n'est pas trop éloigné de 0.0500 la distribution peut être considérée effectivement comme aléatoire.

Indiquons qu'il est possible toujours avec le package **spatstat** de faire des graphs de points distribués selon la loi de Poisson avec la fonction **rpoispp( )** :


R Console
> plot(rpoispp(100))
>
R Graphics: Device 2 (ACTIVE)
rpoispp(100)

31. Text Mining

Le texte mining est un sujet qui a toujours été important dans certains domaines académiques mais depuis l'émergence des réseaux sociaux et de l'analyse de news en temps réel, ces techniques ont connu une véritable explosion et ce particulièrement dans le journalisme automatisé et la finance spéculative.

Un exemple fameux est un algorithme de Text Mining de Tweet qui aurait suite à l'émission d'un faux Tweet publié par un personnage influent du réseau massivement fait chuter le S&P 500 de 136 milliards de dollars[45]. Un autre exemple d'application est le métier d'historien qui va radicalement changer dans les décennies ou siècles à venir puisqu'aujourd'hui les documents officiels ou non ne sont plus imprimés mais générés directement en version électronique et une quantité phénomène d'entre eux est produite chaque jour. L'historien se transforme donc à un analyste de données numériques avec un approche statistique de la lecture de texte par les machines que sont les ordinateurs.

Dans ce chapitre nous allons nous concentrer sur les techniques proposées par le package **tm** (text mining) de R en se basant sur un corpus contenant un seul et unique fichier qui est le pdf de mon e-book *Éléments de Mathématiques appliquées* (~4'900 pages A4) convertit en texte pur (*.txt) ce qui le réduit à environ 2'800 pages A4.



Sinon le Text Mining n'est pas utile que pour analyser des contenus de livres ou des médias sociaux ou de sites web mais aussi pour l'interaction avec des intelligences artificielles (http://www.ipsoft.com/amelia/) ou l'automatisation de la justice par l'analyse de compte rendus de jugements (https://predictice.com) ou encore la détection automatisée de plagiat et l'adaptation intelligente automatisée de contrats.

Remarque: Nous ne traiterons pas ici du package *googleLanguageR* (du type text-to-speech) et de son API pour deux raisons. La première est que l'API n'est pas gratuite, la deuxième est que le compte correspondant est vraiment ennuyeux à créer et à mettre à jour (du moins à chaque fois qu'il y a une mise à jour). Idem du package Rtts (aussi du type text-to-speech) dont le serveur ne répond que rarement…

[45] Source: https://www.buffalo.edu/news/releases/2015/05/037.html

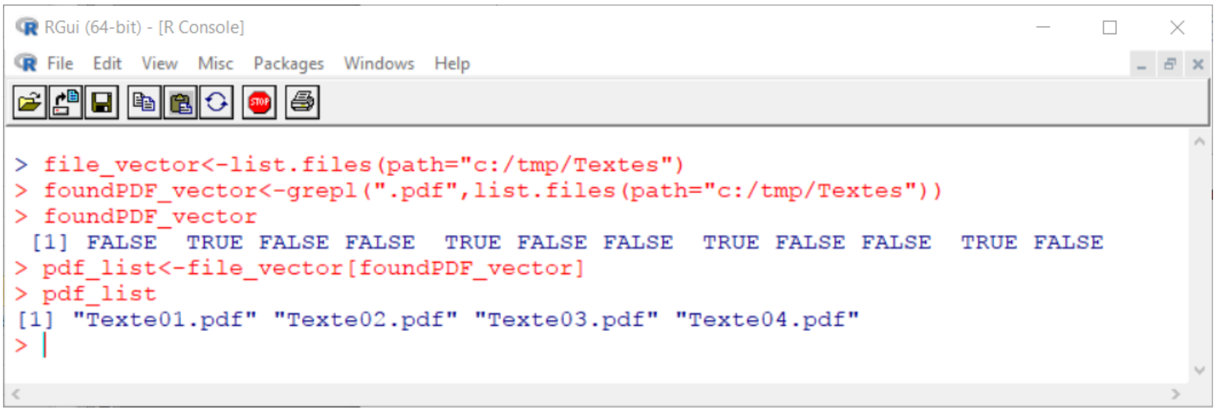
Exemple 427.: Sélectionner des fichiers précis par leur extension

R 3.4.2

D'abord nous lisons le contenu d'un dossier:



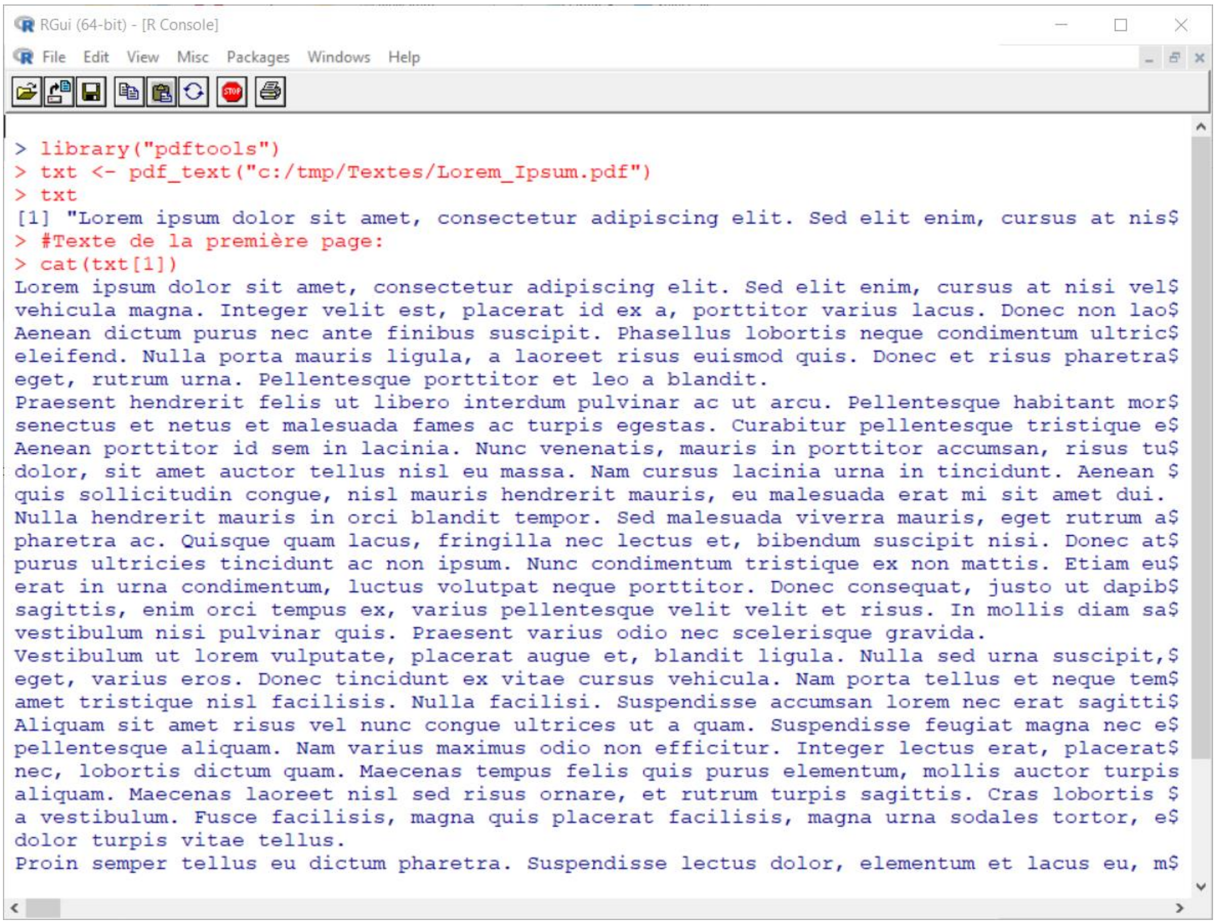
Le problème ici est que nous voulons que les *.pdf. Nous utilisons alors la fonction **grepl( )** comme suit:



Exemple 428.: Lire le contenu d'un unique PDF

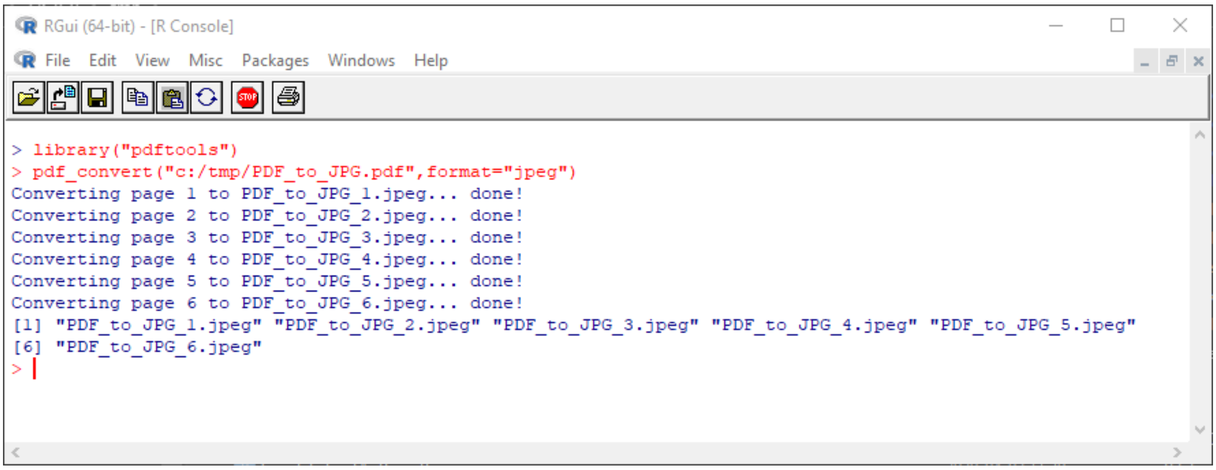
R 3.4.3

Pour cet exercice de style, le package **pdftools** et sa fonction **pdf_text( )** sont tout à fait appropriés:

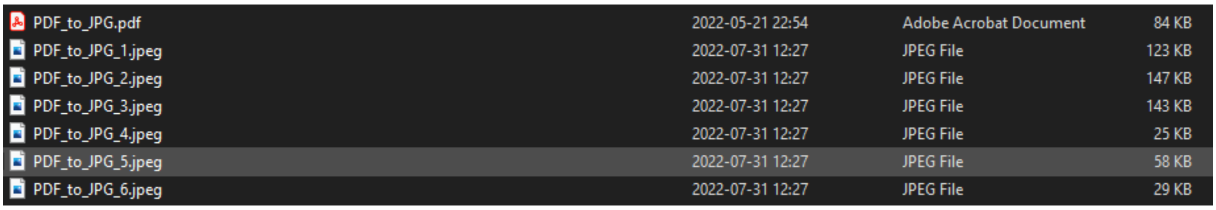


Avec ce que nous avons vu précédemment, il est bien sûr alors possible d'itérer sur une série de fichiers listés dans un vecteur et d'agréger tous les textes!

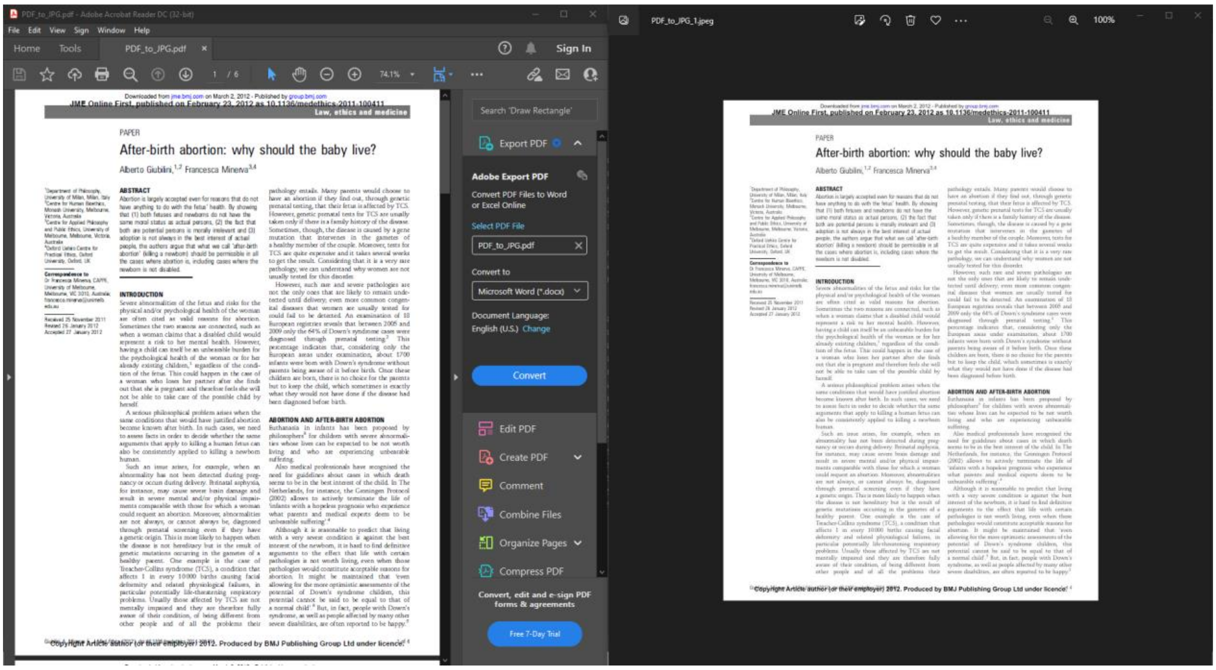
Si jamais vous avez besoin d'extraire le contenu d'un PDF au format JPG, pour utiliser du OCR, vous pouvez toujours utiliser le package **pdftools** et sa fonction **pdf_convert( )**:



Ce qui donne:



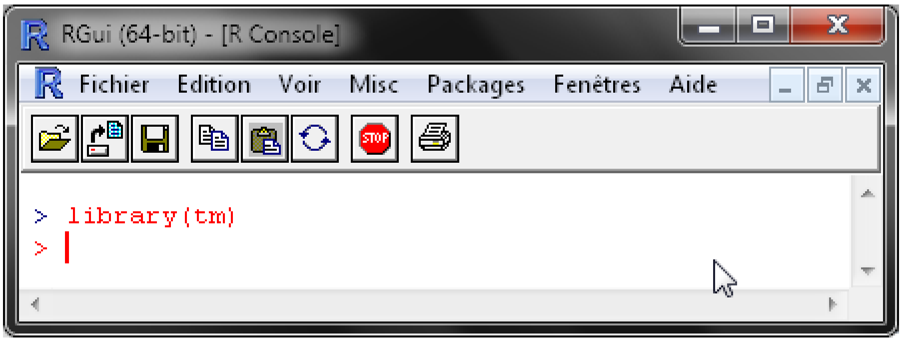
Et on peut comparer le PDF à gauche d'une des images extraites à droite:



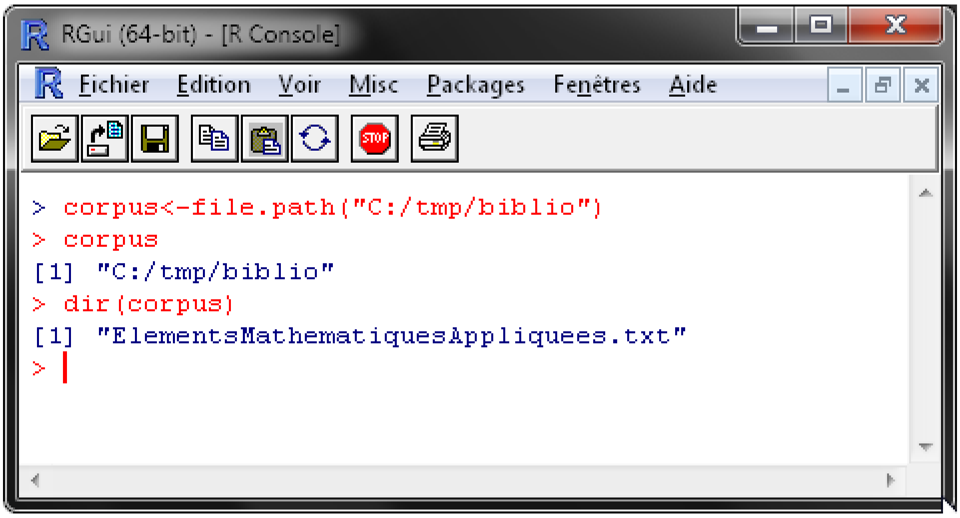
Exemple 429.: Créer et nettoyer le corpus (package tm/SnowballC)

R 3.0.2

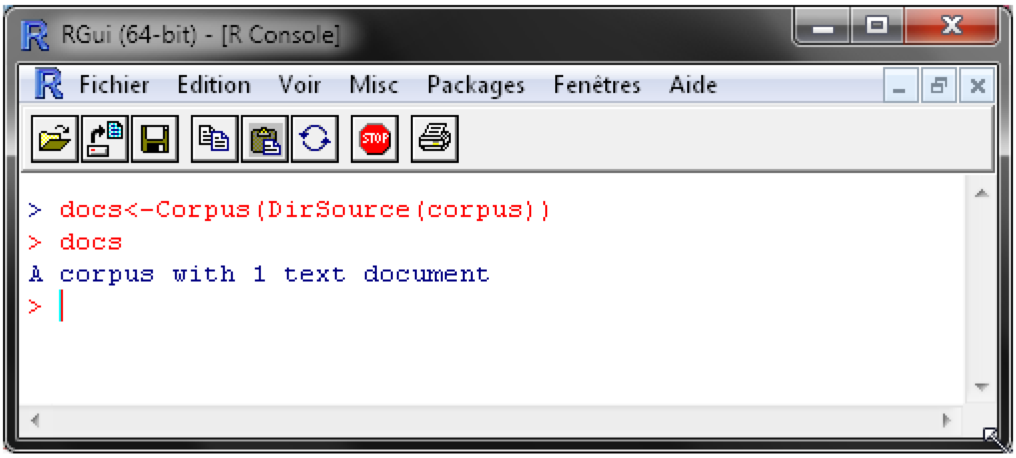
D'abord nous installons et chargeons le package **tm**:



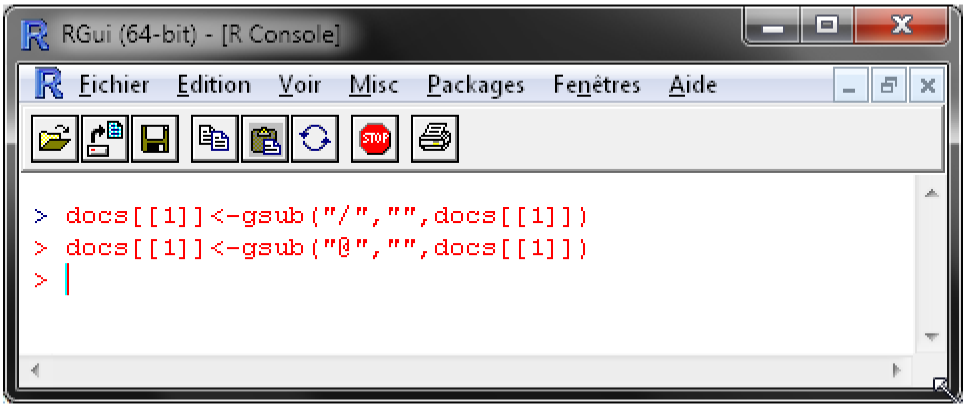
Nous spécifions le dossier avec tous les fichiers à analyser:



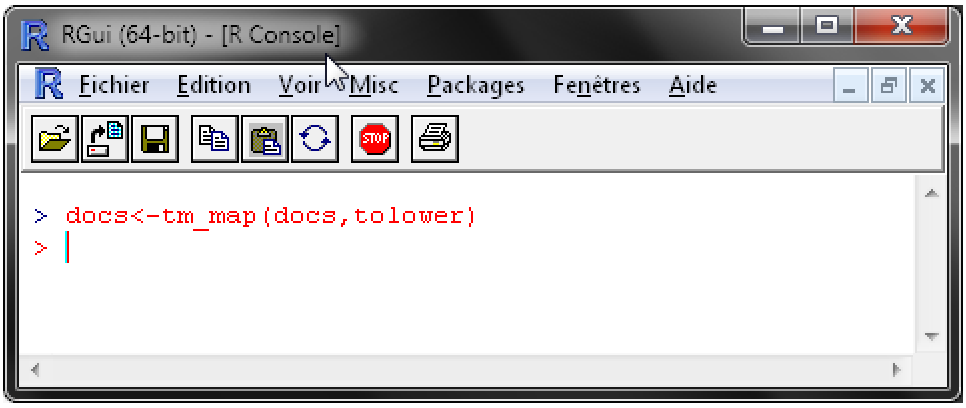
Ensuite, nous commençons seulement à utiliser le package **tm** en construisant d'abord un corpus à l'aide de la commande **Corpus( )** :



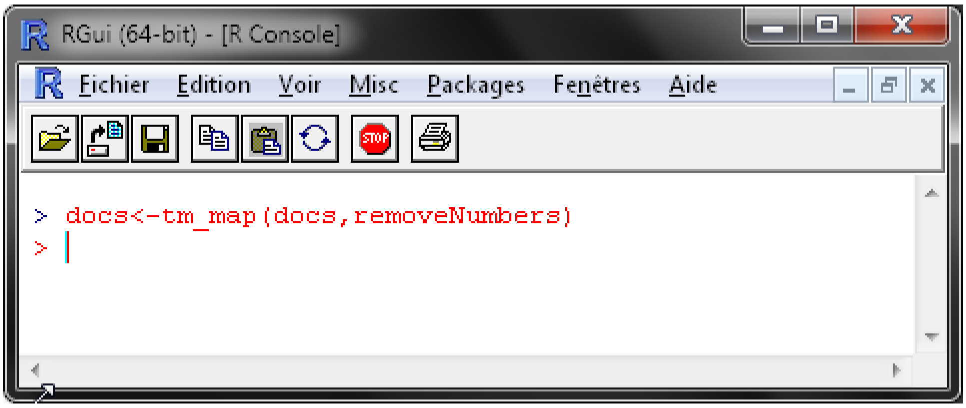
On peut nettoyer le document avec la commande **gsub( )** ou également pour remplacer des termes ou abréviations par autre chose:



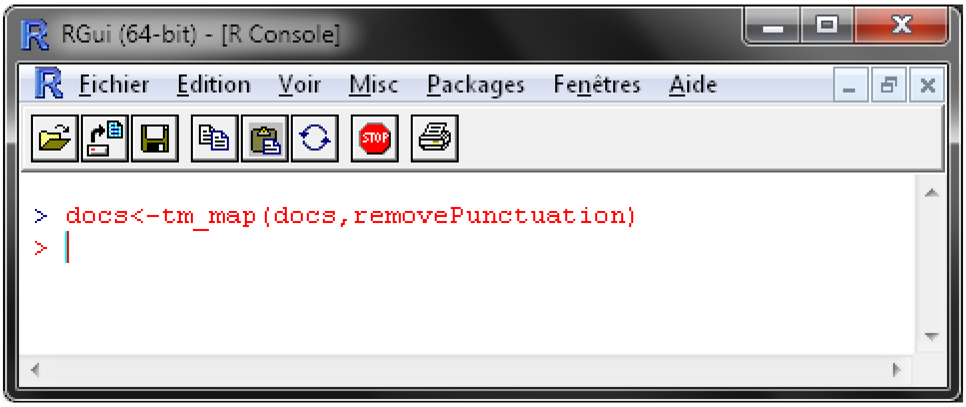
On peut mettre le contenu de l'ensemble du corpus en minuscules en utilisant de multiples paramètres avec la commande **tm_map( )** :



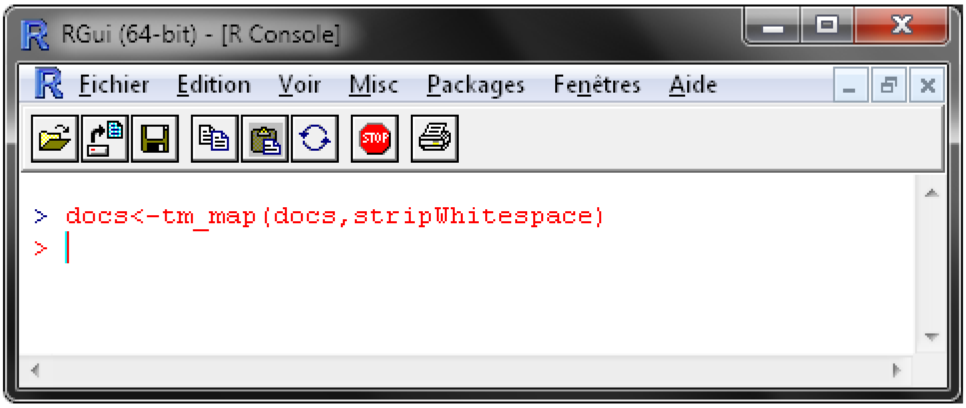
Supprimer tous les nombres:



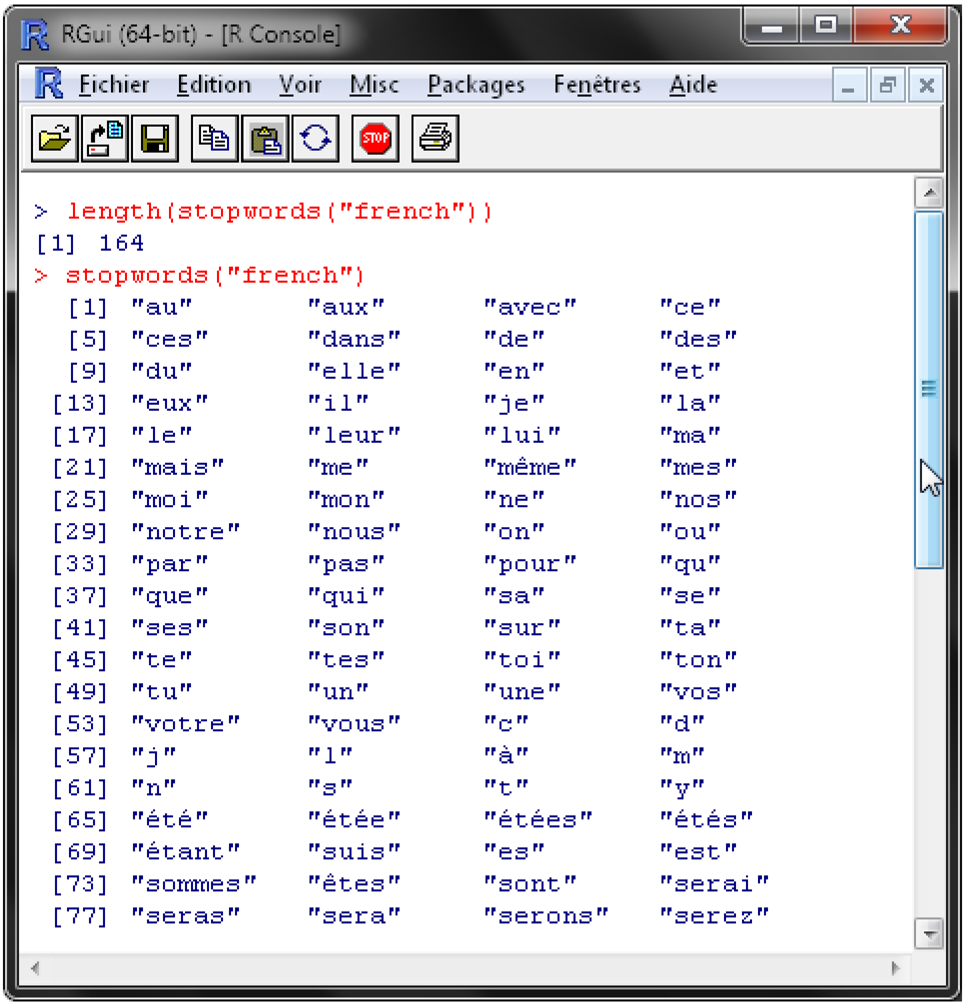
Supprimer la ponctuation:



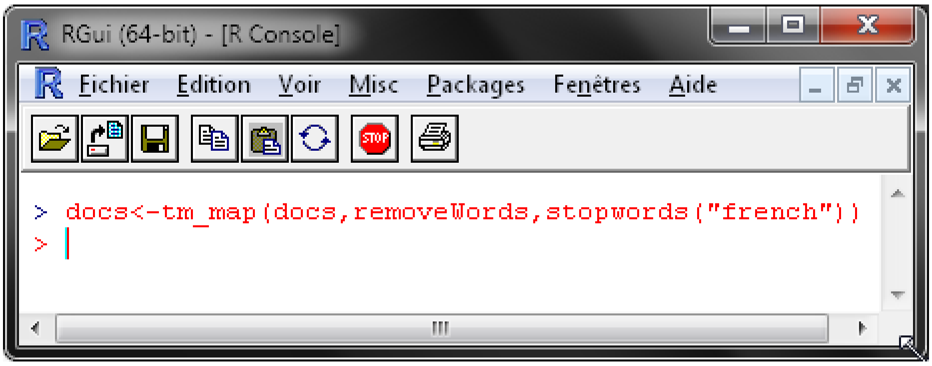
Enlever les espaces vides en trop:



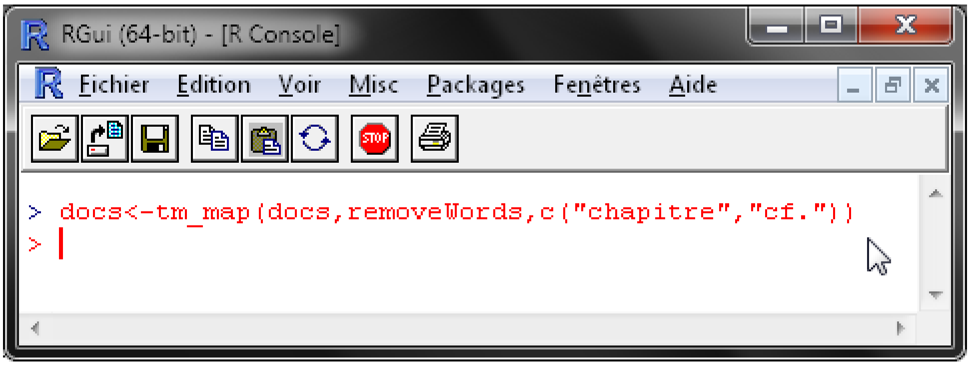
Il y a possibilité de supprimer aussi les mots ultra courants d'une langue donnée et qui n'apportent rien à une analyse. Voyons d'abord quels sont ces mots dans le cas de la langue française:



Nous nettoyons donc le corpus avec ce minimum syndical...:



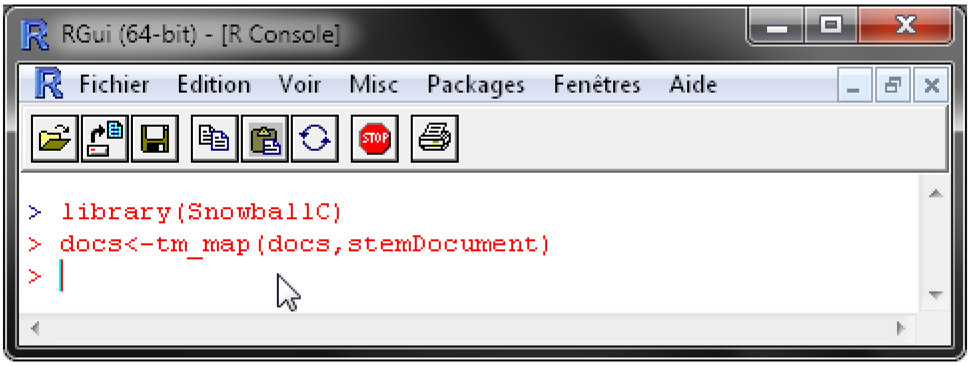
ou nous pouvons aussi enlever nos propres termes (en réalité il vaut mieux avoir un fichier externe avec l'ensemble des termes considérés comme inutiles car en français il y en a un tel nombre que la commande *stopwords* utilisée précédemment est quasiment inutile):



Arrivé à ce point nous ne pouvons pas encore lancer une analyse du corpus. Nous allons devoir effecteur un autre nettoyage (voir ci-après):

De nombreux mots sont identiques à part qu'ils sont au singulier/pluriel ou accordés au féminin/masculin.

Pour ne pas avoir à nettoyer cela manuellement (ce travail étant laborieux). Nous pouvons le laisser au package **SnowballC** mais mixé avec la commande **tm_map( )** du package **tm** (pas besoin de spécifier la langue il s'en occupe) cependant le résultat du travail est à ce jour (2014-07-07) mauvais en français:

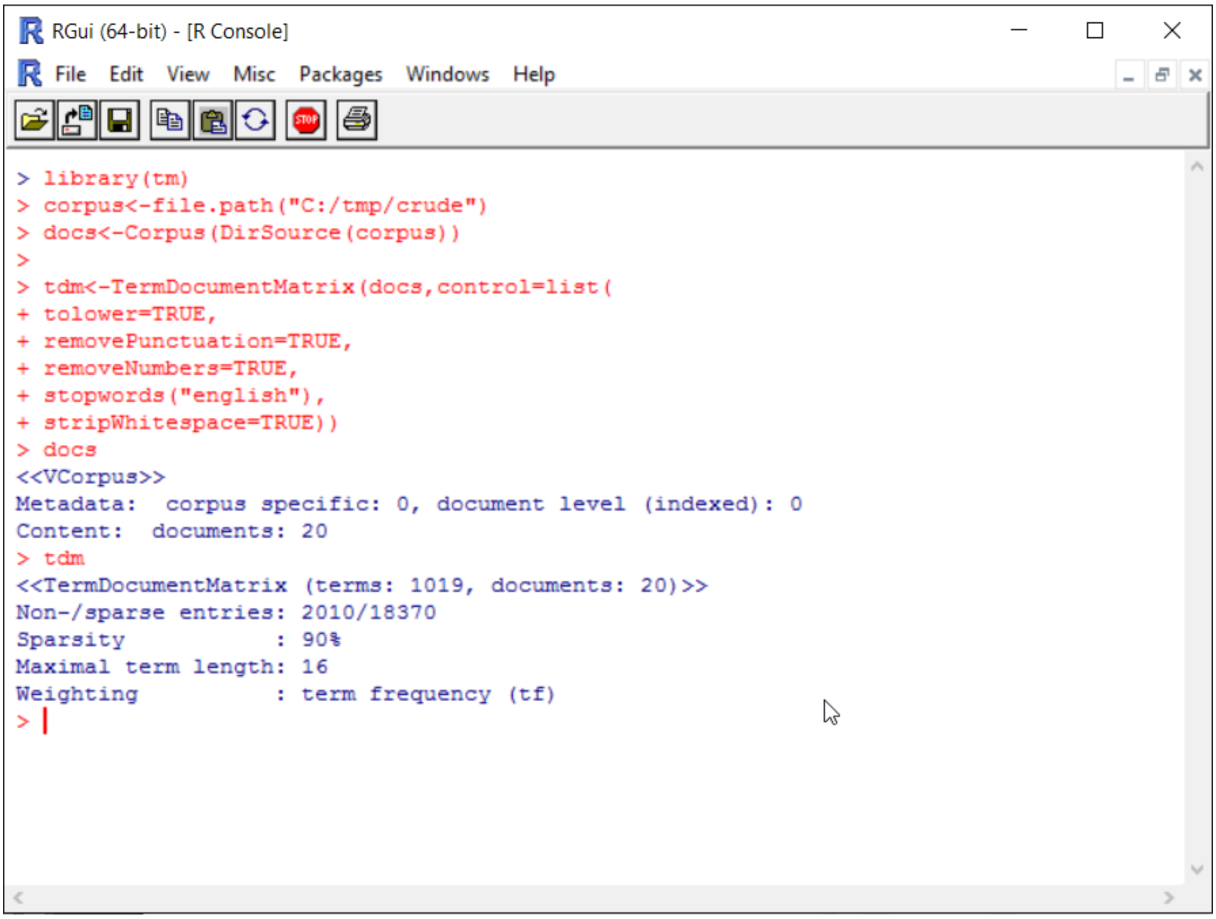


Exemple 430.: Obtenir la matrice d'adjacence d'un corpus

R 3.2.1

Nous avons vu en théorie des graphes ce qu'était une matrice d'adjacence. Nous allons voir comment construire cette dernière dans R avec le corpus précédent.

D'abord nous construisons notre corpus:



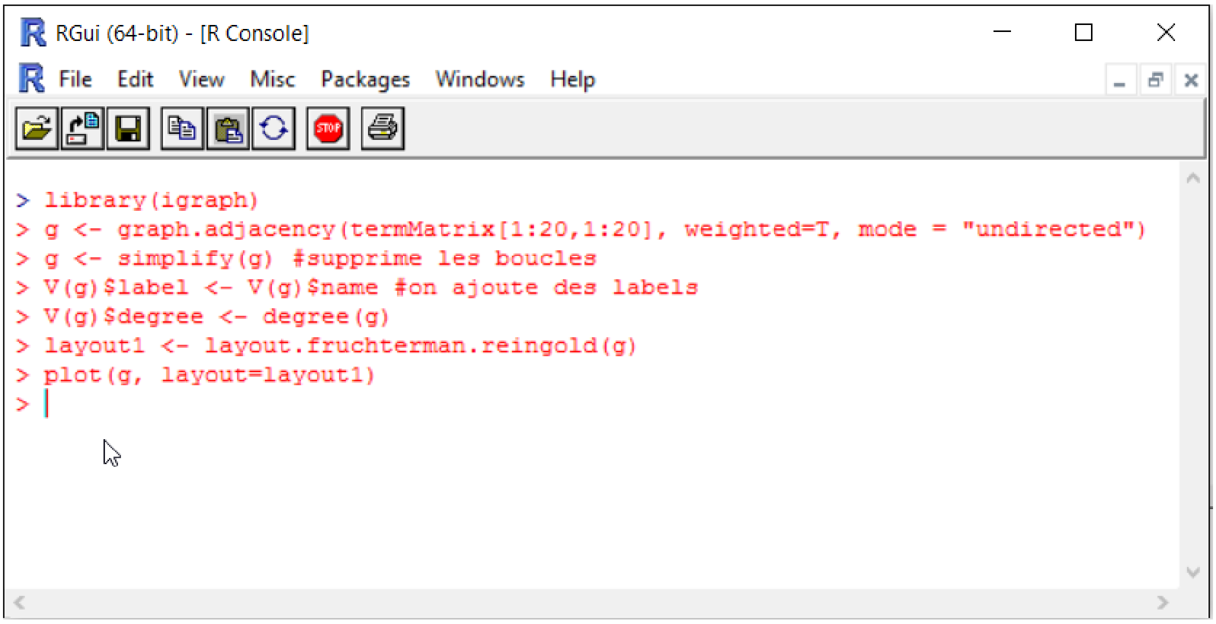
Ensuite nous visualisons la matrice de termes par document du corpus:

```
> termDocMatrix <- as.matrix(tdm)
> termDocMatrix[1:20,1:10]#20 premières lignes des dix premiers fichiers
           Docs
Terms        1.txt 10.txt 11.txt 12.txt 13.txt 14.txt 15.txt 16.txt 17.txt 18.txt
  abdulaziz      0      0      0      0      0      0      0      0      0      0
  ability        0      0      1      0      0      0      0      0      0      0
  able           0      0      0      0      0      0      0      0      0      0
  about          0      2      0      0      1      0      0      2      2      0
  above          0      2      0      0      0      1      0      0      0      0
  abroad         0      0      0      0      0      0      0      0      0      0
  accept         0      0      0      0      0      0      0      0      1      0
  accord         0      5      1      0      2      0      0      0      0      0
  according      0      0      0      0      0      0      0      0      0      0
  across         0      0      0      0      0      0      1      0      0      0
  activity       0      0      0      0      0      0      0      0      0      0
  add            0      0      0      0      0      0      0      0      0      0
  added          0      0      0      0      0      0      0      0      0      0
  address        0      0      0      0      0      0      0      0      0      0
  addressed      0      0      0      0      0      0      0      0      0      0
  adherence      0      1      0      0      0      0      0      0      0      0
  adhering       0      1      0      0      1      0      0      0      0      0
  advantage      0      0      0      0      0      0      0      0      0      0
  advisers       0      0      0      0      0      0      0      0      0      0
  after          0      0      1      0      0      1      2      0      0      0
>
```

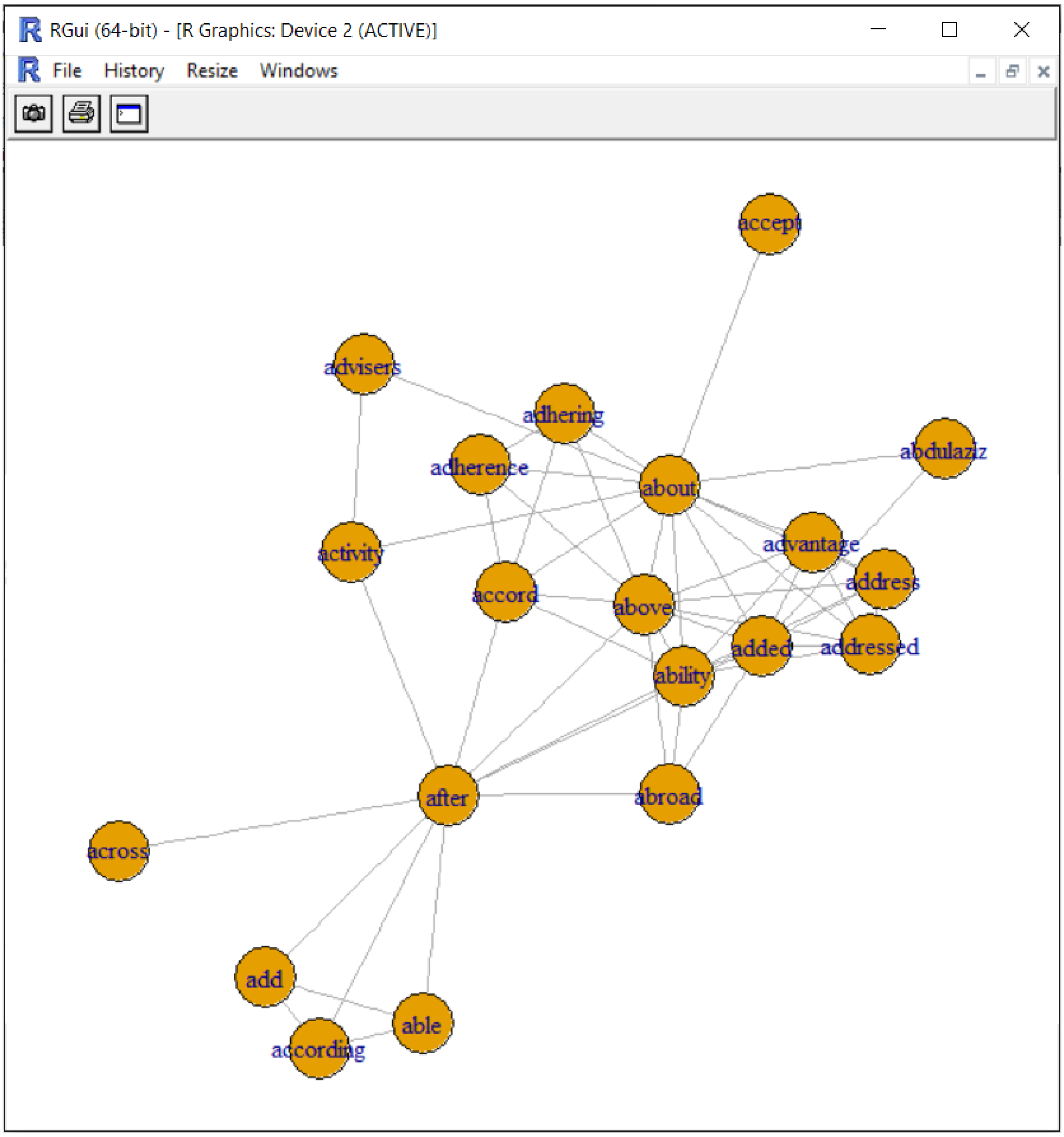
Nous construisons et affichons une partie de la matrice d'adjacence:

```
> termDocMatrix[termDocMatrix>=1] <- 1 #transformation en matrice boléenne
> termMatrix <- termDocMatrix %*% t(termDocMatrix) #transformation en matrice d'adjacence
> termMatrix[1:20,1:10]
           Terms
Terms        abdulaziz ability able about above abroad accept accord according across
  abdulaziz          1       0    0     1     0      0      0      0         0      0
  ability            0       3    0     1     2      1      0      1         0      0
  able               0       0    1     0     0      0      0      0         1      0
  about              1       1    0     8     2      0      1      2         0      0
  above              0       2    0     2     4      1      0      1         0      0
  abroad             0       1    0     0     1      1      0      0         0      0
  accept             0       0    0     1     0      0      1      0         0      0
  accord             0       1    0     2     1      0      0      3         0      0
  according          0       0    1     0     0      0      0      0         1      0
  across             0       0    0     0     0      0      0      0         0      1
  activity           0       0    0     1     0      0      0      0         0      0
  add                0       0    1     0     0      0      0      0         1      0
  added              1       2    0     2     2      1      0      0         0      0
  address            0       1    0     1     1      0      0      0         0      0
  addressed          0       1    0     1     1      0      0      0         0      0
  adherence          0       0    0     1     1      0      0      1         0      0
  adhering           0       0    0     2     1      0      0      2         0      0
  advantage          0       1    0     1     1      0      0      0         0      0
  advisers           0       0    0     1     0      0      0      0         0      0
  after              0       2    1     0     2      1      0      1         1      1
>
```

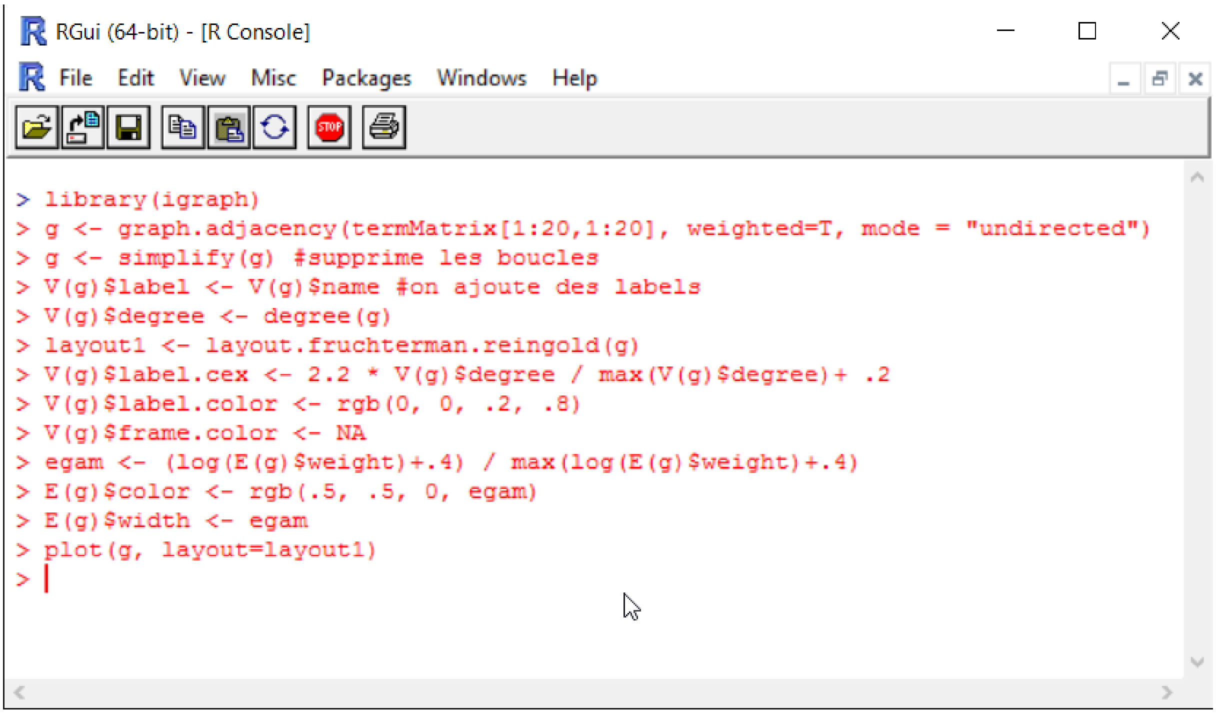
Nous faisons un premier type de graphe de réseau:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(igraph)
> g <- graph.adjacency(termMatrix[1:20,1:20], weighted=T, mode = "undirected")
> g <- simplify(g) #supprime les boucles
> V(g)$label <- V(g)$name #on ajoute des labels
> V(g)$degree <- degree(g)
> layout1 <- layout.fruchterman.reingold(g)
> plot(g, layout=layout1)
>

qui donne:



ou avec un visuel amélioré:



Ce qui donne en toute proportion ;-) :

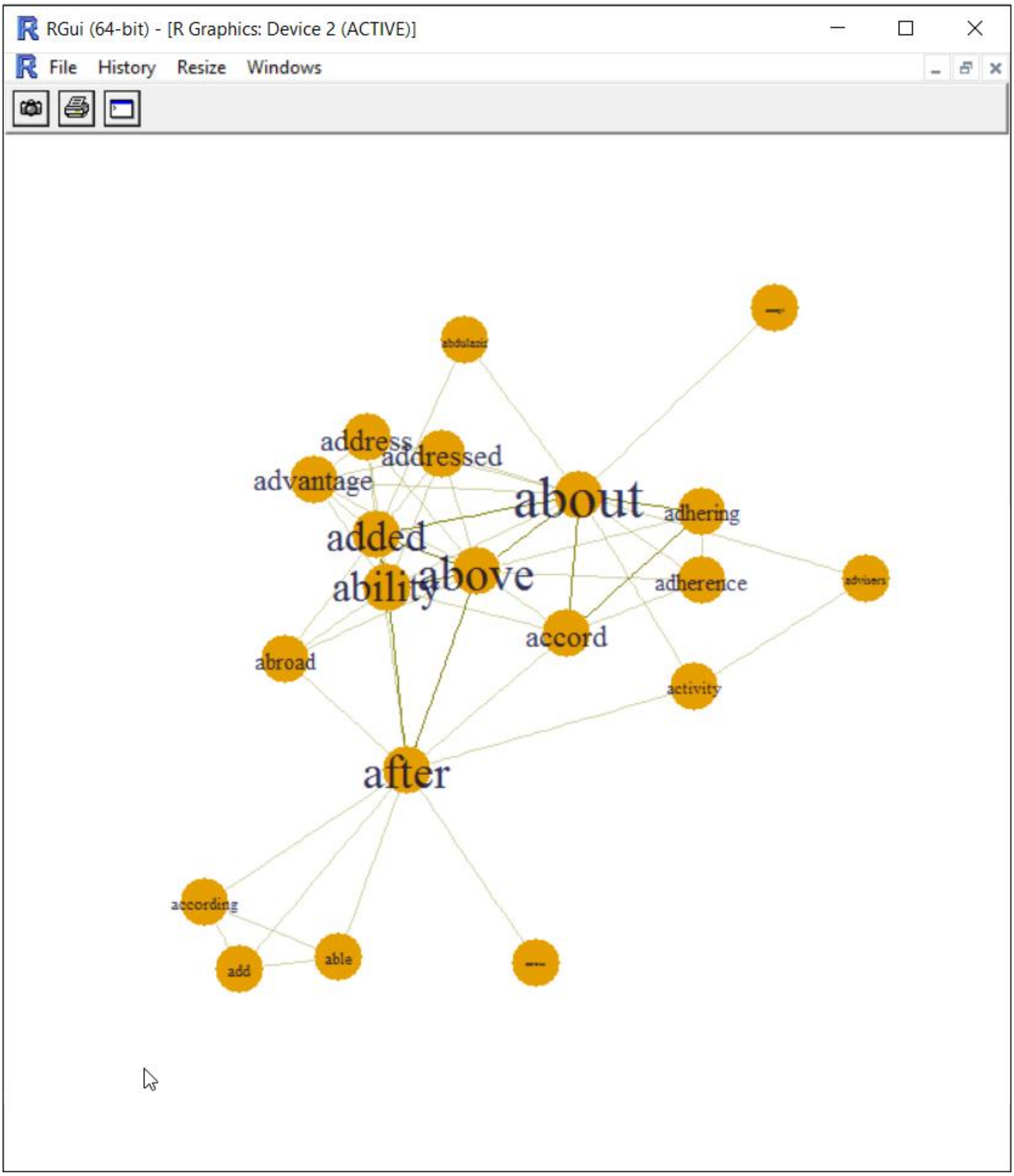
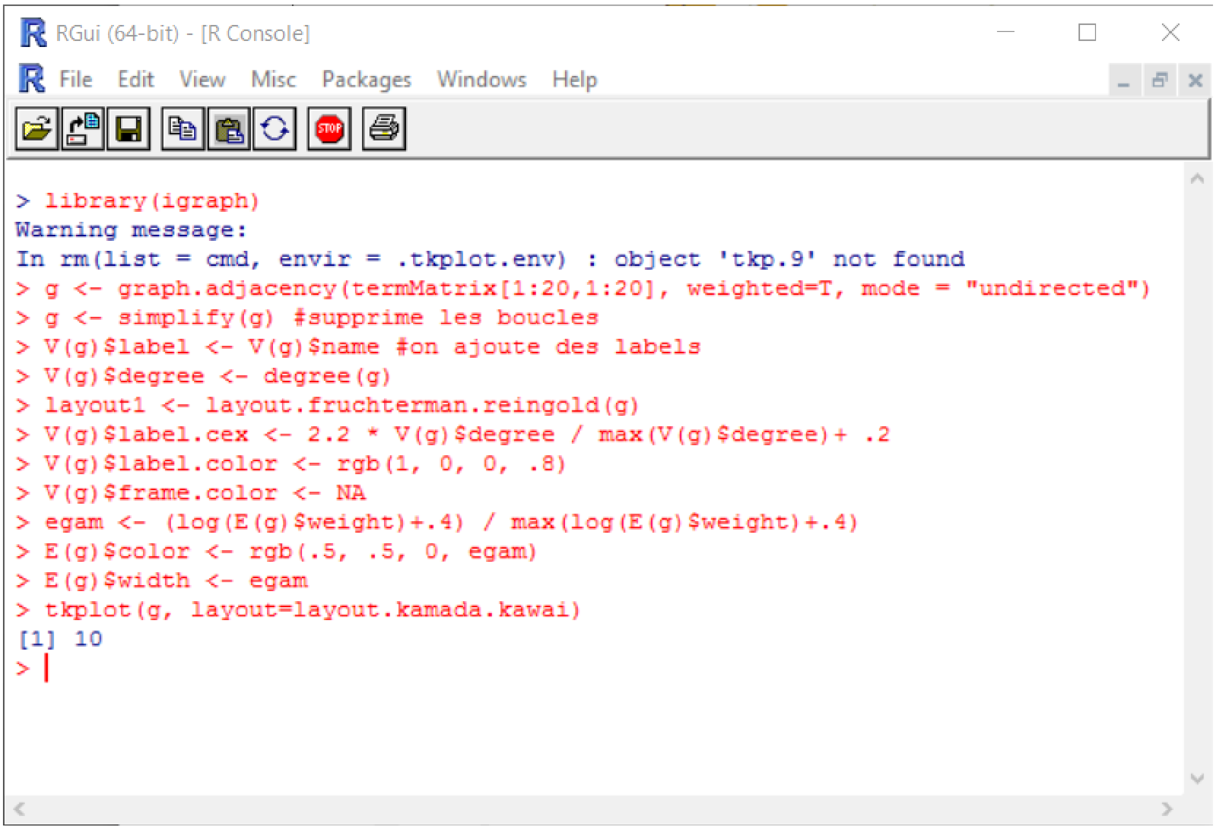
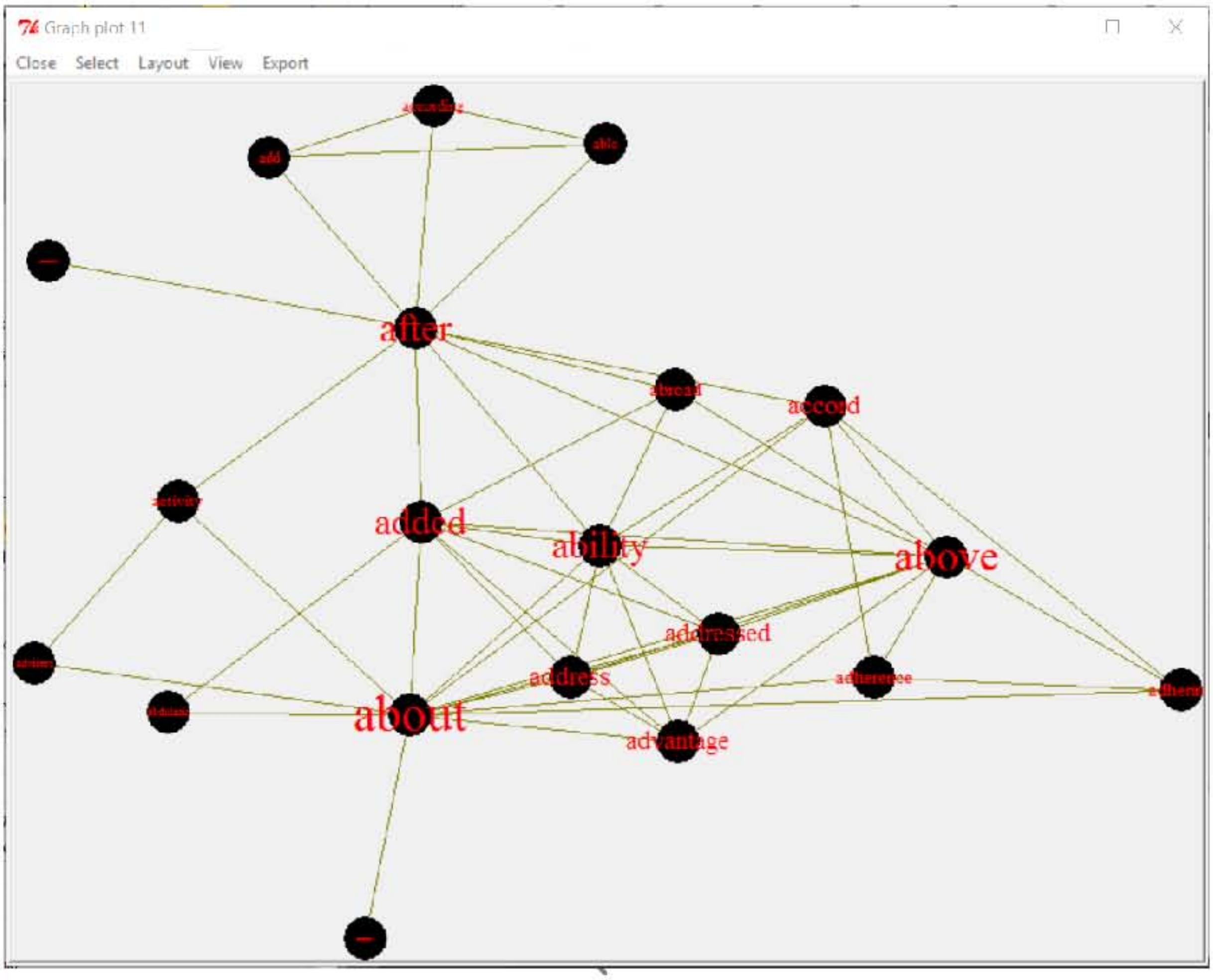


Figure 340 Graph de réseau de mots

Sinon pour avoir le fameux cas interactif que tout le monde à vu dans les reportages sur Data Journalists des Panama Papers il suffit d'utiliser la commande:


RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library(igraph)
Warning message:
In rm(list = cmd, envir = .tkplot.env) : object 'tkp.9' not found
> g <- graph.adjacency(termMatrix[1:20,1:20], weighted=T, mode = "undirected")
> g <- simplify(g) #supprime les boucles
> V(g)$label <- V(g)$name #on ajoute des labels
> V(g)$degree <- degree(g)
> layout1 <- layout.fruchterman.reingold(g)
> V(g)$label.cex <- 2.2 * V(g)$degree / max(V(g)$degree)+ .2
> V(g)$label.color <- rgb(1, 0, 0, .8)
> V(g)$frame.color <- NA
> egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)
> E(g)$color <- rgb(.5, .5, 0, egam)
> E(g)$width <- egam
> tkplot(g, layout=layout.kamada.kawai)
[1] 10
>

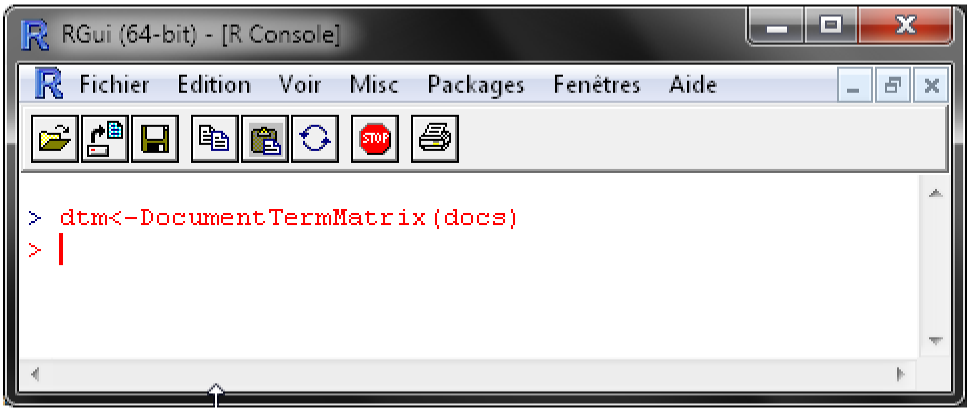
Ce qui donne:


Graph plot 11
Close Select Layout View Export
after
abroad
accord
activity
added
ability
above
addressed
address
adherence
about
advantage

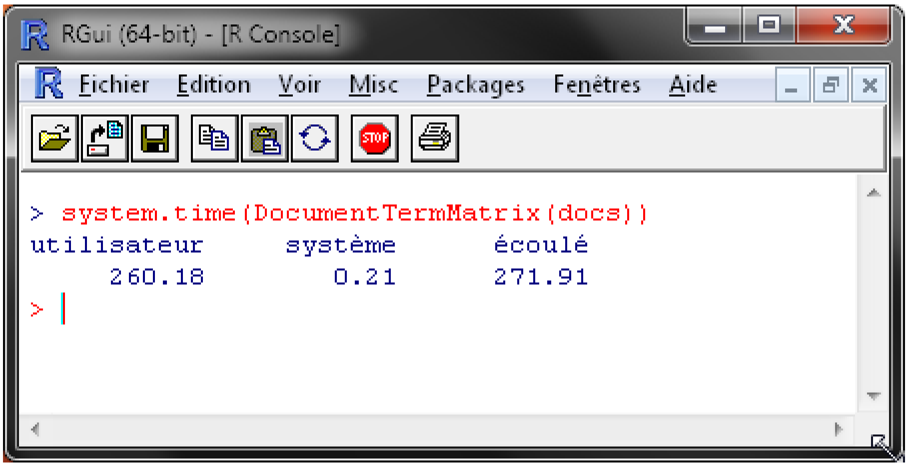
Exemple 431.: Analyser le corpus (package tm/wordcloud)

R 3.0.2

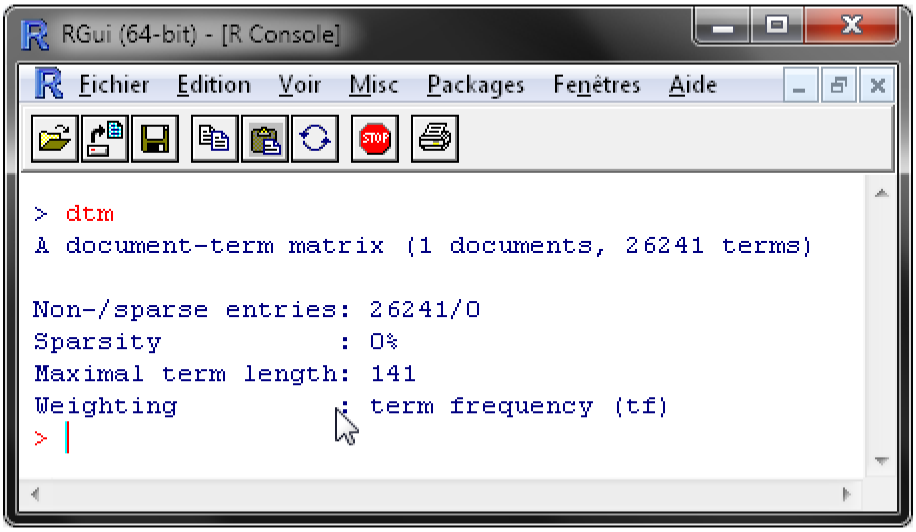
Une fois le nettoyage terminé, pour analyser le corpus nous utilisons la commande **DocumentTermMatrix( )** du package **tm**:



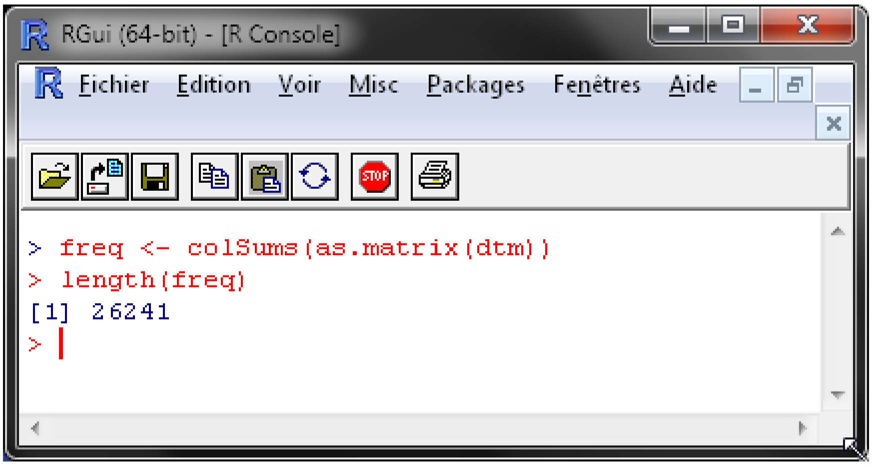
Voici par exemple le temps nécessaire pour analyser la version texte brute de mon livre (pour que le lecteur ait une idée du temps de traitement de fichiers volumineux):



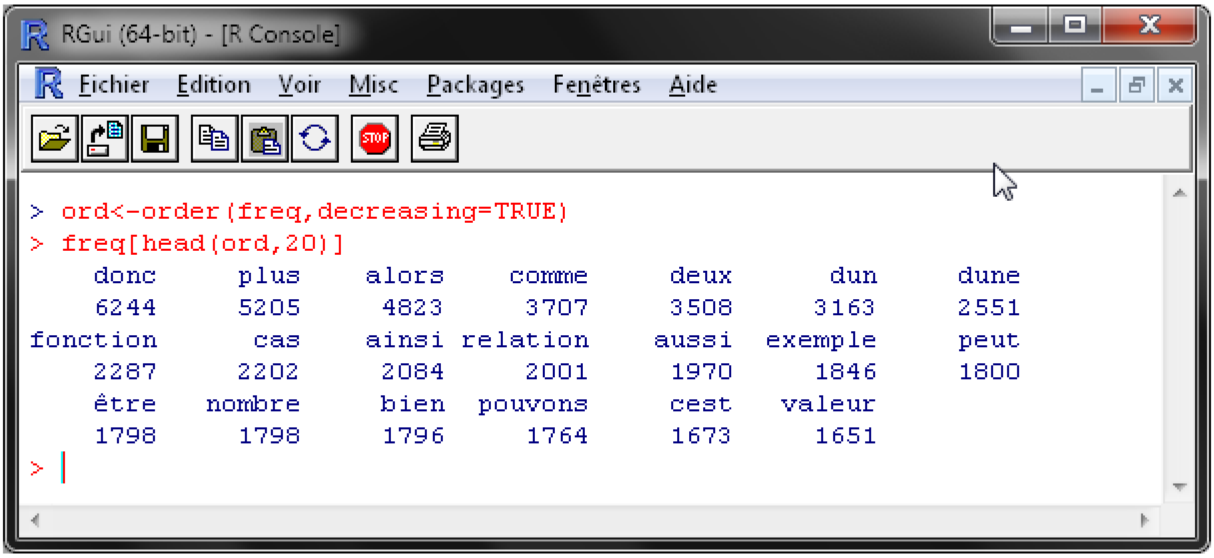
Ensuite, nous affichons un résumé de la variable créée pour voir le nombre de termes analysés:



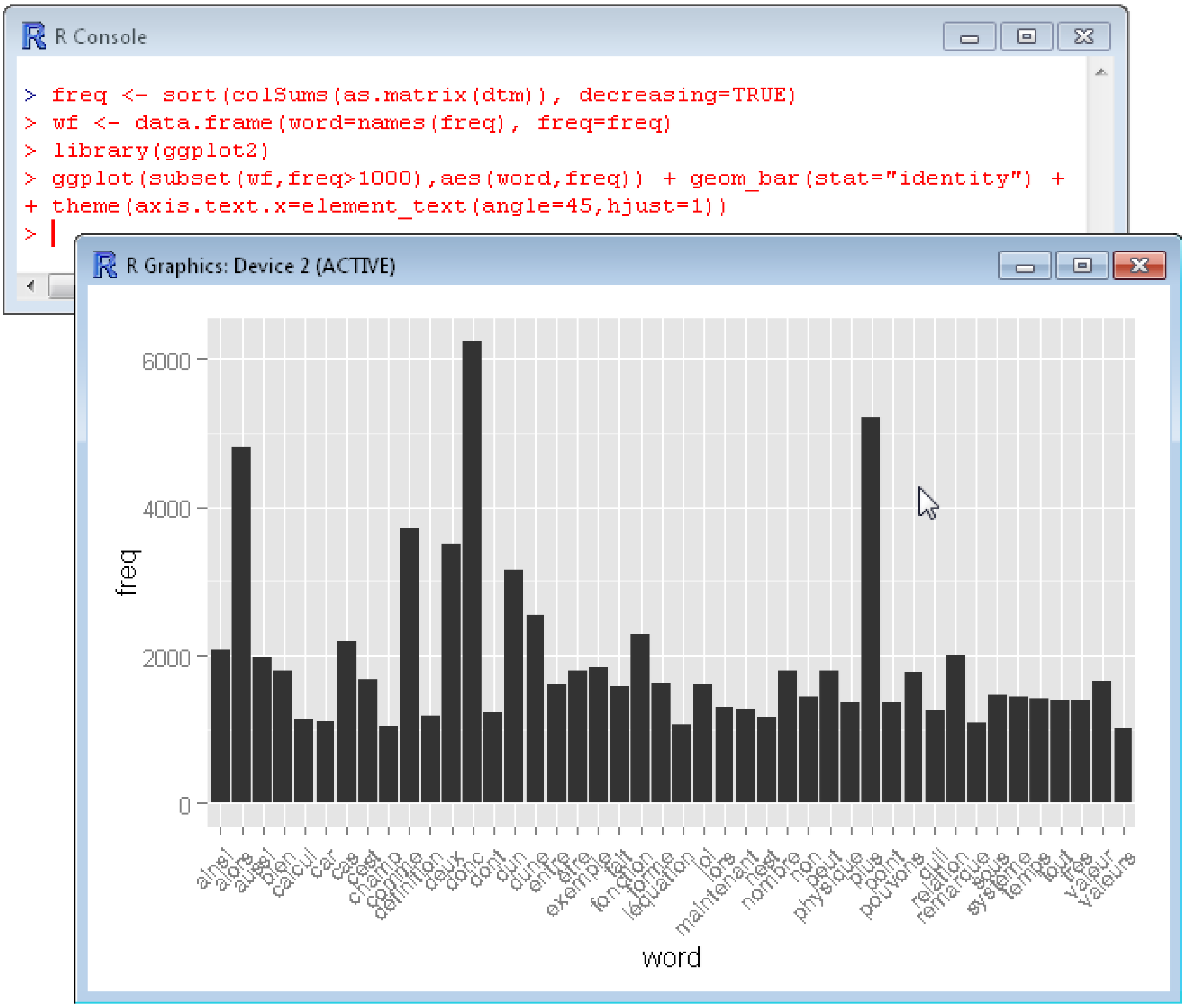
Ensuite, nous transformons la variable *dtm* en tant que matrice et faisons la somme des colonnes afin d'avoir la fréquence d'apparition de chaque mot:



Ensuite, avec les commandes classiques natives de R nous pouvons trier et afficher les mots les plus fréquents (remarquez les apostrophes qui n'apparaissent pas car le package **tm** ne gère par les apostrophes ' mais veut une apostrophe stylisée ´):



Nous pouvons ensuite avec le package **ggplot** faire un histogramme des termes les plus fréquents:



Maintenant fais un nuage de terme à l'aide du package **wordcloud** et de sa commande... **wordcloud( )** :

R Console

```
> #le paramètre scale permet de régler empirique le ratio de taille des mots clés
> #le paramètre rot.per permet de choisir le % de mots verticaux
> library(wordcloud)
> wordcloud(names(freq),freq,max.words=200,colors=brewer.pal(6, "Dark2"),
+ scale=c(5, .1),rot.per=0.2)
```

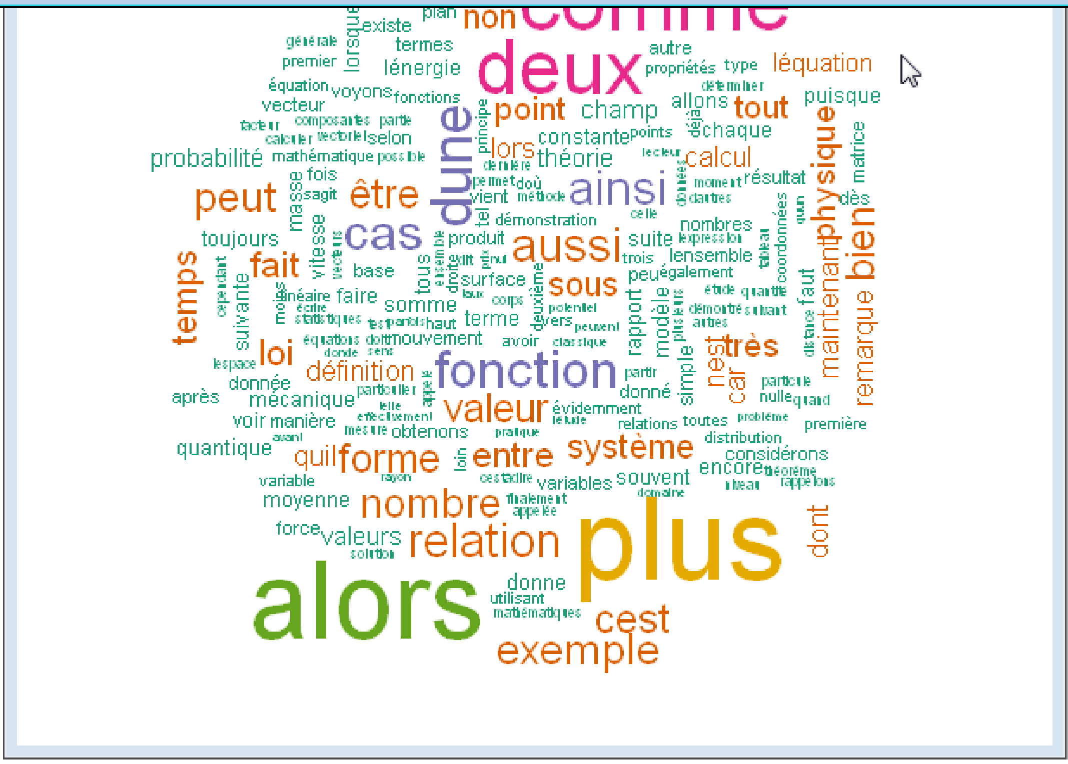


Figure 341 Graph de nuage de mots

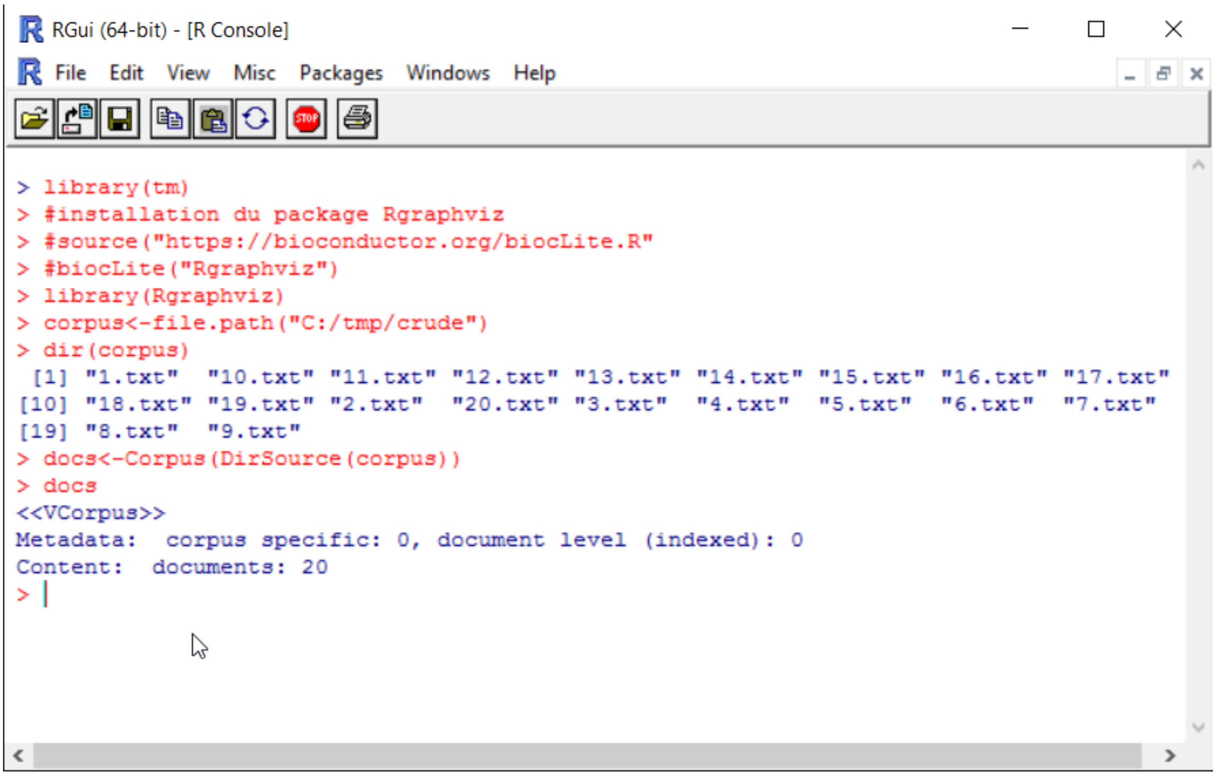
Exemple 432.: Réseau de termes corrélés

R 3.2.1

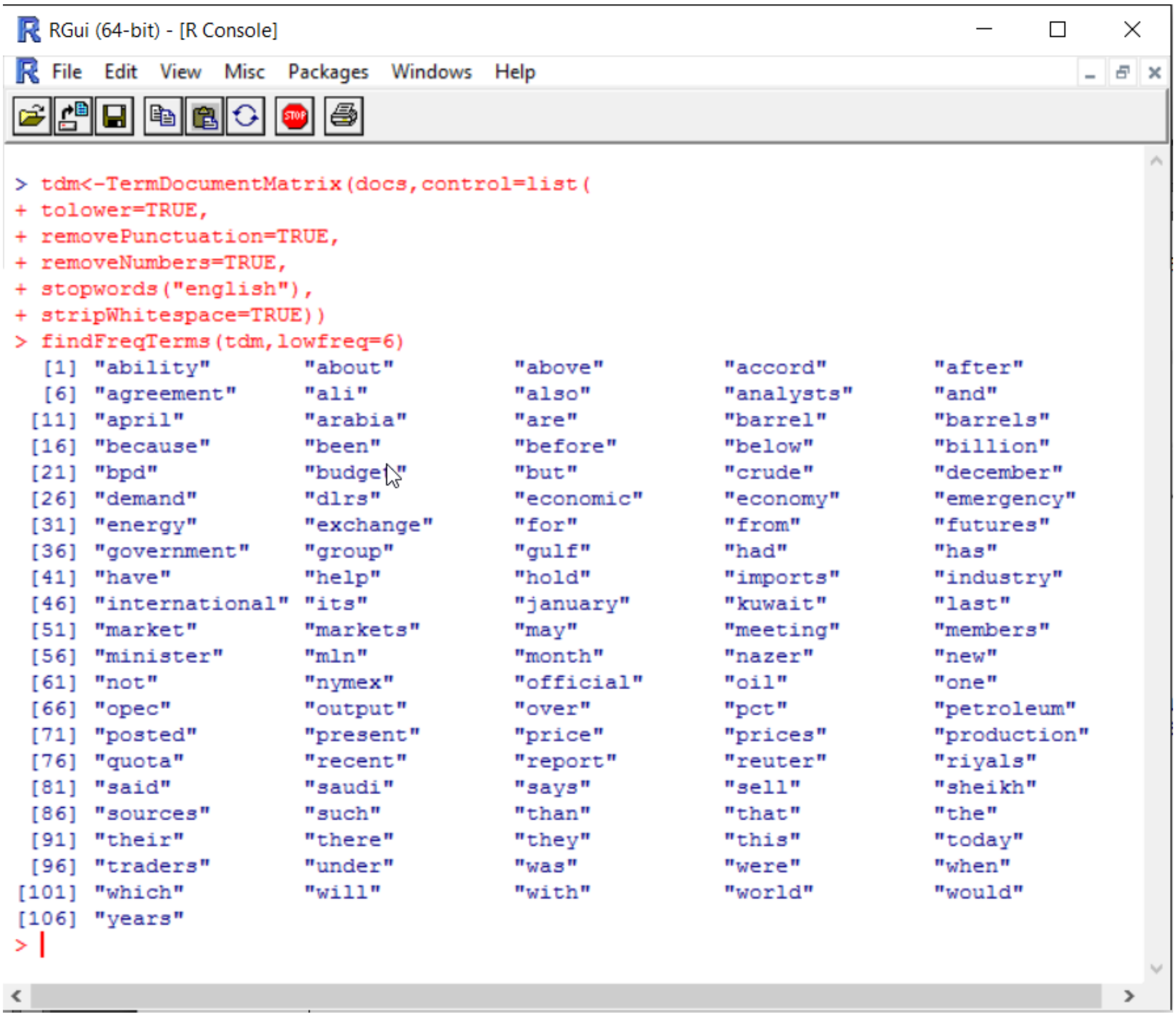
À l'époque des Wikileaks, Swissleaks, Luxembourgleaks et Panama Papers, le data journalisme prendra probablement un place de plus en plus importante vis-à-vis du défis d'analyse ou de corréler les données de plusieurs TeraBytes de fichiers ou courriels.

Voyons un exemple simple d'analyse de réseau de mots avec un corpus ayant 20 documents.

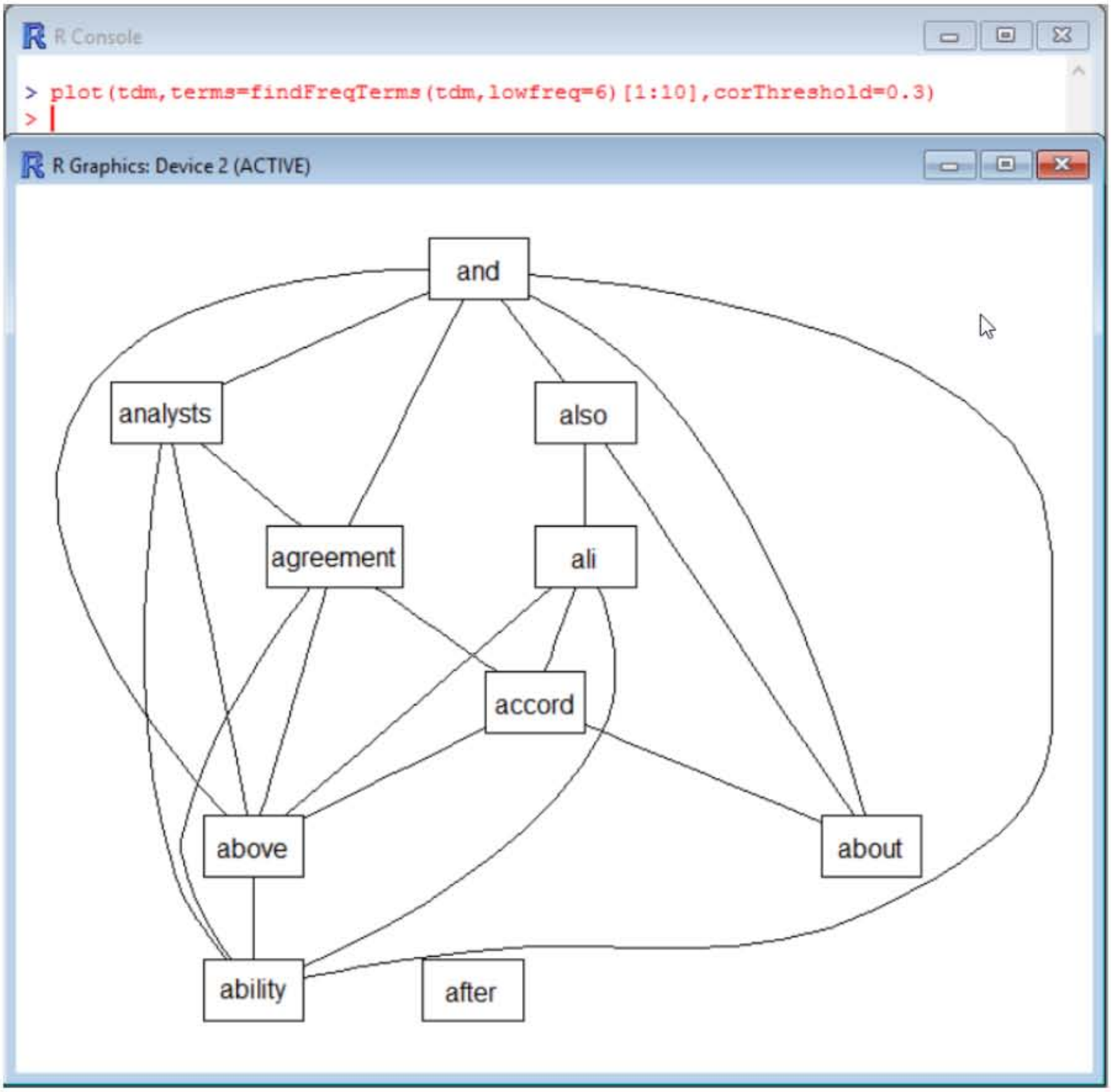
Nous chargeons les librairies, les documents et créons le corpus:



Nous construisons ensuite la matrice des termes:



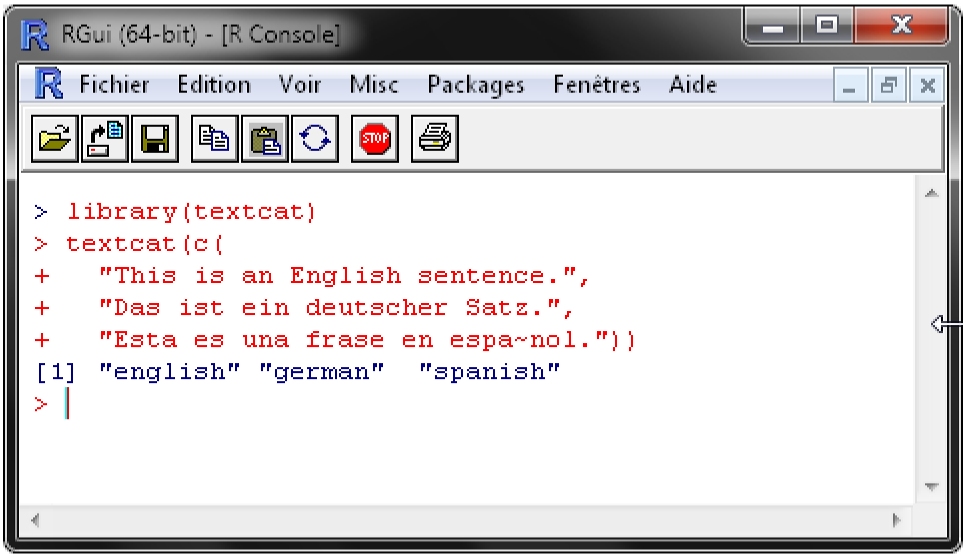
Enfin, nous pouvons faire un plot du réseau:


R Console
> plot(tdm,terms=findFreqTerms(tdm,lowfreq=6)[1:10],corThreshold=0.3)
>
R Graphics: Device 2 (ACTIVE)
and
analysts
also
agreement
ali
accord
above
about
ability
after

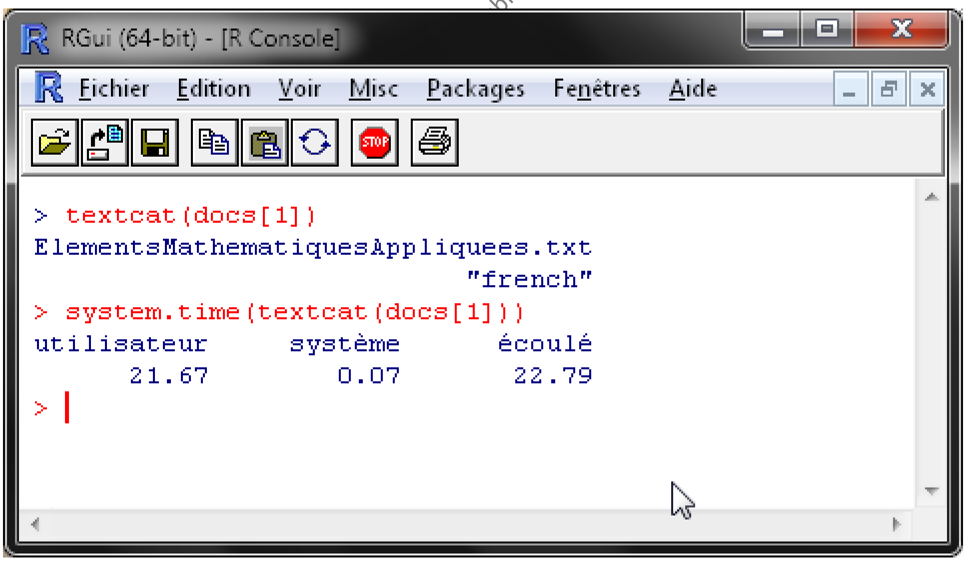
Exemple 433.: Détecter la langue (package textcat)

R 3.0.2

Nous pouvons détecter les langues avec la commande **textcat** du package **textcat( )** nous avons avec un exemple simpliste:



Ou en utilisant notre texte de tout à l'heure (j'ai aussi mis le temps afin que le lecteur puisse se faire une idée):

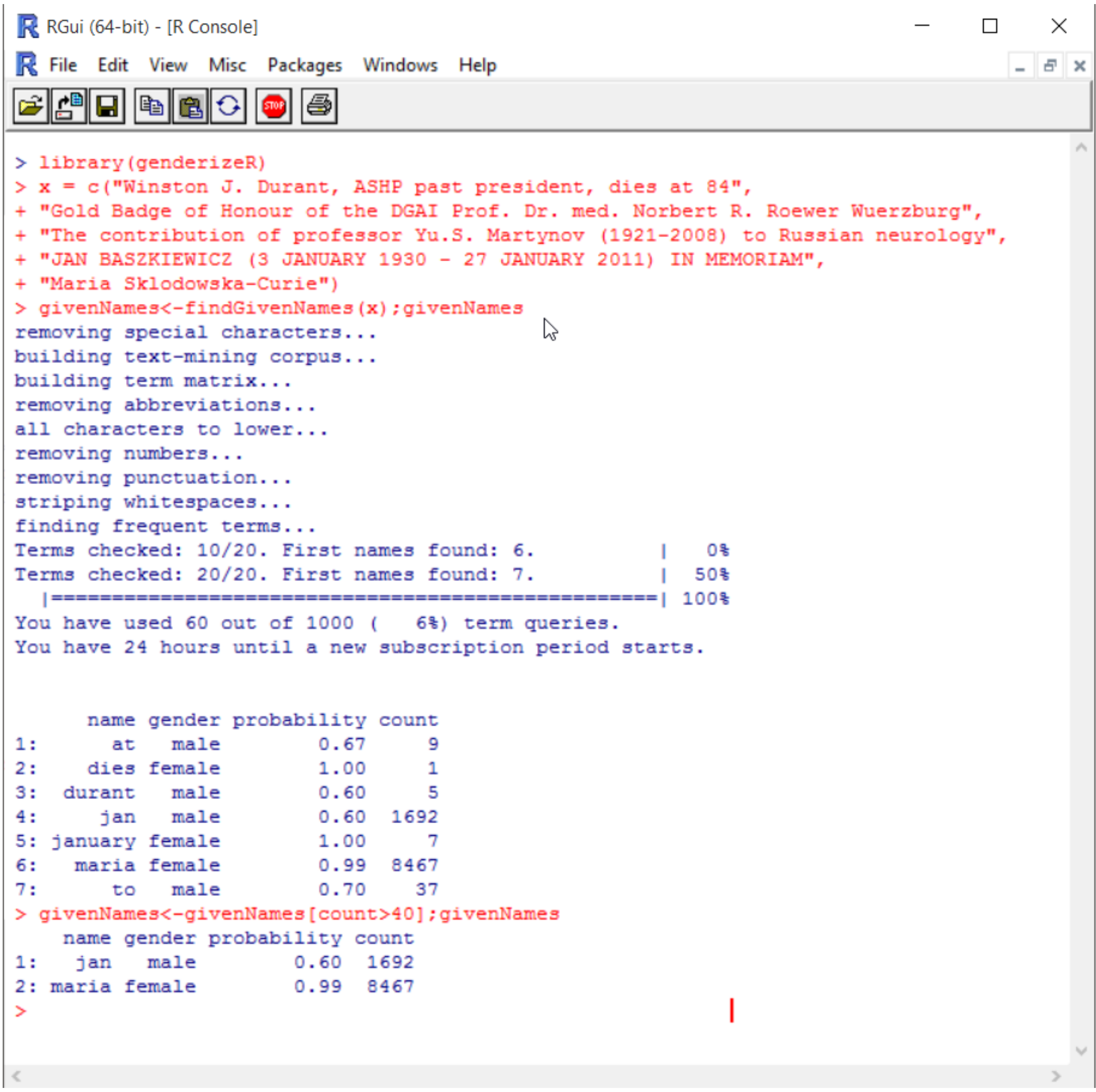


Exemple 434.: Détecter les genres d'un vecteur de textes

R 3.2.1.

Le package **genderizeR** va intéresser probablement plus d'un praticien. Il utilise un API qui se connecte à un site web mais qui autorise par jour un maximum de 1'000 identifications de genres de prénoms ou de noms. Il peut être appliqué à un vecteur aussi bien qu'à un corpus.

Voyons cela avec les fonctions **findGivenNames( )** et **givenNames( )** :



Exemple 435.: Tableau de contingence lexical

R 3.2.1.

Nous partons du fichier texte suivant:

```
Millésime;Vin;Texte
Millésime 1;Vin 1;Acidité, fruit
Millésime 2;Vin 1;Fruité, léger
Millésime 1;Vin 1;Bois
Millésime 2;Vin 1;Acidité
Millésime 1;Vin 2;Sucré
Millésime 2;Vin 2;Sucré liquoreux
Millésime 1;Vin 2;Léger, fruité
Millésime 2;Vin 2;Sucré léger
```

Nous utilisons alors la fonction **textual( )** du package **FactoMineR**:

```
> mydata<-read.csv("C:/tmp/Vins.csv",header=T,sep=";")
> library(FactoMineR)
> textual(mydata,num.text=3,contingence.by=list(2,1:3),sep.word=(", "),maj.in.min=TRUE)
$cont.table
                   acidité bois fruit fruité léger liquoreux sucré
Vin 1                    2    1     1      1     1         0     0
Vin 2                    0    0     0      1     2         1     3
Millésime 1.Vin 1        1    1     1      0     0         0     0
Millésime 1.Vin 2        0    0     0      1     1         0     1
Millésime 2.Vin 1        1    0     0      1     1         0     0
Millésime 2.Vin 2        0    0     0      0     1         1     2

$nb.words
          words nb.list
sucré         3       3
léger         3       3
fruité        2       2
acidité       2       2
liquoreux     1       1
fruit         1       1
bois          1       1

> 
```

où l'objet *nb.list* représente le nombre de lignes dans lequel le mot apparaît.

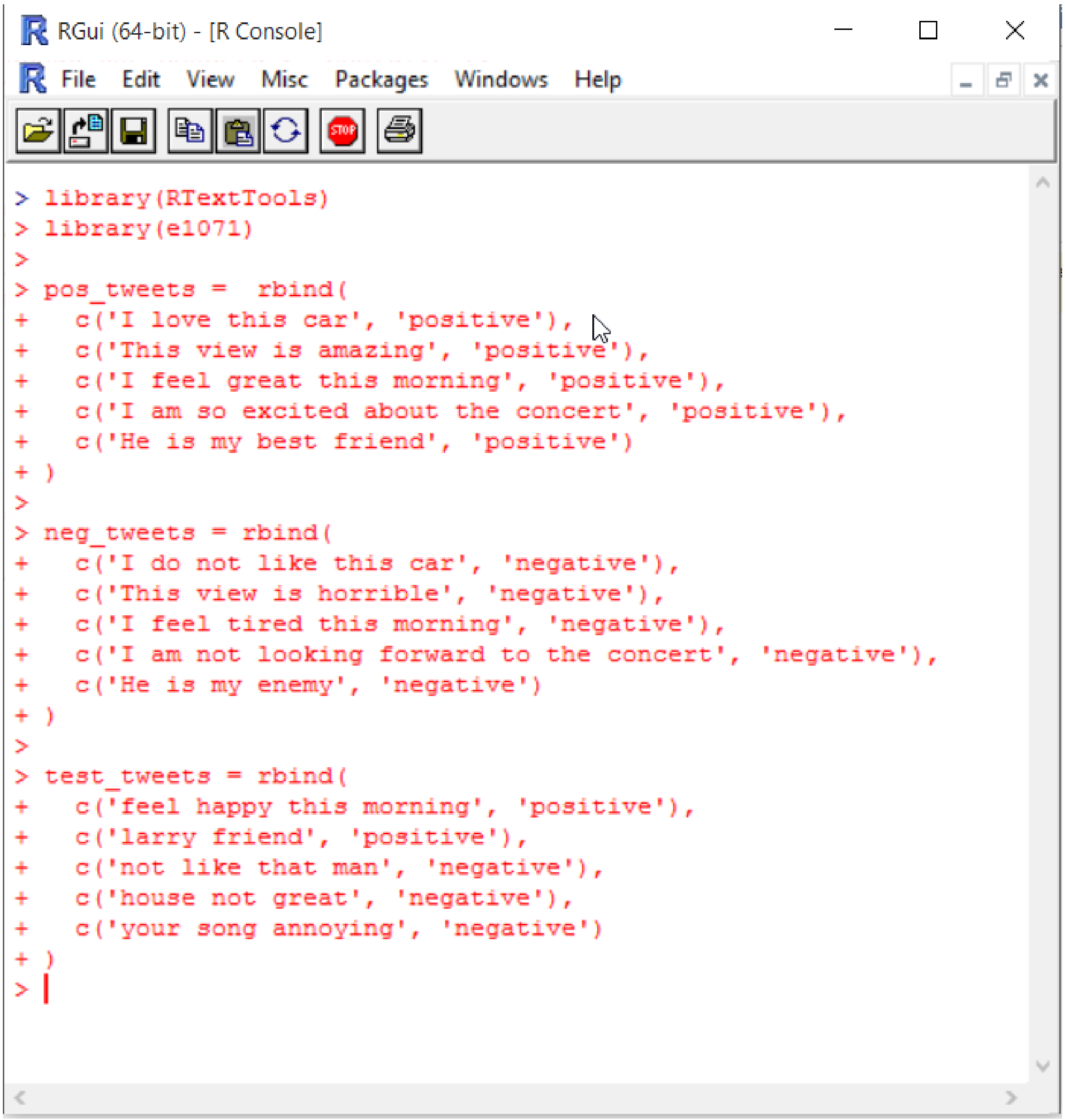
Les valeurs peuvent ensuite être accédées directement puisqu'il s'agit de simple data frames!

Exemple 436.: Analyse de sentiments (avec apprentissage par méthode bayésienne)

R 3.2.1.

Nous allons procéder ici à une analyse de Tweets typique. Pas souci de simplification nous écrirons les Tweets à la main plutôt que de passer par l'API qui a une tendance volatile.

Nous utiliserons d'abord un nouveau package pas utilisé jusqu'ici qui est le package **RTextTools**.



Jusque là rien d'extraordinaire!

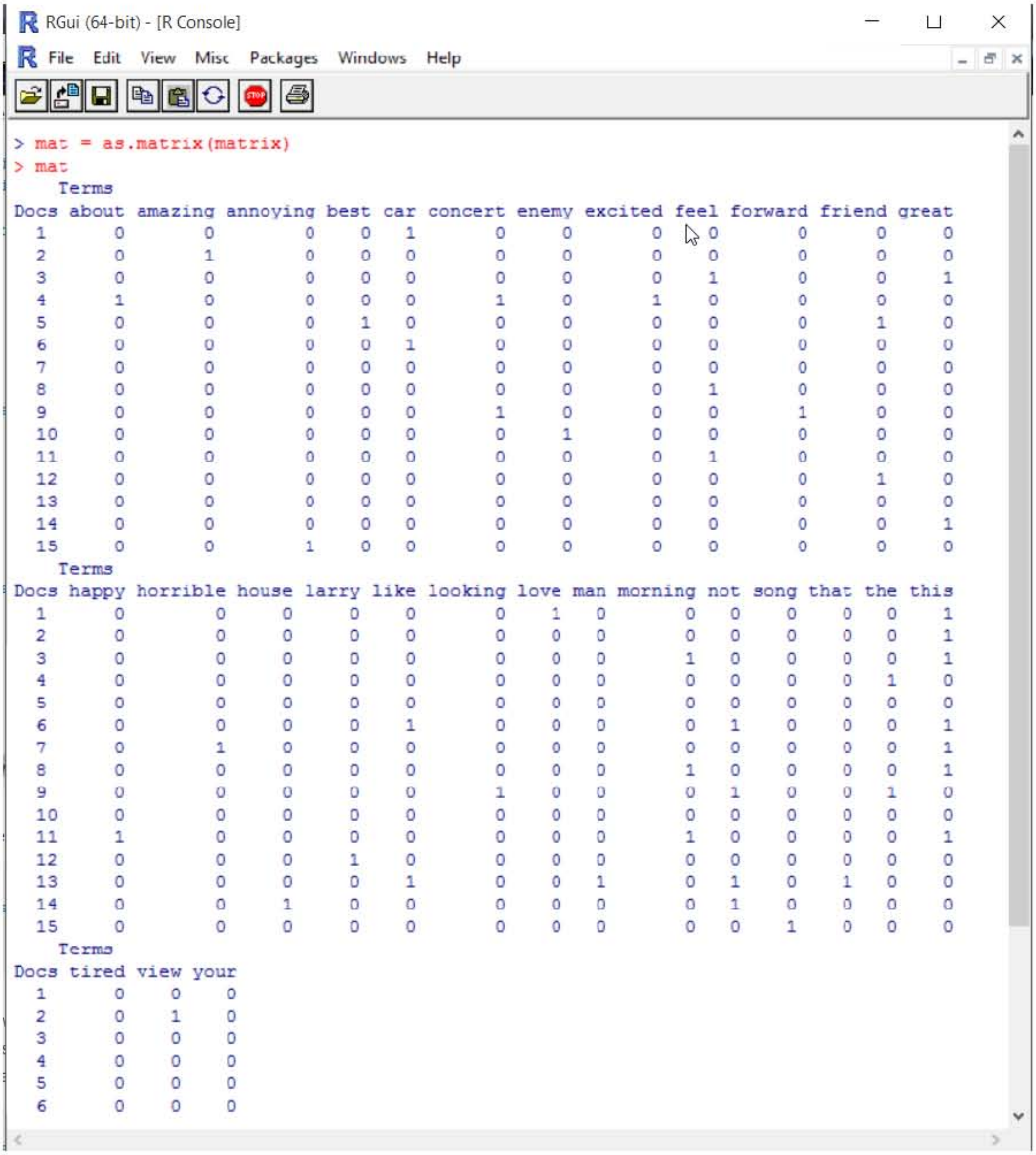
Ensuite, nous fusionnons ce joli petit monde:

```
> tweets = rbind(pos_tweets, neg_tweets, test_tweets)
> tweets
      [,1]                                        [,2]
 [1,] "I love this car"                           "positive"
 [2,] "This view is amazing"                      "positive"
 [3,] "I feel great this morning"                 "positive"
 [4,] "I am so excited about the concert"         "positive"
 [5,] "He is my best friend"                      "positive"
 [6,] "I do not like this car"                    "negative"
 [7,] "This view is horrible"                     "negative"
 [8,] "I feel tired this morning"                 "negative"
 [9,] "I am not looking forward to the concert"   "negative"
[10,] "He is my enemy"                            "negative"
[11,] "feel happy this morning"                   "positive"
[12,] "larry friend"                              "positive"
[13,] "not like that man"                         "negative"
[14,] "house not great"                           "negative"
[15,] "your song annoying"                        "negative"
>
```

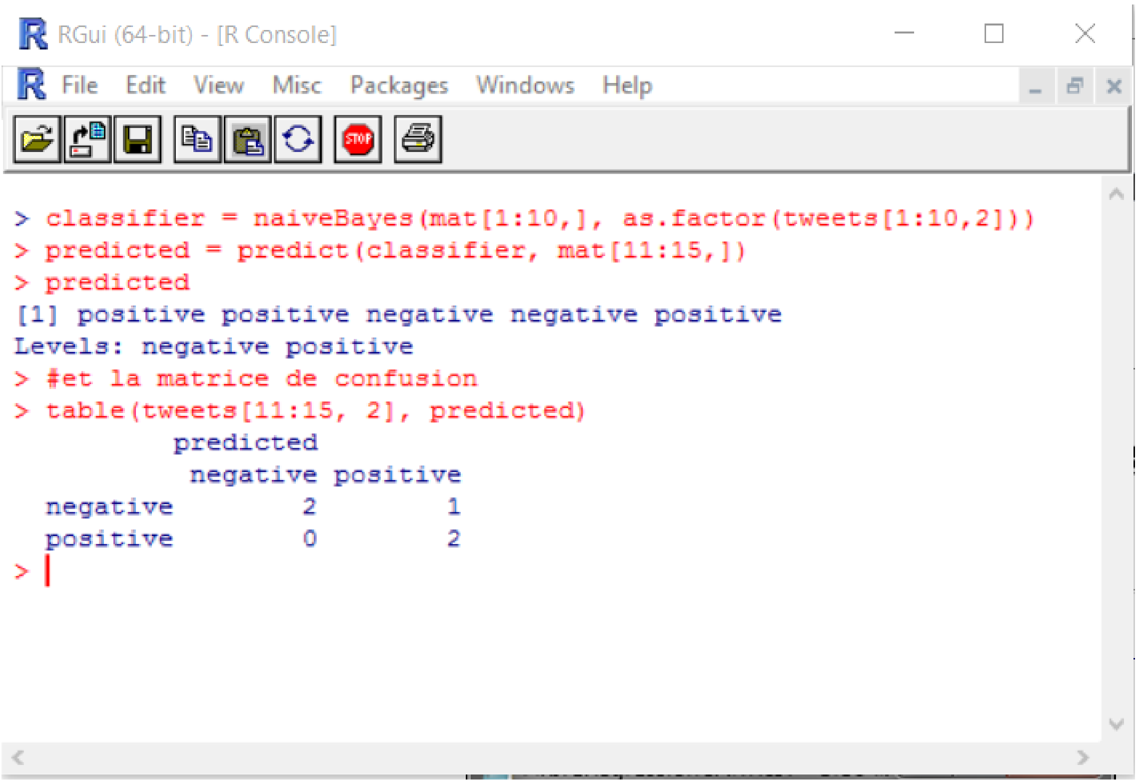
Maintenant nous utilisons la fonction **create_matrix( )** du package **RTextTools**:

```
> matrix= create_matrix(tweets[,1], language="english",
+                       removeStopwords=FALSE, removeNumbers=TRUE,
+                       stemWords=FALSE)
> matrix
<<DocumentTermMatrix (documents: 15, terms: 29)>>
Non-/sparse entries: 49/386
Sparsity           : 89%
Maximal term length: 8
Weighting          : term frequency (tf)
>
```

Bon cela crée un objet mais ce serait mieux d'en voir le contenu détaillé:



Et enfin:

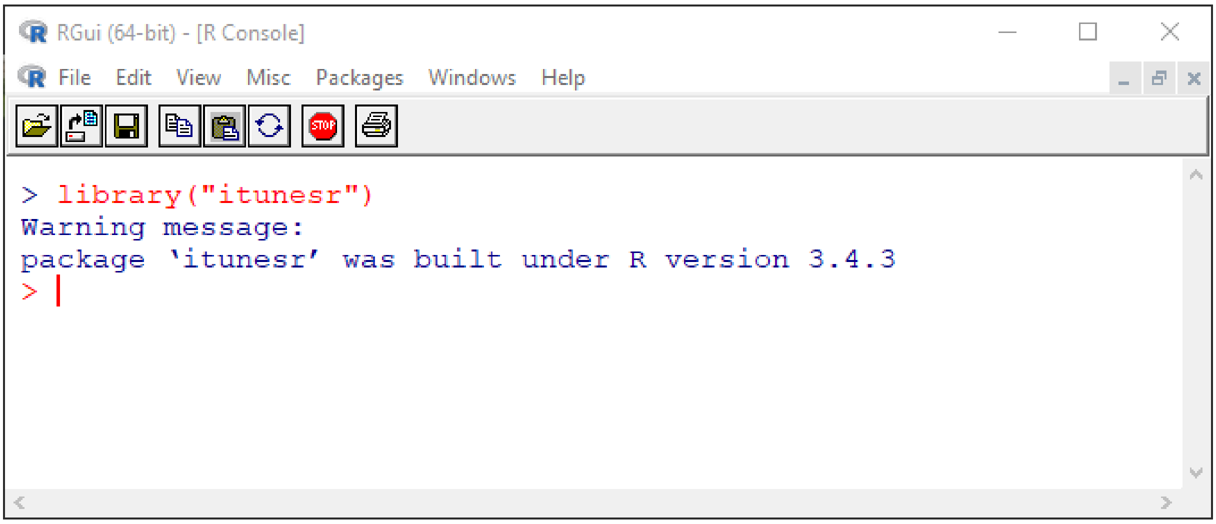


Voilà donc un résultat obtenu en un minimum d'efforts avec un coût minimum!

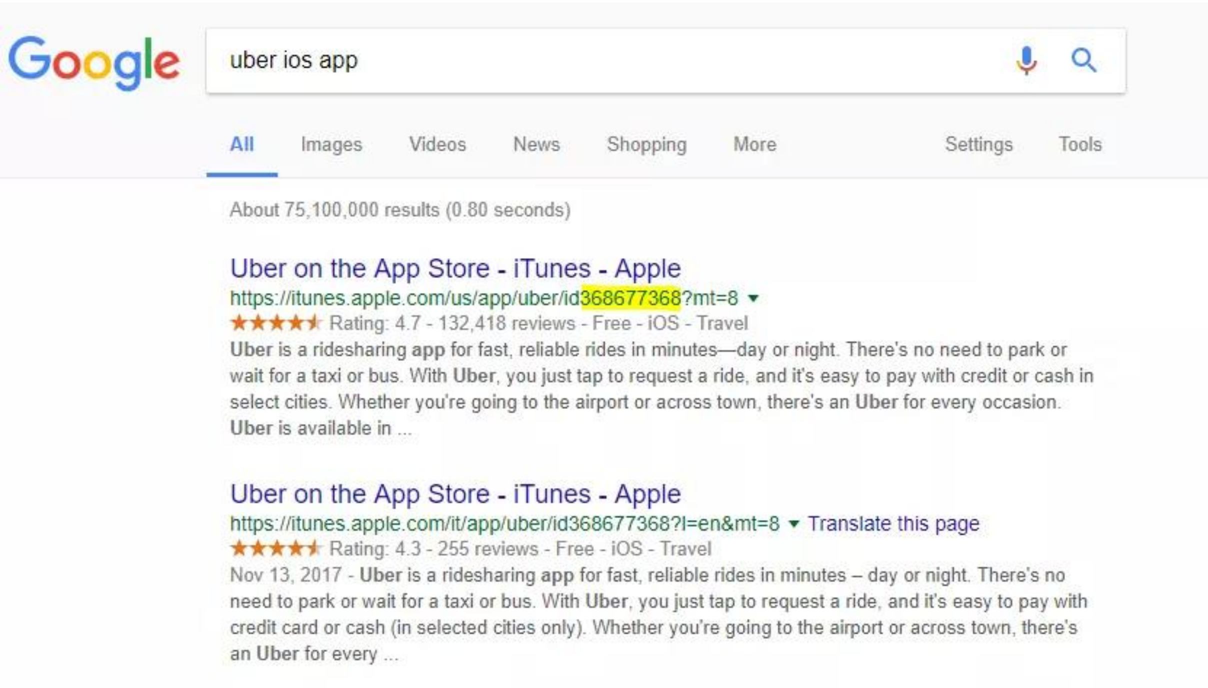
Exemple 437.: Analyse de sentiments sur iTunes Store

R 3.4.2

Nous allons ici voir comment faire un exemple d'analyse sentimentale sur la base des commentaires d'une application iTunes à l'aide du package **itunesr** :



Pour accéder aux avis de toute application iOS, nous devons avoir l'identifiant de l'application. L'identifiant d'application d'un identifiant d'application iOS peut être trouvé dans son URL iTunes (lien). Voici la capture d'écran montrant comment identifier l'App ID de Uber iOS App:



Maintenant, nous avons l'App ID - 368677368 pour lequel nous allons accéder à iTunes Reviews et l'analyser.

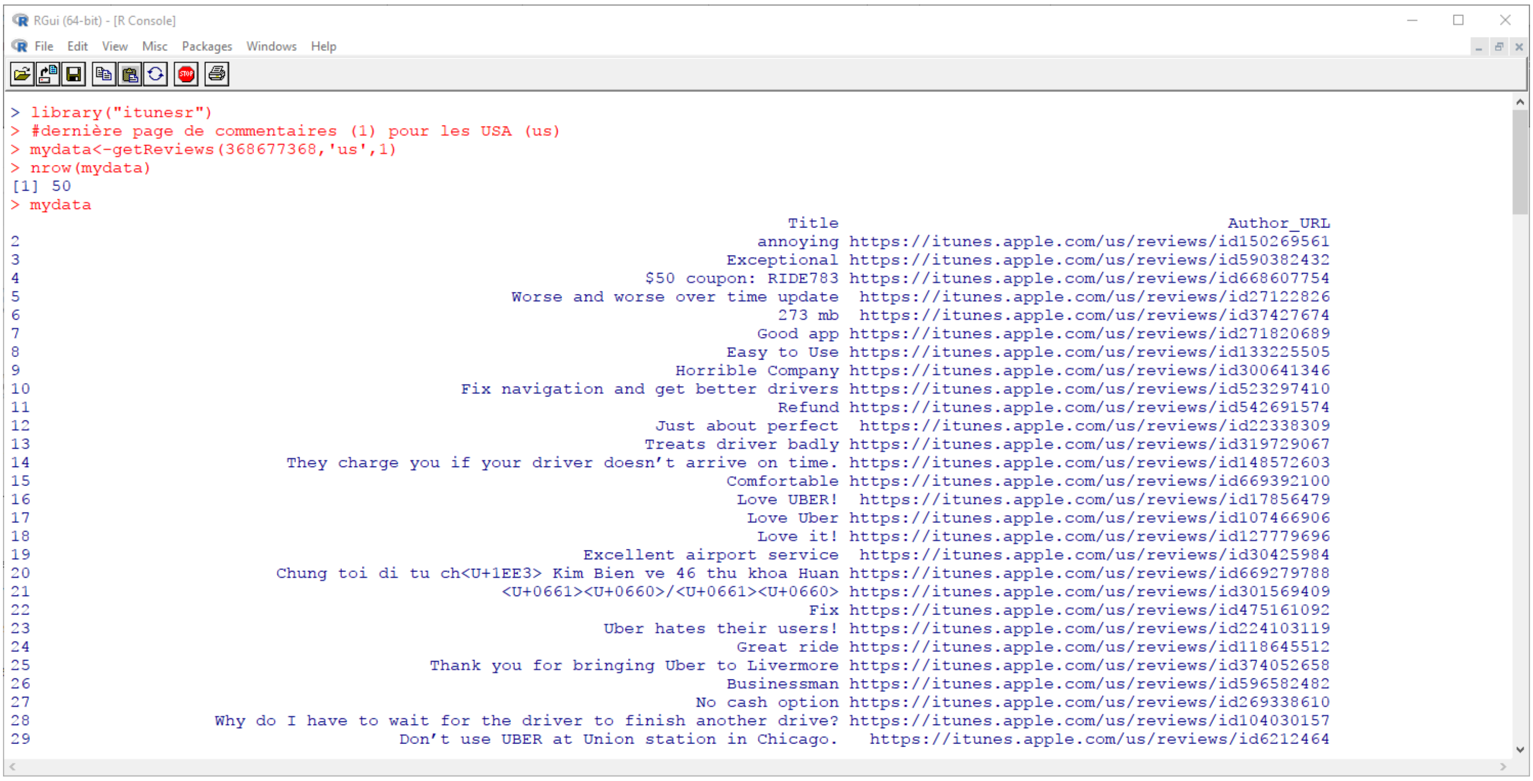
Nous pouvons extraire les avis d'iTunes de n'importe quelle application avec la fonction **getReviews( )** qui prend 3 arguments:

1. ID de l'application

2. Code du pays

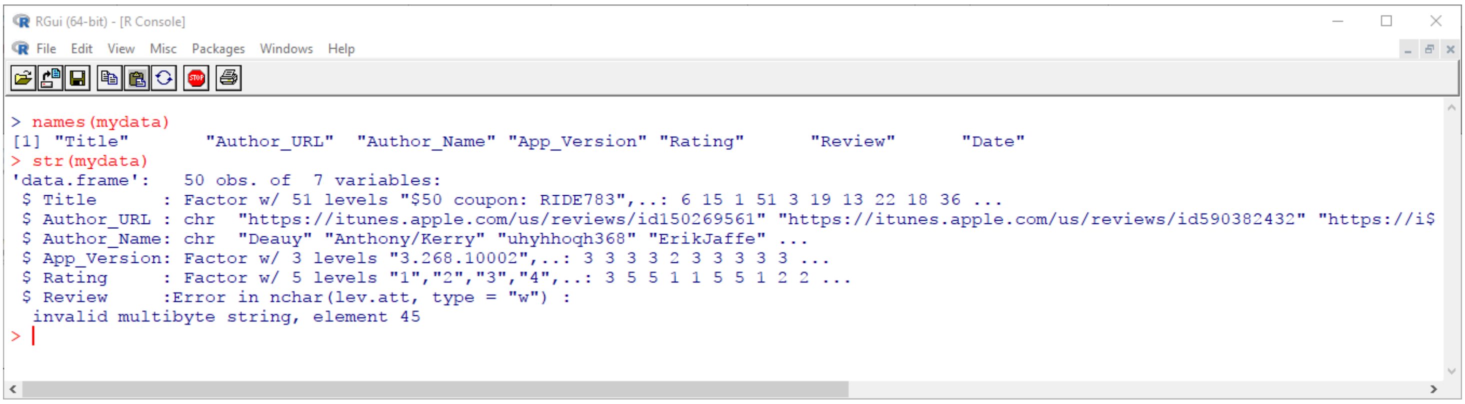
3. Données de la page des avis

Allons-y pour extraire les commentaires de l'app Uber:



```
> library("itunesr")
> #dernière page de commentaires (1) pour les USA (us)
> mydata<-getReviews(368677368,'us',1)
> nrow(mydata)
[1] 50
> mydata
                                                                          Title                                   Author_URL
2                                                                       annoying https://itunes.apple.com/us/reviews/id150269561
3                                                                    Exceptional https://itunes.apple.com/us/reviews/id590382432
4                                                             $50 coupon: RIDE783 https://itunes.apple.com/us/reviews/id668607754
5                                                 Worse and worse over time update  https://itunes.apple.com/us/reviews/id27122826
6                                                                         273 mb  https://itunes.apple.com/us/reviews/id37427674
7                                                                       Good app https://itunes.apple.com/us/reviews/id271820689
8                                                                    Easy to Use https://itunes.apple.com/us/reviews/id133225505
9                                                               Horrible Company https://itunes.apple.com/us/reviews/id300641346
10                                           Fix navigation and get better drivers https://itunes.apple.com/us/reviews/id523297410
11                                                                        Refund https://itunes.apple.com/us/reviews/id542691574
12                                                            Just about perfect  https://itunes.apple.com/us/reviews/id22338309
13                                                             Treats driver badly https://itunes.apple.com/us/reviews/id319729067
14                          They charge you if your driver doesn't arrive on time. https://itunes.apple.com/us/reviews/id148572603
15                                                                   Comfortable https://itunes.apple.com/us/reviews/id669392100
16                                                                    Love UBER!  https://itunes.apple.com/us/reviews/id17856479
17                                                                     Love Uber https://itunes.apple.com/us/reviews/id107466906
18                                                                      Love it! https://itunes.apple.com/us/reviews/id127779696
19                                                      Excellent airport service  https://itunes.apple.com/us/reviews/id30425984
20                         Chung toi di tu ch<U+1EE3> Kim Bien ve 46 thu khoa Huan https://itunes.apple.com/us/reviews/id669279788
21                                        <U+0661><U+0660>/<U+0661><U+0660> https://itunes.apple.com/us/reviews/id301569409
22                                                                           Fix https://itunes.apple.com/us/reviews/id475161092
23                                                           Uber hates their users! https://itunes.apple.com/us/reviews/id224103119
24                                                                    Great ride https://itunes.apple.com/us/reviews/id118645512
25                                        Thank you for bringing Uber to Livermore https://itunes.apple.com/us/reviews/id374052658
26                                                                   Businessman https://itunes.apple.com/us/reviews/id596582482
27                                                                No cash option https://itunes.apple.com/us/reviews/id269338610
28                     Why do I have to wait for the driver to finish another drive? https://itunes.apple.com/us/reviews/id104030157
29                                         Don't use UBER at Union station in Chicago.   https://itunes.apple.com/us/reviews/id6212464
```

Voici un résumé de ce que contient le dataframe:



```
> names(mydata)
[1] "Title"        "Author_URL"  "Author_Name" "App_Version" "Rating"      "Review"      "Date"
> str(mydata)
'data.frame':   50 obs. of  7 variables:
 $ Title      : Factor w/ 51 levels "$50 coupon: RIDE783",..: 6 15 1 51 3 19 13 22 18 36 ...
 $ Author_URL : chr  "https://itunes.apple.com/us/reviews/id150269561" "https://itunes.apple.com/us/reviews/id590382432" "https://i$
 $ Author_Name: chr  "Deauy" "Anthony/Kerry" "uhyhhoqh368" "ErikJaffe" ...
 $ App_Version: Factor w/ 3 levels "3.268.10002",..: 3 3 3 3 2 3 3 3 3 3 ...
 $ Rating     : Factor w/ 5 levels "1","2","3","4",..: 3 5 5 1 1 5 5 1 2 2 ...
 $ Review     :Error in nchar(lev.att, type = "w") :
  invalid multibyte string, element 45
>
```

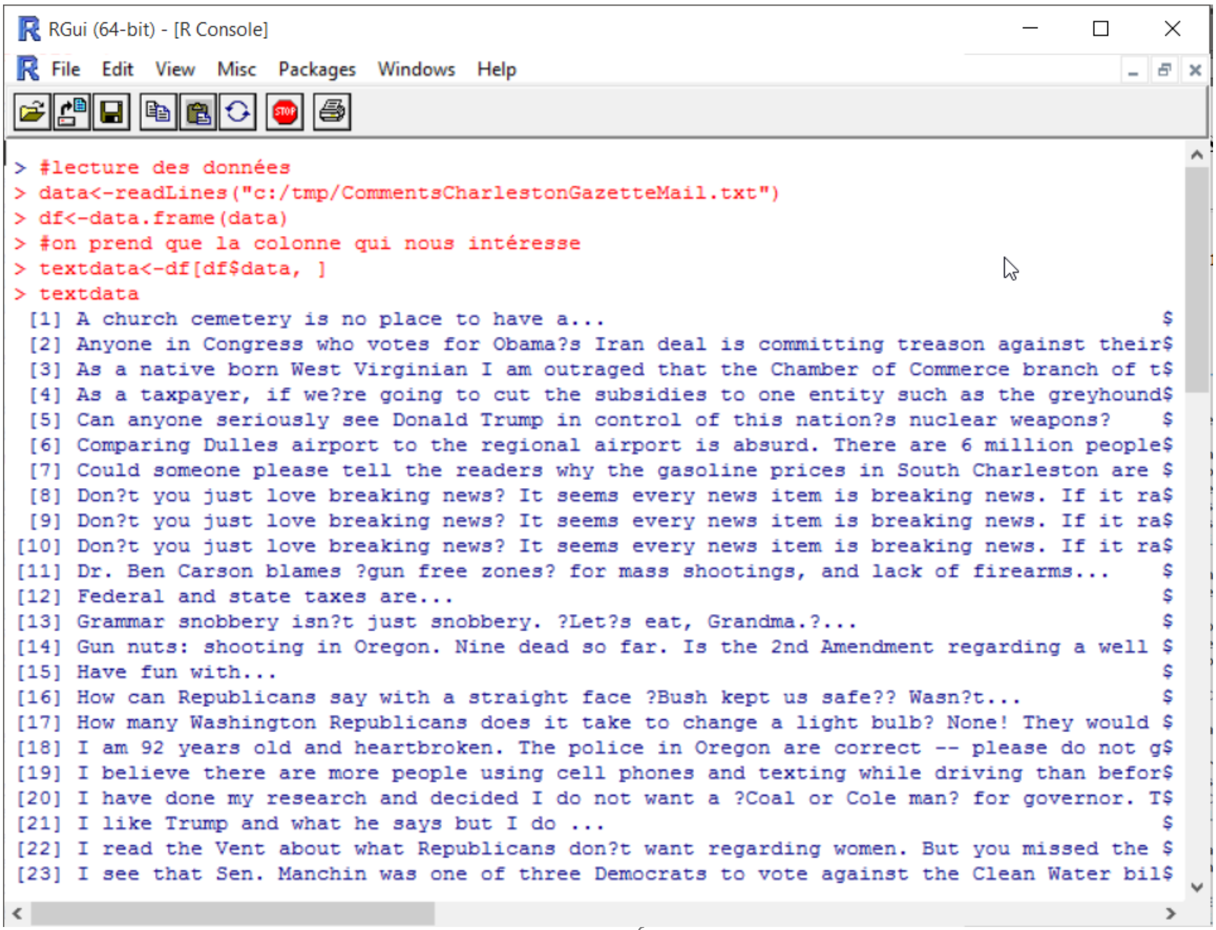
Exemple 438.: Inférence de sentiments (sans apprentissage)

R 3.2.3

Nous allons analysis sans a priori le fichier de commentaires suivant en utilisant le multiples packages:

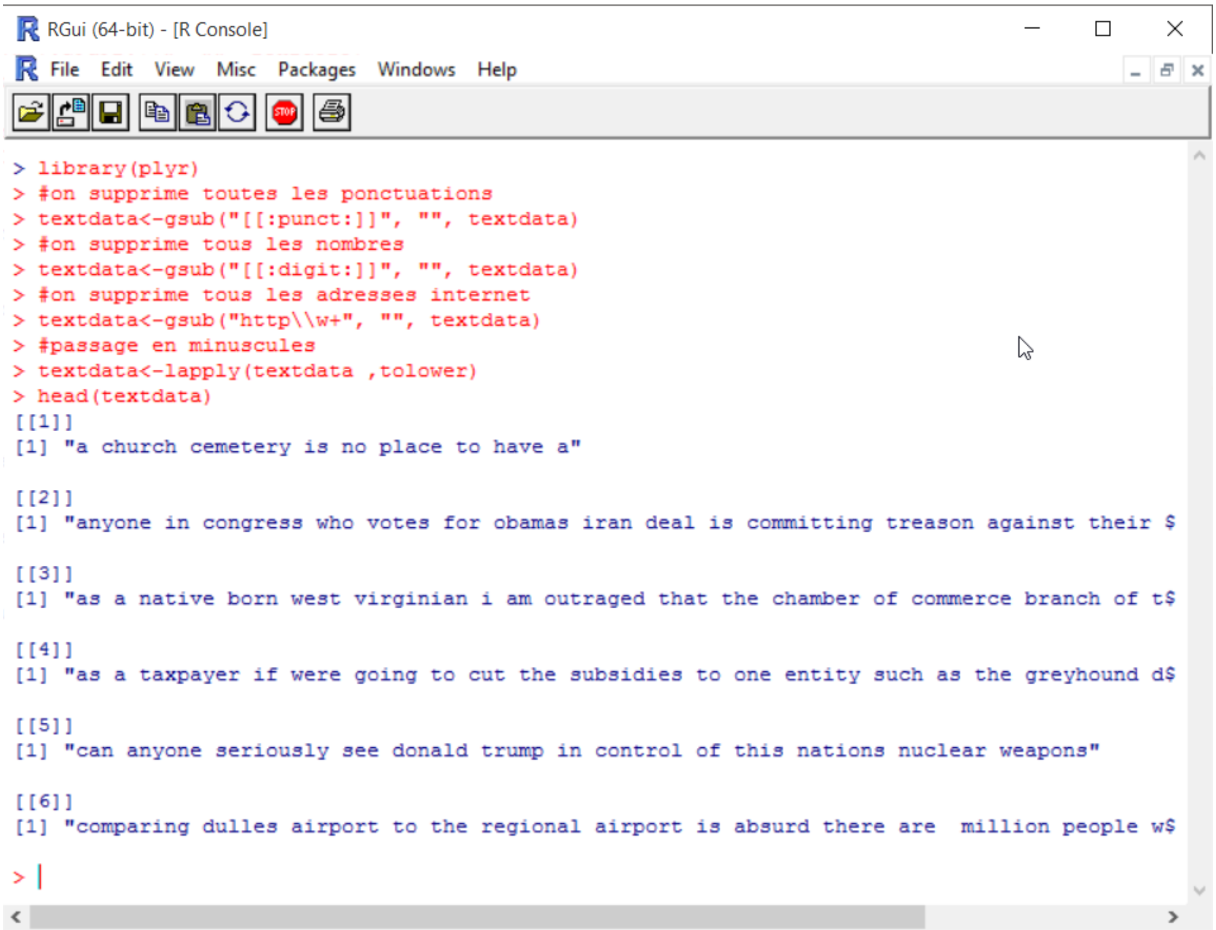
```
CommentsCharlestonGazetteMail.txt - Notepad
File  Edit  Format  View  Help
A church cemetery is no place to have a...
Anyone in Congress who votes for Obama?s Iran deal is committing treason against their own country....
As a native born West Virginian I am outraged that the Chamber of Commerce branch of the Republican Party has...
As a taxpayer, if we?re going to cut the subsidies to one entity such as the greyhound dogs, why...
Can anyone seriously see Donald Trump in control of this nation?s nuclear weapons?
Comparing Dulles airport to the regional airport is absurd. There are 6 million people within an hour-drive...
Could someone please tell the readers why the gasoline prices in South Charleston are $2.64 per gallon and just up the
Don?t you just love breaking news? It seems every news item is breaking news. If it rains too hard,...
Don?t you just love breaking news? It seems every news item is breaking news. If it rains too hard,...
Dr. Ben Carson blames ?gun free zones? for mass shootings, and lack of firearms...
Federal and state taxes are...
Grammar snobbery isn?t just snobbery. ?Let?s eat, Grandma.?...
Gun nuts: shooting in Oregon. Nine dead so far. Is the 2nd Amendment regarding a well armed militia still...
Have fun with...
How can Republicans say with a straight face ?Bush kept us safe?? Wasn?t...
How many Washington Republicans does it take to change a light bulb? None! They would rather sit in the dark and blame
I am 92 years old and heartbroken. The police in Oregon are correct -- please do not give these...
I believe there are more people using cell phones and texting while driving than before the law against it...
I have done my research and decided I do not want a ?Coal or Cole man? for governor. They both...
I like Trump and what he says but I do ...
I read the Vent about what Republicans don?t want regarding women. But you missed the whole point. They don?t want a ?
I see that Sen. Manchin was one of three Democrats to vote against the Clean Water bill. If in fact he wants...
I see where the fall version of FestivALL is getting ready to start. Does it ever end?
I think the state needs to start taking notice of these people who are...
I think this side of Charleston has been shut out of Coonskin Park and no one cares enough to do something...
I was sorry to see the news about Cali. The only problem with having pet animals, especially cats, is that...
I wonder why AEP cannot ...
I would love to read real examples of how Christians are being abused. Give examples or quit being so fragile.
I?m convinced that if Adolf Hitler were alive today and ran for the Democratic Party for president...
I?m glad there are a few who have enough common sense to have Halloween on Halloween.
I?m so proud of the Gazette-Mail and the writer Eric Eyre. That article about Patrick Morissey is just ...
If Joe Manchin expects to get elected again in West Virginia then he needs to vote against the Iranian...
If you want cheap gasoline then you need to come to Nitro or Cross Lanes. It is always several cents lower than anywhe
If you want to know why there are no electric cars available in West Virginia, it?s not c-o-a-l,...
In a job where you have no contact with the public whatsoever, why can you not wear blue jeans?
In response to the person who said that the aluminum workers at Ravenswood went on strike and they are still...
```

D'abord nous lisons les le contenu ligne par ligne avec la fonction native **readLines( )** :

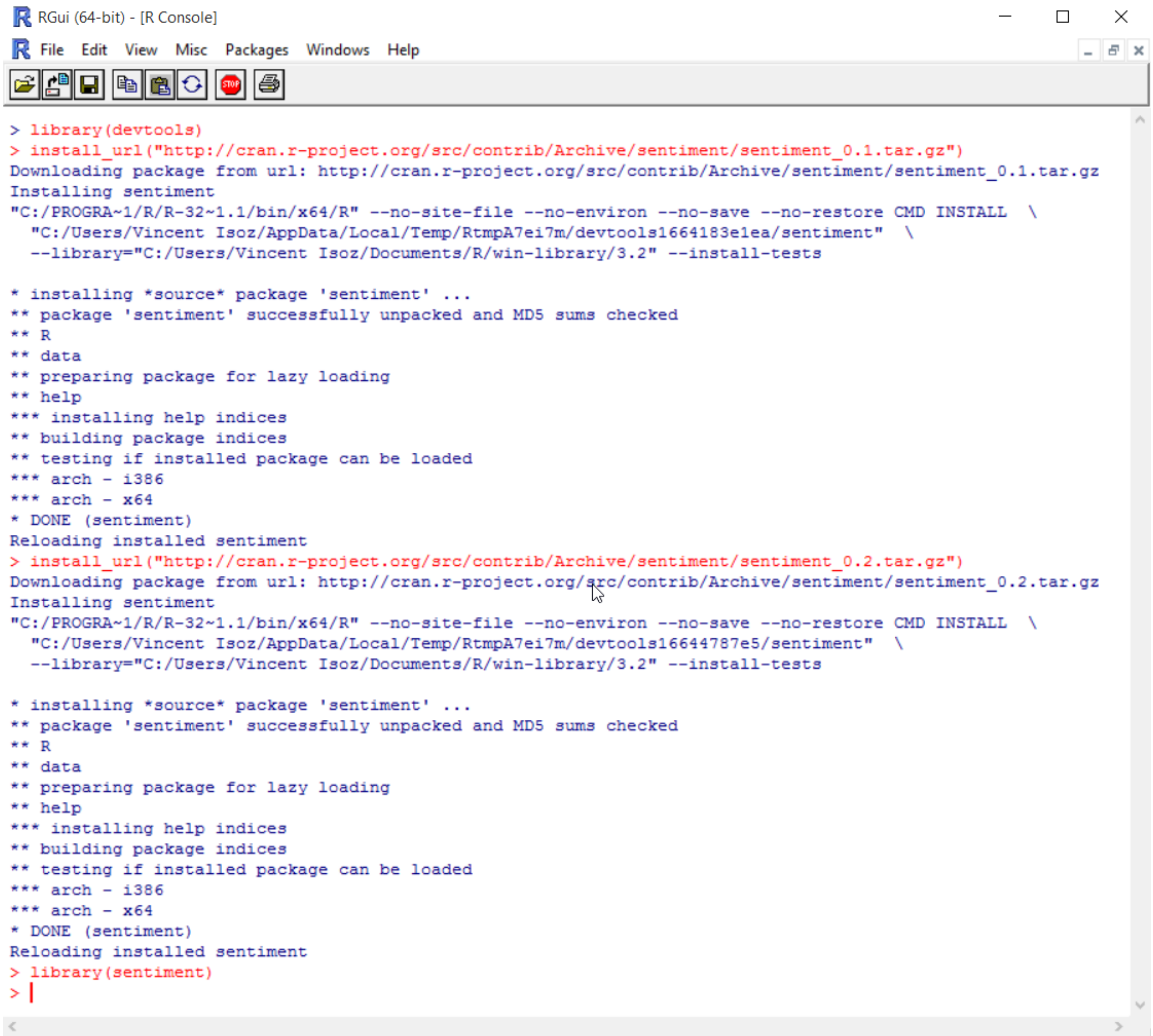


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> #lecture des données
> data<-readLines("c:/tmp/CommentsCharlestonGazetteMail.txt")
> df<-data.frame(data)
> #on prend que la colonne qui nous intéresse
> textdata<-df[df$data, ]
> textdata
 [1] A church cemetery is no place to have a...                                    $
 [2] Anyone in Congress who votes for Obama?s Iran deal is committing treason against their$
 [3] As a native born West Virginian I am outraged that the Chamber of Commerce branch of t$
 [4] As a taxpayer, if we?re going to cut the subsidies to one entity such as the greyhound$
 [5] Can anyone seriously see Donald Trump in control of this nation?s nuclear weapons?    $
 [6] Comparing Dulles airport to the regional airport is absurd. There are 6 million people$
 [7] Could someone please tell the readers why the gasoline prices in South Charleston are $
 [8] Don?t you just love breaking news? It seems every news item is breaking news. If it ra$
 [9] Don?t you just love breaking news? It seems every news item is breaking news. If it ra$
[10] Don?t you just love breaking news? It seems every news item is breaking news. If it ra$
[11] Dr. Ben Carson blames ?gun free zones? for mass shootings, and lack of firearms...    $
[12] Federal and state taxes are...                                                        $
[13] Grammar snobbery isn?t just snobbery. ?Let?s eat, Grandma.?...                        $
[14] Gun nuts: shooting in Oregon. Nine dead so far. Is the 2nd Amendment regarding a well $
[15] Have fun with...                                                                      $
[16] How can Republicans say with a straight face ?Bush kept us safe?? Wasn?t...           $
[17] How many Washington Republicans does it take to change a light bulb? None! They would $
[18] I am 92 years old and heartbroken. The police in Oregon are correct -- please do not g$
[19] I believe there are more people using cell phones and texting while driving than befor$
[20] I have done my research and decided I do not want a ?Coal or Cole man? for governor. T$
[21] I like Trump and what he says but I do ...                                            $
[22] I read the Vent about what Republicans don?t want regarding women. But you missed the $
[23] I see that Sen. Manchin was one of three Democrats to vote against the Clean Water bil$
```


Nous faisons un nettoyage avec la classique fonction **gsub( )** et une mise en minuscule avec **lapply( )** :



Pour la suite nous installons et nous chargeons le package **sentiment** qui n'est plus à ce jour sur le CRAN:



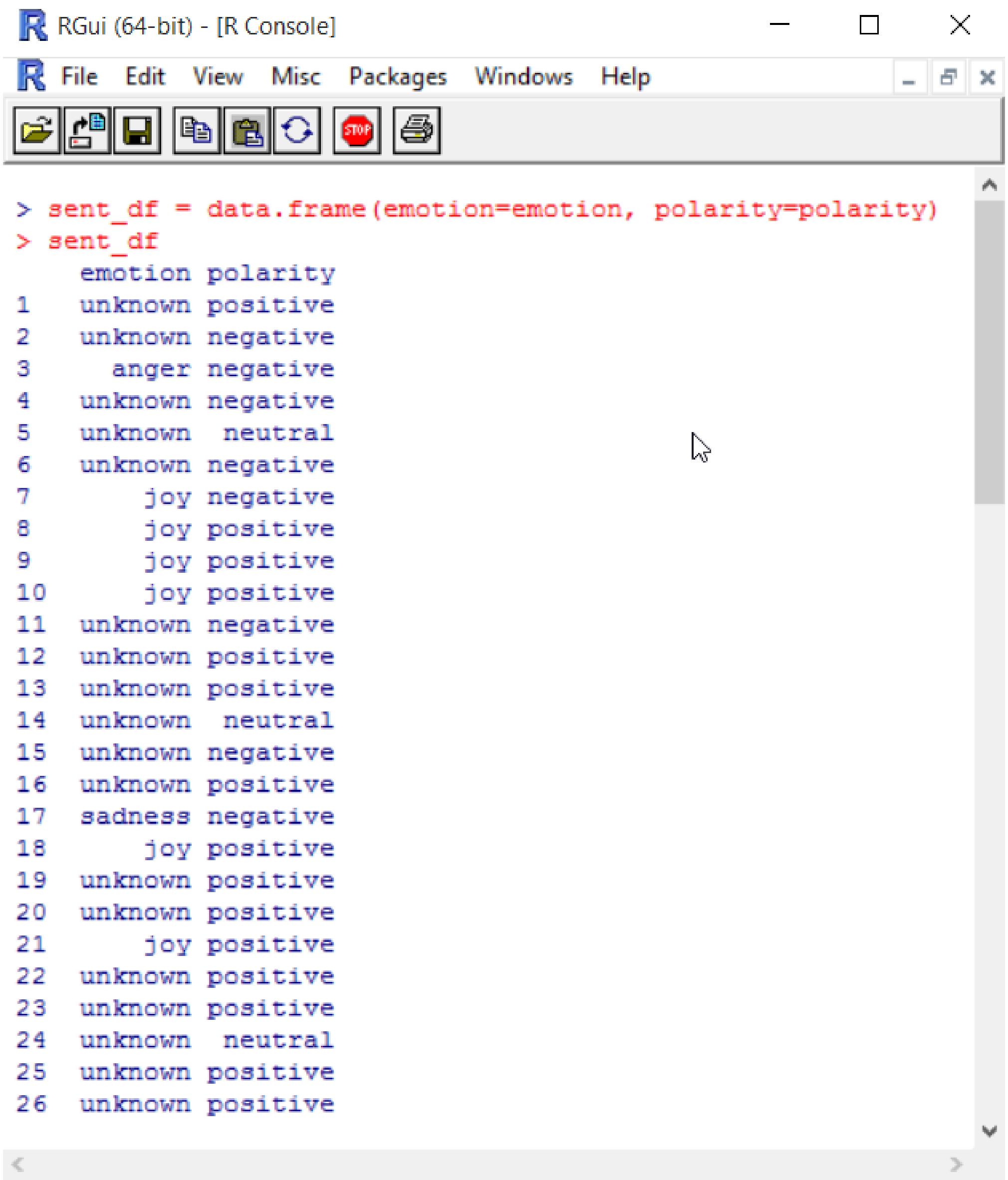
Maintenant avec la fonction **classify_emotion( )** nous classifions les messages par sentiments:

```
> #on tente de classifier les messages par émotion (joie, colère, etc.)
> class_emo = classify_emotion(textdata, algorithm="bayes", prior=1.0)
> #on prend que la partie émotions qui est dans la colonne 7
> emotion = class_emo[,7];emotion
 [1] NA         NA         "anger"    NA         NA         NA         "joy"      "joy"      "joy"
[10] "joy"      NA         NA         NA         NA         NA         NA         "sadness"  "joy"
[19] NA         NA         "joy"      NA         NA         NA         NA         NA         "sadness"
[28] "surprise" "joy"      NA         "joy"      "joy"      "sadness"  NA         NA         "sadness"
[37] "joy"      "surprise" "fear"     "fear"     NA         NA         NA         "anger"    NA
[46] NA         NA         "anger"    NA         NA         NA         NA         "joy"      "joy"
[55] "joy"      "anger"    NA         NA         NA         NA         NA         NA         NA
[64] NA         "fear"     NA         NA         "joy"      "joy"      "joy"      NA         NA
[73] "joy"      NA         "joy"      "joy"      NA         NA         NA         NA         "joy"
[82] "joy"      NA
> #on fait un peu plus propre
> emotion[is.na(emotion)] = "unknown";emotion
 [1] "unknown"  "unknown"  "anger"    "unknown"  "unknown"  "unknown"  "joy"      "joy"      "joy"
[10] "joy"      "unknown"  "unknown"  "unknown"  "unknown"  "unknown"  "unknown"  "sadness"  "joy"
[19] "unknown"  "unknown"  "joy"      "unknown"  "unknown"  "unknown"  "unknown"  "unknown"  "sadness"
[28] "surprise" "joy"      "unknown"  "joy"      "joy"      "sadness"  "unknown"  "unknown"  "sadness"
[37] "joy"      "surprise" "fear"     "fear"     "unknown"  "unknown"  "unknown"  "anger"    "unknown"
[46] "unknown"  "unknown"  "anger"    "unknown"  "unknown"  "unknown"  "unknown"  "joy"      "joy"
[55] "joy"      "anger"    "unknown"  "unknown"  "unknown"  "unknown"  "unknown"  "unknown"  "unknown"
[64] "unknown"  "fear"     "unknown"  "unknown"  "joy"      "joy"      "joy"      "unknown"  "unknown"
[73] "joy"      "unknown"  "joy"      "joy"      "unknown"  "unknown"  "unknown"  "unknown"  "joy"
[82] "joy"      "unknown"
>
```

On fait de même avec les polarités en utilisant la fonction **classify_polarity( )**:

```
> #on tente de classier les messages par polarité (négatif, positif, neutre)
> class_pol = classify_polarity(textdata, algorithm="bayes")
> #on prend que la partie polarité qui est dans la colonne 4
> polarity = class_pol[,4];polarity
 [1] "positive" "negative" "negative" "neutral"  "negative" "negative" "positive" "positive"
[10] "positive" "negative" "positive" "positive" "neutral"  "negative" "positive" "negative" "positive"
[19] "positive" "positive" "positive" "positive" "positive" "neutral"  "positive" "positive" "negative"
[28] "positive" "neutral"  "positive" "positive" "positive" "negative" "negative" "positive" "negative"
[37] "negative" "positive" "neutral"  "neutral"  "negative" "negative" "positive" "neutral"  "positive"
[46] "neutral"  "neutral"  "negative" "neutral"  "positive" "positive" "negative" "positive" "positive"
[55] "positive" "negative" "negative" "negative" "negative" "positive" "positive" "positive" "positive"
[64] "positive" "negative" "positive" "positive" "negative" "negative" "negative" "positive" "positive"
[73] "negative" "positive" "negative" "negative" "positive" "neutral"  "negative" "positive" "positive"
[82] "positive" "positive"
>
```

On rassemble le tout:



On fait un premier plot des sentiments:

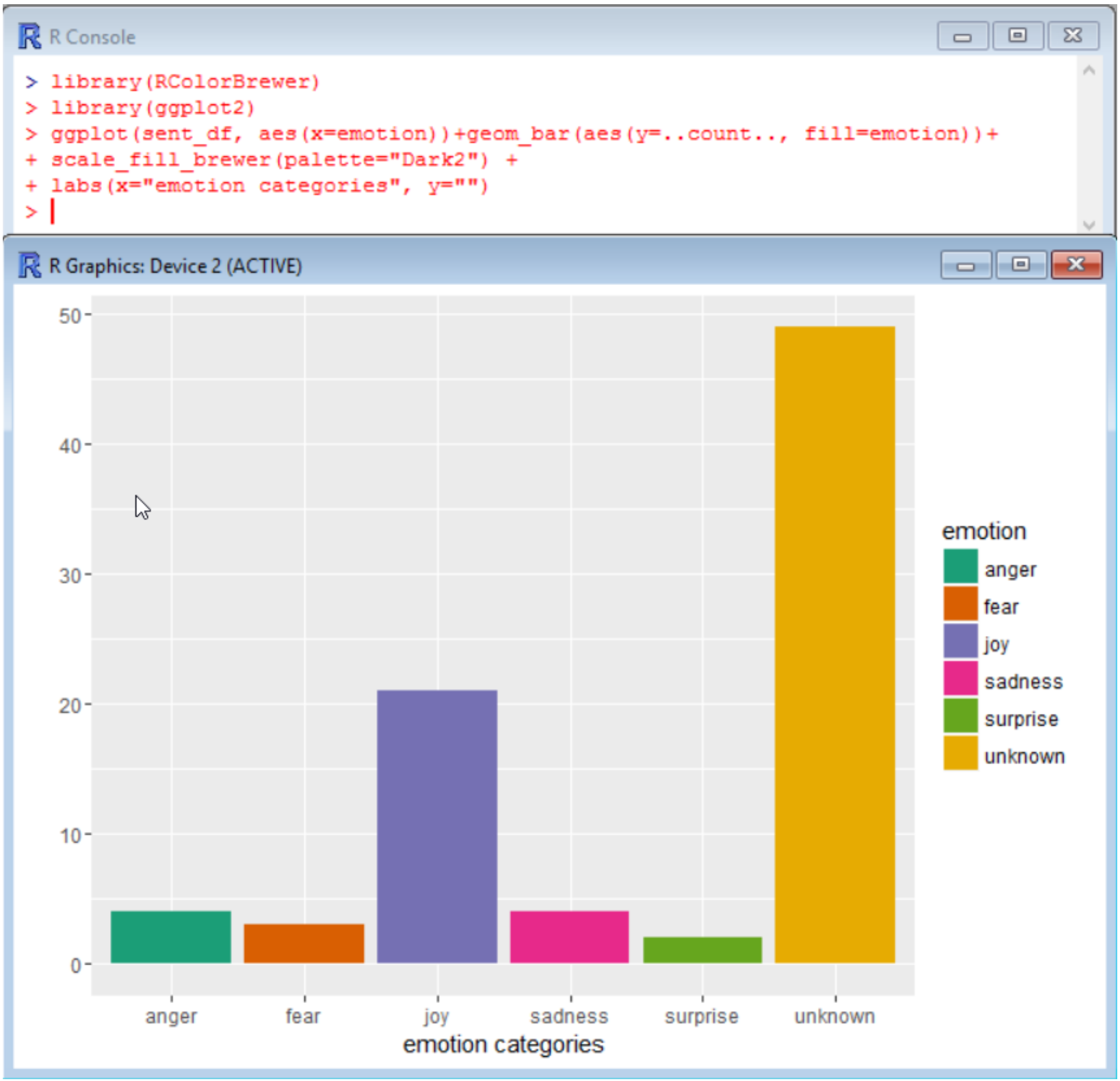
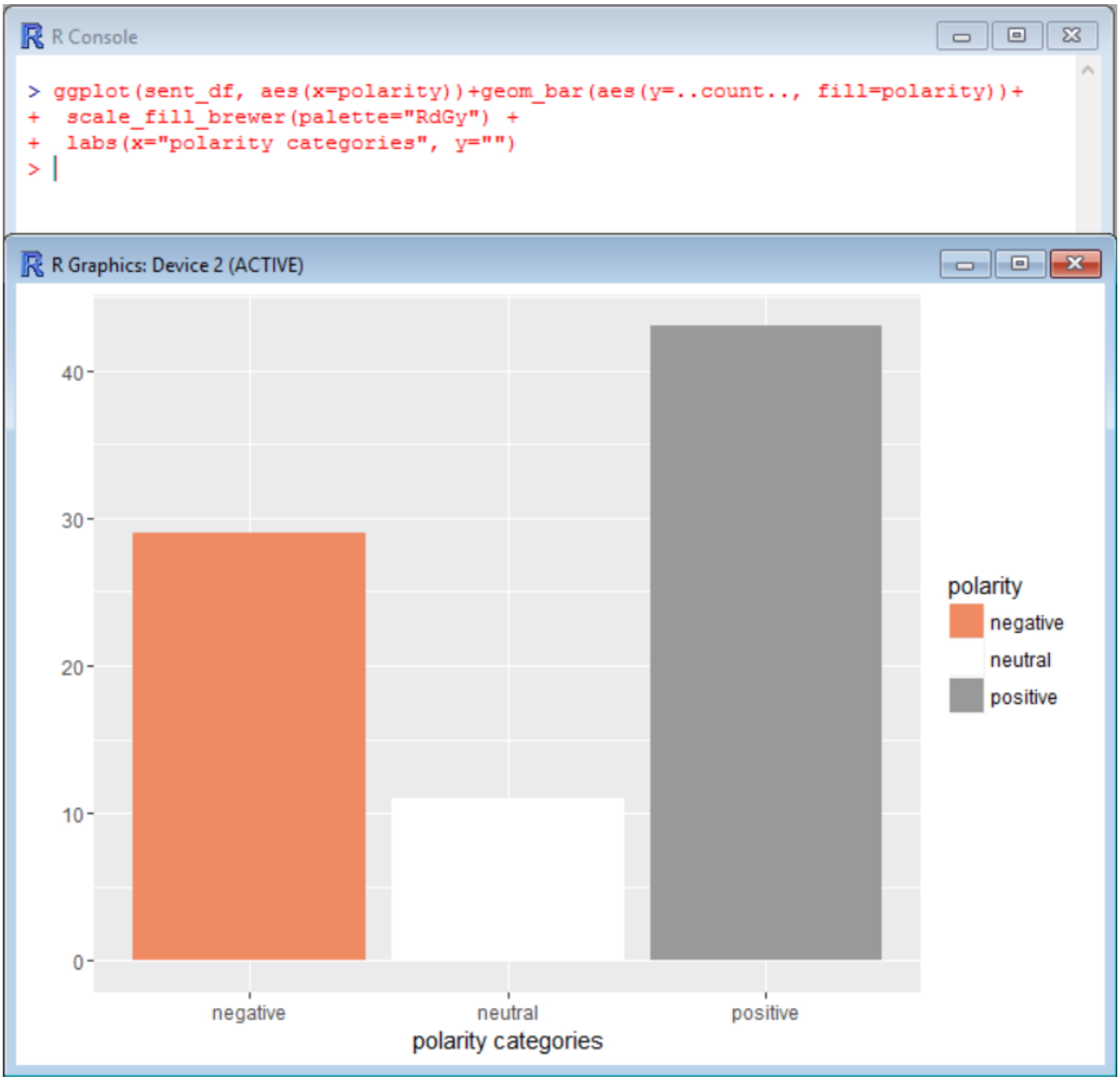


Figure 342 Graph à barres d'analyse de sentiments

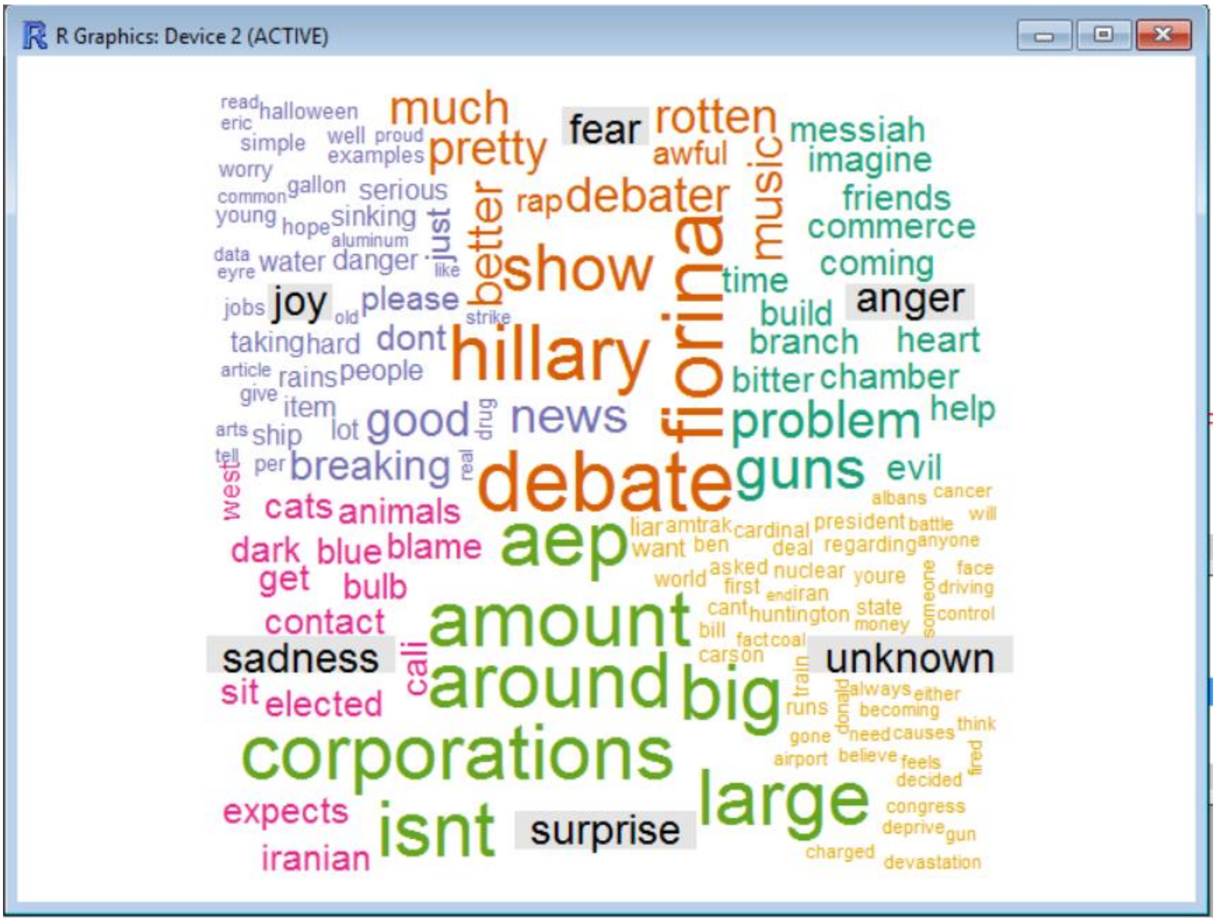
Et idem pour la polarité:



Et enfin le nuage:

```
> library(wordcloud)
> library(tm)
> library(SnowballC)
> emos = levels(factor(sent_df$emotion))
> nemo = length(emos)
> emo.docs = rep("", nemo)
> for (i in 1:nemo)
+ {
+   tmp = textdata[emotion == emos[i]]
+   emo.docs[i] = paste(tmp, collapse=" ")
+ }
> emo.docs = removeWords(emo.docs, stopwords("english"))
> corpus = Corpus(VectorSource(emo.docs))
> tdm = TermDocumentMatrix(corpus)
> tdm = as.matrix(tdm)
> colnames(tdm) = emos
> comparison.cloud(tdm, colors = brewer.pal(nemo, "Dark2"),scale = c(3,0.5), random.order = FALSE,title.size = 1.5)
```

Ce qui donne:



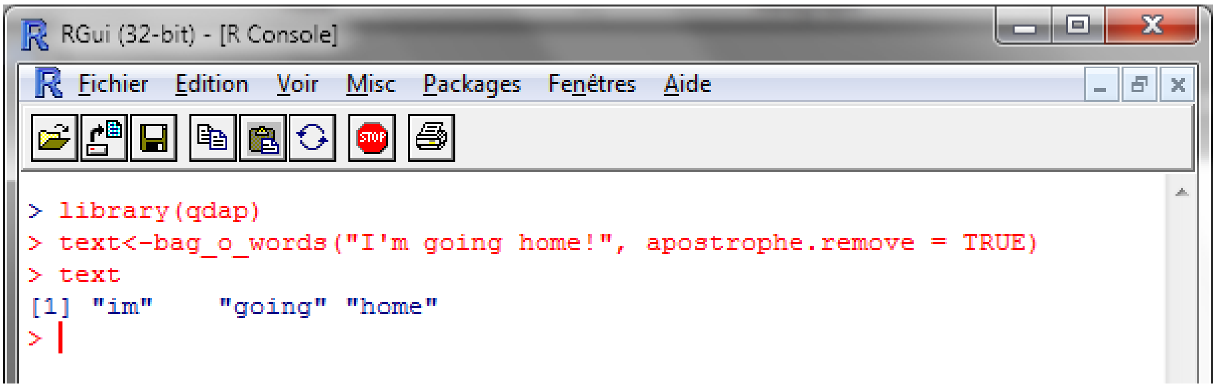
Exemple 439.: Bag-of-Words (package qdap)

R 3.2.1. **x86**

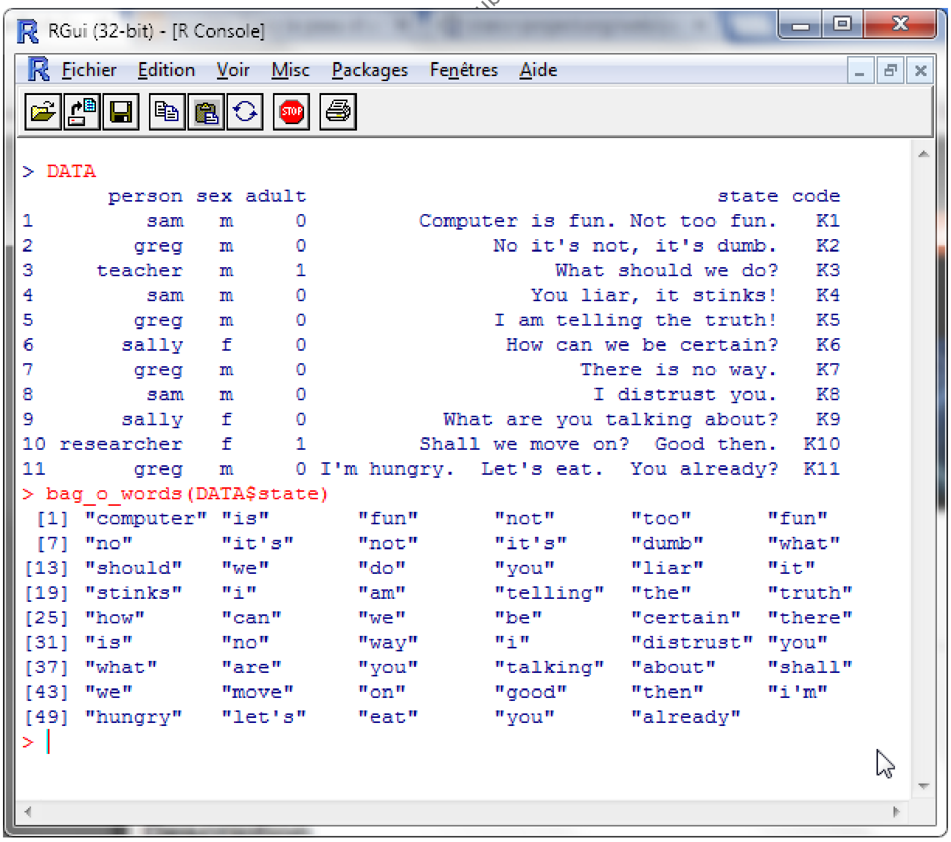
Le Bag-Of-Words est beaucoup utilisé dans le Data Mining que ce soit en finance ou dans l'analyse des échanges sur les réseaux sociaux.

Voyons quelques exemples avec le package **qdap** et attention il faut la version 32bits car utilise rJava!!

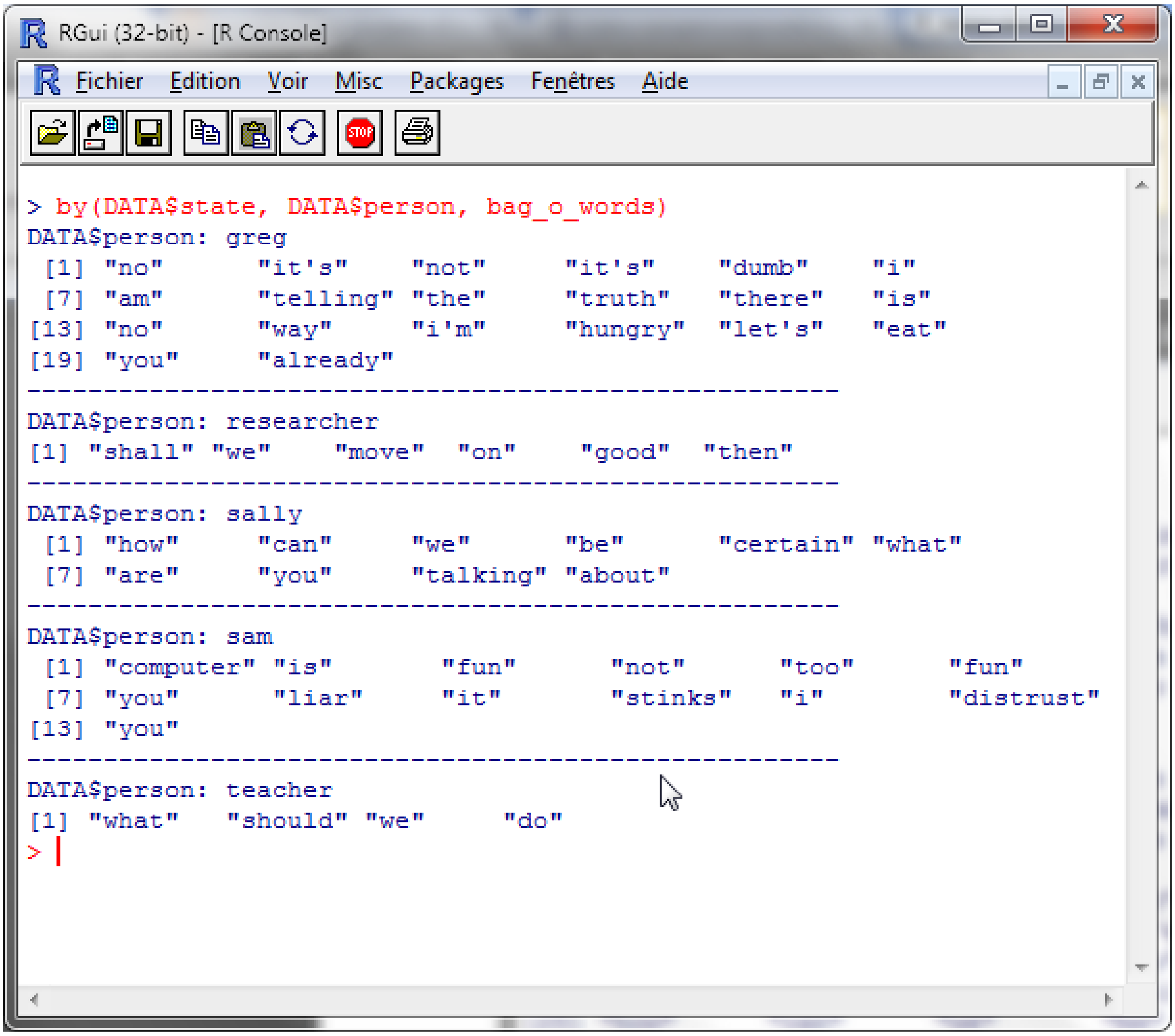
D'abord voyons la fonction **bag_o_words( )** qui nous intéresse ici particulièrement et qui renvoie un vecteur de mots (il devient alors aisé d'en faire toutes sortes de statistiques):



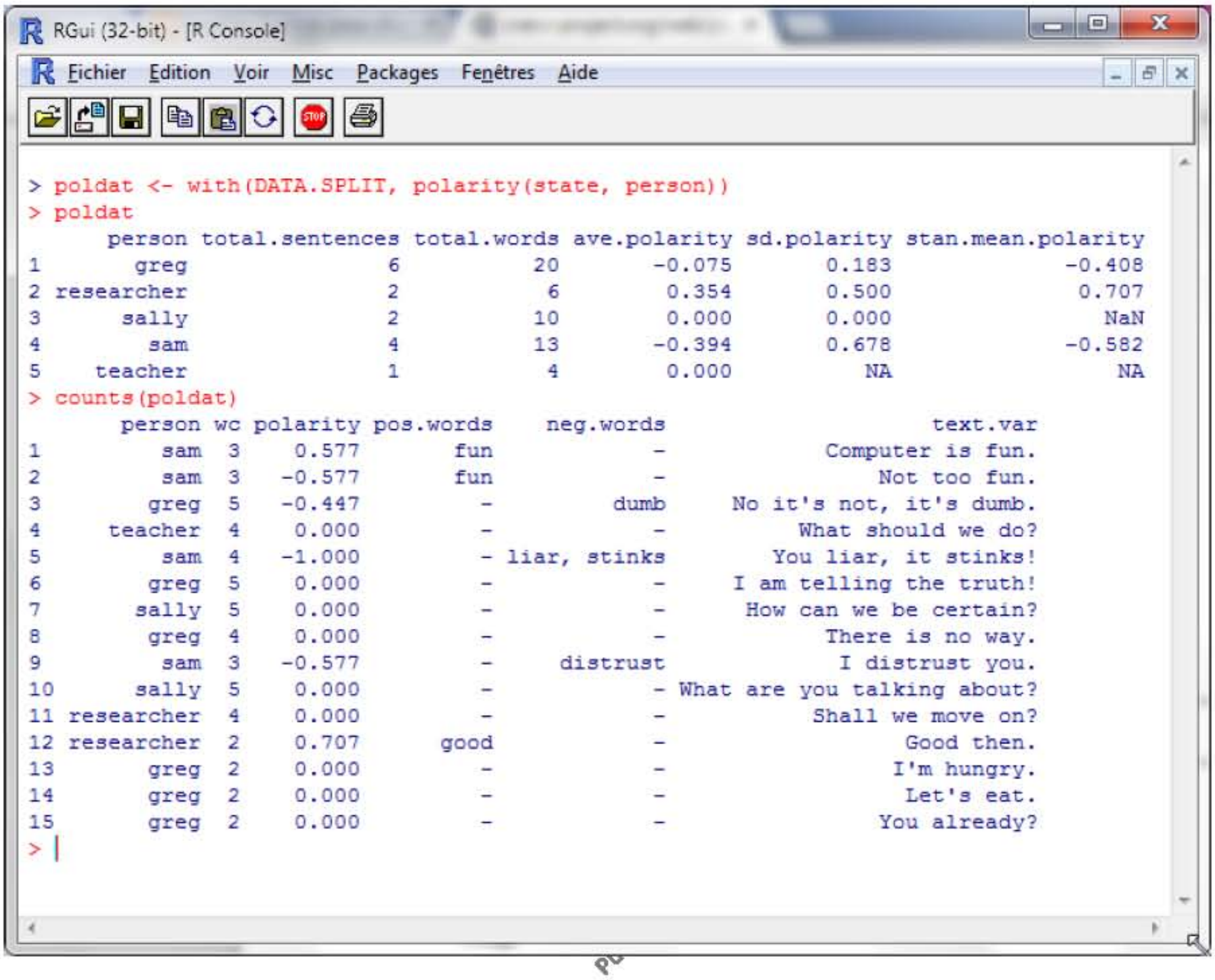
Ou sur un data frame directement:



Ensuite nous pouvons faire un regroupement par variable en utilisant la méthode **by**:



Ou faire encore plus subtil en prenant en compte la polarité (méthode empirique d'attribution de négativité ou de positivité de phrase si ces dernières sont en anglais):


RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> poldat <- with(DATA.SPLIT, polarity(state, person))
> poldat
person total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
1 greg 6 20 -0.075 0.183 -0.408
2 researcher 2 6 0.354 0.500 0.707
3 sally 2 10 0.000 0.000 NaN
4 sam 4 13 -0.394 0.678 -0.582
5 teacher 1 4 0.000 NA NA
> counts(poldat)
person wc polarity pos.words neg.words text.var
1 sam 3 0.577 fun - Computer is fun.
2 sam 3 -0.577 fun - Not too fun.
3 greg 5 -0.447 - dumb No it's not, it's dumb.
4 teacher 4 0.000 - - What should we do?
5 sam 4 -1.000 - liar, stinks You liar, it stinks!
6 greg 5 0.000 - - I am telling the truth!
7 sally 5 0.000 - - How can we be certain?
8 greg 4 0.000 - - There is no way.
9 sam 3 -0.577 - distrust I distrust you.
10 sally 5 0.000 - - What are you talking about?
11 researcher 4 0.000 - - Shall we move on?
12 researcher 2 0.707 good - Good then.
13 greg 2 0.000 - - I'm hungry.
14 greg 2 0.000 - - Let's eat.
15 greg 2 0.000 - - You already?
>

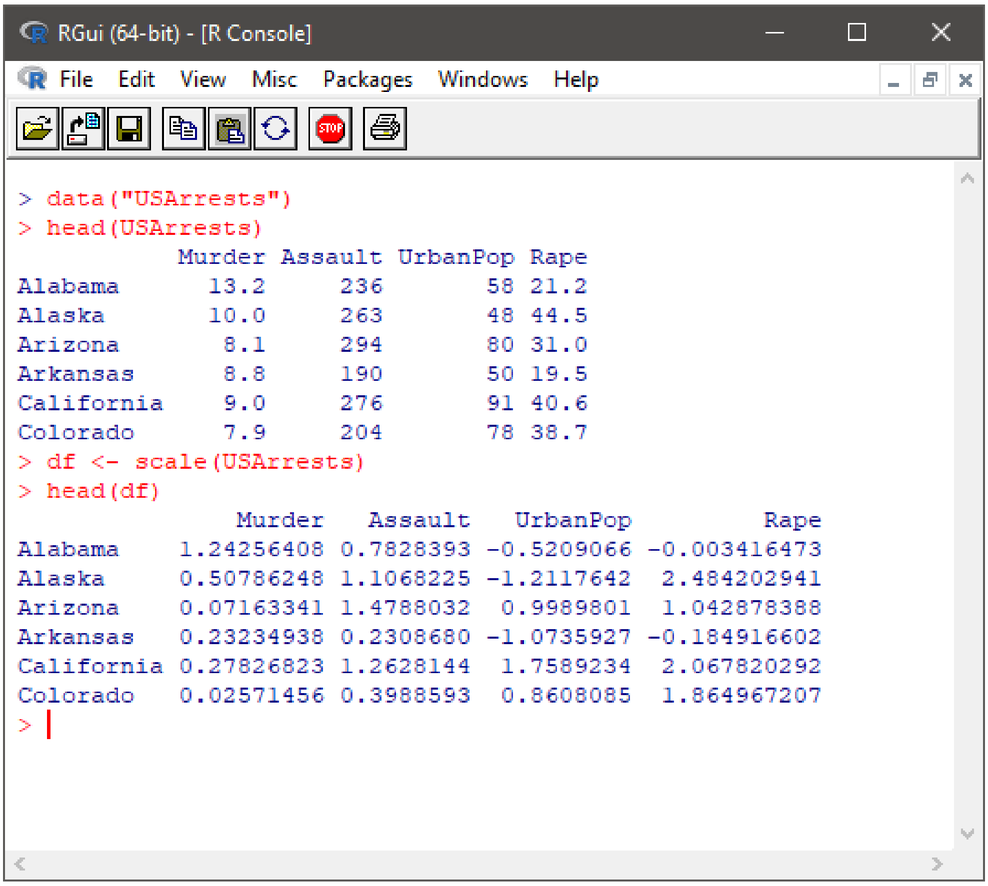
Exemple 440.: Clustering d'un matrice numérique catégorielle

R 4.1

L'exemple qui suit est repris de:

https://www.datanovia.com/en/blog/cluster-analysis-in-r-simplified-and-enhanced/

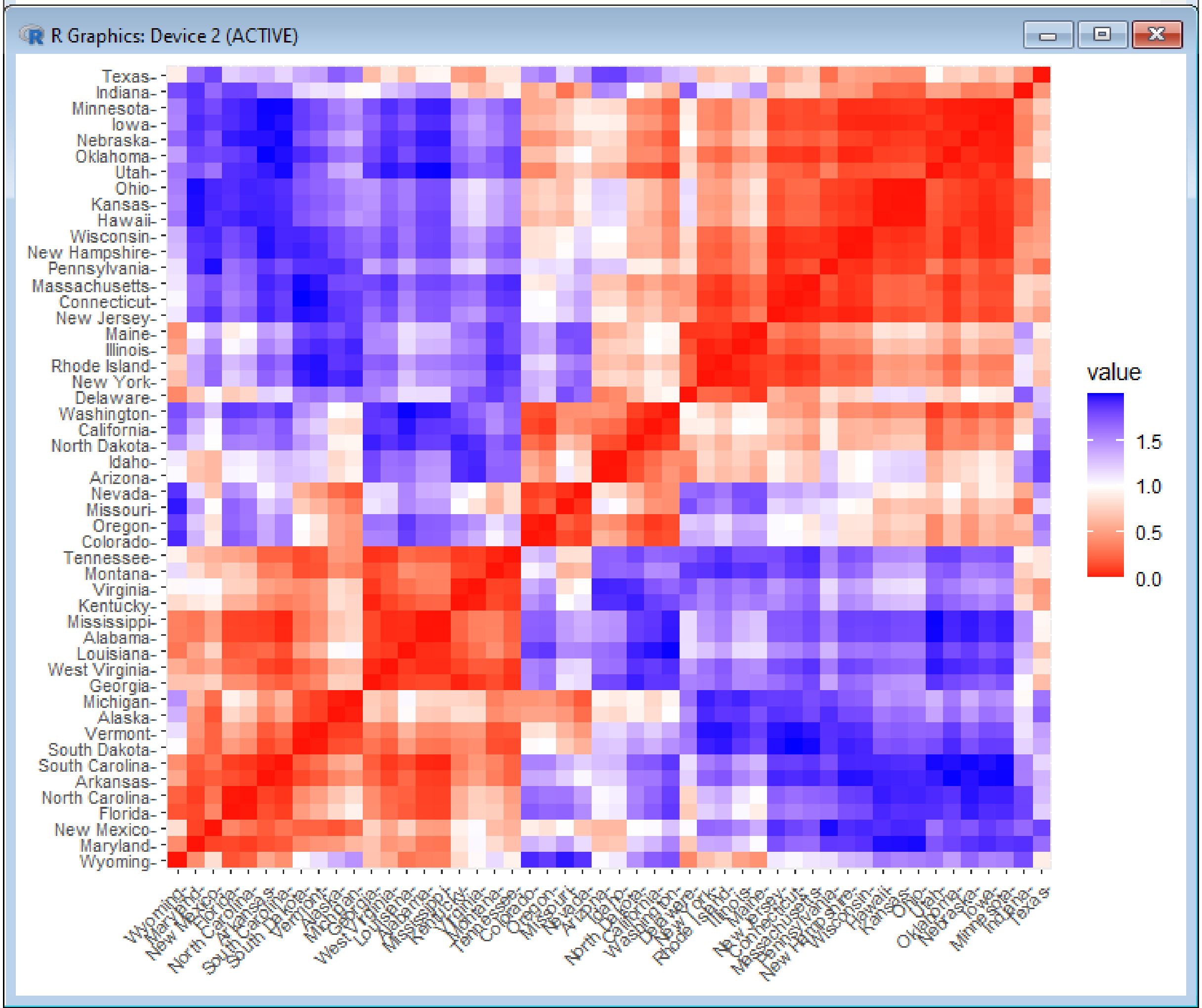
Considérons le jeu de donnés suivant que nous normalisons:



Nous souhaiterions regrouper les villes les plus semblables entre elles entre utilisant un groupement en 4 clusters (vu qu'il y a 4 types facteurs).

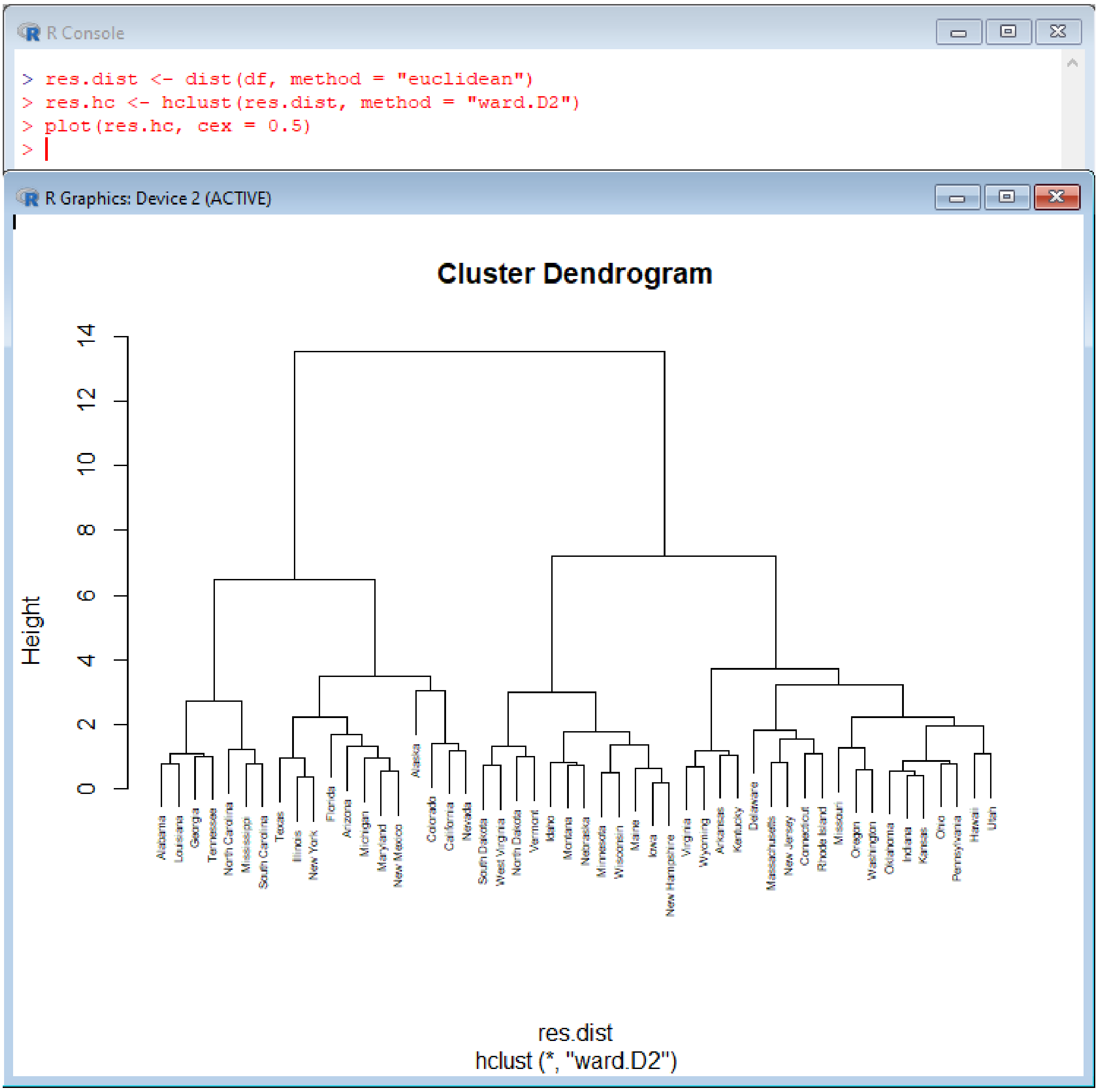
Faisons d'abord une analyse de la corrélation via la métrique de Pearson en utilisant les fonctions **get_dist( )** et **fviz_dist( )** de factoextra:

```
R Console
> library("factoextra")
> #création de la matrice des distances
> res.dist <- factoextra::get_dist(df, method = "pearson")
> head(round(as.matrix(res.dist), 2))[, 1:6]
           Alabama Alaska Arizona Arkansas California Colorado
Alabama       0.00   0.71    1.45     0.09       1.87     1.69
Alaska        0.71   0.00    0.83     0.37       0.81     0.52
Arizona       1.45   0.83    0.00     1.18       0.29     0.60
Arkansas      0.09   0.37    1.18     0.00       1.59     1.37
California    1.87   0.81    0.29     1.59       0.00     0.11
Colorado      1.69   0.52    0.60     1.37       0.11     0.00
> factoextra::fviz_dist(res.dist, lab_size = 8)
>
```

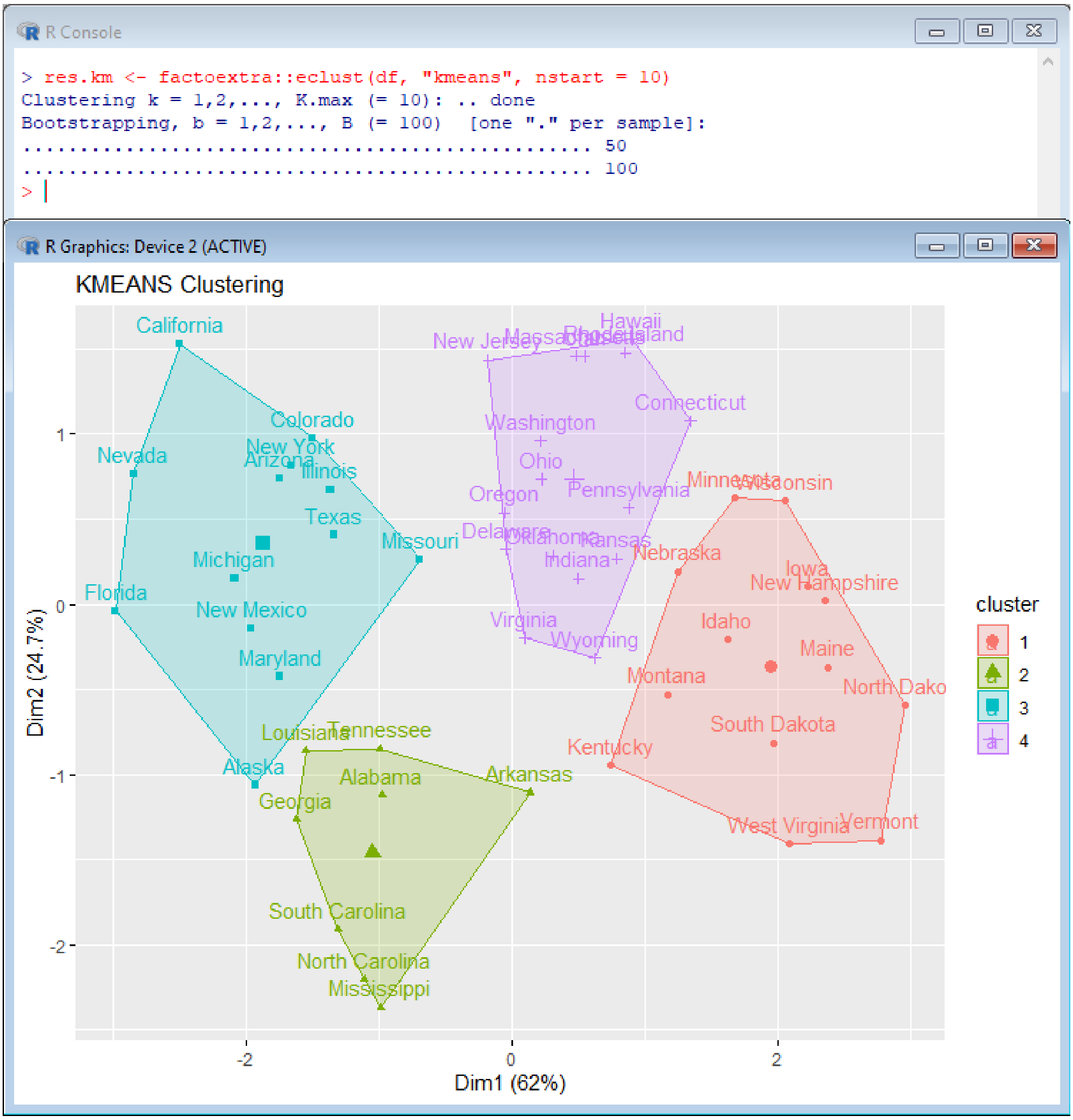


Ok c'est sympa cela nous donne une première idée de quelle vielle ressemble à laquelle. Mais on peut faire mieux!

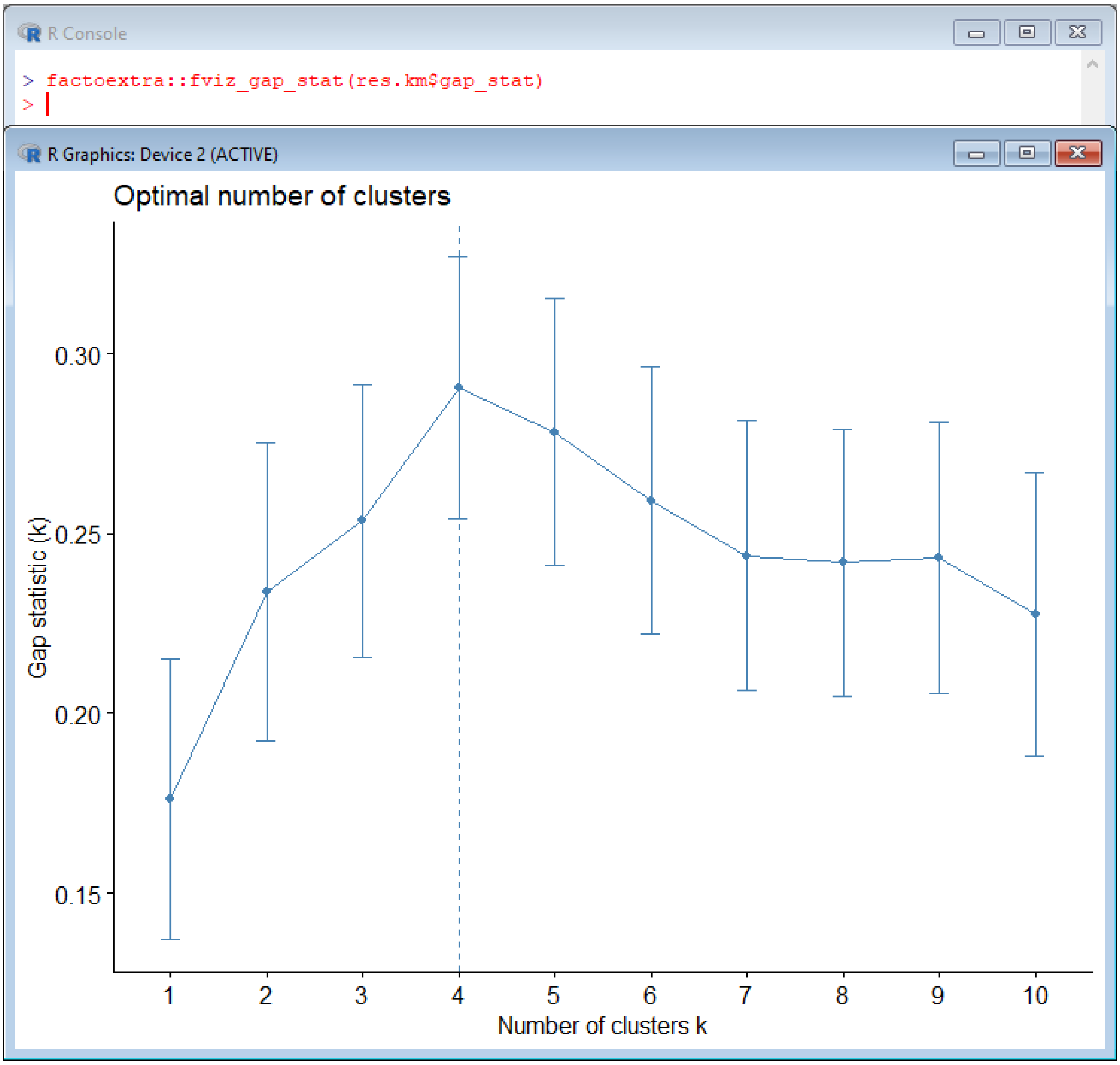
Faisons un clustering ascendant hiérarchique:



Et un clustering par la méthode des K-means:



On peut vérifier le nombre de cluster optimaux avec la fonction fviz_gap_stat( ):



On peut observer aussi le contenu de l'objet de clustering par la méthode des k-means:

RGui (64-bit) - [R Console]

```
> res.km
K-means clustering with 4 clusters of sizes 13, 8, 13, 16

Cluster means:
      Murder    Assault   UrbanPop         Rape
1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
2  1.4118898  0.8743346 -0.8145211  0.01927104
3  0.6950701  1.0394414  0.7226370  1.27693964
4 -0.4894375 -0.3826001  0.5758298 -0.26165379

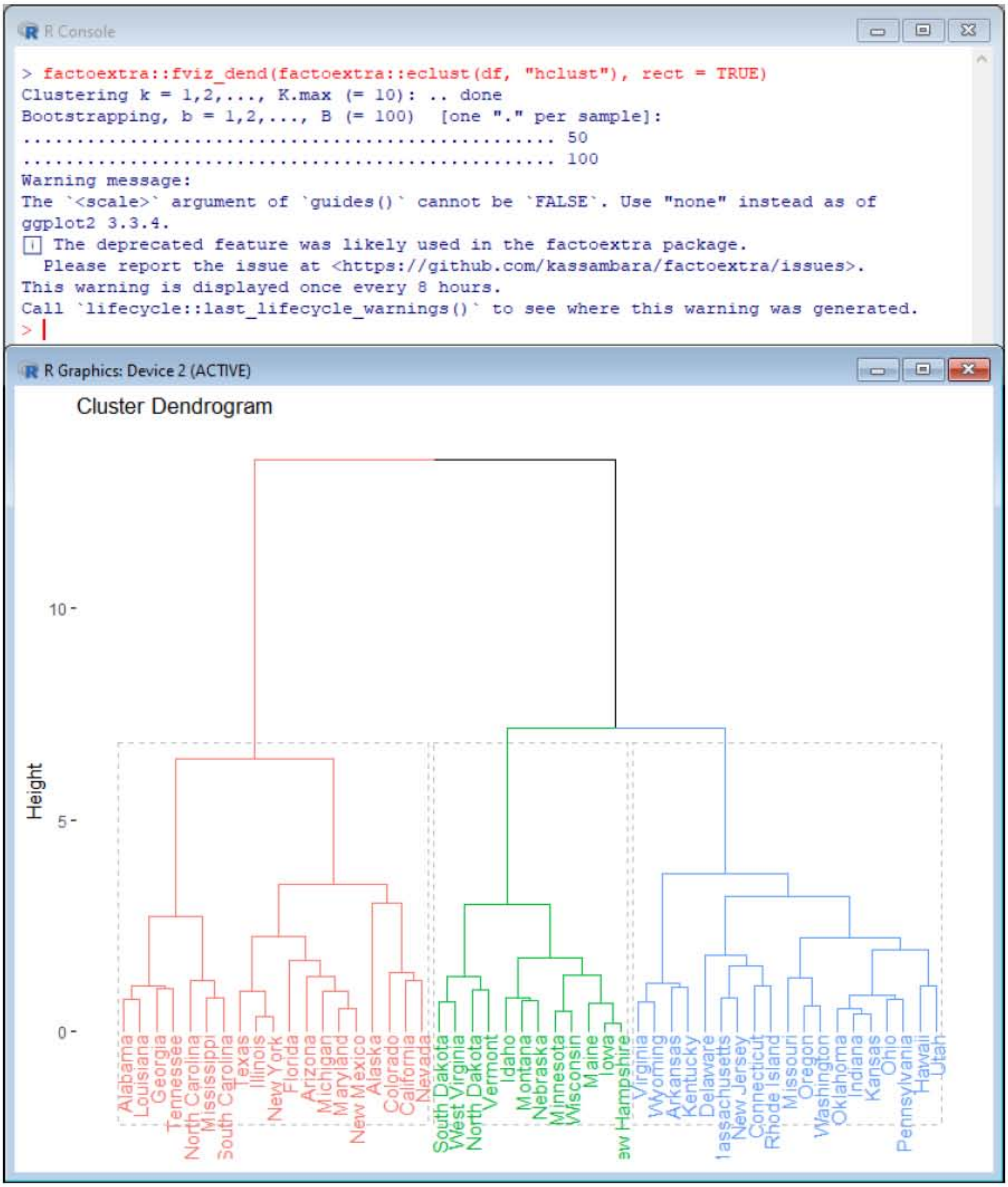
Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California
             2              3              3              2              3
      Colorado    Connecticut       Delaware        Florida        Georgia
             3              4              4              3              2
        Hawaii          Idaho       Illinois        Indiana           Iowa
             4              1              3              4              1
        Kansas       Kentucky      Louisiana          Maine       Maryland
             4              1              2              1              3
 Massachusetts       Michigan      Minnesota    Mississippi       Missouri
             4              3              1              2              3
       Montana       Nebraska         Nevada  New Hampshire     New Jersey
             1              1              3              1              4
    New Mexico       New York North Carolina   North Dakota           Ohio
             3              3              2              1              4
      Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
             4              4              4              4              2
  South Dakota      Tennessee          Texas           Utah        Vermont
             1              2              3              4              1
      Virginia     Washington  West Virginia      Wisconsin        Wyoming
             4              4              1              1              4

Within cluster sum of squares by cluster:
[1] 11.952463  8.316061 19.922437 16.212213
 (between_SS / total_SS =  71.2 %)

Available components:

 [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
 [6] "betweenss"    "size"         "iter"         "ifault"       "clust_plot"
[11] "silinfo"      "nbclust"      "data"         "gap_stat"
> 
```

On peut faire un clustering ascendant hiérarchique en utilisant les fonctions native de factoextra dont **fviz_dend( )** et **eclust( )**:


R Console
> factoextra::fviz_dend(factoextra::eclust(df, "hclust"), rect = TRUE)
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 100) [one "." per sample]:
.... 50
.... 100
Warning message:
The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as of ggplot2 3.3.4.
i The deprecated feature was likely used in the factoextra package.
Please report the issue at <https://github.com/kassambara/factoextra/issues>.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
R Graphics: Device 2 (ACTIVE)
Cluster Dendrogram
Height
10
5
0
Alabama
Louisiana
Georgia
Tennessee
North Carolina
Mississippi
South Carolina
Texas
Illinois
New York
Florida
Arizona
Michigan
Maryland
New Mexico
Alaska
Colorado
California
Nevada
South Dakota
West Virginia
North Dakota
Vermont
Idaho
Montana
Nebraska
Minnesota
Wisconsin
Maine
Iowa
Virginia
Wyoming
Arkansas
Kentucky
Delaware
New Jersey
Connecticut
Rhode Island
Missouri
Oregon
Washington
Oklahoma
Indiana
Kansas
Ohio
Pennsylvania
Hawaii
Utah

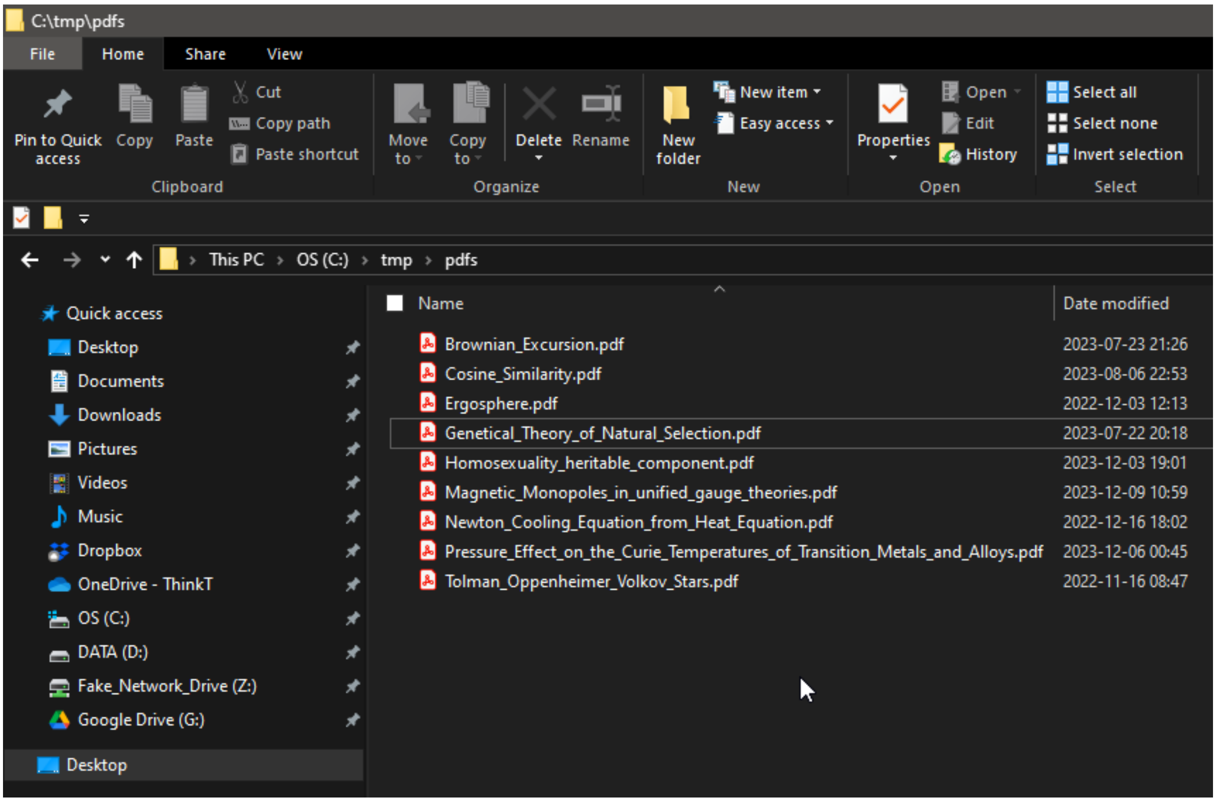
Exemple 441.: Clustering et ACP de multiples PDFs

R 4.1

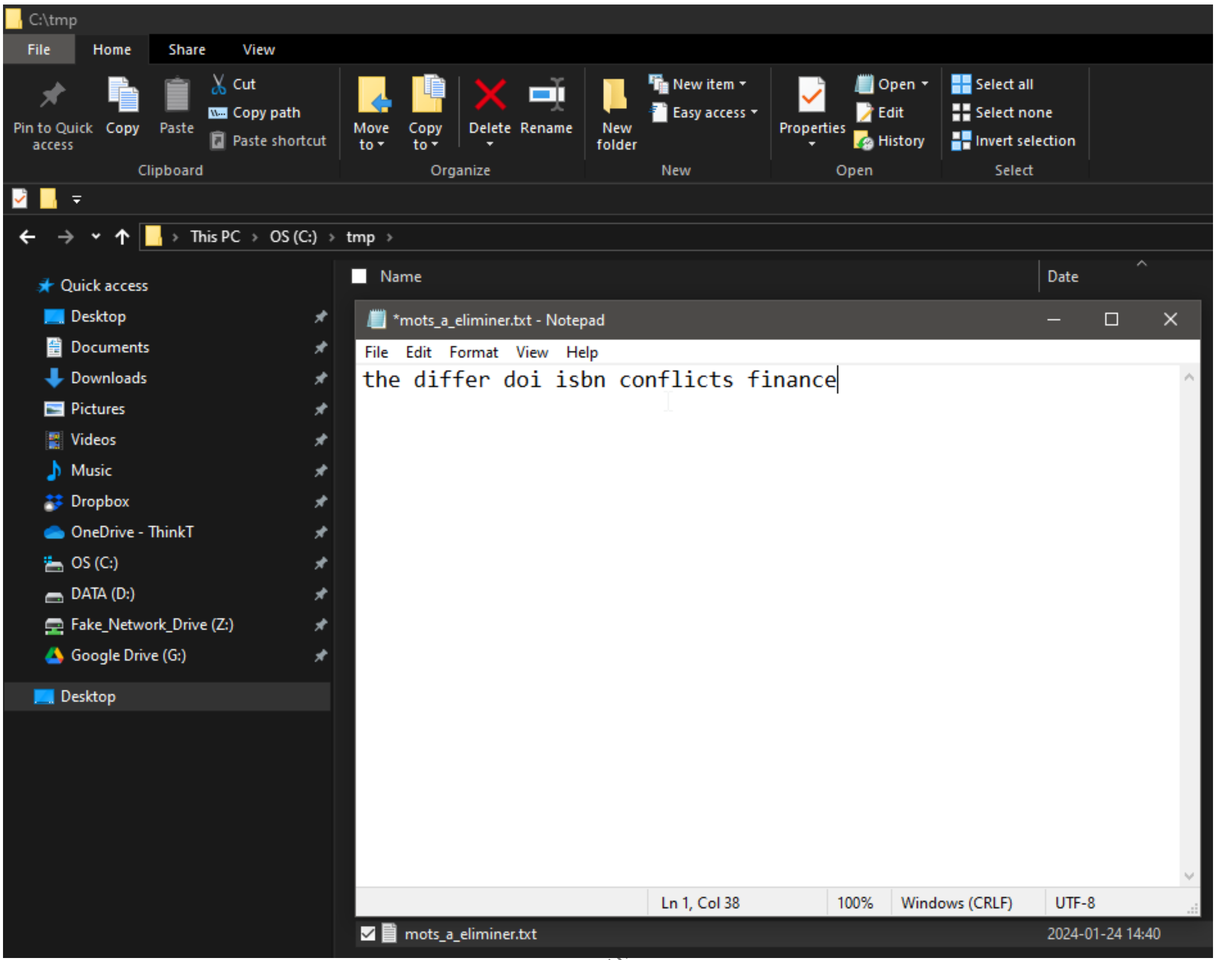
Nous allons voir ici comment lire un dossier de PDFs multiples, analyser la fréquences des termes dans chaque PDF, afficher sous forme arbre hiérarchique la similarité entre les PDFs et enfin faire de même en utilisant une Analyse en Composantes Principales (ACP).

Exploration des données

Pour cela nous allons d'abord préparer un dossier de PDFs comme par exemple:



Et un fichier texte dans un autre dossier avec les mots que nous voudrons ignorer lors de notre analyse (mots séparés par des espaces):



Ensuite, nous utilisons le package **tm** et sa fonction VCorpus( ) pour lire tous les PDFs et leurs contenus:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("tm")
Loading required package: NLP
> rm(list = ls())
>
> #on lit le dossier avec les PDFs
> directory <- "c:/tmp/pdfs"
> my.corpus <- VCorpus(DirSource(directory, pattern = ".pdf"),
+                                  readerControl = list(reader = readPDF))
>
> #on stock à part les noms des fichiers pour les graphiques plus tard
> (file.names <- meta(my.corpus,tag='id'))
$Brownian_Excursion.pdf
[1] "Brownian_Excursion.pdf"

$Cosine_Similarity.pdf
[1] "Cosine_Similarity.pdf"

$Ergosphere.pdf
[1] "Ergosphere.pdf"

$Genetical_Theory_of_Natural_Selection.pdf
[1] "Genetical_Theory_of_Natural_Selection.pdf"

$Homosexuality_heritable_component.pdf
[1] "Homosexuality_heritable_component.pdf"

$Magnetic_Monopoles_in_unified_gauge_theories.pdf
[1] "Magnetic_Monopoles_in_unified_gauge_theories.pdf"

$Newton_Cooling_Equation_from_Heat_Equation.pdf
[1] "Newton_Cooling_Equation_from_Heat_Equation.pdf"

$Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf
[1] "Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf"

$Tolman_Oppenheimer_Volkov_Stars.pdf
[1] "Tolman_Oppenheimer_Volkov_Stars.pdf"

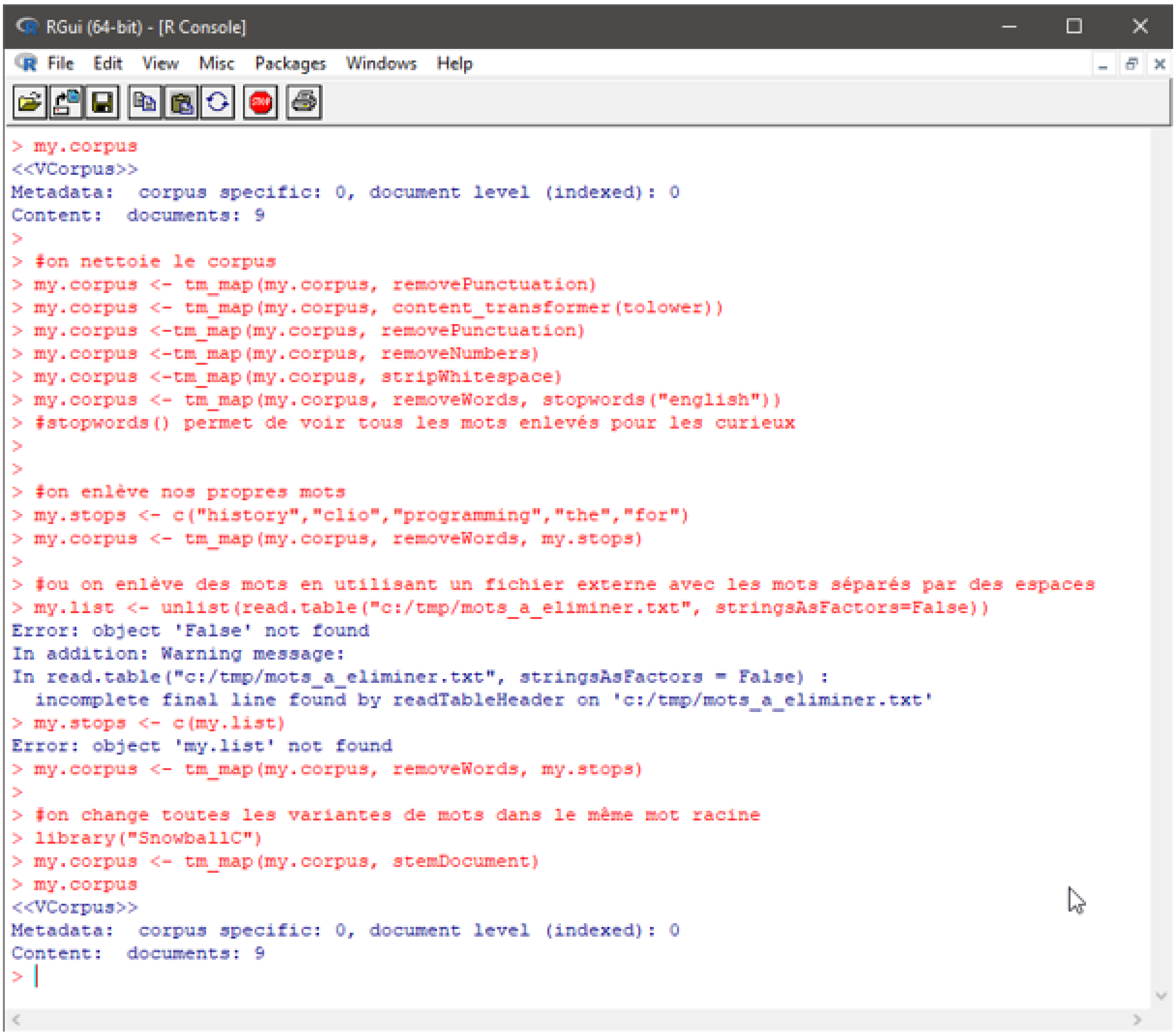
> |
```

Nous avons également montré comme vous pouvez récupérer les noms des fichiers au cas où dans les graphiques plus tard vous souhaiteriez les utiliser comme légendes (mais c'est pas une bonne idée vu la longueur des noms de fichiers).

Voyons maintenant comment préparer le corpus!

Préparation du corpus

Encore une fois, nous utilisons le package **tm** pour cela et sa fonction tm_map( ) mais aussi le package SnowballC pour changer toutes les variantes de mots dans la même mot racine:



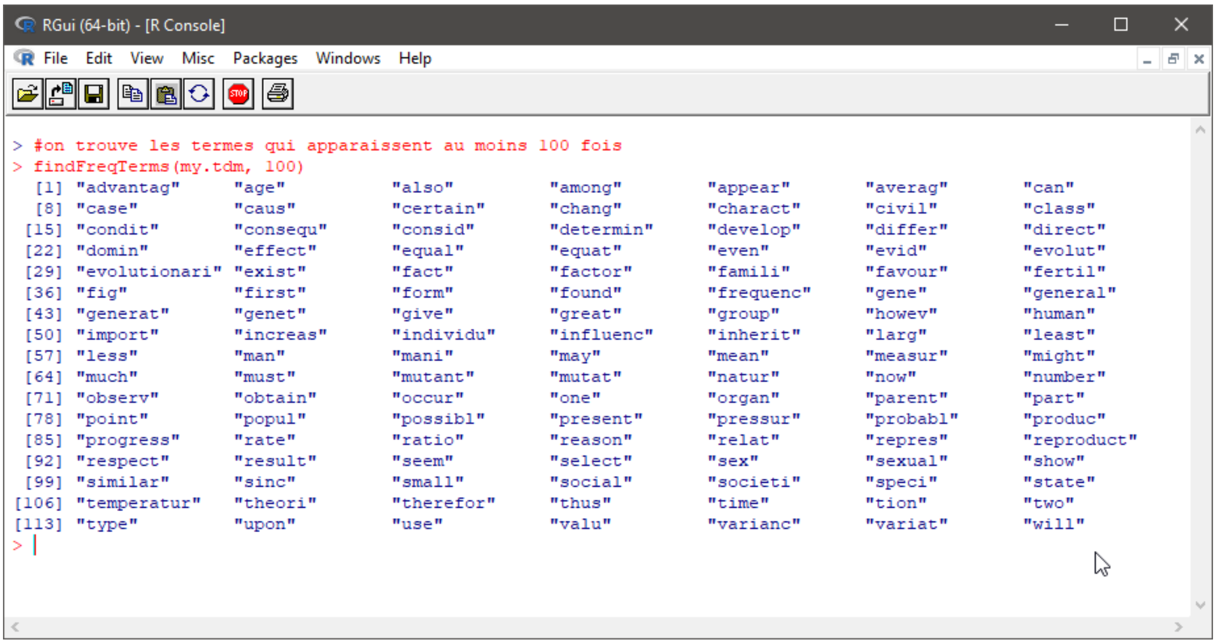
Nous transformons le corpus en matrice de termes documentaires et observons ce qu'elle contient en termes de fréquence de termes à l'aide de TermDocumentMatrix( ) et inspect( ):

```
> #on inspecte un peu le corpus
> my.tdm <- TermDocumentMatrix(my.corpus)
> inspect(my.tdm)
<<TermDocumentMatrix (terms: 8276, documents: 9)>>
Non-/sparse entries: 11222/63262
Sparsity           : 85%
Maximal term length: 55
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms     Brownian_Excursion.pdf Cosine_Similarity.pdf Ergosphere.pdf Genetical_Theory_of_Natural_Selection.pdf
  differ                       0                    20              1                                       415
  effect                       0                     3              5                                       306
  may                          0                     5              0                                       419
  mutat                        0                     0              0                                       362
  natur                        0                     0              0                                       310
  number                       0                     2              0                                       308
  one                          0                     7              8                                       276
  select                       0                     0              0                                       666
  valu                         0                     7              1                                       254
  will                         0                    12              5                                       604
        Docs
Terms     Homosexuality_heritable_component.pdf Magnetic_Monopoles_in_unified_gauge_theories.pdf
  differ                                     15                                                0
  effect                                      3                                                0
  may                                         8                                                8
  mutat                                       1                                                0
  natur                                       2                                                0
  number                                      0                                                0
  one                                        10                                               18
  select                                      3                                                0
  valu                                        0                                                7
  will                                        3                                                7
        Docs
Terms     Newton_Cooling_Equation_from_Heat_Equation.pdf
  differ                                              11
  effect                                               1
  may                                                  1
  mutat                                                0
  natur                                                1
  number                                               2
  one                                                  3
  select                                               0
  valu                                                28
  will                                                20
        Docs
Terms     Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf
  differ                                                                             12
  effect                                                                             27
  may                                                                                 2
  mutat                                                                               0
  natur                                                                               0
  number                                                                              4
  one                                                                                20
  select                                                                              0
  valu                                                                               23
  will                                                                                6
```

Ou en utilisant le TfIdf pondéré:

```
> #autre vue possible du corpus
> my.dtm <- DocumentTermMatrix(my.corpus, control = list(weighting = weightTfIdf, stopwords = TRUE))
> inspect(my.dtm)
<<DocumentTermMatrix (documents: 9, terms: 8271)>>
Non-/sparse entries: 11214/63225
Sparsity           : 85%
Maximal term length: 55
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
                                                                            Terms
Docs                                                                               access     annal
  Brownian_Excursion.pdf                                                      0.1320802084 0.2641604
  Cosine_Similarity.pdf                                                       0.0027035608 0.0000000
  Ergosphere.pdf                                                              0.0000000000 0.0000000
  Genetical_Theory_of_Natural_Selection.pdf                                   0.0001170362 0.0000000
  Homosexuality_heritable_component.pdf                                       0.0000000000 0.0000000
  Magnetic_Monopoles_in_unified_gauge_theories.pdf                            0.0000000000 0.0000000
  Newton_Cooling_Equation_from_Heat_Equation.pdf                              0.0000000000 0.0000000
  Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf 0.0000000000 0.0000000
  Tolman_Oppenheimer_Volkov_Stars.pdf                                         0.0000000000 0.0000000
                                                                            Terms
Docs                                                                            collabor     digit
  Brownian_Excursion.pdf                                                      0.180827083 0.2641604
  Cosine_Similarity.pdf                                                       0.001850682 0.0000000
  Ergosphere.pdf                                                              0.000000000 0.0000000
  Genetical_Theory_of_Natural_Selection.pdf                                   0.000000000 0.0000000
  Homosexuality_heritable_component.pdf                                       0.000000000 0.0000000
  Magnetic_Monopoles_in_unified_gauge_theories.pdf                            0.000000000 0.0000000
  Newton_Cooling_Equation_from_Heat_Equation.pdf                              0.000000000 0.0000000
  Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf 0.000000000 0.0000000
  Tolman_Oppenheimer_Volkov_Stars.pdf                                         0.000000000 0.0000000
                                                                            Terms
Docs                                                                             institut      jstor
  Brownian_Excursion.pdf                                                      0.0974937501 0.18082708
  Cosine_Similarity.pdf                                                       0.0004989019 0.00000000
  Ergosphere.pdf                                                              0.0000000000 0.00000000
  Genetical_Theory_of_Natural_Selection.pdf                                   0.0001943756 0.00000000
  Homosexuality_heritable_component.pdf                                       0.0000000000 0.01192266
  Magnetic_Monopoles_in_unified_gauge_theories.pdf                            0.0019147709 0.00000000
  Newton_Cooling_Equation_from_Heat_Equation.pdf                              0.0000000000 0.00000000
  Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf 0.0000000000 0.00000000
  Tolman_Oppenheimer_Volkov_Stars.pdf                                         0.0000000000 0.00000000
                                                                            Terms
Docs                                                                              preserv       simx
  Brownian_Excursion.pdf                                                      0.1808270835 0.00000000
  Cosine_Similarity.pdf                                                       0.0000000000 0.08651394
  Ergosphere.pdf                                                              0.0000000000 0.00000000
  Genetical_Theory_of_Natural_Selection.pdf                                   0.0003204615 0.00000000
  Homosexuality_heritable_component.pdf                                       0.0000000000 0.00000000
  Magnetic_Monopoles_in_unified_gauge_theories.pdf                            0.0000000000 0.00000000
  Newton_Cooling_Equation_from_Heat_Equation.pdf                              0.0000000000 0.00000000
  Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf 0.0000000000 0.00000000
```

Nous pouvons lister les 100 termes les plus fréquents avec findFreqTerms( ):



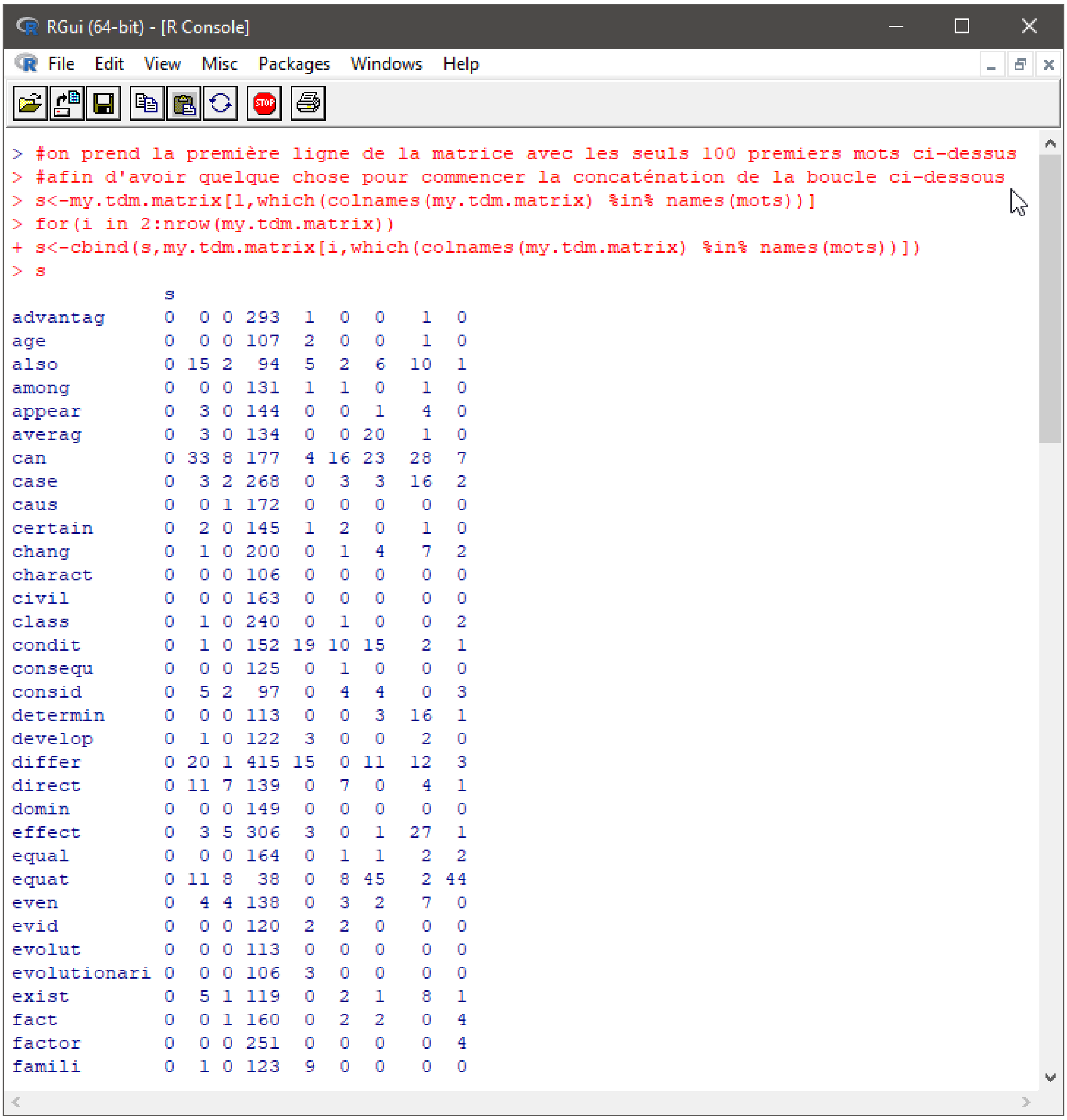
Nous préparons maintenant spécifiquement les données pour l'analyse en composantes principales et pour le clustering hiérarchique:

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

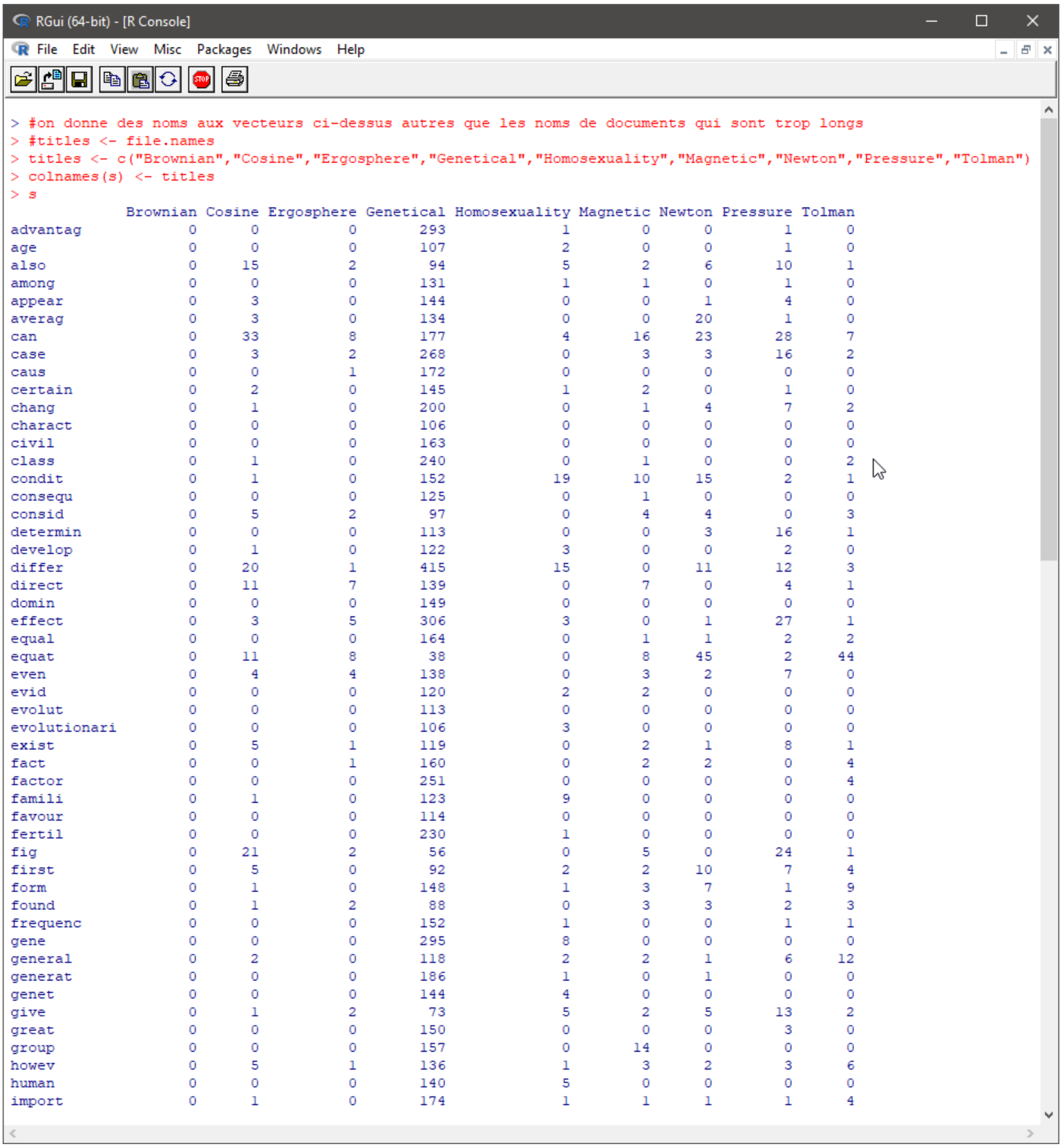
```
> #on prépare tout ce qu'il faut pour l'ACP et le HClut des documents
>
> #on prépare la matrice des termes par documents
> my.tdm.matrix <- t(as.matrix(my.tdm))
> dim(t(my.tdm.matrix))
[1] 8276    9
> #voyons ce qu'elle contient just pour les 5 premières colonnes
> my.tdm.matrix[,1:5]
                                                                         Terms
Docs                                                                       \017eφ \022∂r \023ni -ba -λk
  Brownian_Excursion.pdf                                                        0      0      0   0   0
  Cosine_Similarity.pdf                                                         0      0      0   0   0
  Ergosphere.pdf                                                                0      0      0   1   0
  Genetical_Theory_of_Natural_Selection.pdf                                     0      0      0   0   0
  Homosexuality_heritable_component.pdf                                         0      0      0   0   0
  Magnetic_Monopoles_in_unified_gauge_theories.pdf                              0      0      0   0   0
  Newton_Cooling_Equation_from_Heat_Equation.pdf                                0      0      0   0   1
  Pressure_Effect_on_the_Curie_Temperatures_of_Transition_Metals_and_Alloys.pdf 0      0      0   0   0
  Tolman_Oppenheimer_Volkov_Stars.pdf                                           1      1      2   0   0
>
> #on garde que les mots qui se répètent les plus souvent globalement
> frequency <- colSums(my.tdm.matrix)
> frequency <- sort(frequency, decreasing=TRUE)
> #on choisit arbitrairement une limite de 100
> (mots<-frequency[frequency>100])
      select         will       differ          may        mutat       effect          one         valu
         669          659          477          452          363          346          344          322
      number        natur        speci         must         gene         case          can     advantag
         317          314          307          306          303          297          296          295
     increas    reproduct       social          two       factor     individu        class       fertil
         293          269          259          257          255          245          244          231
        rate       sexual        chang        organ       theori         upon       condit         less
         230          222          215          214          208          207          200          200
       popul          use      generat       import        relat         caus        group        equal
         198          192          188          183          183          173          171          170
        form       direct         fact      possibl         seem         time        civil       averag
         170          169          169          165          165          165          163          158
        even        howev        equat       observ     frequenc         tion        great       appear
         158          157          156          156          155          155          153          152
     certain         show        domin        genet      societi       variat        human      general
         151          151          149          148          147          147          145          143
     varianc       produc        exist         also      similar        among     determin       famili
         139          138          137          135          135          134          133          133
         man       measur         sinc   temperatur     therefor      develop      inherit         larg
         133          133          132          129          129          128          128          128
        mean      consequ         type          sex         evid        point        first         much
         127          126          126          125          124          124          122          121
     pressur      respect       consid       favour        least       obtain       evolut        occur
         121          121          115          114          114          114          113          113
     probabl          age      present     progress evolutionari          fig       mutant     influenc
         112          110          110          110          109          109          108          107
         now      charact       result         mani       parent        small         give        might
         107          106          106          105          104          104          103          103
       found         part        ratio       reason       repres         thus
         102          102          102          102          102          101
>
> |
```

On prépare la matrice avec une structure particulière adaptée à l'ACP et au HClust à l'aide d'une boucle:



```
> #on prend la première ligne de la matrice avec les seuls 100 premiers mots ci-dessus
> #afin d'avoir quelque chose pour commencer la concaténation de la boucle ci-dessous
> s<-my.tdm.matrix[1,which(colnames(my.tdm.matrix) %in% names(mots))]
> for(i in 2:nrow(my.tdm.matrix))
+ s<-cbind(s,my.tdm.matrix[i,which(colnames(my.tdm.matrix) %in% names(mots))])
> s
             s
advantag     0  0 0 293  1  0  0   1  0
age          0  0 0 107  2  0  0   1  0
also         0 15 2  94  5  2  6  10  1
among        0  0 0 131  1  1  0   1  0
appear       0  3 0 144  0  0  1   4  0
averag       0  3 0 134  0  0 20   1  0
can          0 33 8 177  4 16 23  28  7
case         0  3 2 268  0  3  3  16  2
caus         0  0 1 172  0  0  0   0  0
certain      0  2 0 145  1  2  0   1  0
chang        0  1 0 200  0  1  4   7  2
charact      0  0 0 106  0  0  0   0  0
civil        0  0 0 163  0  0  0   0  0
class        0  1 0 240  0  1  0   0  2
condit       0  1 0 152 19 10 15   2  1
consequ      0  0 0 125  0  1  0   0  0
consid       0  5 2  97  0  4  4   0  3
determin     0  0 0 113  0  0  3  16  1
develop      0  1 0 122  3  0  0   2  0
differ       0 20 1 415 15  0 11  12  3
direct       0 11 7 139  0  7  0   4  1
domin        0  0 0 149  0  0  0   0  0
effect       0  3 5 306  3  0  1  27  1
equal        0  0 0 164  0  1  1   2  2
equat        0 11 8  38  0  8 45   2 44
even         0  4 4 138  0  3  2   7  0
evid         0  0 0 120  2  2  0   0  0
evolut       0  0 0 113  0  0  0   0  0
evolutionari 0  0 0 106  3  0  0   0  0
exist        0  5 1 119  0  2  1   8  1
fact         0  0 1 160  0  2  2   0  4
factor       0  0 0 251  0  0  0   0  4
famili       0  1 0 123  9  0  0   0  0
```

C'est pas mal mais le titre des colonnes est catastrophique (inexistant), nous remédions à cela:



```
> #on donne des noms aux vecteurs ci-dessus autres que les noms de documents qui sont trop longs
> #titles <- file.names
> titles <- c("Brownian","Cosine","Ergosphere","Genetical","Homosexuality","Magnetic","Newton","Pressure","Tolman")
> colnames(s) <- titles
> s
```

	Brownian	Cosine	Ergosphere	Genetical	Homosexuality	Magnetic	Newton	Pressure	Tolman
advantag	0	0	0	293	1	0	0	1	0
age	0	0	0	107	2	0	0	1	0
also	0	15	2	94	5	2	6	10	1
among	0	0	0	131	1	1	0	1	0
appear	0	3	0	144	0	0	1	4	0
averag	0	3	0	134	0	0	20	1	0
can	0	33	8	177	4	16	23	28	7
case	0	3	2	268	0	3	3	16	2
caus	0	0	1	172	0	0	0	0	0
certain	0	2	0	145	1	2	0	1	0
chang	0	1	0	200	0	1	4	7	2
charact	0	0	0	106	0	0	0	0	0
civil	0	0	0	163	0	0	0	0	0
class	0	1	0	240	0	1	0	0	2
condit	0	1	0	152	19	10	15	2	1
consequ	0	0	0	125	0	1	0	0	0
consid	0	5	2	97	0	4	4	0	3
determin	0	0	0	113	0	0	3	16	1
develop	0	1	0	122	3	0	0	2	0
differ	0	20	1	415	15	0	11	12	3
direct	0	11	7	139	0	7	0	4	1
domin	0	0	0	149	0	0	0	0	0
effect	0	3	5	306	3	0	1	27	1
equal	0	0	0	164	0	1	1	2	2
equat	0	11	8	38	0	8	45	2	44
even	0	4	4	138	0	3	2	7	0
evid	0	0	0	120	2	2	0	0	0
evolut	0	0	0	113	0	0	0	0	0
evolutionari	0	0	0	106	3	0	0	0	0
exist	0	5	1	119	0	2	1	8	1
fact	0	0	1	160	0	2	2	0	4
factor	0	0	0	251	0	0	0	0	4
famili	0	1	0	123	9	0	0	0	0
favour	0	0	0	114	0	0	0	0	0
fertil	0	0	0	230	1	0	0	0	0
fig	0	21	2	56	0	5	0	24	1
first	0	5	0	92	2	2	10	7	4
form	0	1	0	148	1	3	7	1	9
found	0	1	2	88	0	3	3	2	3
frequenc	0	0	0	152	1	0	0	1	1
gene	0	0	0	295	8	0	0	0	0
general	0	2	0	118	2	2	1	6	12
generat	0	0	0	186	1	0	1	0	0
genet	0	0	0	144	4	0	0	0	0
give	0	1	2	73	5	2	5	13	2
great	0	0	0	150	0	0	0	3	0
group	0	0	0	157	0	14	0	0	0
howev	0	5	1	136	1	3	2	3	6
human	0	0	0	140	5	0	0	0	0
import	0	1	0	174	1	1	1	1	4

Maintenant que notre jeu de données est prêt, nous allons pouvoir passer à l'ACP et au clustering hiérarchique.

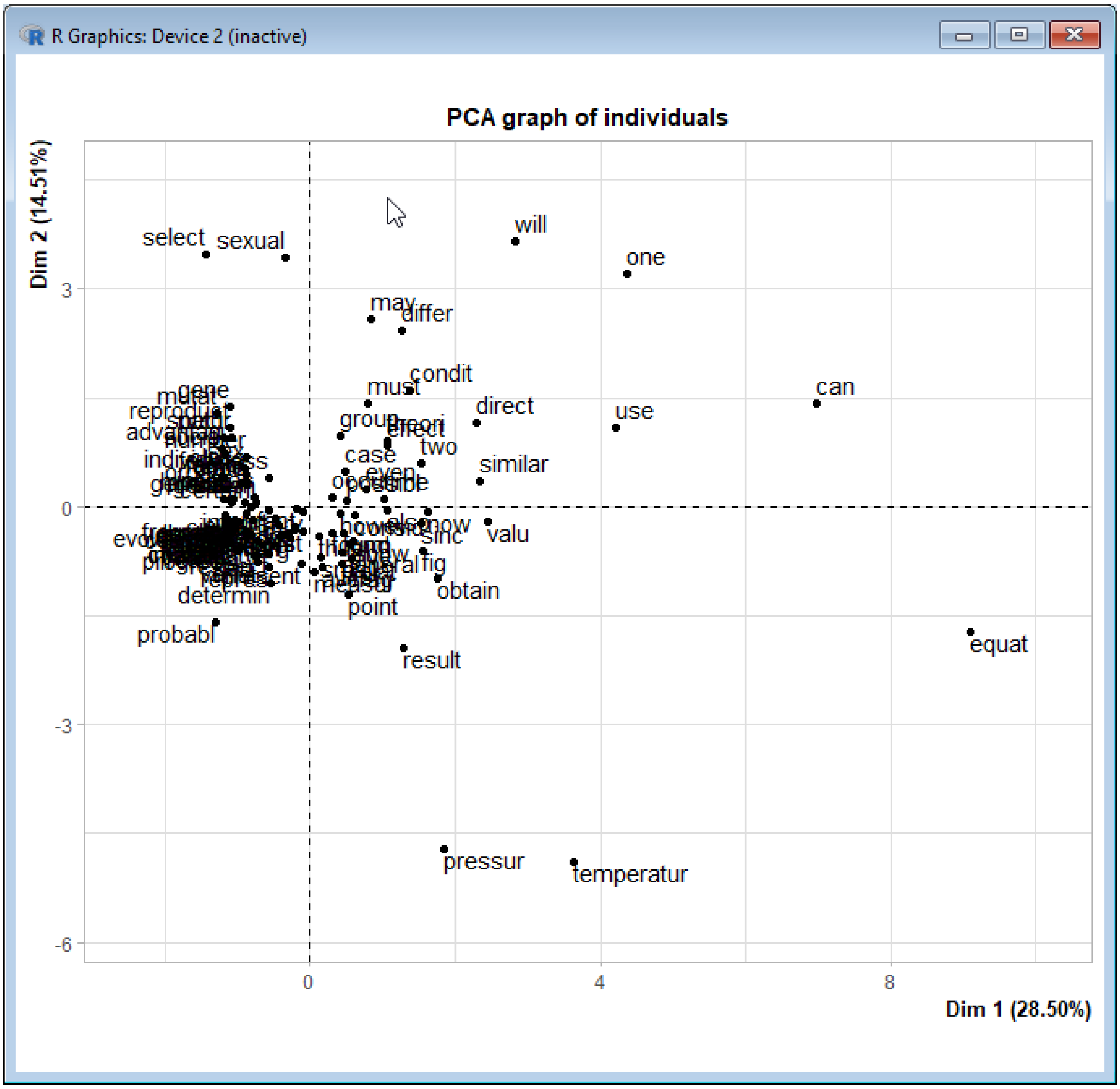
ACP et Clustering Hiérarchique

La suite va être rapide grâce au package FactoMineR et sa fonction PCA( ):

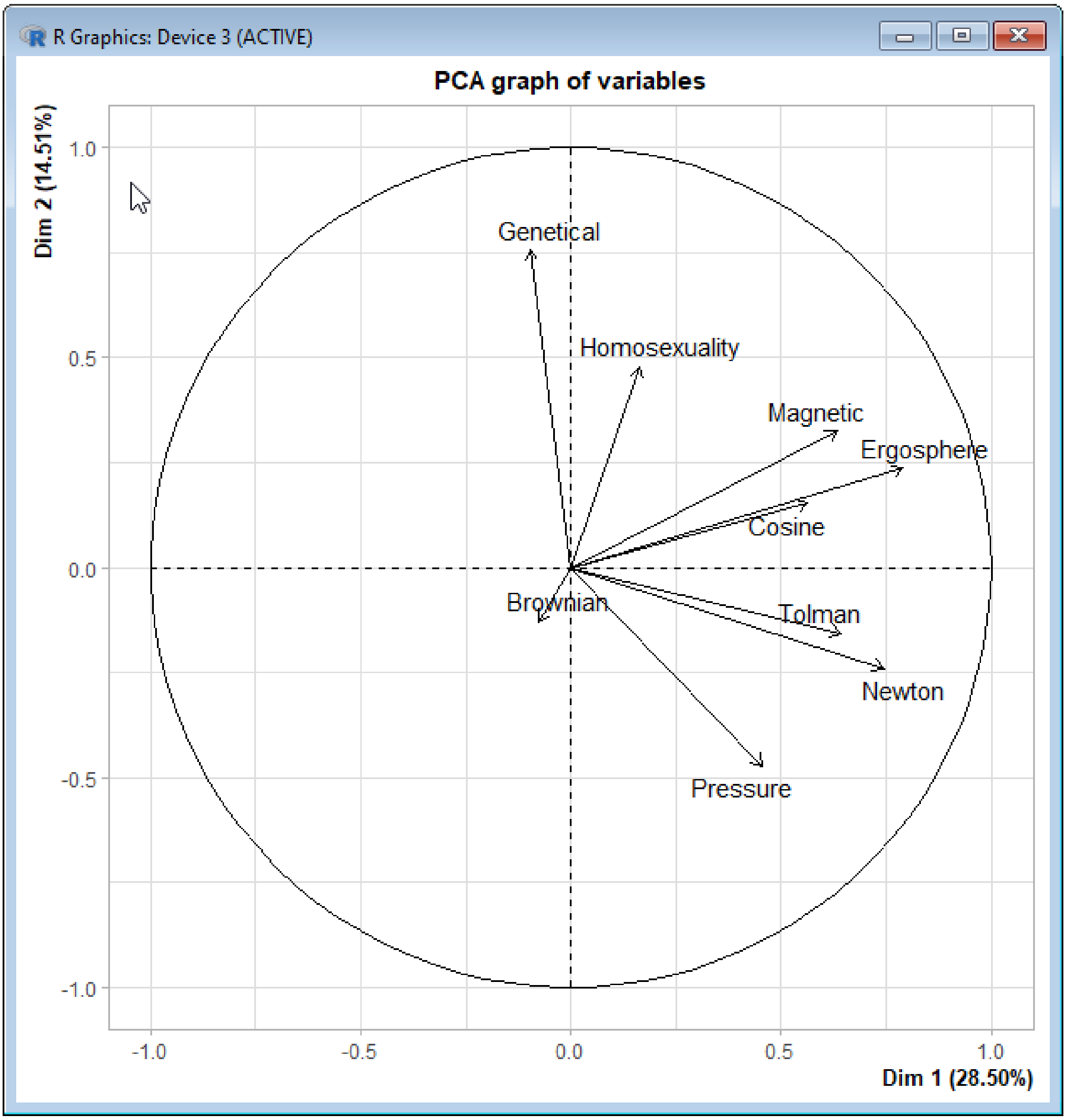
```
> library("FactoMineR")
> PCA(s)
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 118 individuals, described by 9 variables
*The results are available in the following objects:

   name               description
1  "$eig"             "eigenvalues"
2  "$var"             "results for the variables"
3  "$var$coord"       "coord. for the variables"
4  "$var$cor"         "correlations variables - dimensions"
5  "$var$cos2"        "cos2 for the variables"
6  "$var$contrib"     "contributions of the variables"
7  "$ind"             "results for the individuals"
8  "$ind$coord"       "coord. for the individuals"
9  "$ind$cos2"        "cos2 for the individuals"
10 "$ind$contrib"     "contributions of the individuals"
11 "$call"            "summary statistics"
12 "$call$centre"     "mean of the variables"
13 "$call$ecart.type" "standard error of the variables"
14 "$call$row.w"      "weights for the individuals"
15 "$call$col.w"      "weights for the variables"
>
```

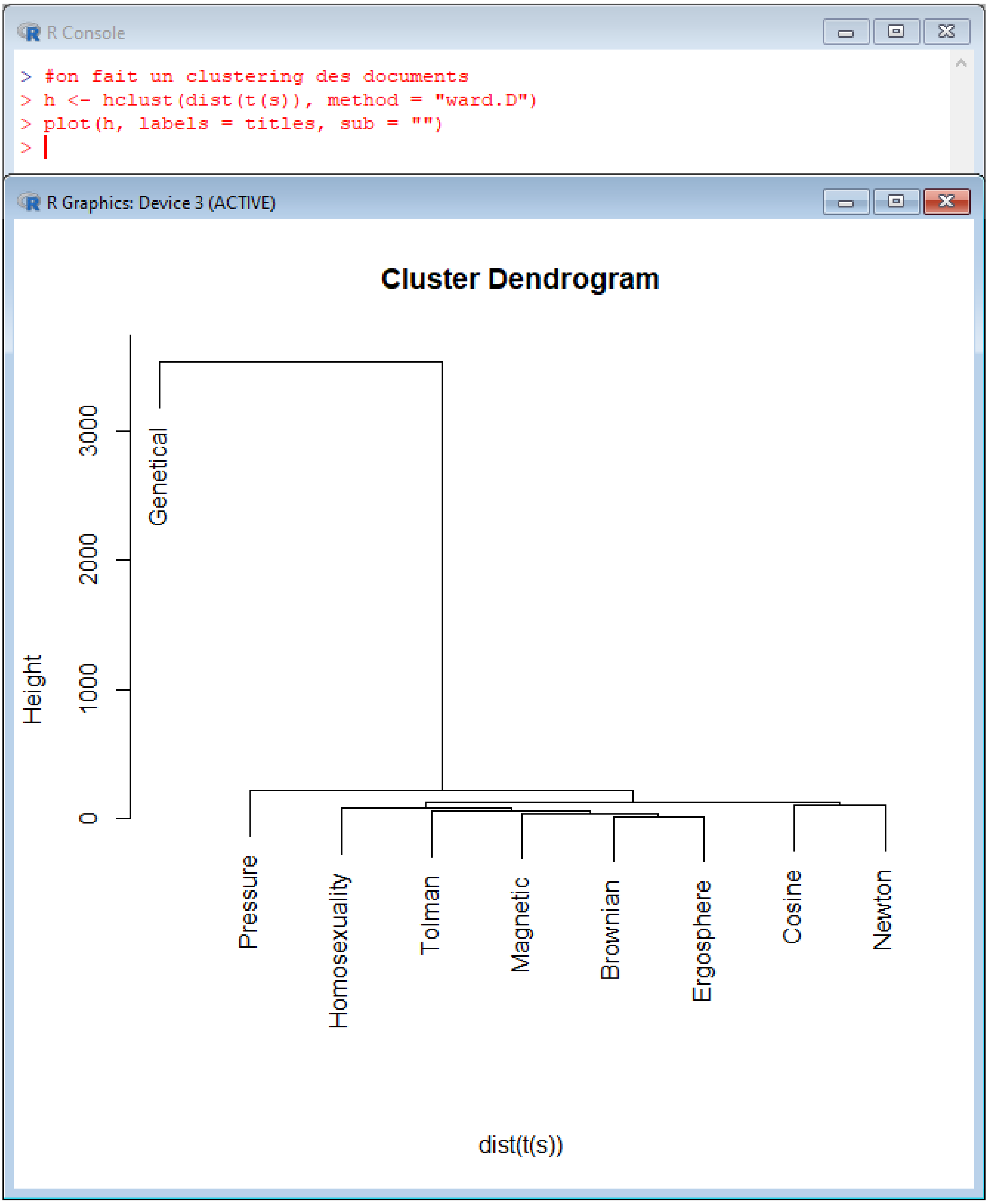
Ce qui donne sur les deux axes les plus explicatifs:



Et nous avons pour les vecteurs propres:



Et comme nous le savons, plusieurs méthodes ne donnent pas toujours les mêmes résultats, faisons une analyse par clustering hiérarchique. Ce qui va nous donner avec la fonction hclust( ) native de R:



Voici le code complet de tout ce que nous avons vu ci-dessus:

```
library("tm")
rm(list = ls())

#on lit le dossier avec les PDFs
directory <- "c:/tmp/pdfs"
my.corpus <- VCorpus(DirSource(directory, pattern = ".pdf"),
                              readerControl = list(reader =
readPDF))
```

```
#on stock à part les noms des fichiers pour les graphiques plus tard
(file.names <- meta(my.corpus,tag='id'))

my.corpus

#on nettoie le corpus
my.corpus <- tm_map(my.corpus, removePunctuation)
my.corpus <- tm_map(my.corpus, content_transformer(tolower))
my.corpus <-tm_map(my.corpus, removePunctuation)
my.corpus <-tm_map(my.corpus, removeNumbers)
my.corpus <-tm_map(my.corpus, stripWhitespace)
my.corpus <- tm_map(my.corpus, removeWords, stopwords("english"))
#stopwords() permet de voir tous les mots enlevés pour les curieux


#on enlève nos propres mots
my.stops <- c("history","clio","programming","the","for")
my.corpus <- tm_map(my.corpus, removeWords, my.stops)

#ou on enlève des mots en utilisant un fichier externe avec les mots
séparés par des espaces
my.list <- unlist(read.table("c:/tmp/mots_a_eliminer.txt",
stringsAsFactors=False))
my.stops <- c(my.list)
my.corpus <- tm_map(my.corpus, removeWords, my.stops)

#on change toutes les variantes de mots dans le même mot racine
library("SnowballC")
my.corpus <- tm_map(my.corpus, stemDocument)
my.corpus

#on inspecte un peu le corpus
my.tdm <- TermDocumentMatrix(my.corpus)
inspect(my.tdm)

#autre vue possible du corpus
my.dtm <- DocumentTermMatrix(my.corpus, control = list(weighting =
weightTfIdf, stopwords = TRUE))
inspect(my.dtm)

#on trouve les termes qui apparaissent au moins 100 fois
findFreqTerms(my.tdm, 100)

#on prépare tout ce qu'il faut pour l'ACP et le HClut des documents

#on prépare la matrice des termes par documents
my.tdm.matrix <- t(as.matrix(my.tdm))
dim(t(my.tdm.matrix))
#voyons ce qu'elle contient just pour les 5 premières colonnes
my.tdm.matrix[,1:5]

#on garde que les mots qui se répètent les plus souvent globalement
frequency <- colSums(my.tdm.matrix)
frequency <- sort(frequency, decreasing=TRUE)
#on choisit arbitrairement une limite de 100
```

```
(mots<-frequency[frequency>100])

#on prend la première ligne de la matrice avec les seuls 100
premiers mots ci-dessus
#afin d'avoir quelque chose pour commencer la concaténation de la
boucle ci-dessous
s<-my.tdm.matrix[1,which(colnames(my.tdm.matrix) %in% names(mots))]
for(i in 2:nrow(my.tdm.matrix))
      s<-cbind(s,my.tdm.matrix[i,which(colnames(my.tdm.matrix) %in%
names(mots))])
s
#on donne des noms aux vecteurs ci-dessus autres que les noms de
documents qui sont trop longs
#titles <- file.names
titles <-
c("Brownian","Cosine","Ergosphere","Genetical","Homosexuality","Magn
etic","Newton","Pressure","Tolman")
colnames(s) <- titles
s
library("FactoMineR")
PCA(s)

#on fait un clustering des documents
h <- hclust(dist(t(s)), method = "ward.D")
plot(h, labels = titles, sub = "")
```

Exemple 442.: Clustering de corpus de textes

R 4.1

Nous allons regrouper un ensemble de données de tweets du *News Health*. L'objectif n'est pas d'exécuter une technique de pointe pour regrouper des ensembles de données de texte court. Nous allons faire des raccourcis de programmation et d'exécution pour ne pas compliquer ce tutoriel.

Quels que soient les résultats, ce tutoriel vous donnera une bonne idée de la façon de faire de la science des données de base en R et, espérons-le, vous donnera l'envie d'aller plus loin !

L'exemple qui suit est totalement inspiré du tutorial disponible ici (Tanguy Lucci, 2018):

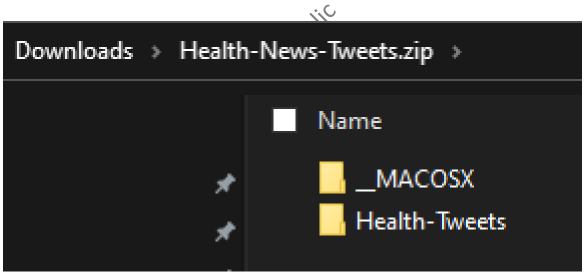
https://medium.com/@SAPCAI/text-clustering-with-r-an-introduction-for-data-scientists-c406e7454e76

Et merci à Shivam P. pour avoir comblé les trous manquants du tutorial!

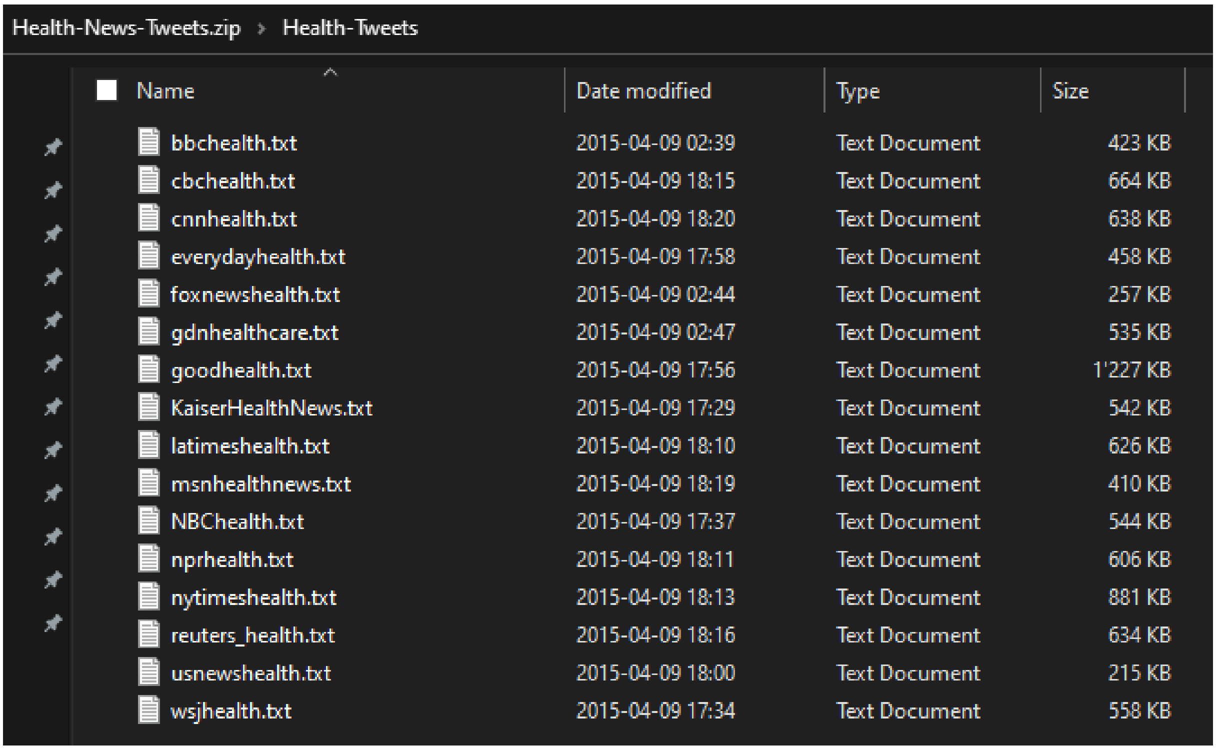
Exploration des données

Nous allons charger le zip directement à partir de la source, décompresser, créer notre bloc de données et supprimer les fichiers temporaires.

Le fichier zip contient:



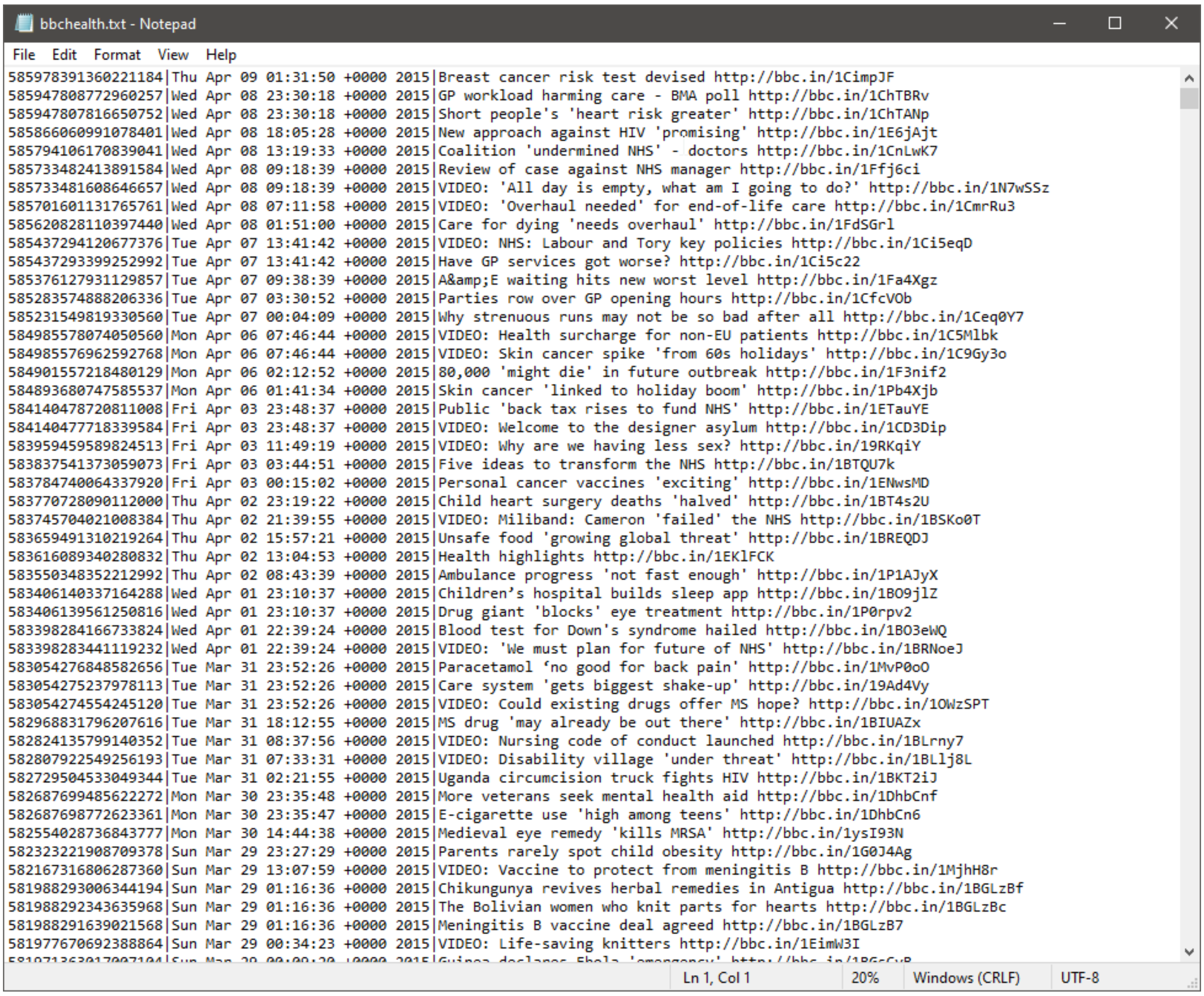
Plus exactement:



Health-News-Tweets.zip › Health-Tweets

Name	Date modified	Type	Size
bbchealth.txt	2015-04-09 02:39	Text Document	423 KB
cbchealth.txt	2015-04-09 18:15	Text Document	664 KB
cnnhealth.txt	2015-04-09 18:20	Text Document	638 KB
everydayhealth.txt	2015-04-09 17:58	Text Document	458 KB
foxnewshealth.txt	2015-04-09 02:44	Text Document	257 KB
gdnhealthcare.txt	2015-04-09 02:47	Text Document	535 KB
goodhealth.txt	2015-04-09 17:56	Text Document	1'227 KB
KaiserHealthNews.txt	2015-04-09 17:29	Text Document	542 KB
latimeshealth.txt	2015-04-09 18:10	Text Document	626 KB
msnhealthnews.txt	2015-04-09 18:19	Text Document	410 KB
NBChealth.txt	2015-04-09 17:37	Text Document	544 KB
nprhealth.txt	2015-04-09 18:11	Text Document	606 KB
nytimeshealth.txt	2015-04-09 18:13	Text Document	881 KB
reuters_health.txt	2015-04-09 18:16	Text Document	634 KB
usnewshealth.txt	2015-04-09 18:00	Text Document	215 KB
wsjhealth.txt	2015-04-09 17:34	Text Document	558 KB

Qui contiennent typiquement:



bbchealth.txt - Notepad

File Edit Format View Help

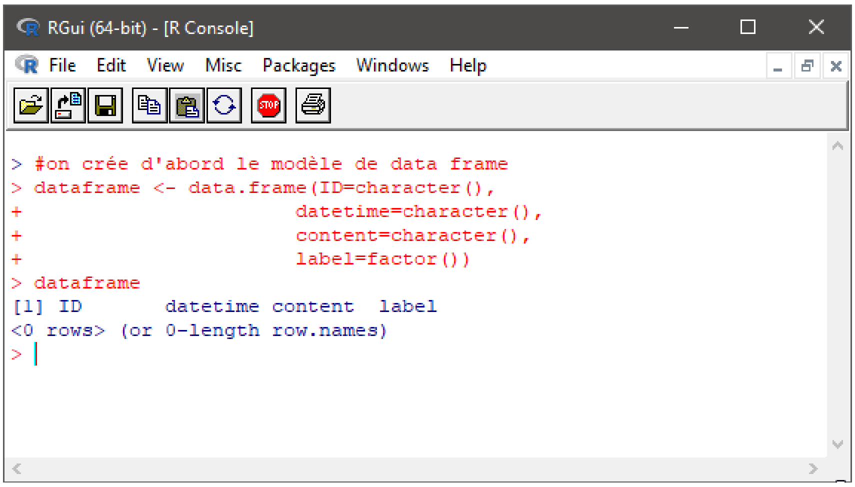
```
585978391360221184|Thu Apr 09 01:31:50 +0000 2015|Breast cancer risk test devised http://bbc.in/1CimpJF
585947808772960257|Wed Apr 08 23:30:18 +0000 2015|GP workload harming care - BMA poll http://bbc.in/1ChTBRv
585947807816650752|Wed Apr 08 23:30:18 +0000 2015|Short people's 'heart risk greater' http://bbc.in/1ChTANp
585866060991078401|Wed Apr 08 18:05:28 +0000 2015|New approach against HIV 'promising' http://bbc.in/1E6jAjt
585794106170839041|Wed Apr 08 13:19:33 +0000 2015|Coalition 'undermined NHS' - doctors http://bbc.in/1CnLwK7
585733482413891584|Wed Apr 08 09:18:39 +0000 2015|Review of case against NHS manager http://bbc.in/1Ffj6ci
585733481608646657|Wed Apr 08 09:18:39 +0000 2015|VIDEO: 'All day is empty, what am I going to do?' http://bbc.in/1N7wSSz
585701601131765761|Wed Apr 08 07:11:58 +0000 2015|VIDEO: 'Overhaul needed' for end-of-life care http://bbc.in/1CmrRu3
585620828110397440|Wed Apr 08 01:51:00 +0000 2015|Care for dying 'needs overhaul' http://bbc.in/1FdSGrl
585437294120677376|Tue Apr 07 13:41:42 +0000 2015|VIDEO: NHS: Labour and Tory key policies http://bbc.in/1Ci5eqD
585437293399252992|Tue Apr 07 13:41:42 +0000 2015|Have GP services got worse? http://bbc.in/1Ci5c22
585376127931129857|Tue Apr 07 09:38:39 +0000 2015|A&amp;E waiting hits new worst level http://bbc.in/1Fa4Xgz
585283574888206336|Tue Apr 07 03:30:52 +0000 2015|Parties row over GP opening hours http://bbc.in/1CfcVOb
585231549819330560|Tue Apr 07 00:04:09 +0000 2015|Why strenuous runs may not be so bad after all http://bbc.in/1Ceq0Y7
584985578074050560|Mon Apr 06 07:46:44 +0000 2015|VIDEO: Health surcharge for non-EU patients http://bbc.in/1C5Mlbk
584985576962592768|Mon Apr 06 07:46:44 +0000 2015|VIDEO: Skin cancer spike 'from 60s holidays' http://bbc.in/1C9Gy3o
584901557218480129|Mon Apr 06 02:12:52 +0000 2015|80,000 'might die' in future outbreak http://bbc.in/1F3nif2
584893680747585537|Mon Apr 06 01:41:34 +0000 2015|Skin cancer 'linked to holiday boom' http://bbc.in/1Pb4Xjb
584140478720811008|Fri Apr 03 23:48:37 +0000 2015|Public 'back tax rises to fund NHS' http://bbc.in/1ETauYE
584140477718339584|Fri Apr 03 23:48:37 +0000 2015|VIDEO: Welcome to the designer asylum http://bbc.in/1CD3Dip
583959459589824513|Fri Apr 03 11:49:19 +0000 2015|VIDEO: Why are we having less sex? http://bbc.in/19RKqiY
583837541373059073|Fri Apr 03 03:44:51 +0000 2015|Five ideas to transform the NHS http://bbc.in/1BTQU7k
583784740064337920|Fri Apr 03 00:15:02 +0000 2015|Personal cancer vaccines 'exciting' http://bbc.in/1ENwsMD
583770728090112000|Thu Apr 02 23:19:22 +0000 2015|Child heart surgery deaths 'halved' http://bbc.in/1BT4s2U
583745704021008384|Thu Apr 02 21:39:55 +0000 2015|VIDEO: Miliband: Cameron 'failed' the NHS http://bbc.in/1BSKo0T
583659491310219264|Thu Apr 02 15:57:21 +0000 2015|Unsafe food 'growing global threat' http://bbc.in/1BREQDJ
583616089340280832|Thu Apr 02 13:04:53 +0000 2015|Health highlights http://bbc.in/1EKlFCK
583550348352212992|Thu Apr 02 08:43:39 +0000 2015|Ambulance progress 'not fast enough' http://bbc.in/1P1AJyX
583406140337164288|Wed Apr 01 23:10:37 +0000 2015|Children’s hospital builds sleep app http://bbc.in/1BO9jlZ
583406139561250816|Wed Apr 01 23:10:37 +0000 2015|Drug giant 'blocks' eye treatment http://bbc.in/1P0rpv2
583398284166733824|Wed Apr 01 22:39:24 +0000 2015|Blood test for Down's syndrome hailed http://bbc.in/1BO3eWQ
583398283441119232|Wed Apr 01 22:39:24 +0000 2015|VIDEO: 'We must plan for future of NHS' http://bbc.in/1BRNoeJ
583054276848582656|Tue Mar 31 23:52:26 +0000 2015|Paracetamol ‘no good for back pain' http://bbc.in/1MvP0oO
583054275237978113|Tue Mar 31 23:52:26 +0000 2015|Care system 'gets biggest shake-up' http://bbc.in/19Ad4Vy
583054274554245120|Tue Mar 31 23:52:26 +0000 2015|VIDEO: Could existing drugs offer MS hope? http://bbc.in/1OWzSPT
582968831796207616|Tue Mar 31 18:12:55 +0000 2015|MS drug 'may already be out there' http://bbc.in/1BIUAZx
582824135799140352|Tue Mar 31 08:37:56 +0000 2015|VIDEO: Nursing code of conduct launched http://bbc.in/1BLrny7
582807922549256193|Tue Mar 31 07:33:31 +0000 2015|VIDEO: Disability village 'under threat' http://bbc.in/1BLlj8L
582729504533049344|Tue Mar 31 02:21:55 +0000 2015|Uganda circumcision truck fights HIV http://bbc.in/1BKT2iJ
582687699485622272|Mon Mar 30 23:35:48 +0000 2015|More veterans seek mental health aid http://bbc.in/1DhbCnf
582687698772623361|Mon Mar 30 23:35:47 +0000 2015|E-cigarette use 'high among teens' http://bbc.in/1DhbCn6
582554028736843777|Mon Mar 30 14:44:38 +0000 2015|Medieval eye remedy 'kills MRSA' http://bbc.in/1ysI93N
582323221908709378|Sun Mar 29 23:27:29 +0000 2015|Parents rarely spot child obesity http://bbc.in/1G0J4Ag
582167316806287360|Sun Mar 29 13:07:59 +0000 2015|VIDEO: Vaccine to protect from meningitis B http://bbc.in/1MjhH8r
581988293006344194|Sun Mar 29 01:16:36 +0000 2015|Chikungunya revives herbal remedies in Antigua http://bbc.in/1BGLzBf
581988292343635968|Sun Mar 29 01:16:36 +0000 2015|The Bolivian women who knit parts for hearts http://bbc.in/1BGLzBc
581988291639021568|Sun Mar 29 01:16:36 +0000 2015|Meningitis B vaccine deal agreed http://bbc.in/1BGLzB7
581977670692388864|Sun Mar 29 00:34:23 +0000 2015|VIDEO: Life-saving knitters http://bbc.in/1EimW3I
```

Ln 1, Col 1 | 20% | Windows (CRLF) | UTF-8



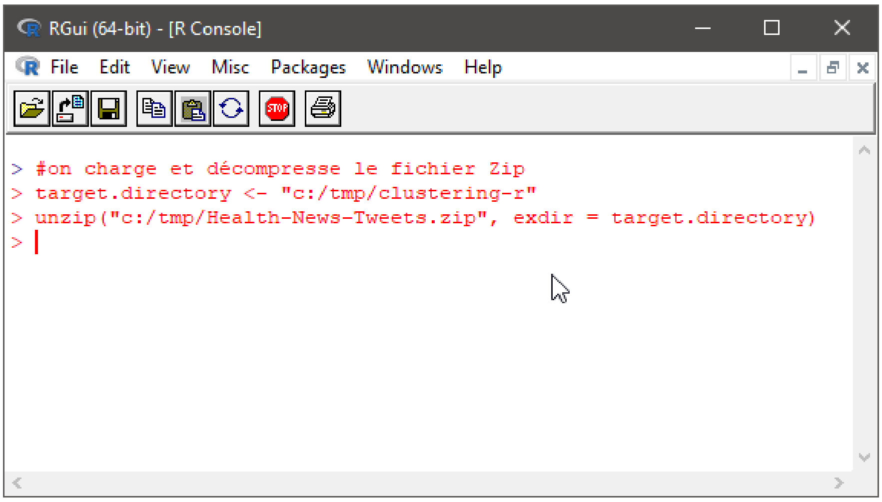
```
nytimeshealth.txt - Notepad
File Edit Format View Help
548662191340421120|Sat Dec 27 02:10:34 +0000 2014|Risks in Using Social Media to Spot Signs of Mental Distress http://nyti.ms/1rqi9I1
548579831169163265|Fri Dec 26 20:43:18 +0000 2014|RT @paula_span: The most effective nationwide diabetes prevention program you've probably never heard of:
http://newoldage.blogs.nytimes.com/2014/12/26/diabetes-prevention-that-works/
548579045269852161|Fri Dec 26 20:40:11 +0000 2014|The New Old Age Blog: Diabetes Prevention That Works http://nyti.ms/1xm7fTi
548444679529041920|Fri Dec 26 11:46:15 +0000 2014|Well: Comfort Casseroles for Winter Dinners http://nyti.ms/1xTNoO0
548311901227474944|Fri Dec 26 02:58:39 +0000 2014|High-Level Knowledge Before Veterans Affairs Scandal http://nyti.ms/13yCpvS
548305625449787392|Fri Dec 26 02:33:42 +0000 2014|Your Money: Affordable Care Act’s Tax Effects Now Loom for Filers http://nyti.ms/13yAtUf
548283182853160960|Fri Dec 26 01:04:32 +0000 2014|Well: Christmas in the Hospital http://nyti.ms/1vtPNcm
548278414504108033|Fri Dec 26 00:45:35 +0000 2014|The Texas Tribune: For Members of Health Ministries in Texas, Caring Means Sharing the Bills http://nyti.ms/1wLtAen
548176399299649536|Thu Dec 25 18:00:12 +0000 2014|A q&amp;a with Laura Hillenbrand, author of "Unbroken," who suffers from chronic fatigue syndrome.
http://bit.ly/1ziCflj
547949409288609792|Thu Dec 25 02:58:14 +0000 2014|Race to Deliver Nicotine’s Punch, With Less Risk  http://nyti.ms/1xNzyg4
547904193961656321|Wed Dec 24 23:58:34 +0000 2014|The New Health Care: People Are Shopping for Health Insurance, Surprisingly http://nyti.ms/1vpIjqE
547904193001185280|Wed Dec 24 23:58:33 +0000 2014|Machine Learning: Bedtime Technology for a Better Night’s Sleep http://nyti.ms/16RCgpE
547872136304394242|Wed Dec 24 21:51:10 +0000 2014|Breaking: Lab Error May Have Exposed U.S. Technician to Ebola Virus http://nyti.ms/1ziIYf6
547870756592300034|Wed Dec 24 21:45:41 +0000 2014|C.D.C. Ebola Error in Lab May Have Exposed Technician to Virus http://nyti.ms/1CDkLqc
547870381160140800|Wed Dec 24 21:44:12 +0000 2014|RT @nytimes: Breaking News: Lab Error May Have Exposed U.S. Technician to Ebola Virus http://nyti.ms/1B2ThGD
547802935141232642|Wed Dec 24 17:16:12 +0000 2014|A q&amp;a with Laura Hillenbrand, author of "Unbroken," who suffers from chronic fatigue syndrome.
http://nyti.ms/1B5PO9Q
547765704829964289|Wed Dec 24 14:48:15 +0000 2014|RT @cslnyt: Virus That Killed a Kansas Farmer is identified @CDCgov  @nytDeniseGrady
http://www.nytimes.com/2014/12/24/science/mysterious-virus-that-killed-a-farmer-in-kansas-is-identified.html?smid=nytcore-iphone-share&smprod=nytcore-iphone
547624914531127296|Wed Dec 24 05:28:48 +0000 2014|Well: Exercise to Lose Weight? Stay Warm http://nyti.ms/1Hz7Pj2
547599936452583427|Wed Dec 24 03:49:33 +0000 2014|Business Briefing: Diabetes Drug Gains Approval for Treatment of Obesity http://nyti.ms/1ta9nPw
547599935517241344|Wed Dec 24 03:49:33 +0000 2014|So Far, 6.4 Million Obtain Health Care Coverage for 2015 in Federal Marketplace http://nyti.ms/1vaJ1ad
547595227197087745|Wed Dec 24 03:30:50 +0000 2014|DealBook: Inquiry Into Foreign Bribes at Biomet Hangs Over $13 Billion Merger http://nyti.ms/1vaHQYn
547595226215632896|Wed Dec 24 03:30:50 +0000 2014|Firing of V.A. Clinic Chief Is Upheld Over Gifts, Not Wait Lists http://nyti.ms/1ta9nPu
547576403836035072|Wed Dec 24 02:16:02 +0000 2014|World Briefing: Ireland: Doctors Seek to Remove Pregnant Woman From Life Support http://nyti.ms/13uMfPo
547562964153536512|Wed Dec 24 01:22:38 +0000 2014|Plastic Surgery Tourism Brings Chinese to South Korea  http://nyti.ms/1wDB6Yq
547561821373075456|Wed Dec 24 01:18:06 +0000 2014|Looking for a better night's sleep? Put down that e-book and read a paperback instead. http://nyti.ms/1whtwhd
547559173974921218|Wed Dec 24 01:07:34 +0000 2014|Mysterious Virus That Killed a Farmer in Kansas Is Identified http://nyti.ms/1AZ0qaL
547515883518980096|Tue Dec 23 22:15:33 +0000 2014|Well: Pain Relievers Tied to Reduced Skin Cancer Risk http://nyti.ms/13ukGpr
547501217086062592|Tue Dec 23 21:17:16 +0000 2014|The best gift? The humble gift card. http://nyti.ms/1A1pDmz
547500871689711616|Tue Dec 23 21:15:54 +0000 2014|What do most people say they want for a gift? http://pbs.twimg.com/media/B5kdIQXCQAAtcgA.jpg
547460287289634822|Tue Dec 23 18:34:38 +0000 2014|F.D.A. Lifting Ban on Gay Blood Donors http://nyti.ms/16NAVjH
547459549360574466|Tue Dec 23 18:31:42 +0000 2014|F.D.A. to Lift Lifetime Ban on Blood Donation by Gay Men http://nyti.ms/1A12TTG
547440575545614336|Tue Dec 23 17:16:18 +0000 2014|Well: Think You Found the Perfect Gift? Think Again http://nyti.ms/13B6Amt
547410904414167040|Tue Dec 23 15:18:24 +0000 2014|RT @nytDeniseGrady: C.D.C. Head Says Fight on Ebola Will Be Long http://nyti.ms/1vhYN4g
547232205605707776|Tue Dec 23 03:28:19 +0000 2014|Gunman May Have Used Garner and Brown Cases to Justify Police Killings, Experts Say http://nyti.ms/1ATCm9m
547227742849470464|Tue Dec 23 03:10:35 +0000 2014|U.S. Court Strikes Down North Carolina Ultrasound Abortion Law http://nyti.ms/1xcoPMu
547227741960278016|Tue Dec 23 03:10:35 +0000 2014|Group Sues Aetna, Claiming Discrimination Against H.I.V. Patients http://nyti.ms/1ATxhOj
547223894130978817|Tue Dec 23 02:55:17 +0000 2014|Obama Administration to Investigate Insurers for Bias Against Costly Conditions http://nyti.ms/1xcnYeX
547184000411127808|Tue Dec 23 00:16:46 +0000 2014|C.D.C. Head Says Fight on Ebola Will Be Long http://nyti.ms/1CxIFR9
547179177829756929|Mon Dec 22 23:57:36 +0000 2014|Op-Ed Contributors: Sugar Season. It’s Everywhere, and Addictive http://nyti.ms/16KEwPg
547144384672976896|Mon Dec 22 21:39:21 +0000 2014|Letters: Flavors From the Past  http://nyti.ms/1HruX2S
547140214125506560|Mon Dec 22 21:22:47 +0000 2014|Global Health: For Heroin Users, Less May Be Worse for H.I.V.  http://nyti.ms/1AAlJB2
547130056959860736|Mon Dec 22 20:42:25 +0000 2014|Q&amp;A:  The Danger of Going Off Course http://nyti.ms/13WclvS
547129518683475968|Mon Dec 22 20:40:17 +0000 2014|Video: Is farmed fish better than no fish at all?  #askwell http://nyti.ms/1weXRwR
Ln 30, Col 78    20%    Windows (CRLF)    UTF-8
```

Ensuite, nous initialisons la structure du dataframe dans la mémoire qui accueillera les données:

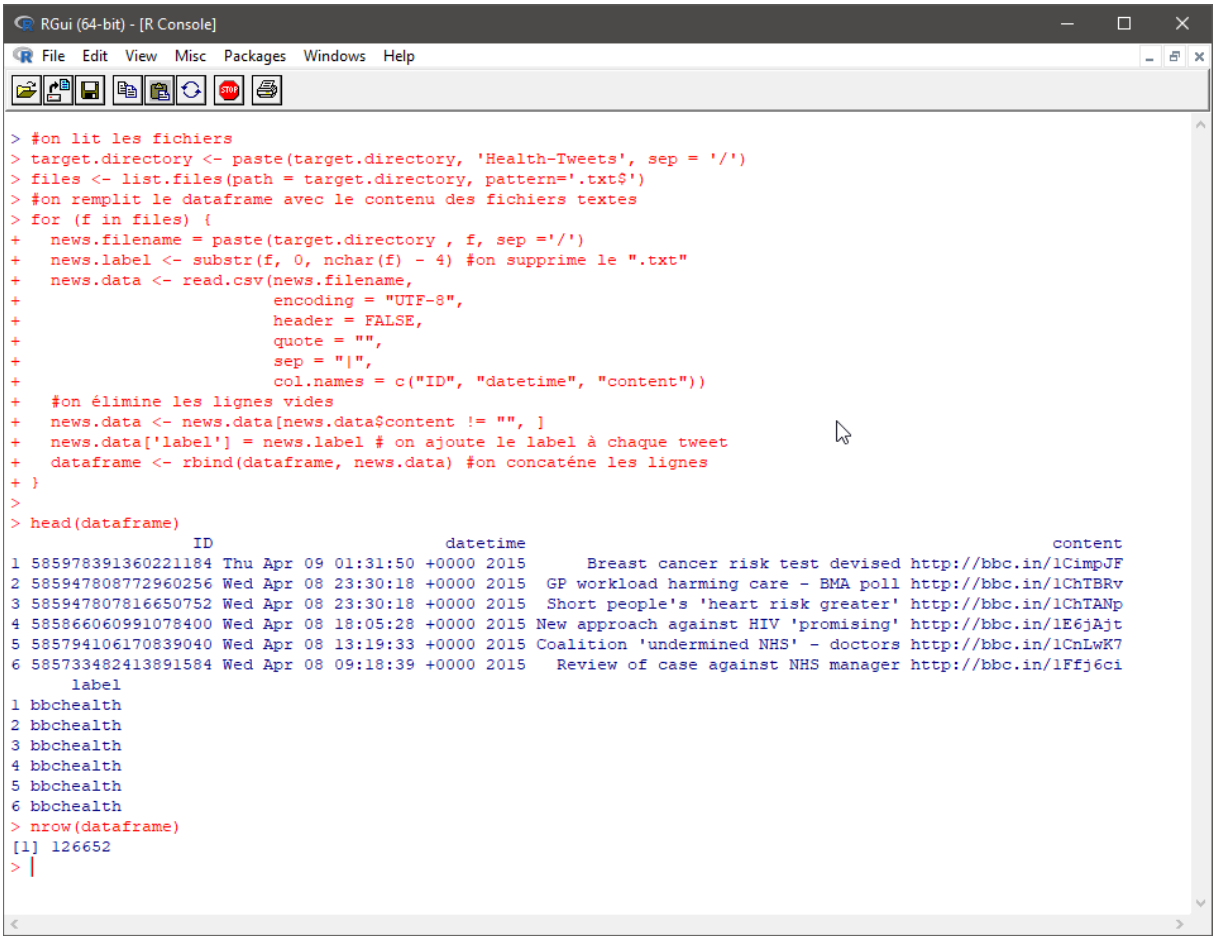


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> #on crée d'abord le modèle de data frame
> dataframe <- data.frame(ID=character(),
+                         datetime=character(),
+                         content=character(),
+                         label=factor())
> dataframe
[1] ID       datetime content  label
<0 rows> (or 0-length row.names)
>
```

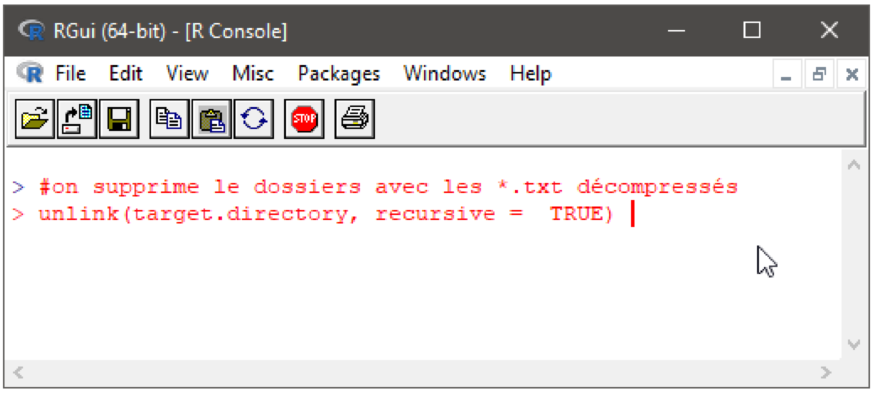
Nous dézippons le fichier:



Nous bouclons sur chacun des fichiers:



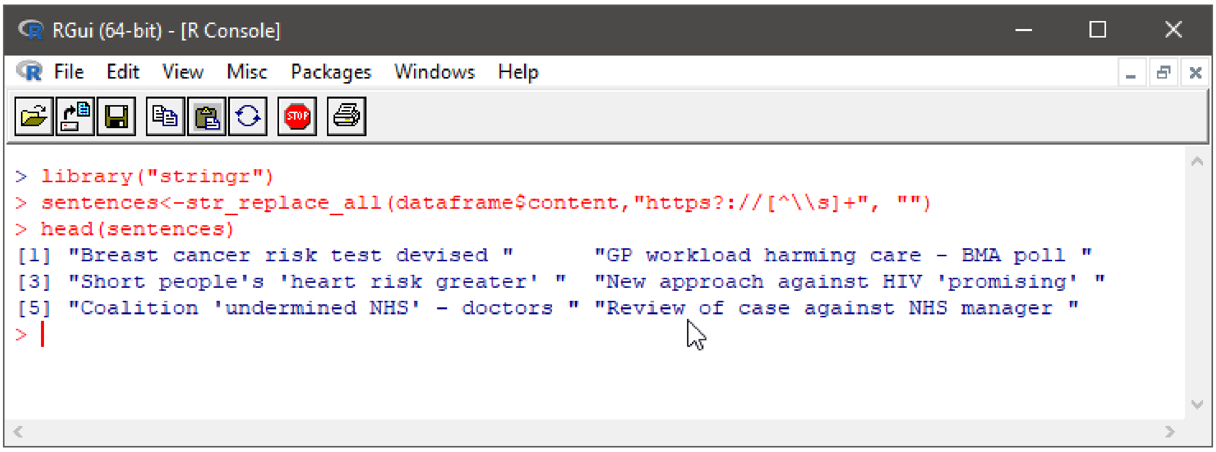
Et donc une fois tous les textes mis dans la mémoire via le dataframe nous pouvons supprimer les fichiers extraits:



Vient maintenant le temps de nettoyer nos phrases. Après un rapide temps d'observation, nous pouvons remarquer que nos tweets sont tous terminés par une URL raccourcie renvoyant vers l'article d'actualité.

Ce type d'informations est en fait une poubelle d'un point de vue NLP, malgré le fait qu'il pourrait rendre les classes facilement distinguables si un raccourcisseur spécifique est utilisé par l'un des comptes Twitter.

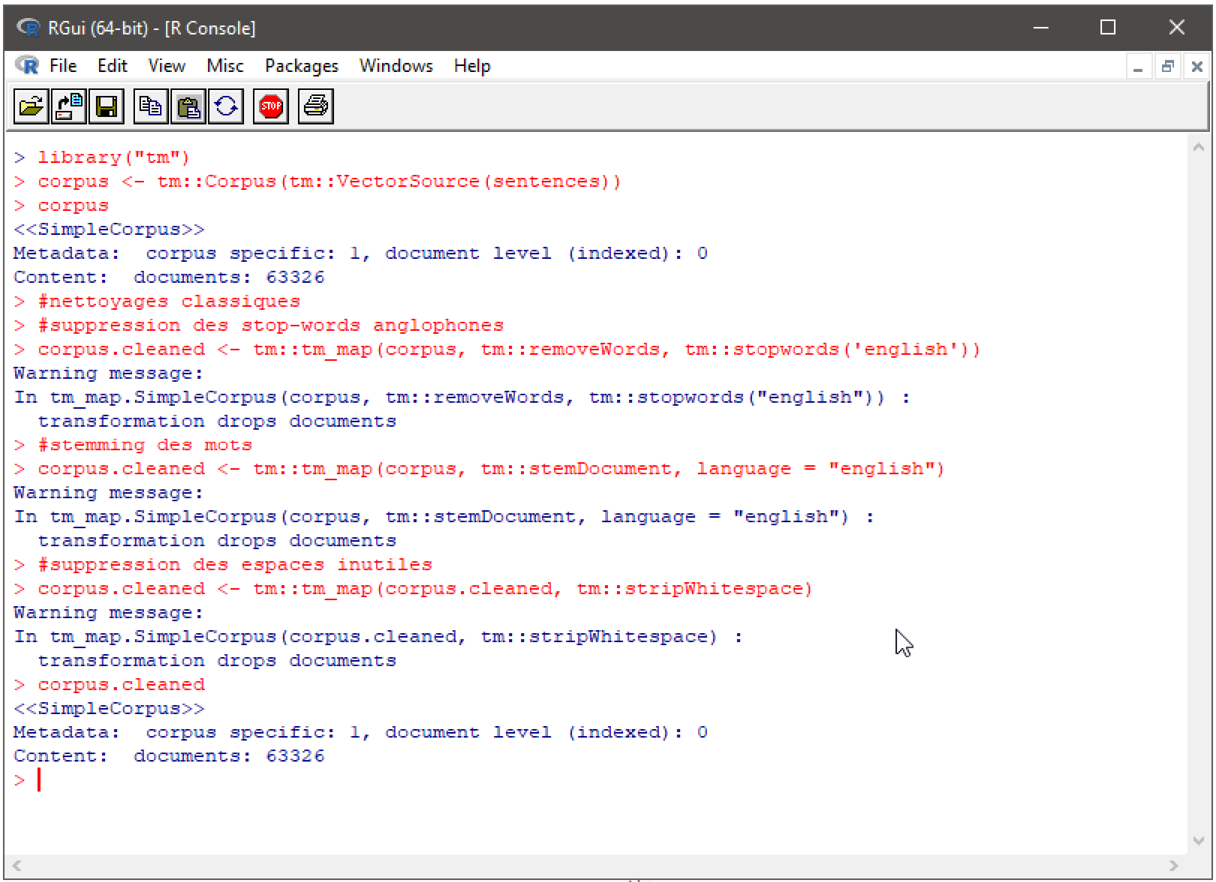
Pour se débarrasser de ces URL, utilisons des substitutions de regex :



Pour la partie restante des tâches de prétraitement régulières, nous utiliserons encore un package dédié. La manipulation de texte est coûteuse en termes de codage ou d'exécution ou les deux. Lorsque les données sont autres que des entités numériques, R peut devenir pénible pour les débutants.

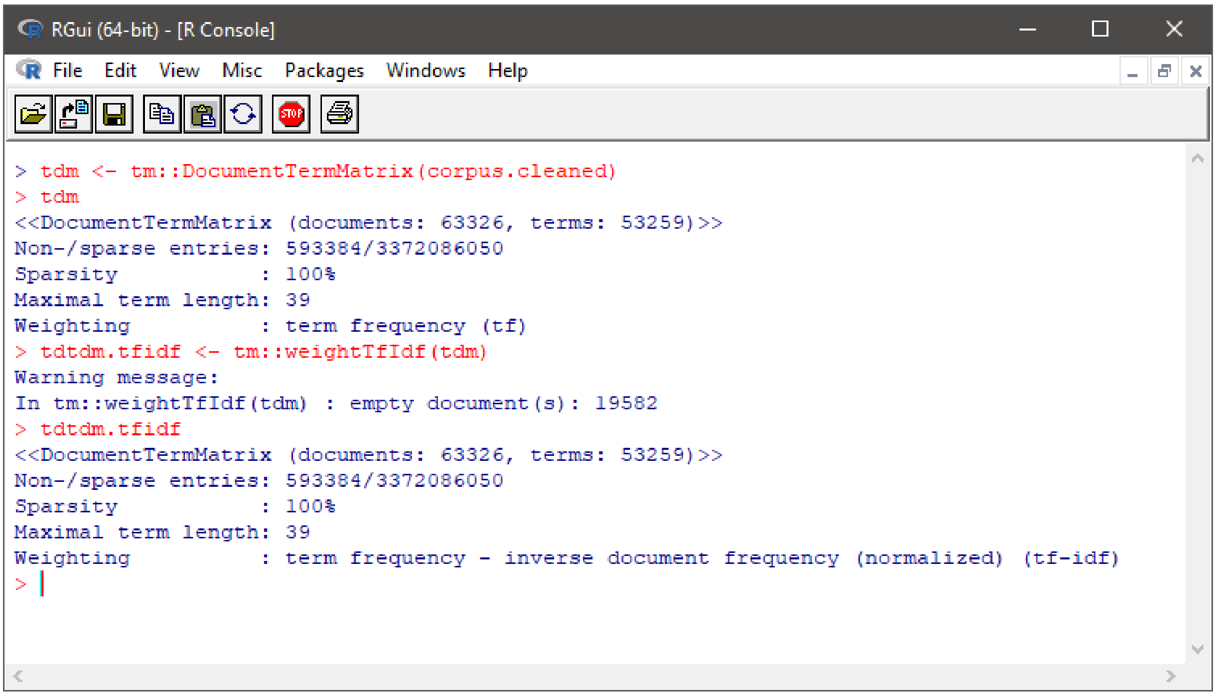
Préparation du corpus

Le package qui nous sauvera la vie est **tm** (pour text mining). À partir de nos phrases résultantes, nous allons créer un objet *Corpus*, nous permettant d'appeler des méthodes sur celui-ci pour effectuer le nettoyage des mots vides, le stemming, le découpage des espaces blancs, …

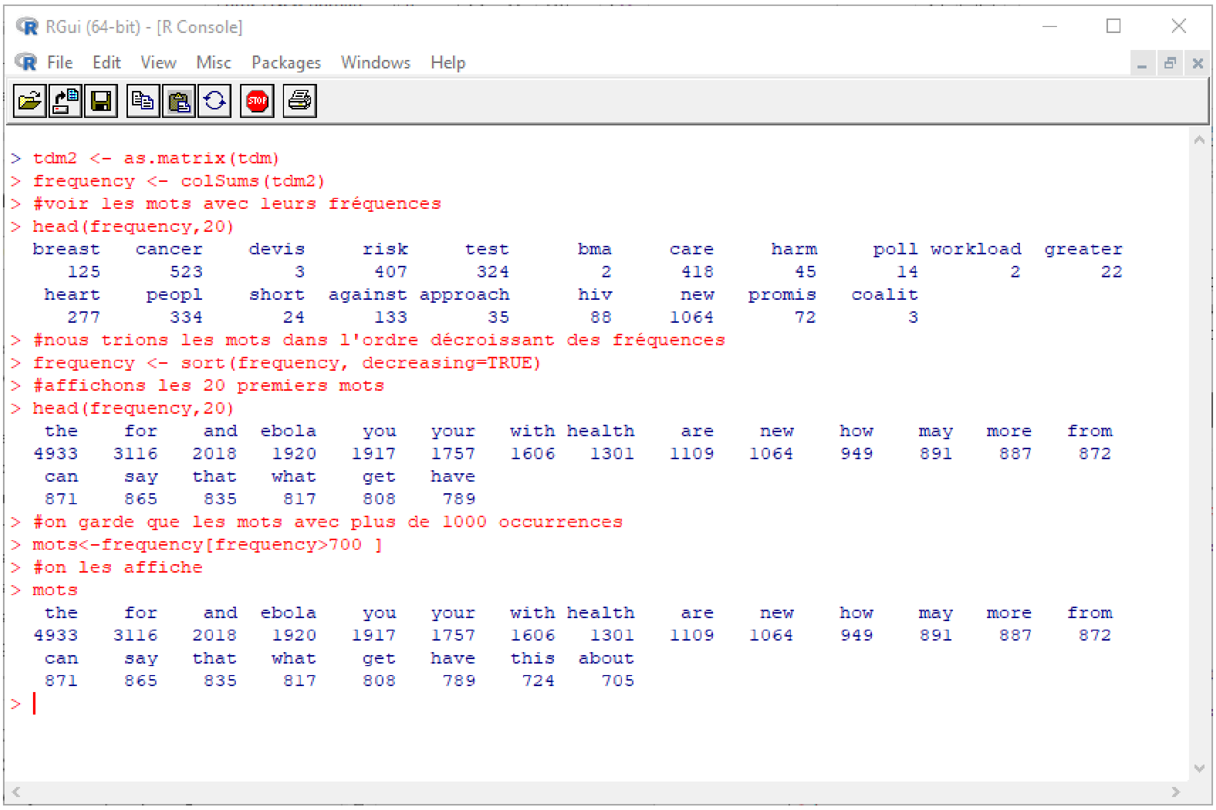


Maintenant que nous avons tout un ensemble de phrases nettoyées, nous devons les représenter numériquement. Le moyen le plus efficace parmi les plus faciles est le TF-IDF (de l'anglais term frequency-inverse document frequency) pondéré[46]. Ainsi, chaque phrase est un vecteur dont la longueur est la taille du vocabulaire restant ; et ses composantes sont valorisées en fonction de la fréquence du mot dans la phrase et de sa signification dans le corpus:

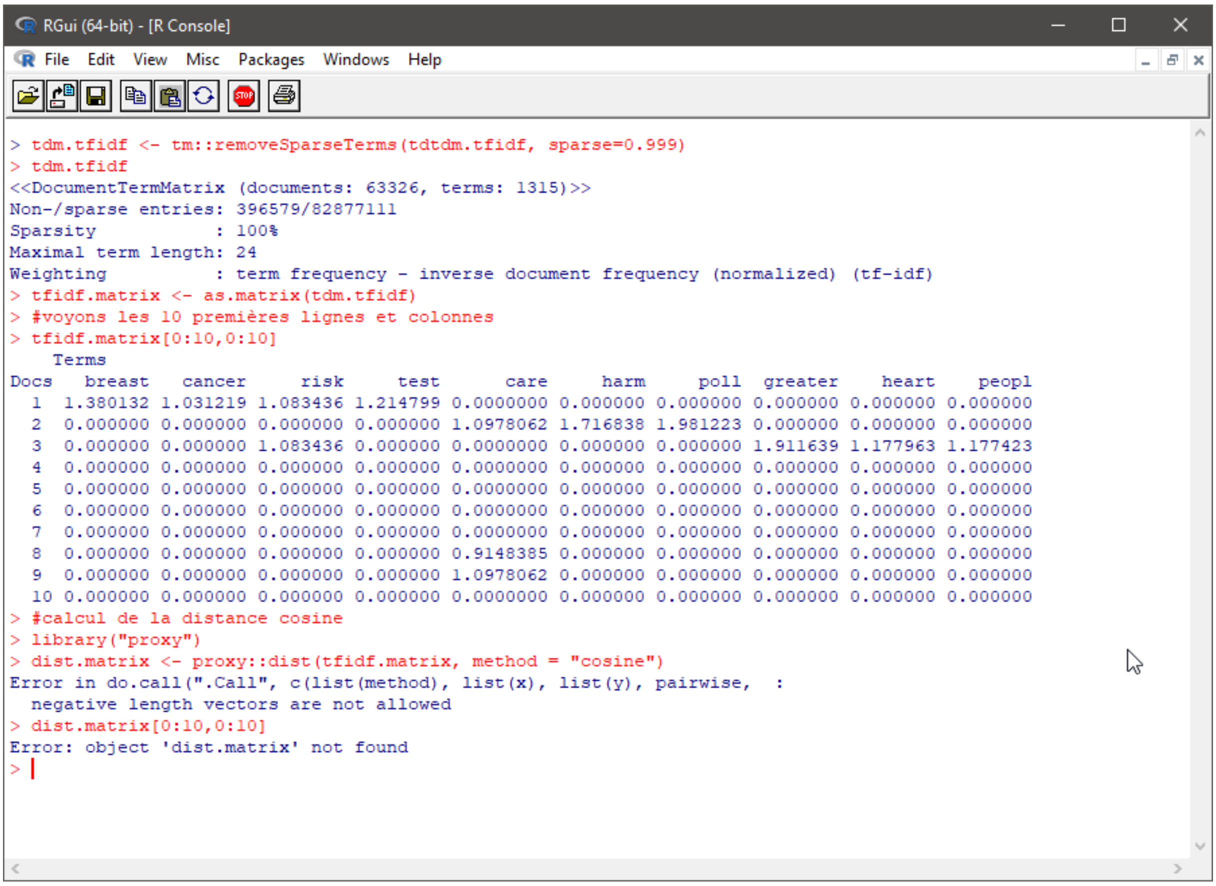
[46] Rappelons que c'est une mesure totalement empirique (parmi d'autres) et très discutable.



Nous pouvons explorer un peu le corpus pour le fun:

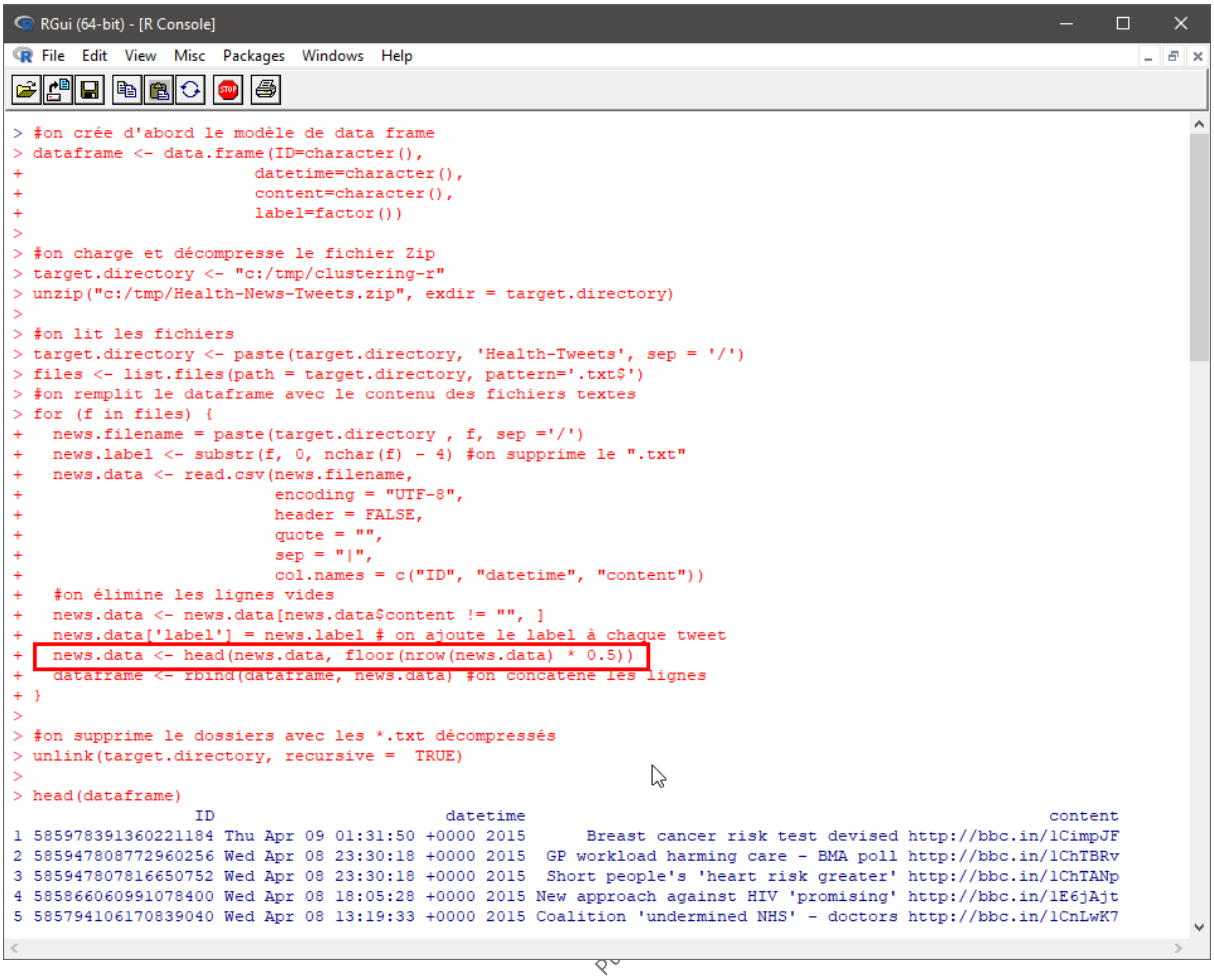


Maintenant que nos phrases sont représentées numériquement, nous sommes capables de calculer une distance entre elles. Dans notre situation, nous devons appliquer une coupe supplémentaire dans les fonctionnalités à des fins de calcul : la parcimonie n'étant pas bien gérée globalement dans R, nous devons supprimer les colonnes qui sont clairsemées parmi les documents. De cette façon, nous réduisons le nombre de fonctionnalités, puis la taille de la matrice de données.

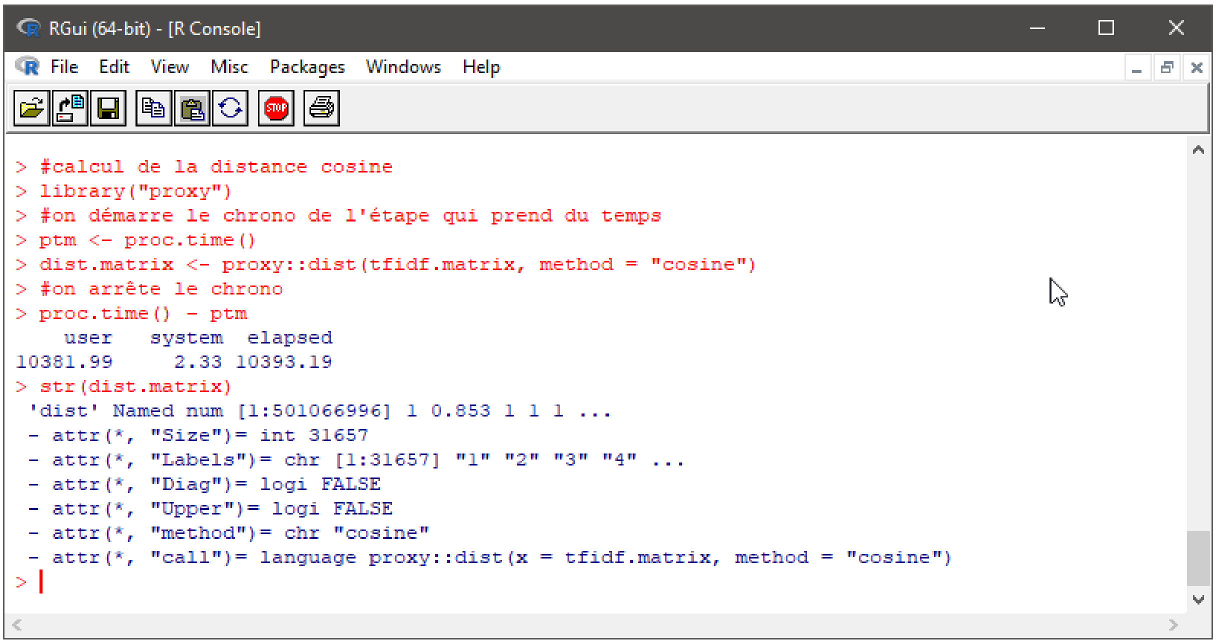


```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> tdm.tfidf <- tm::removeSparseTerms(tdtdm.tfidf, sparse=0.999)
> tdm.tfidf
<<DocumentTermMatrix (documents: 63326, terms: 1315)>>
Non-/sparse entries: 396579/82877111
Sparsity           : 100%
Maximal term length: 24
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
> tfidf.matrix <- as.matrix(tdm.tfidf)
> #voyons les 10 premières lignes et colonnes
> tfidf.matrix[0:10,0:10]
    Terms
Docs   breast   cancer     risk     test      care     harm     poll  greater    heart    peopl
  1  1.380132 1.031219 1.083436 1.214799 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
  2  0.000000 0.000000 0.000000 0.000000 1.0978062 1.716838 1.981223 0.000000 0.000000 0.000000
  3  0.000000 0.000000 1.083436 0.000000 0.0000000 0.000000 0.000000 1.911639 1.177963 1.177423
  4  0.000000 0.000000 0.000000 0.000000 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
  5  0.000000 0.000000 0.000000 0.000000 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
  6  0.000000 0.000000 0.000000 0.000000 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
  7  0.000000 0.000000 0.000000 0.000000 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
  8  0.000000 0.000000 0.000000 0.000000 0.9148385 0.000000 0.000000 0.000000 0.000000 0.000000
  9  0.000000 0.000000 0.000000 0.000000 1.0978062 0.000000 0.000000 0.000000 0.000000 0.000000
  10 0.000000 0.000000 0.000000 0.000000 0.0000000 0.000000 0.000000 0.000000 0.000000 0.000000
> #calcul de la distance cosine
> library("proxy")
> dist.matrix <- proxy::dist(tfidf.matrix, method = "cosine")
Error in do.call(".Call", c(list(method), list(x), list(y), pairwise,  :
  negative length vectors are not allowed
> dist.matrix[0:10,0:10]
Error: object 'dist.matrix' not found
>
```

Le message d'erreur ci-dessus signale que nous n'aurions **pas assez de mémoire** sur l'ordinateur sur lequel le script est exécuté (et pourtant nous avons 64GB de RAM… mais pour 40% du jeu de données il faut 21GB de RAM pour info…). Donc reprenons notre script depuis le tout début et limitons un peu le nombre de tweets en rajoutant la ligne ci-dessous mise en évidence dans la boucle du début:



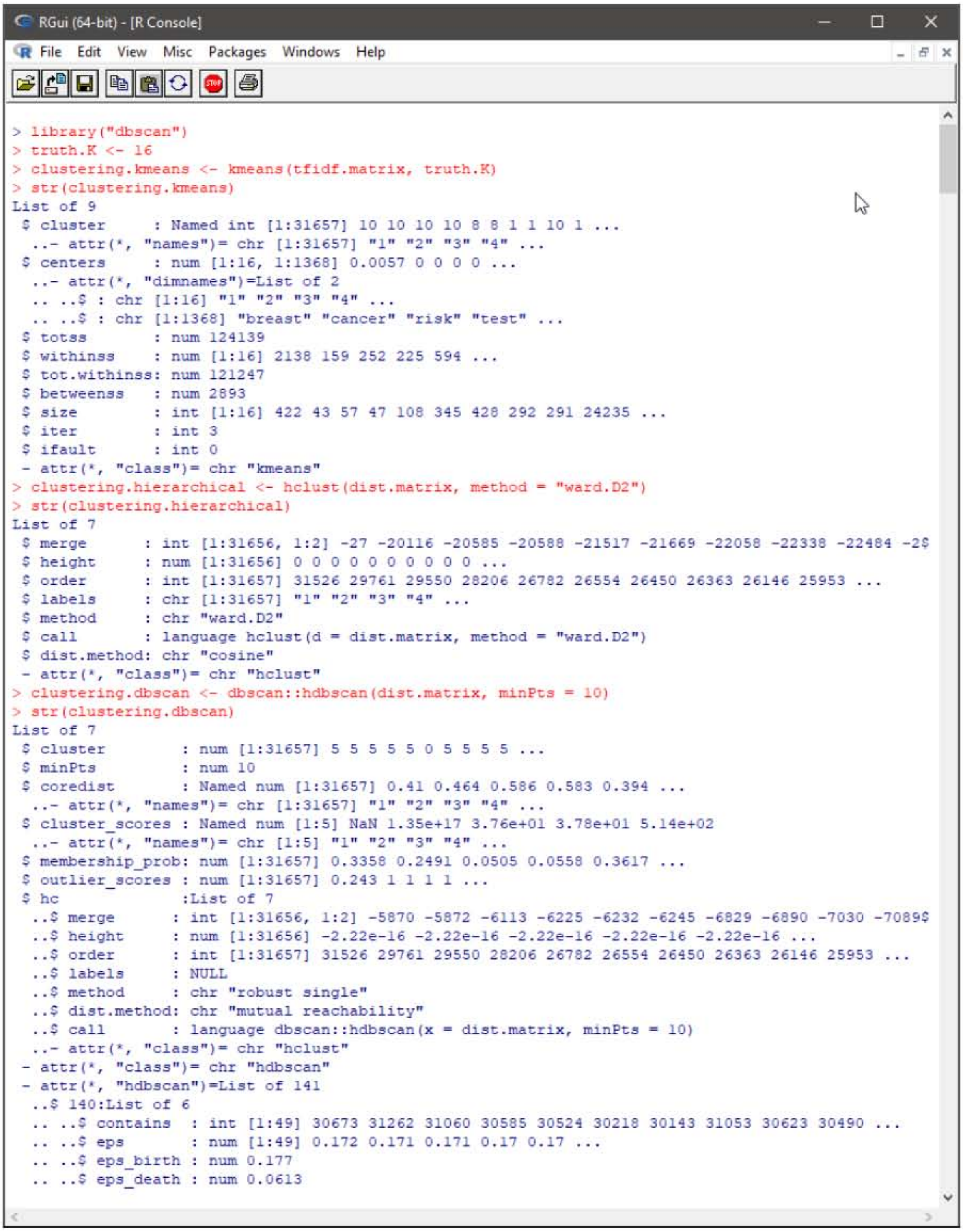
Après 2 heures et 45 minutes sur un Alienware Area 51M avec 64GB de RAM et un processeur 8 cœurs à 4.6 GHz (Windows 10, 64bits) nous avons:



Ok nous y sommes! La partie survivante du jeu de données est interprétable par nos algorithmes de clustering ! Mais comment allons-nous regrouper tout cela ?

Clustering K-Means, Hiérarchique et HDBScan

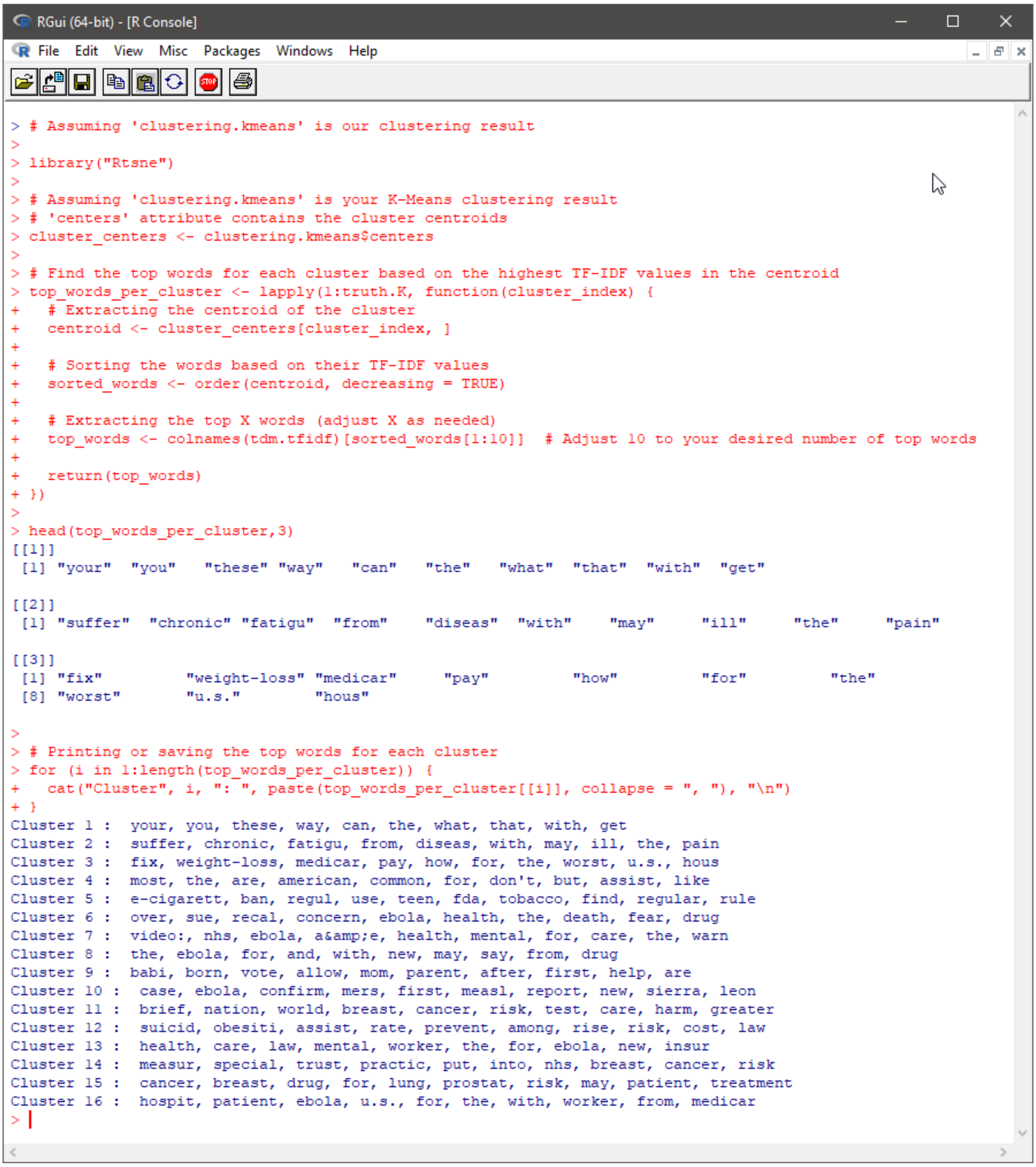
Nous allons exécuter la plupart des différentes techniques dans les précédents exemple de ce livre : une de partition (K-Means), une hiérarchique (Bottom-up avec fusion) et une basée sur la densité (HDBScan). À l'exception du dernier, les fonctions natives de R peuvent être utilisés pour exécuter les algorithmes respectifs:



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("dbscan")
> truth.K <- 16
> clustering.kmeans <- kmeans(tfidf.matrix, truth.K)
> str(clustering.kmeans)
List of 9
 $ cluster     : Named int [1:31657] 10 10 10 10 8 8 1 1 10 1 ...
  ..- attr(*, "names")= chr [1:31657] "1" "2" "3" "4" ...
 $ centers     : num [1:16, 1:1368] 0.0057 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "1" "2" "3" "4" ...
  .. ..$ : chr [1:1368] "breast" "cancer" "risk" "test" ...
 $ totss       : num 124139
 $ withinss    : num [1:16] 2138 159 252 225 594 ...
 $ tot.withinss: num 121247
 $ betweenss   : num 2893
 $ size        : int [1:16] 422 43 57 47 108 345 428 292 291 24235 ...
 $ iter        : int 3
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
> clustering.hierarchical <- hclust(dist.matrix, method = "ward.D2")
> str(clustering.hierarchical)
List of 7
 $ merge      : int [1:31656, 1:2] -27 -20116 -20585 -20588 -21517 -21669 -22058 -22338 -22484 -2$
 $ height     : num [1:31656] 0 0 0 0 0 0 0 0 0 0 ...
 $ order      : int [1:31657] 31526 29761 29550 28206 26782 26554 26450 26363 26146 25953 ...
 $ labels     : chr [1:31657] "1" "2" "3" "4" ...
 $ method     : chr "ward.D2"
 $ call       : language hclust(d = dist.matrix, method = "ward.D2")
 $ dist.method: chr "cosine"
 - attr(*, "class")= chr "hclust"
> clustering.dbscan <- dbscan::hdbscan(dist.matrix, minPts = 10)
> str(clustering.dbscan)
List of 7
 $ cluster         : num [1:31657] 5 5 5 5 5 0 5 5 5 5 ...
 $ minPts          : num 10
 $ coredist        : Named num [1:31657] 0.41 0.464 0.586 0.583 0.394 ...
  ..- attr(*, "names")= chr [1:31657] "1" "2" "3" "4" ...
 $ cluster_scores : Named num [1:5] NaN 1.35e+17 3.76e+01 3.78e+01 5.14e+02
  ..- attr(*, "names")= chr [1:5] "1" "2" "3" "4" ...
 $ membership_prob: num [1:31657] 0.3358 0.2491 0.0505 0.0558 0.3617 ...
 $ outlier_scores : num [1:31657] 0.243 1 1 1 1 ...
 $ hc              :List of 7
  ..$ merge      : int [1:31656, 1:2] -5870 -5872 -6113 -6225 -6232 -6245 -6829 -6890 -7030 -7089$
  ..$ height     : num [1:31656] -2.22e-16 -2.22e-16 -2.22e-16 -2.22e-16 -2.22e-16 ...
  ..$ order      : int [1:31657] 31526 29761 29550 28206 26782 26554 26450 26363 26146 25953 ...
  ..$ labels     : NULL
  ..$ method     : chr "robust single"
  ..$ dist.method: chr "mutual reachability"
  ..$ call       : language dbscan::hdbscan(x = dist.matrix, minPts = 10)
  ..- attr(*, "class")= chr "hclust"
 - attr(*, "class")= chr "hdbscan"
 - attr(*, "hdbscan")=List of 141
  ..$ 140:List of 6
  .. ..$ contains  : int [1:49] 30673 31262 31060 30585 30524 30218 30143 31053 30623 30490 ...
  .. ..$ eps       : num [1:49] 0.172 0.171 0.171 0.17 0.17 ...
  .. ..$ eps_birth : num 0.177
  .. ..$ eps_death : num 0.0613
```

Analyse des mots appartenant aux clusters

Avec 30% du jeu total de données nous avons pour voir quels mots appartiennent à quels clusters de la méthodes des K-means:



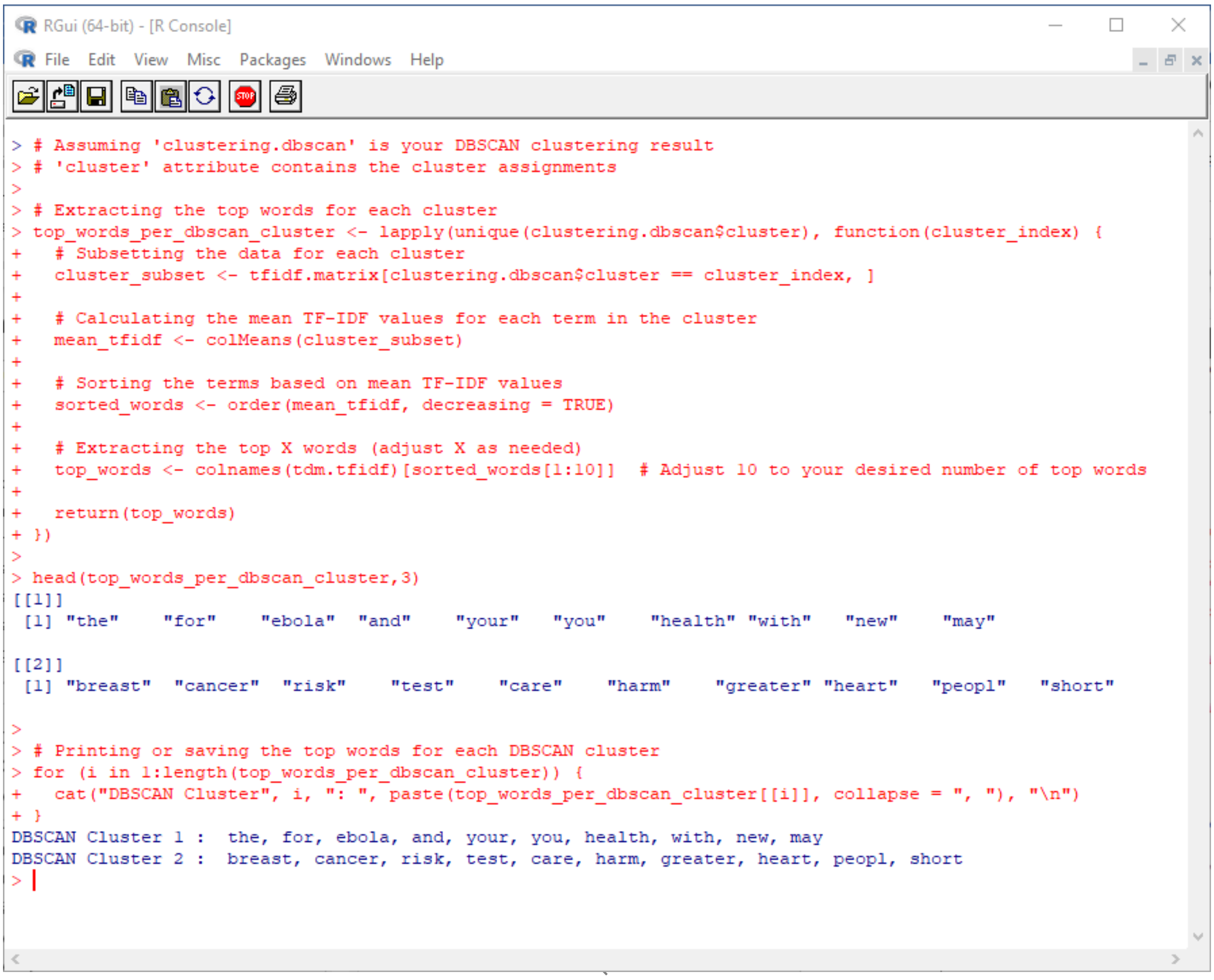
```
> # Assuming 'clustering.kmeans' is our clustering result
>
> library("Rtsne")
>
> # Assuming 'clustering.kmeans' is your K-Means clustering result
> # 'centers' attribute contains the cluster centroids
> cluster_centers <- clustering.kmeans$centers
>
> # Find the top words for each cluster based on the highest TF-IDF values in the centroid
> top_words_per_cluster <- lapply(1:truth.K, function(cluster_index) {
+   # Extracting the centroid of the cluster
+   centroid <- cluster_centers[cluster_index, ]
+
+   # Sorting the words based on their TF-IDF values
+   sorted_words <- order(centroid, decreasing = TRUE)
+
+   # Extracting the top X words (adjust X as needed)
+   top_words <- colnames(tdm.tfidf)[sorted_words[1:10]]  # Adjust 10 to your desired number of top words
+
+   return(top_words)
+ })
>
> head(top_words_per_cluster,3)
[[1]]
 [1] "your"   "you"    "these" "way"    "can"    "the"    "what"   "that"   "with"   "get"

[[2]]
 [1] "suffer"  "chronic" "fatigu"  "from"    "diseas"  "with"    "may"     "ill"     "the"     "pain"

[[3]]
 [1] "fix"         "weight-loss" "medicar"     "pay"         "how"         "for"         "the"
 [8] "worst"       "u.s."        "hous"

>
> # Printing or saving the top words for each cluster
> for (i in 1:length(top_words_per_cluster)) {
+   cat("Cluster", i, ": ", paste(top_words_per_cluster[[i]], collapse = ", "), "\n")
+ }
Cluster 1 :  your, you, these, way, can, the, what, that, with, get
Cluster 2 :  suffer, chronic, fatigu, from, diseas, with, may, ill, the, pain
Cluster 3 :  fix, weight-loss, medicar, pay, how, for, the, worst, u.s., hous
Cluster 4 :  most, the, are, american, common, for, don't, but, assist, like
Cluster 5 :  e-cigarett, ban, regul, use, teen, fda, tobacco, find, regular, rule
Cluster 6 :  over, sue, recal, concern, ebola, health, the, death, fear, drug
Cluster 7 :  video:, nhs, ebola, a&amp;e, health, mental, for, care, the, warn
Cluster 8 :  the, ebola, for, and, with, new, may, say, from, drug
Cluster 9 :  babi, born, vote, allow, mom, parent, after, first, help, are
Cluster 10 :  case, ebola, confirm, mers, first, measl, report, new, sierra, leon
Cluster 11 :  brief, nation, world, breast, cancer, risk, test, care, harm, greater
Cluster 12 :  suicid, obesiti, assist, rate, prevent, among, rise, risk, cost, law
Cluster 13 :  health, care, law, mental, worker, the, for, ebola, new, insur
Cluster 14 :  measur, special, trust, practic, put, into, nhs, breast, cancer, risk
Cluster 15 :  cancer, breast, drug, for, lung, prostat, risk, may, patient, treatment
Cluster 16 :  hospit, patient, ebola, u.s., for, the, with, worker, from, medicar
>
```

Et pour la méthode des dbscan:



Et pour la méthode hiérarchique:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> # Assuming 'clustering.hierarchical' is your hierarchical clustering result
> # You may need to cut the dendrogram into clusters using 'cutree'
>
> # Cutting the dendrogram into clusters
> hierarchical_clusters <- cutree(clustering.hierarchical, k = truth.K)
>
> # Extracting the top words for each hierarchical cluster
> top_words_per_hierarchical_cluster <- lapply(unique(hierarchical_clusters), function(cluster_index) {
+   # Subsetting the data for each cluster
+   cluster_subset <- tfidf.matrix[hierarchical_clusters == cluster_index, ]
+
+   # Calculating the mean TF-IDF values for each term in the cluster
+   mean_tfidf <- colMeans(cluster_subset)
+
+   # Sorting the terms based on mean TF-IDF values
+   sorted_words <- order(mean_tfidf, decreasing = TRUE)
+
+   # Extracting the top X words (adjust X as needed)
+   top_words <- colnames(tdm.tfidf)[sorted_words[1:10]]  # Adjust 10 to your desired number of top words
+
+   return(top_words)
+ })
>
> head(top_words_per_hierarchical_cluster,3)
[[1]]
 [1] "cancer"  "fda"     "approv"  "drug"    "breast"  "prostat" "lung"    "win"     "for"     "novarti"

[[2]]
 [1] "the"     "for"     "and"     "ebola"   "may"     "with"    "health"  "say"     "studi"   "video:"

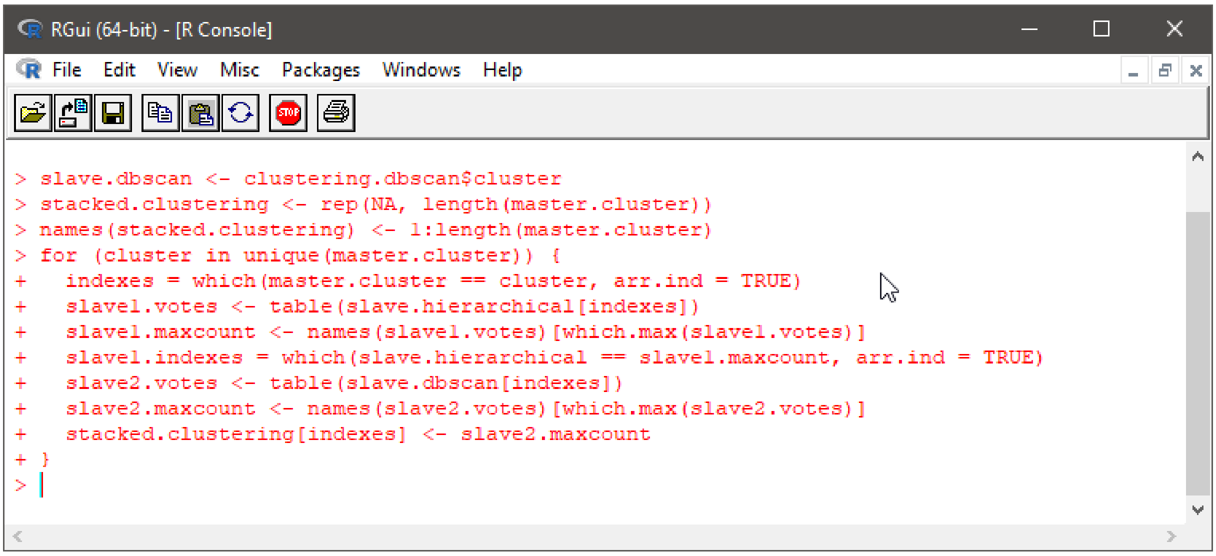
[[3]]
 [1] "your"    "you"     "the"     "these"   "for"     "way"     "and"     "with"    "how"     "food"

>
> # Printing or saving the top words for each hierarchical cluster
> for (i in 1:length(top_words_per_hierarchical_cluster)) {
+   cat("Hierarchical Cluster", i, ": ", paste(top_words_per_hierarchical_cluster[[i]], collapse = ", "), "\n")
+ }
Hierarchical Cluster 1 :  cancer, fda, approv, drug, breast, prostat, lung, win, for, novarti
Hierarchical Cluster 2 :  the, for, and, ebola, may, with, health, say, studi, video:
Hierarchical Cluster 3 :  your, you, the, these, for, way, and, with, how, food
Hierarchical Cluster 4 :  health, law, court, obamacar, medicaid, #obamacar, insuranc, million, suprem, abortion
Hierarchical Cluster 5 :  ebola, test, case, for, brief, worker, vaccin, patient, trial, u.s.
Hierarchical Cluster 6 :  leon, sierra, ebola, case, doctor, die, report, from, quarantin, down
Hierarchical Cluster 7 :  flu, outbreak, measl, bird, disneyland, vaccin, shot, season, farm, this
Hierarchical Cluster 8 :  #healthtalk, love, join, @everydayhealth, @eatsmartbd:, @rmichlermd:, chat, &amp;, vegan, for
Hierarchical Cluster 9 :  cell, stem, research, therapi, scientist, eye, diseas, into, human, treat
Hierarchical Cluster 10 :  breast, cancer, risk, test, care, harm, greater, heart, peopl, short
Hierarchical Cluster 11 :  know, about, you, thing, what, fact, should, need, the, and
Hierarchical Cluster 12 :  new, old, york, blog:, age, the, citi, ebola, quarantin, patient
Hierarchical Cluster 13 :  @cynthiasass:, @goodhealth, &amp;, goal, you, …, weight, can, #talknutrit, set
Hierarchical Cluster 14 :  #healthtotalwel, weekend, @canyonranch!, healthtotalwel, exclusiv, aah-maz, code, detail, here:, offer
Hierarchical Cluster 15 :  @wagdotcom, furri, purchas, foods:, pet, gift, premium, friend, spend, when
Hierarchical Cluster 16 :  pharmalot.., #pharma, @pharmalot:, pharmalittle.., morning.., headlin, good, &amp;, weekend, coffe
>
```

Graphiques des clusters

Avec ces 3 méthodes de clustering, nous pouvons même essayer une méthode d'empilement : fusionner les résultats avec une simple technique de vote dur. Considérant les K-Means comme un clustering maître, chacun de ses clusters sera affecté au cluster majeur représenté parmi leurs points dans le clustering esclave. Nous répétons cette opération à partir du cluster d'esclaves majeur résultant avec l'autre résultat de clustering (le deuxième esclave).

Ce n'est pas la manière la plus efficace de faire du clustering par empilement car cela conduit, au mieux, à conserver des clusters et, au pire, à fusionner des clusters dans le clustering maître. Mais c'est simple à mettre en oeuvre :



Nos quatre clusters sont terminés ! Nous l'avons fait! Waouh ! Mais comment pouvons-nous voir les résultats?

Nous devons définir une façon de voir nos phrases de grande dimension. Puisque nous avons une matrice de distance (utilisée pour le regroupement basé sur la densité), nous pouvons effectuer la technique de mise à l'échelle multidimensionnelle (multidimensional scaling) pour cartographier nos données dans un espace bidimensionnel en choisissant les axes les plus significatifs grâce à une analyse en composantes principales (ACP).

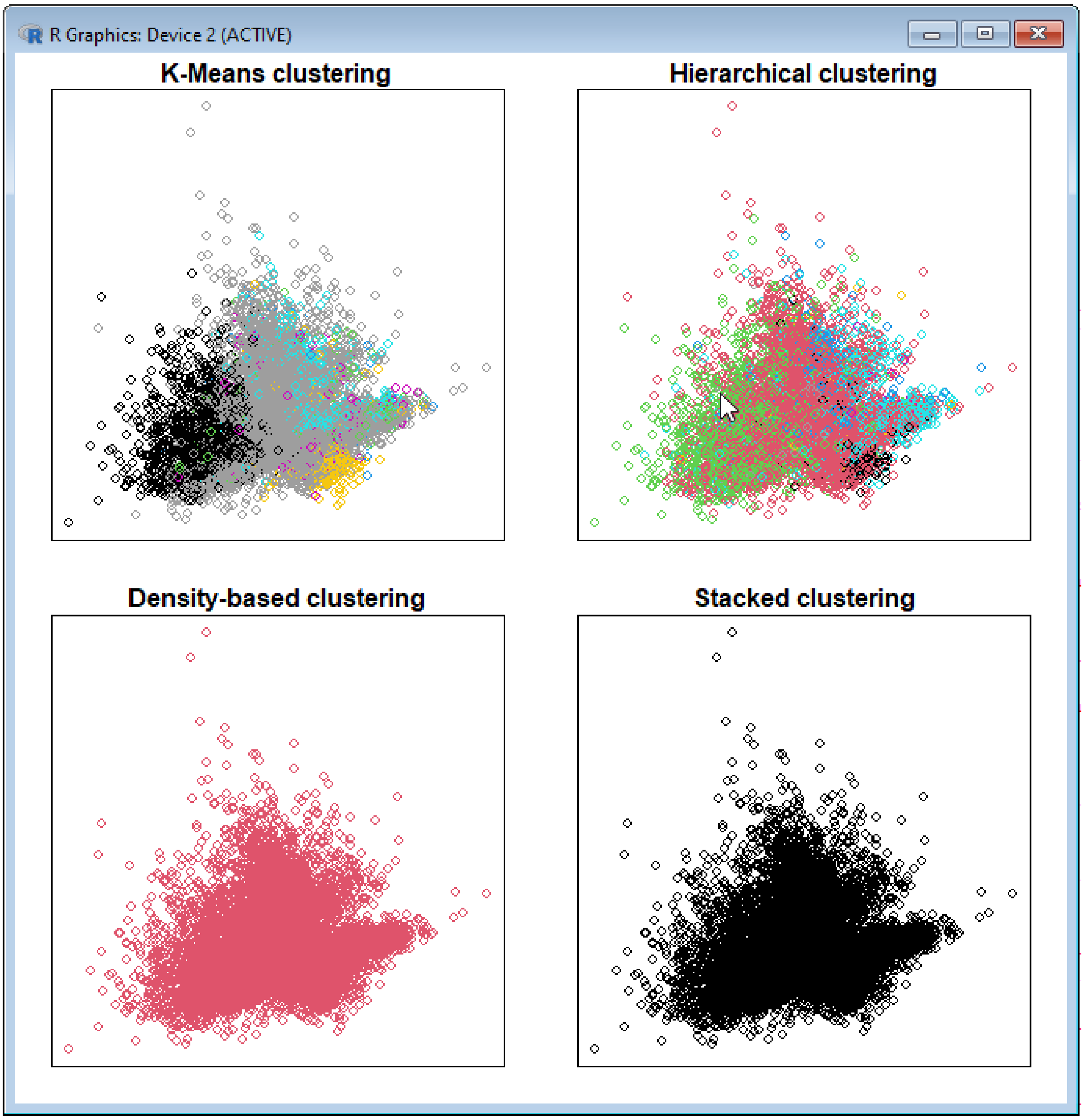
Après cela, R est livré avec des fonctions natives simples pour tracer ces résultats mais attention, nous vous conseillons de réduire la taille du jeu de données initial à 30% sinon le calcul prend environ 1 semaine sur un Alienware Area 51M avec 64 GB de RAM et un processeur 8 cœurs à 4.6 GHz (Windows 10, 64bits). Donc modifiez la ligne de la boucle du début comme-dessous:

```
news.data <- head(news.data, floor(nrow(news.data) * 0.3))
```

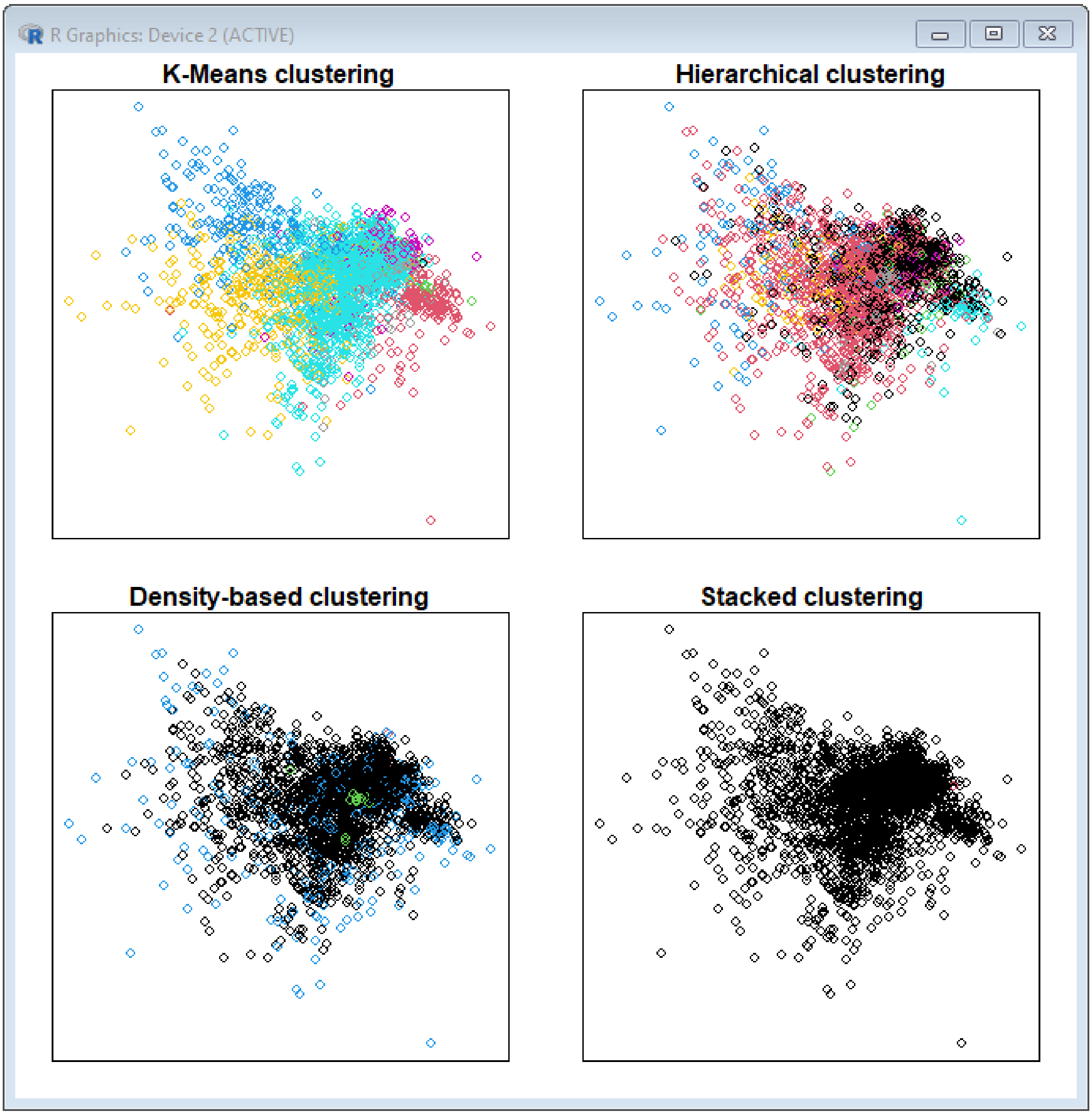
Ensuite, nous faisons l'ACP ce qui durera 5.38 heures comme le montre la capture d'écran ci-dessous sur l'ordinateur susmentionné:

```
> #on démarre le chrono de l'étape qui prend du temps
> ptm <- proc.time()
> #PCA pour choisir les deux meilleures dimensions
> points <- cmdscale(dist.matrix, k = 2)
> #création d'une palette de couleurs
> palette <- colorspace::diverge_hcl(truth.K)
> previous.par <- par(mfrow=c(2,2), mar = rep(1.5, 4))
>
> #partie graphique
> previous.par <- par(mfrow=c(2,2), mar = rep(1.5, 4))
> plot(points, main = "K-Means clustering", col = as.factor(master.cluster),
+      mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
+      xaxt = "n", yaxt = "n", xlab = "", ylab = "")
> plot(points, main = "Hierarchical clustering", col = as.factor(slave.hierarchical),
+      mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
+      xaxt = "n", yaxt = "n", xlab = "", ylab = "")
> plot(points, main = "Density-based clustering", col = as.factor(slave.dbscan),
+      mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
+      xaxt = "n", yaxt = "n", xlab = "", ylab = "")
> plot(points, main = "Stacked clustering", col = as.factor(stacked.clustering),
+      mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
+      xaxt = "n", yaxt = "n", xlab = "", ylab = "")
> #on arrête le chrono
> proc.time() - ptm
    user   system  elapsed
19370.42     6.00 19390.42
>
>
```

Et cela donne les graphiques suivants:



Ou avec seulement 5% du jeu de donnés initial:



Résumé (code entier)

Voici le code entier si jamais en une seul traite:

```
#original code taken from:
#https://medium.com/@SAPCAI/text-clustering-with-r-an-introduction-for-
data-scientists-c406e7454e76

library("tm")
library("stringr")


#we first create the dataframe structure in the memory
dataframe <- data.frame(ID=character(),
                        datetime=character(),
                        content=character(),
                        label=factor())

#we load the data (zip file) and unzip them on the hardrive
```

```
source.url <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/00438/Health-News-Tweets.zip'
target.directory <- '/tmp/clustering-r'
temporary.file <- tempfile()
download.file(source.url, temporary.file)
unzip(temporary.file, exdir = target.directory)

#we read the files
target.directory <- paste(target.directory, 'Health-Tweets', sep = '/')
files <- list.files(path = target.directory, pattern='.txt$')
#filling the dataframe by reading the text content
for (f in files) {
  news.filename = paste(target.directory , f, sep ='/')
  news.label <- substr(f, 0, nchar(f) - 4) #on supprime le ".txt"
  news.data <- read.csv(news.filename,
                        encoding = "UTF-8",
                        header = FALSE,
                        quote = "",
                        sep = "|",
                        col.names = c("ID", "datetime", "content"))
  #we get rid of empty rows
  news.data <- news.data[news.data$content != "", ]
  news.data['label'] = news.label #we add a label to each tweet
  #if your computer is not powerful enough, decrease the percentage
  #from the initial dataset size
  percent_of_dataset <- 0.4
  news.data <- head(news.data, floor(nrow(news.data) * percent_of_dataset))
  dataframe <- rbind(dataframe, news.data) #row appending
}

#deleting the temporary directory
unlink(target.directory, recursive = TRUE)

#just look at a part of the content
head(dataframe)

#a little bit of cleaning to get rid of most URLs

sentences<-str_replace_all(dataframe$content,"https?://[^\\s]+", "")
#we look at a part of the result
head(sentences)

#part where we prepare the corpus
corpus <- tm::Corpus(tm::VectorSource(sentences))
corpus
#classical cleaning part
#suppression of english stop-words
corpus.cleaned <- tm::tm_map(corpus, tm::removeWords,
tm::stopwords("english"))
#stemming of words
corpus.cleaned <- tm::tm_map(corpus, tm::stemDocument, language =
"english")
#suppression of blank spaces
corpus.cleaned <- tm::tm_map(corpus.cleaned, tm::stripWhitespace)
#we look to the content of the resulting object
corpus.cleaned

tdm <- tm::DocumentTermMatrix(corpus.cleaned)
#we look a the tdm structure
tdm
dim(tdm) #its dimension to have an idea of the number of words involved
```

```
#we compute the TfIdf Weight
tdtdm.tfidf <- tm::weightTfIdf(tdm)
tdtdm.tfidf

tdm.tfidf <- tm::removeSparseTerms(tdtdm.tfidf, sparse=0.999)
tdm.tfidf
tfidf.matrix <- as.matrix(tdm.tfidf)
#we take a look to the content of the first 10 rows and 10 columns
tfidf.matrix[0:10,0:10]

#we will now compute the cosine distance

#we start a timer because it's interesting to see how long it take
#depending on the initial percentage of the dataset
#100% takes 2 days of computation
ptm <- proc.time()
dist.matrix <- proxy::dist(tfidf.matrix, method = "cosine")
#we stop the timer
proc.time() - ptm
str(dist.matrix)

#now we run the clustering part
library("dbscan")
truth.K <- 16 #because why not...
clustering.kmeans <- kmeans(tfidf.matrix, truth.K)
str(clustering.kmeans) #just to see the result structure
clustering.hierarchical <- hclust(dist.matrix, method = "ward.D2")
str(clustering.hierarchical) #just to see the result structure
clustering.dbscan <- dbscan::hdbscan(dist.matrix, minPts = 10)
str(clustering.dbscan) #just to see the result structure

#now comes the part where we want to know which of the top X  most frequent

#################################################################
#################################################################
# Assuming 'clustering.kmeans' is our clustering result

library("Rtsne")

# Assuming 'clustering.kmeans' is your K-Means clustering result
# 'centers' attribute contains the cluster centroids
cluster_centers <- clustering.kmeans$centers

# Find the top words for each cluster based on the highest TF-IDF values in
the centroid
top_words_per_cluster <- lapply(1:truth.K, function(cluster_index) {
  # Extracting the centroid of the cluster
  centroid <- cluster_centers[cluster_index, ]

  # Sorting the words based on their TF-IDF values
  sorted_words <- order(centroid, decreasing = TRUE)

  # Extracting the top X words (adjust X as needed)
  top_words <- colnames(tdm.tfidf)[sorted_words[1:10]]  # Adjust 10 to your
desired number of top words

  return(top_words)
})

head(top_words_per_cluster,3)
```

```
# Printing or saving the top words for each cluster
for (i in 1:length(top_words_per_cluster)) {
  cat("Cluster", i, ": ", paste(top_words_per_cluster[[i]], collapse = ",
"), "\n")
}


#################################################################
#################################################################
# Assuming 'clustering.dbscan' is your DBSCAN clustering result
# 'cluster' attribute contains the cluster assignments

# Extracting the top words for each cluster
top_words_per_dbscan_cluster <- lapply(unique(clustering.dbscan$cluster),
function(cluster_index) {
  # Subsetting the data for each cluster
  cluster_subset <- tfidf.matrix[clustering.dbscan$cluster ==
cluster_index, ]

  # Calculating the mean TF-IDF values for each term in the cluster
  mean_tfidf <- colMeans(cluster_subset)

  # Sorting the terms based on mean TF-IDF values
  sorted_words <- order(mean_tfidf, decreasing = TRUE)

  # Extracting the top X words (adjust X as needed)
  top_words <- colnames(tdm.tfidf)[sorted_words[1:10]]  # Adjust 10 to your
desired number of top words

  return(top_words)
})

# Printing or saving the top words for each DBSCAN cluster
for (i in 1:length(top_words_per_dbscan_cluster)) {
  cat("DBSCAN Cluster", i, ": ", paste(top_words_per_dbscan_cluster[[i]],
collapse = ", "), "\n")
}



#################################################################
#################################################################

# Assuming 'clustering.hierarchical' is your hierarchical clustering result
# You may need to cut the dendrogram into clusters using 'cutree'

# Cutting the dendrogram into clusters
hierarchical_clusters <- cutree(clustering.hierarchical, k = truth.K)

# Extracting the top words for each hierarchical cluster
top_words_per_hierarchical_cluster <- lapply(unique(hierarchical_clusters),
function(cluster_index) {
  # Subsetting the data for each cluster
  cluster_subset <- tfidf.matrix[hierarchical_clusters == cluster_index, ]

  # Calculating the mean TF-IDF values for each term in the cluster
  mean_tfidf <- colMeans(cluster_subset)

  # Sorting the terms based on mean TF-IDF values
  sorted_words <- order(mean_tfidf, decreasing = TRUE)
```

```
  # Extracting the top X words (adjust X as needed)
  top_words <- colnames(tdm.tfidf)[sorted_words[1:10]]  # Adjust 10 to your
desired number of top words

  return(top_words)
})

# Printing or saving the top words for each hierarchical cluster
for (i in 1:length(top_words_per_hierarchical_cluster)) {
  cat("Hierarchical Cluster", i, ": ",
paste(top_words_per_hierarchical_cluster[[i]], collapse = ", "), "\n")
}

#################################################################
#################################################################
# partie graphiques

#with 30% of the dataset it takes 5 hours and 20 minutes
ptm <- proc.time()
points <- cmdscale(dist.matrix, k = 2)
#we stop the timer
proc.time() - ptm

master.cluster <- clustering.kmeans$cluster
slave.hierarchical <- cutree(clustering.hierarchical, k = truth.K)
slave.dbscan <- clustering.dbscan$cluster
stacked.clustering <- rep(NA, length(master.cluster))
names(stacked.clustering) <- 1:length(master.cluster)
for (cluster in unique(master.cluster)) {
  indexes = which(master.cluster == cluster, arr.ind = TRUE)
  slave1.votes <- table(slave.hierarchical[indexes])
  slave1.maxcount <- names(slave1.votes)[which.max(slave1.votes)]
  slave1.indexes = which(slave.hierarchical == slave1.maxcount, arr.ind =
TRUE)
  slave2.votes <- table(slave.dbscan[indexes])
  slave2.maxcount <- names(slave2.votes)[which.max(slave2.votes)]
  stacked.clustering[indexes] <- slave2.maxcount
}

palette <- colorspace::diverge_hcl(truth.K) # Creating a color palette
previous.par <- par(mfrow=c(2,2), mar = rep(1.5, 4))

plot(points, main = 'K-Means clustering', col = as.factor(master.cluster),
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
plot(points, main = 'Hierarchical clustering', col =
as.factor(slave.hierarchical),
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
plot(points, main = 'Density-based clustering', col =
as.factor(slave.dbscan),
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
plot(points, main = 'Stacked clustering', col =
as.factor(stacked.clustering),
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
par(previous.par) # recovering the original plot space parameters
```

32. Bioinformatique/Biostatistiques

La bioinformatique (dont nous retrouvons aussi des Toolbox dans MATLAB) est aussi excellent exemple d'application pratique des outils statistiques et de machine learning.

Exceptionnellement un petit rappel de culture générale s'impose.

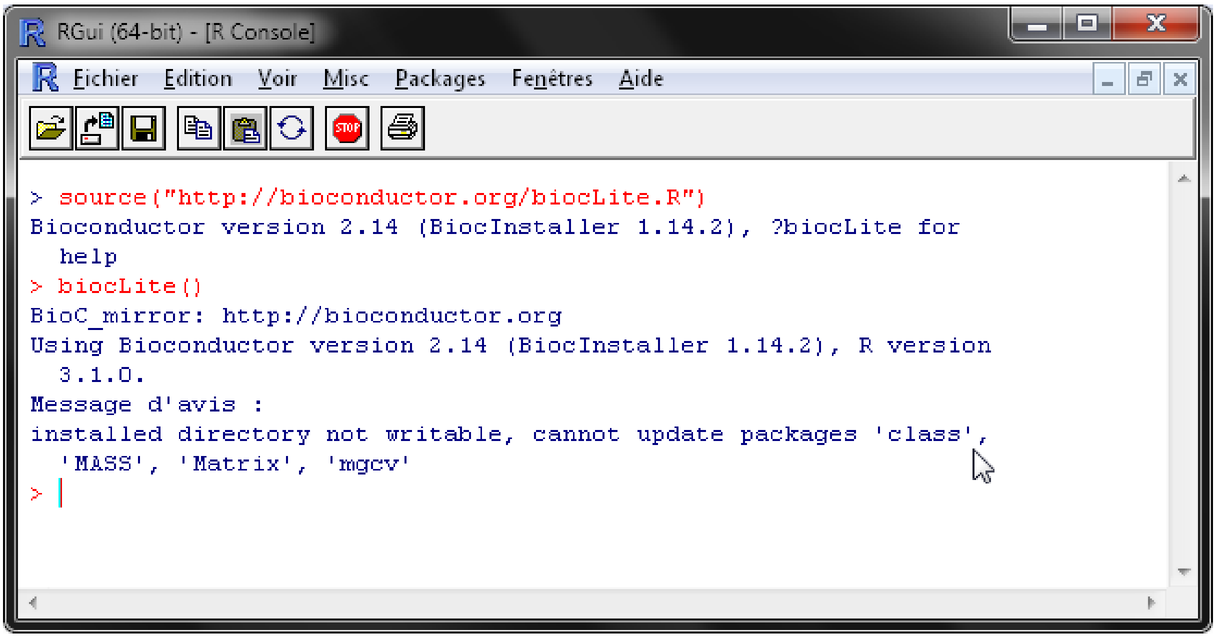
La technologie des microarray utilise les propriétés de l'hybridation des acides nucléiques et utilise des molécules complémentaires attachées à une surface solide, nommées "probes" dans le domaine, pour mesurer la quantité d'une acide nucléique spécifique (ou d'une chaîne particulière) présent dans un échantillon souvent nommé "cible".

Les molécules se retrouvant dans les "probes" sont par la suite labélisées à l'aide de scanners spécialisés et la quantité d'acides nucléiques dans chaque probe est donnée en termes "d'intensité" (fluorescence de l'activité chimique) ce qui se traduit souvent sur les ordinateurs en une image où chaque "probe" est un pixel ou ensemble constant de pixels ayant une couleur donnée avec un certaine intensité (souvent du noir – rien - au blanc pour le maximum d'intensité). La matrice (grille) de couleurs obtenues se lit parfois en ligne pour un type de gène donné et en colonne des conditions d'analyse du gène d'intérêt.

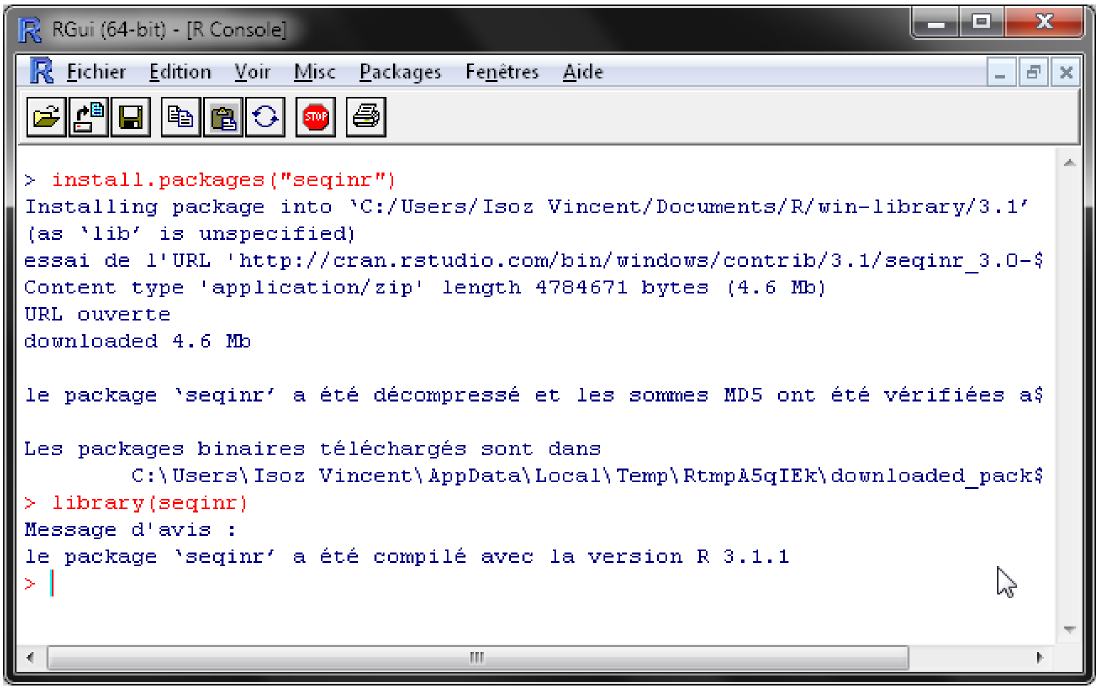
Les industriels[47] proposent une large palette de plateformes d'analyses différentes dont le critère majeur est souvent le bruit de mesure (et en deuxième le type de cible d'intérêt: génome humain, rat ou autre parfois sur mesure…) qui dans la pratique nécessitent des répétitions d'analyses et manipulations statistiques des données extraites avant de pouvoir conclure sur quoi que ce soit de manière un peu scientifique (donc pour un même échantillon on obtient des résultats sensiblement différents à cause du bruit, du protocole expérimental et de l'opérateur). Ces manipulations sont souvent nommées "preprocessing" (prétraitement) dans le domaine.

Nous allons pour survoler ce sujet d'abord installer le package **bioconductor** (http://bioconductor.org/). L'installation est un peu spéciale car en réalité nous appelons des fonctions plutôt que des les installer:

[47] En ce début de 21ème siècle Affymetrix serait le leader du marché des microarray.



Nous avons aussi besoin du package **seqinr** pour obtenir les séquences ADN et extraire des informations des bases de données de protéines en lignes et aussi pour leur analyse élémentaire:



Exemple 443.: Modèle SIR de propagation des épidémies

R 3.6.3

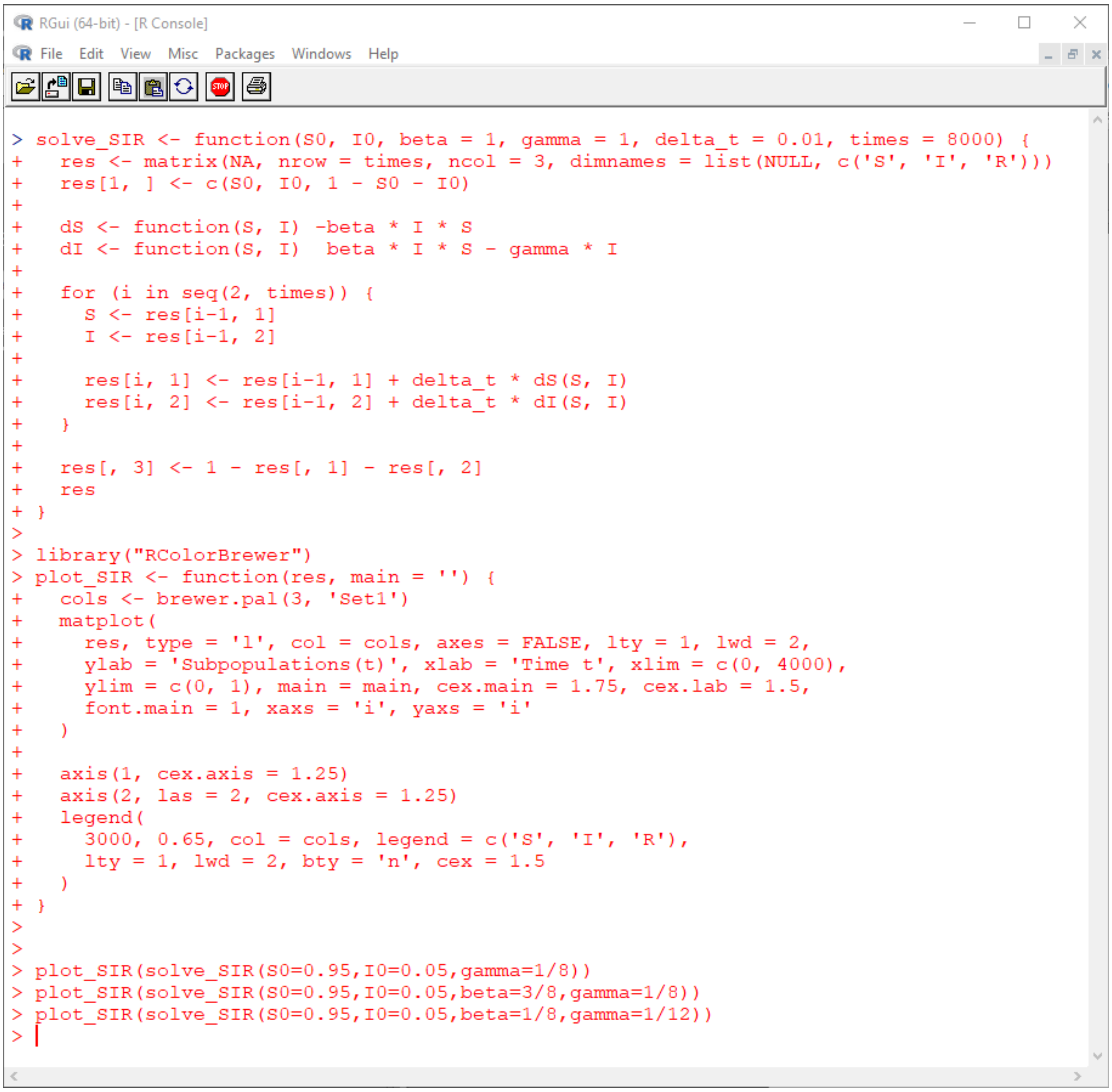
Il existe plusieurs packages pour modéliser le fameux modèle SIR:

- Le package EpiDynamics
- Le package shinySIR
- Le package desolve

Cependant nous préférons montrer le code explicite de comment modéliser ce modèle car c'est très instructif et plutôt court.

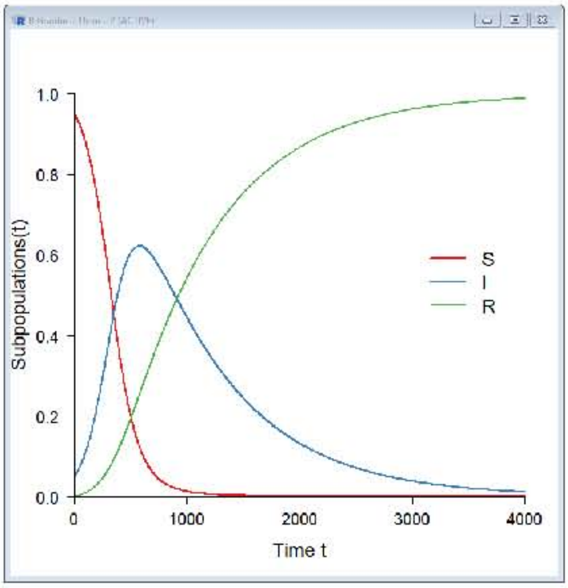
Donc voyons un exemple de code pris sur:

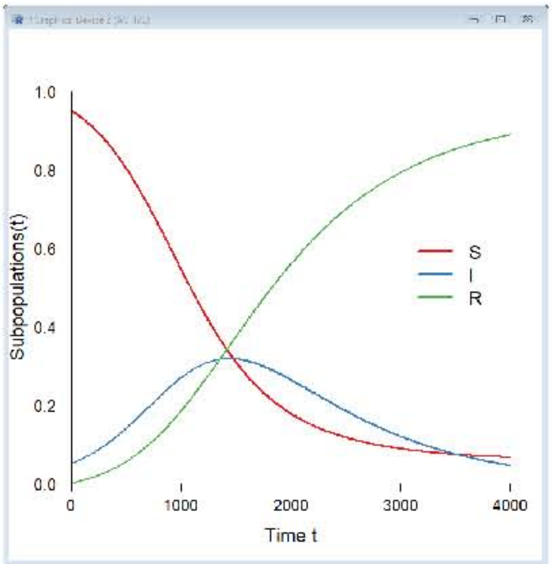
https://fabiandablander.com/r/Nonlinear-Infection.html

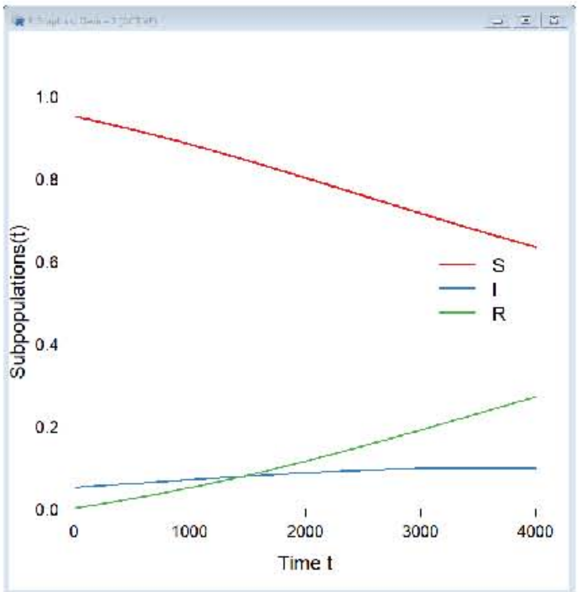


```
> solve_SIR <- function(S0, I0, beta = 1, gamma = 1, delta_t = 0.01, times = 8000) {
+   res <- matrix(NA, nrow = times, ncol = 3, dimnames = list(NULL, c('S', 'I', 'R')))
+   res[1, ] <- c(S0, I0, 1 - S0 - I0)
+
+   dS <- function(S, I) -beta * I * S
+   dI <- function(S, I)  beta * I * S - gamma * I
+
+   for (i in seq(2, times)) {
+     S <- res[i-1, 1]
+     I <- res[i-1, 2]
+
+     res[i, 1] <- res[i-1, 1] + delta_t * dS(S, I)
+     res[i, 2] <- res[i-1, 2] + delta_t * dI(S, I)
+   }
+
+   res[, 3] <- 1 - res[, 1] - res[, 2]
+   res
+ }
>
> library("RColorBrewer")
> plot_SIR <- function(res, main = '') {
+   cols <- brewer.pal(3, 'Set1')
+   matplot(
+     res, type = 'l', col = cols, axes = FALSE, lty = 1, lwd = 2,
+     ylab = 'Subpopulations(t)', xlab = 'Time t', xlim = c(0, 4000),
+     ylim = c(0, 1), main = main, cex.main = 1.75, cex.lab = 1.5,
+     font.main = 1, xaxs = 'i', yaxs = 'i'
+   )
+
+   axis(1, cex.axis = 1.25)
+   axis(2, las = 2, cex.axis = 1.25)
+   legend(
+     3000, 0.65, col = cols, legend = c('S', 'I', 'R'),
+     lty = 1, lwd = 2, bty = 'n', cex = 1.5
+   )
+ }
>
>
> plot_SIR(solve_SIR(S0=0.95,I0=0.05,gamma=1/8))
> plot_SIR(solve_SIR(S0=0.95,I0=0.05,beta=3/8,gamma=1/8))
> plot_SIR(solve_SIR(S0=0.95,I0=0.05,beta=1/8,gamma=1/12))
>
```

Ce qui donne les trois graphiques suivants:





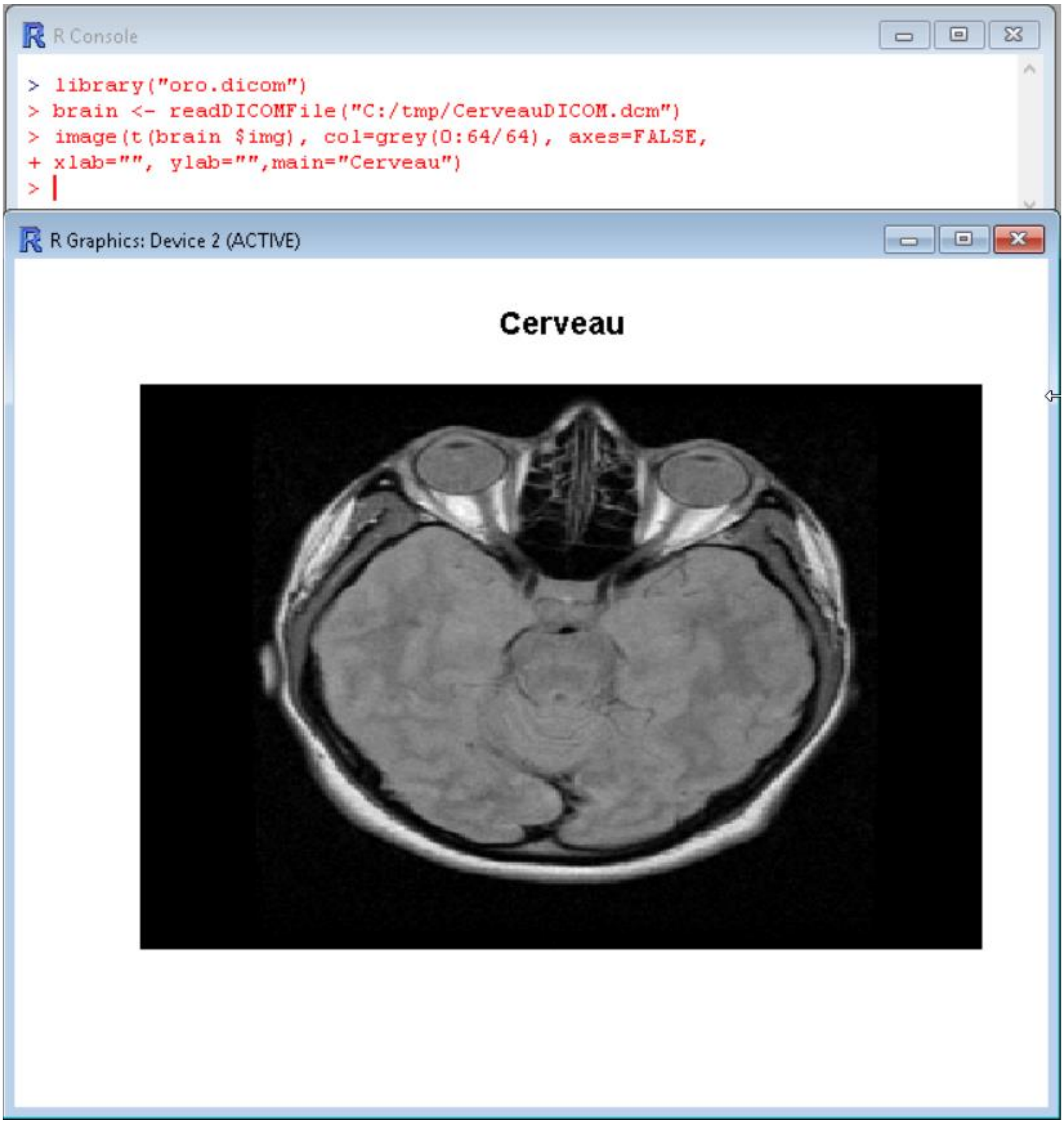


Exemple 444.: Lire un fichier *.DICOM (Digital Imaging and Communications in Medicine)

R 3.2.1

Dans le cours Photoshop nous avons vu les nombreuses possibilités d'ouvrir et d'éditer un fichier DICOM.

Voyons ce que nous pouvons tirer de R à l'aide du package **oro.dicom** et la fonction **readDICOMFile( )**:



ToDo: Trouver comment choisir le calque (couche) à la lire.

Exemple 445.: Obtenir des séquençages sur les BDD en ligne
R 3.1.0

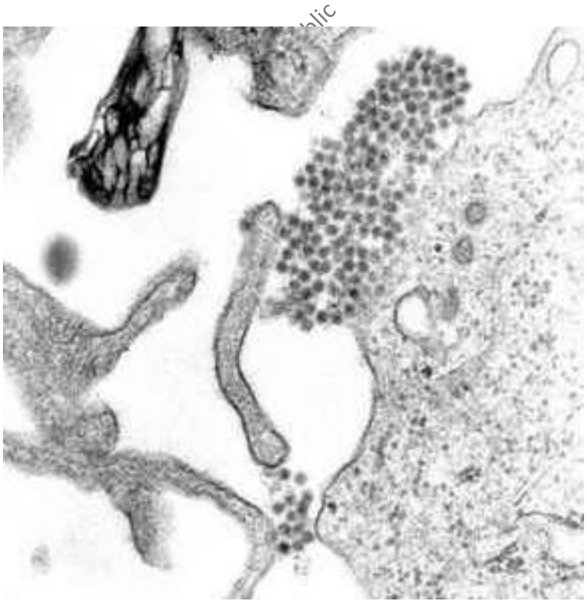
Le Centre national d'information sur la biotechnologie (NCBI; www.ncbi.nlm.nih.gov) aux États-Unis maintient une vaste base de données de toutes les données de séquence d'ADN et de protéines qui ont été recueillis jusqu'à aujourd'hui.

Il y a aussi une base de données similaire en Europe, le European Molecular Biology Laboratory (EMBL; www.ebi.ac.uk/embl), et également une base de données similaire au Japon, la DNA Data Bank of Jaban (DDBJ; www.ddbj.nig.ac.jp).

Ces trois échangent des données des bases de données chaque nuit, donc en un point quelconque dans le temps, elles contiennent des données presque identiques.

Chaque séquence dans la base de données NCBI séquence est mémorisée dans un registre séparé, et se voit attribuer un identifiant unique qui peut être utilisé pour se référer à cet enregistrement de la séquence. L'identifiant est connu comme une adhésion, et consiste en une combinaison de chiffres et de lettres.

Par exemple, le virus de la dengue (grippe tropicale) provoque de la fièvre de la dengue, qui est classée comme une maladie tropicale et désignée par l'un de quatre types de virus de la dengue: DEN-1, DEN-2, DEN-3 et DEN-4. Les adhésions NCBI pour les séquences d'ADN de la DEN-1, DEN-2, DEN-3 et DEN-4 sont respectivement NC_001477, NC_001474, NC_001475 et NC_002640.



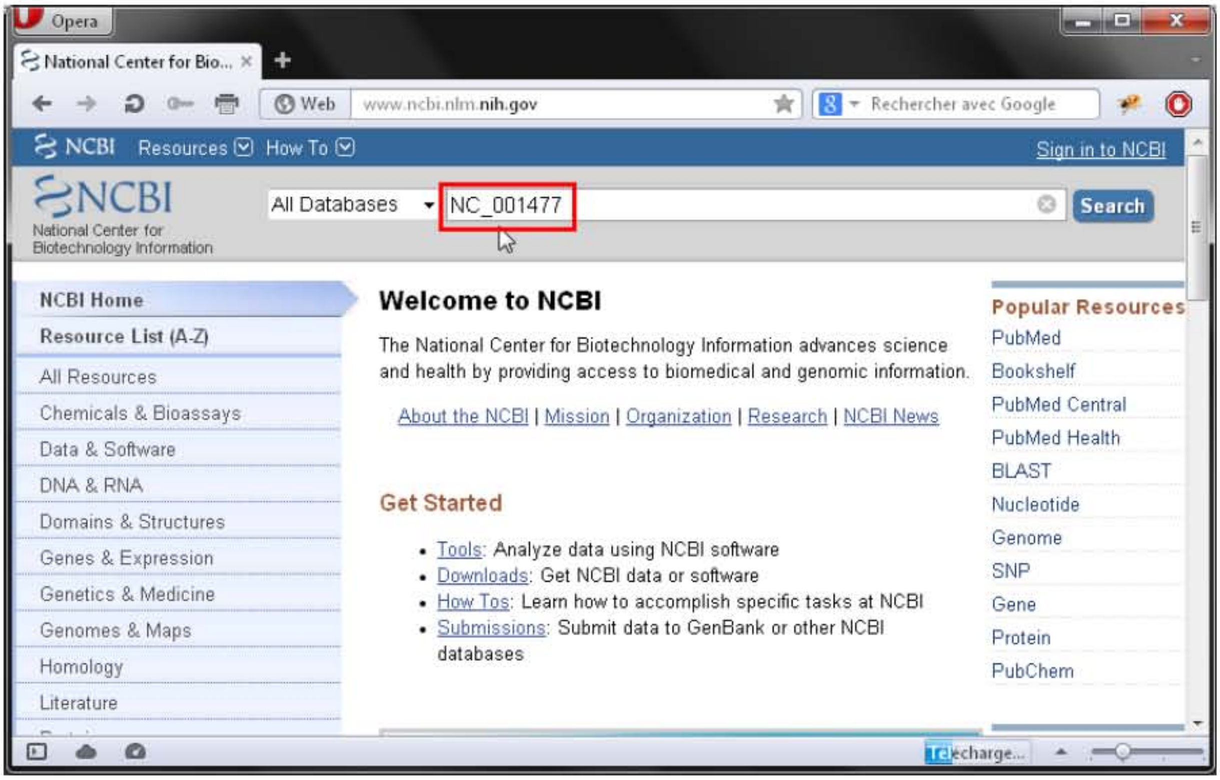
Virus de la dengue au microscope électronique

Note that because the NCBI Sequence Database, the EMBL Sequence Database, and DDBJ exchange data every night, the DEN-1 (and DEN-2, DEN-3, DEN-4) Dengue virus sequence will be present in all three databases, but it will have different accessions in each database, as they each use their own numbering systems for referring to

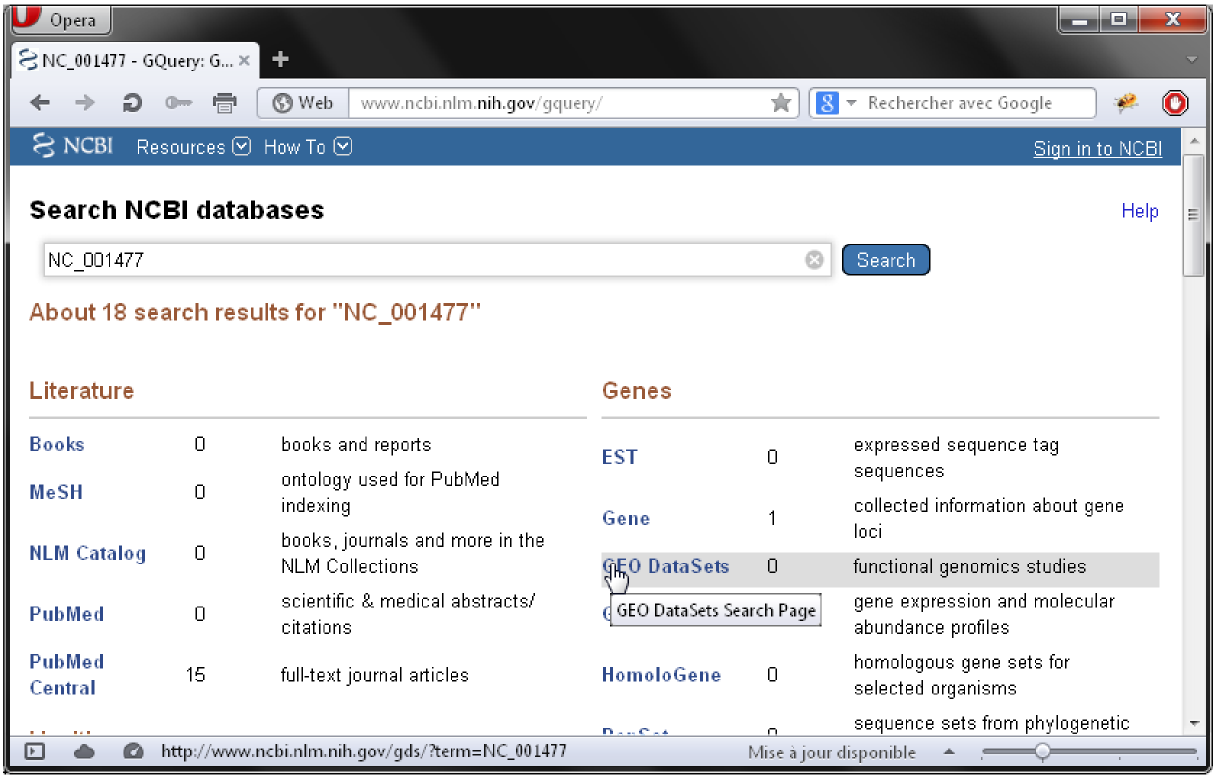
Notez que parce que la base de données de séquences NCBI, EMBL et DDBJ synchronises leurs tous les soirs, les séquences du virus DEN-1 (et DEN-2, DEN-3, DEN-4) sera présent

dans les trois bases de données, mais il aura différentes accessions dans chaque base de données, car ils utilisent chacun leurs propres systèmes de numérotation pour faire référence à leurs propres enregistrements de séquence...

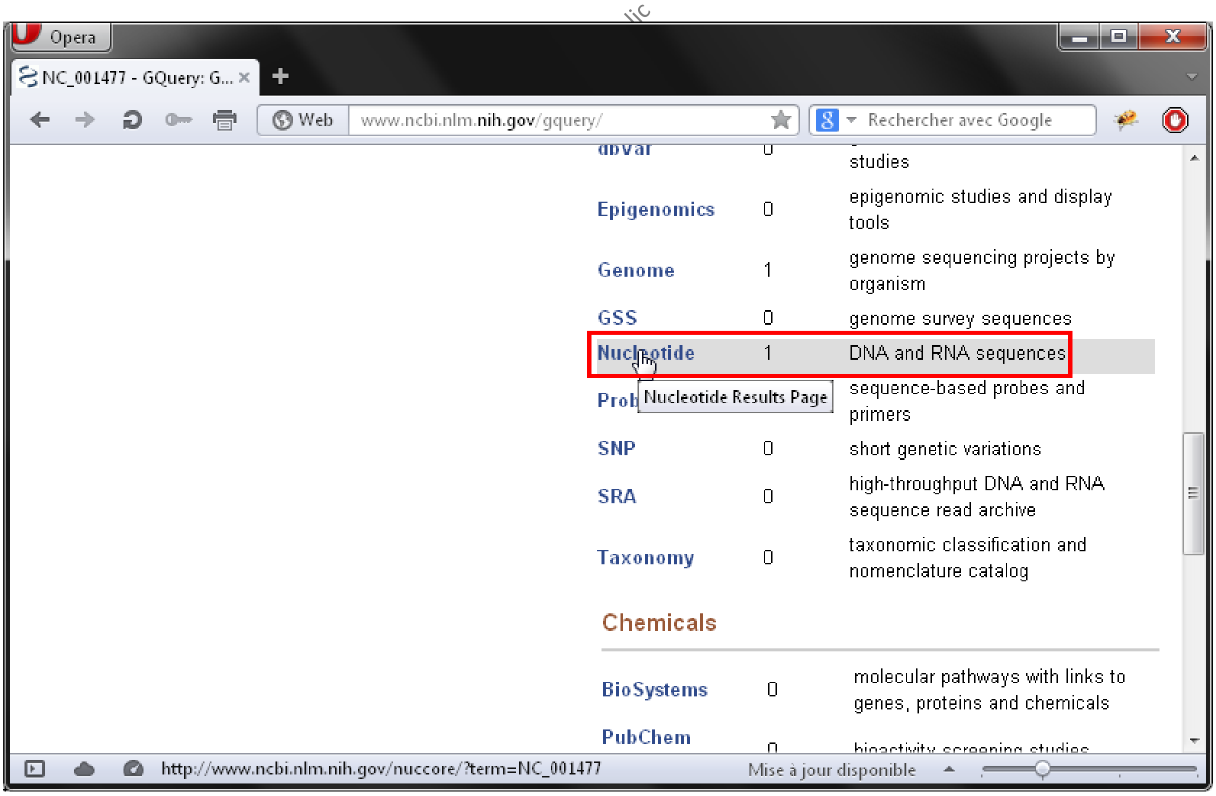
Nous allons voir maintenant comment extraire la séquence NC_001477 du NCBI par exemple. Pour cela nous allons sur le site Internet du NCBI et nous tapons le séquence qui nous intéresse dans le champ de recherche:



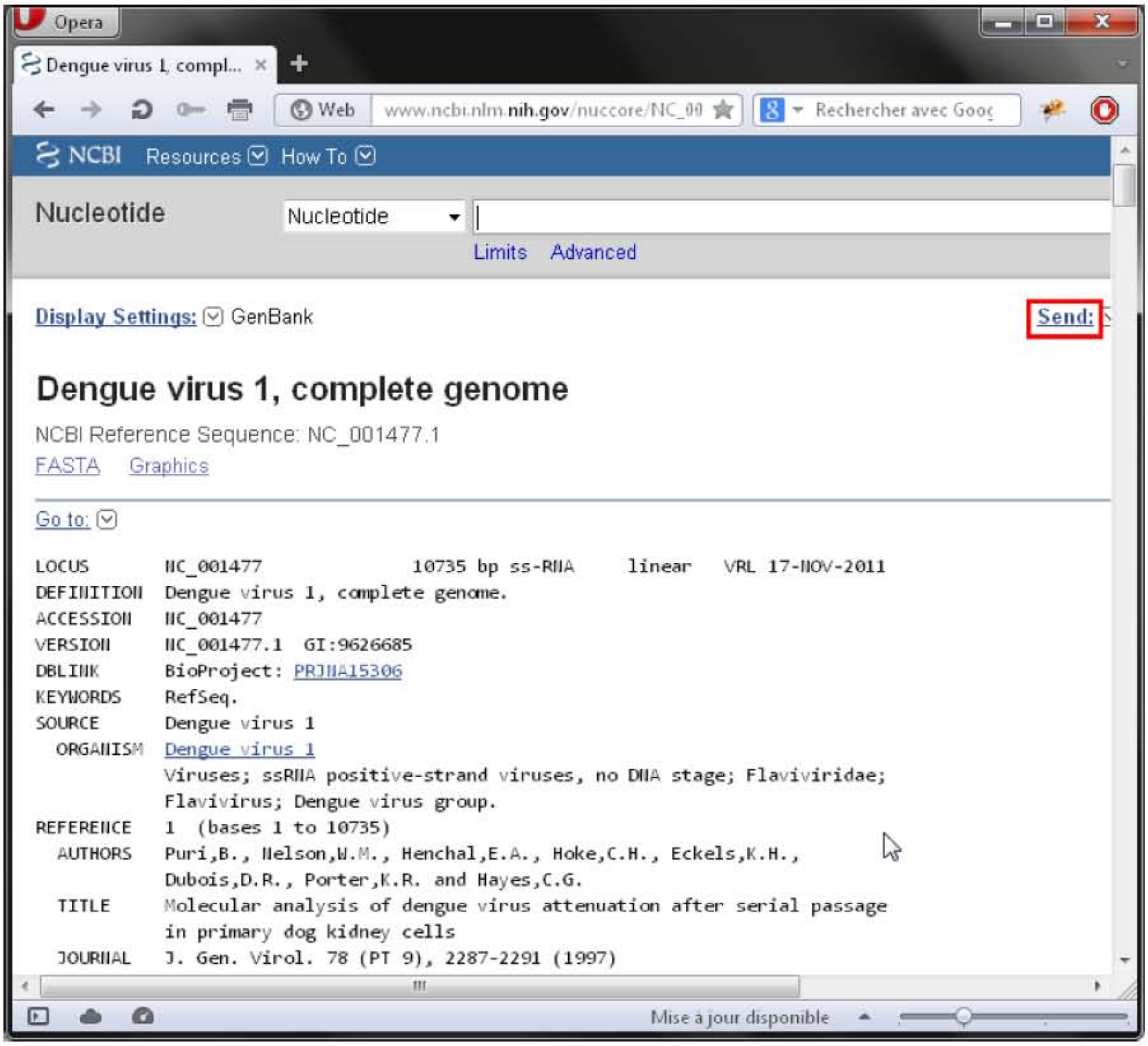
Il y aura de nombreux résultats quand vous cliquerez sur **Search**:



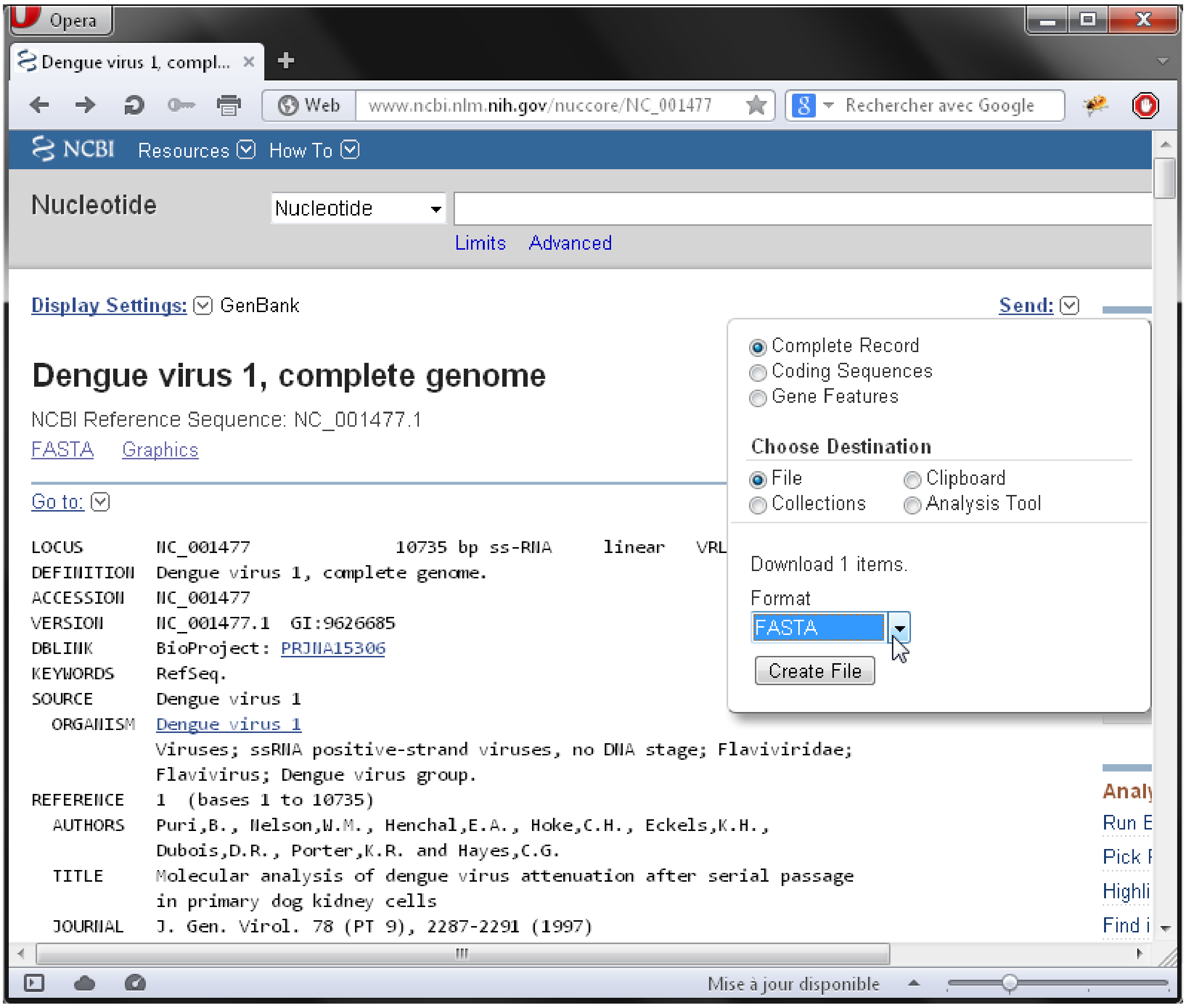
Il faut descendre dans la page jusqu'à trouver:



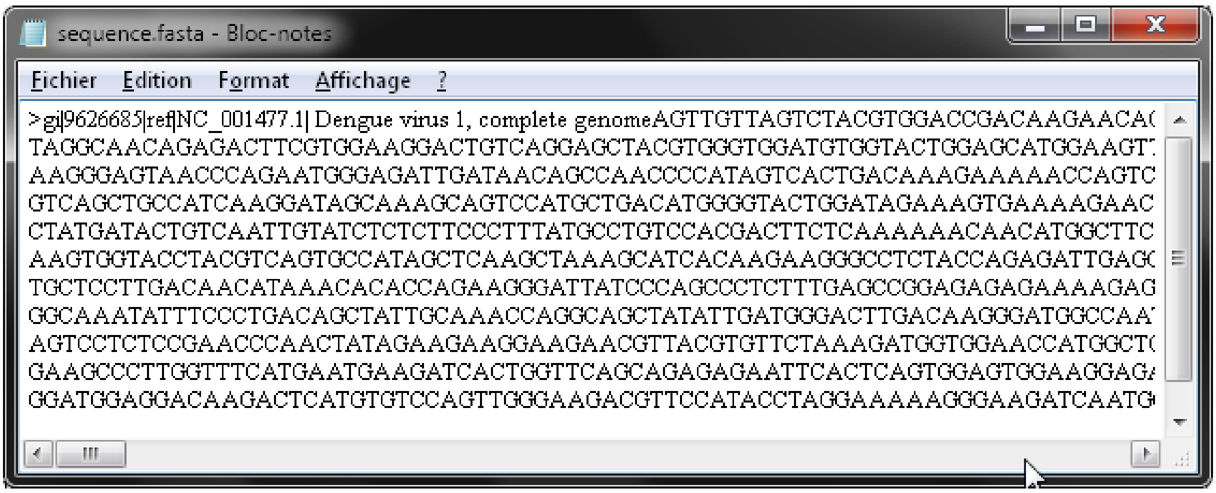
Vous cliquez dessus pour arriver sur:



Cliquez sur le **Send** qui est en haut à droite et choisissez les options suivantes:



Cliquez sur **Create File**, pour que ce fichier texte à l'extension *.FASTA se télécharge sur votre ordinateur. Vous aurez alors:

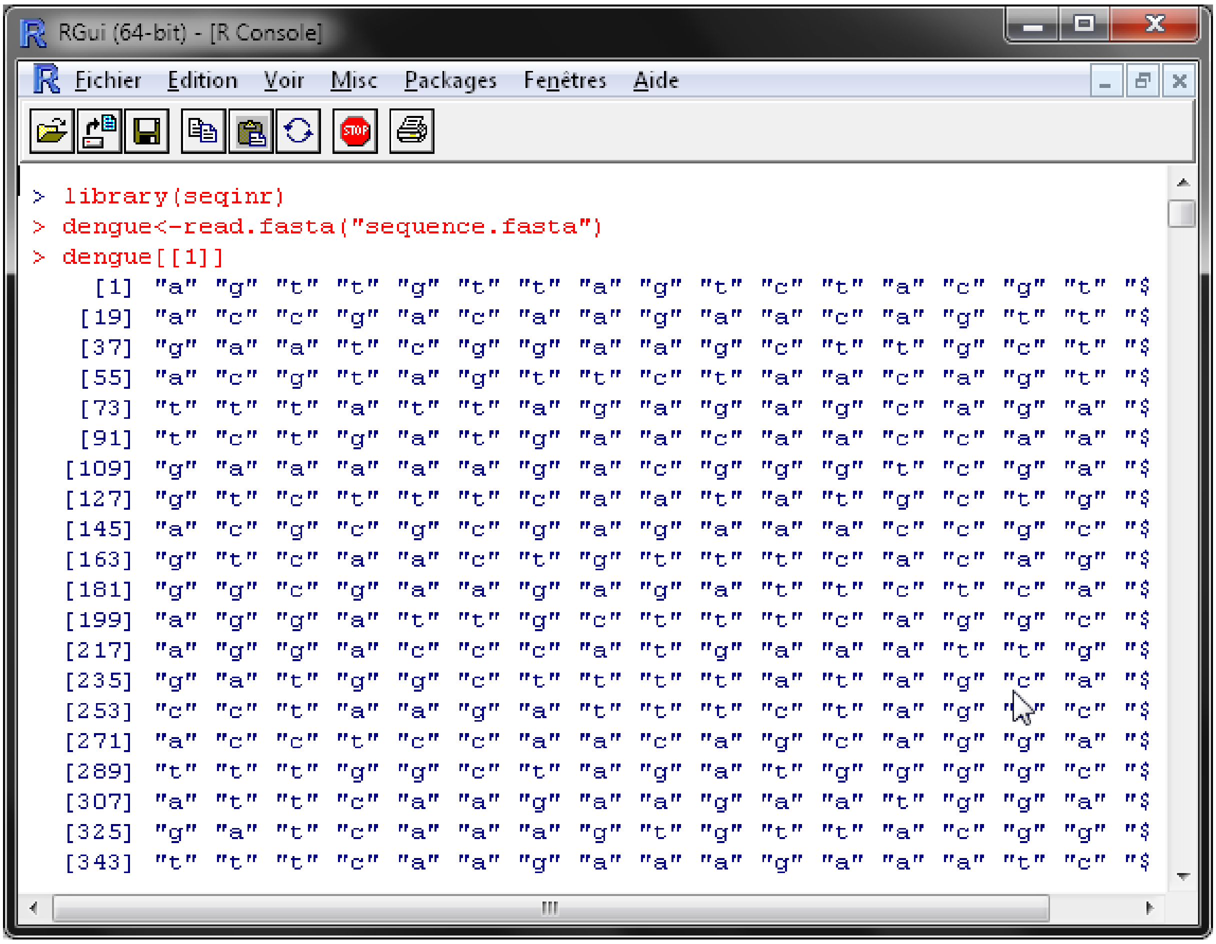


Exemple 446.: Lire et traiter un fichier *.FASTA

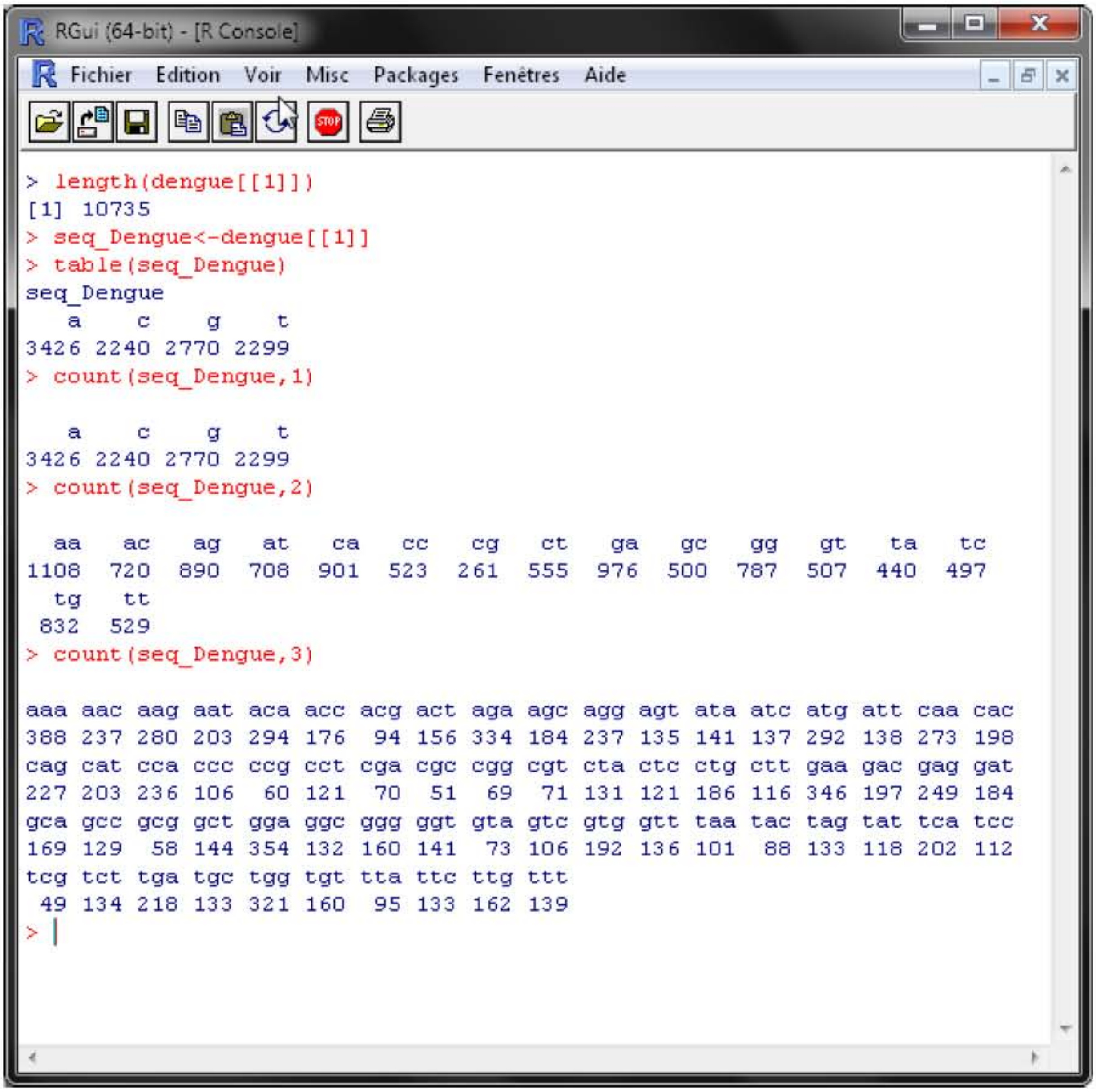
R 3.1.0

Nous allons voir maintenant comment faire quelques petites traitement et analyses élémentaires du fichier FASTA que nous avons téléchargé.

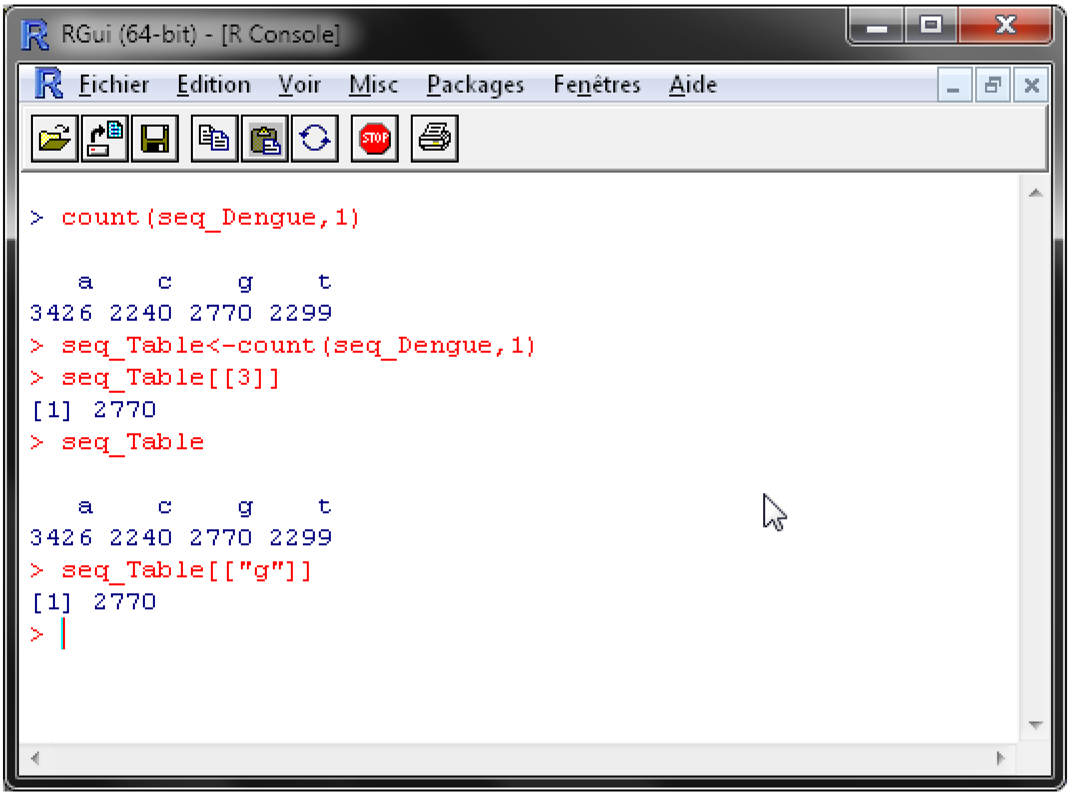
D'abord regardons un peu le contenu en utilisant la commande **read.fasta( )** :



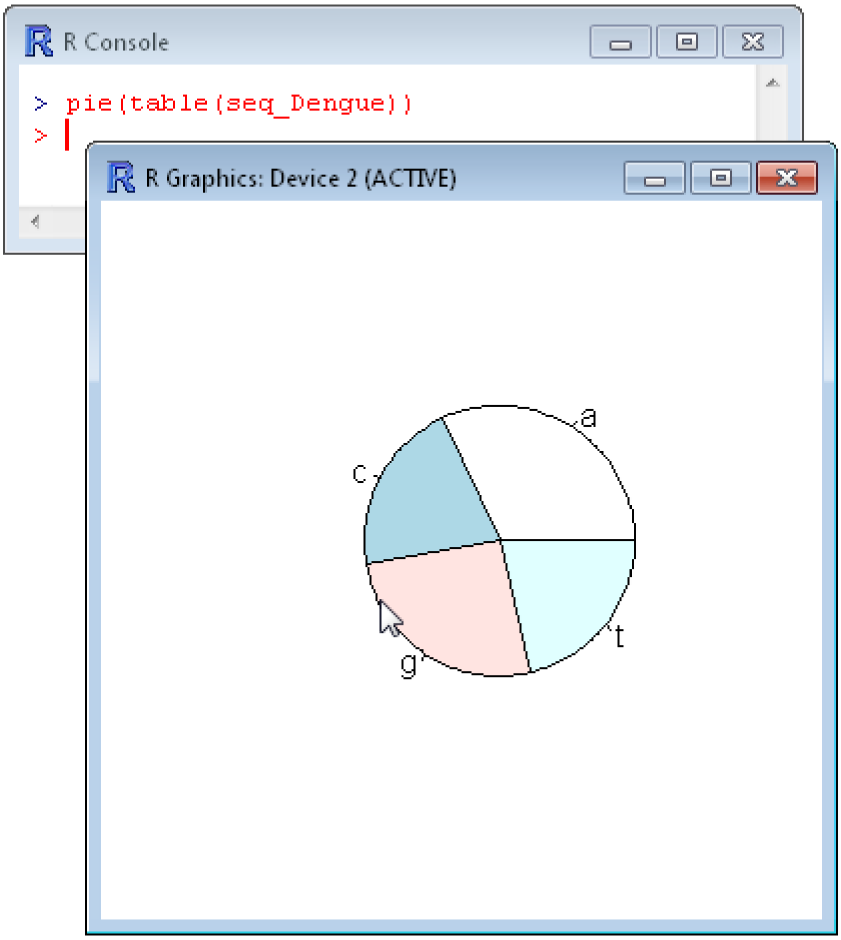
On extrait quelques petites informations élémentaires autres:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> length(dengue[[1]])
[1] 10735
> seq_Dengue<-dengue[[1]]
> table(seq_Dengue)
seq_Dengue
a c g t
3426 2240 2770 2299
> count(seq_Dengue,1)
a c g t
3426 2240 2770 2299
> count(seq_Dengue,2)
aa ac ag at ca cc cg ct ga gc gg gt ta tc
1108 720 890 708 901 523 261 555 976 500 787 507 440 497
tg tt
832 529
> count(seq_Dengue,3)
aaa aac aag aat aca acc acg act aga agc agg agt ata atc atg att caa cac
388 237 280 203 294 176 94 156 334 184 237 135 141 137 292 138 273 198
cag cat cca ccc ccg cct cga cgc cgg cgt cta ctc ctg ctt gaa gac gag gat
227 203 236 106 60 121 70 51 69 71 131 121 186 116 346 197 249 184
gca gcc gcg gct gga ggc ggg ggt gta gtc gtg gtt taa tac tag tat tca tcc
169 129 58 144 354 132 160 141 73 106 192 136 101 88 133 118 202 112
tcg tct tga tgc tgg tgt tta ttc ttg ttt
49 134 218 133 321 160 95 133 162 139
>

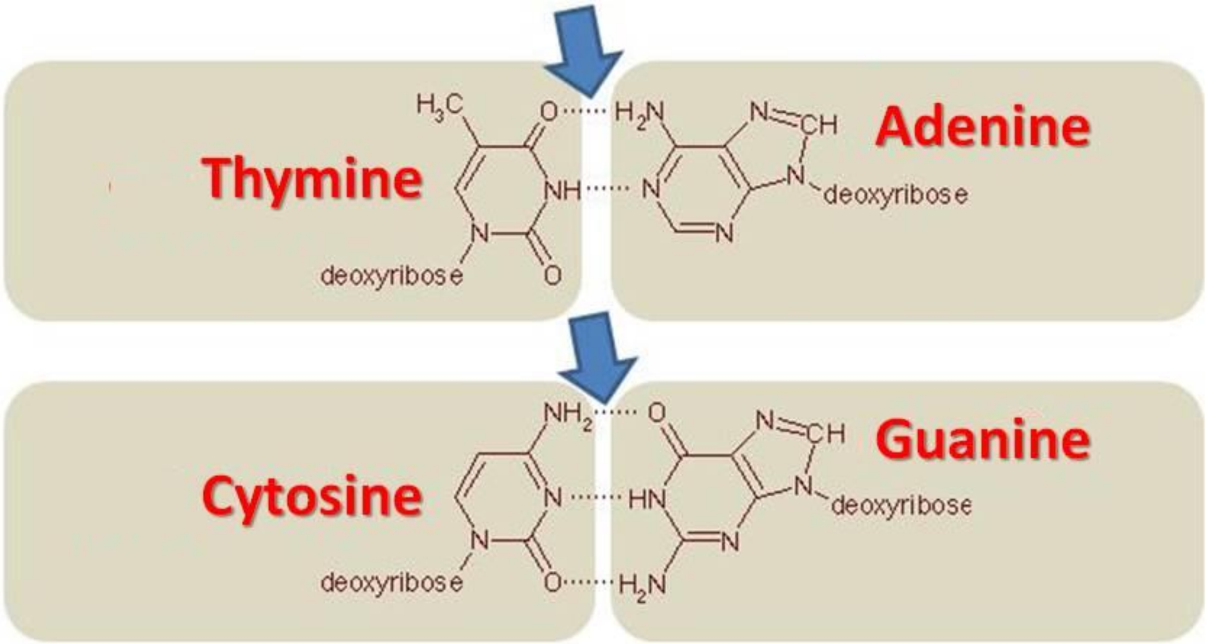
ou aussi:



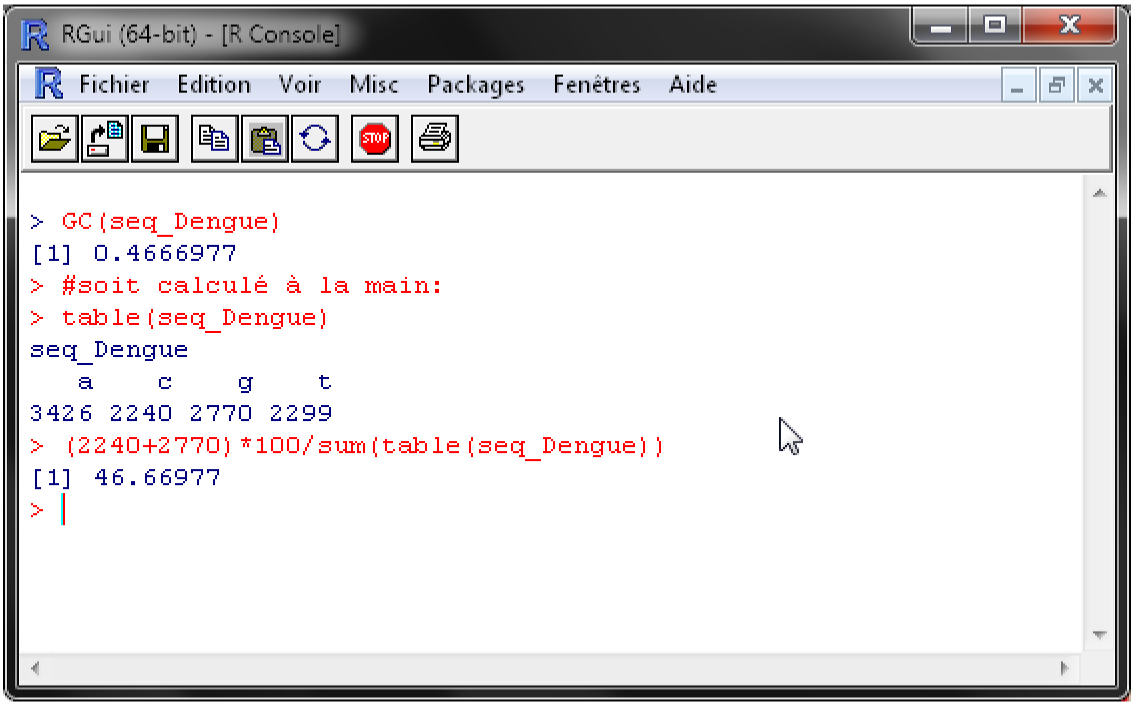
Ou un petit graphique vite fait mal fait:



Comme les G et les C impliquant des triples liaisons et donc des propriétés physico-chimiques parfois intéressantes:



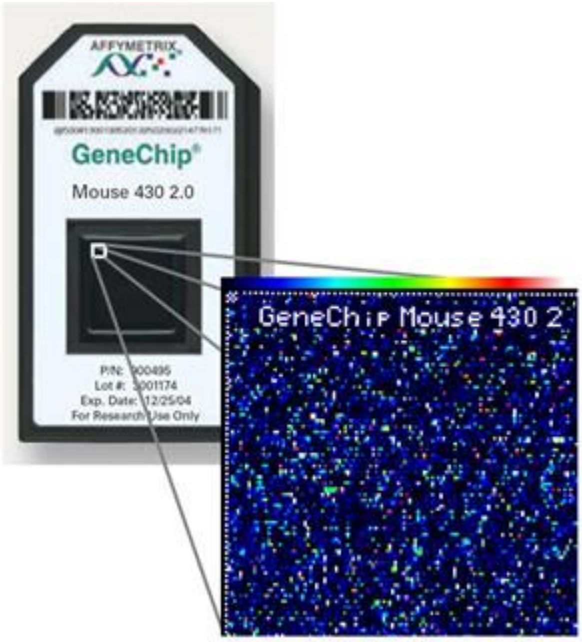
il peut être utile de connaître la proportion de leur présence:



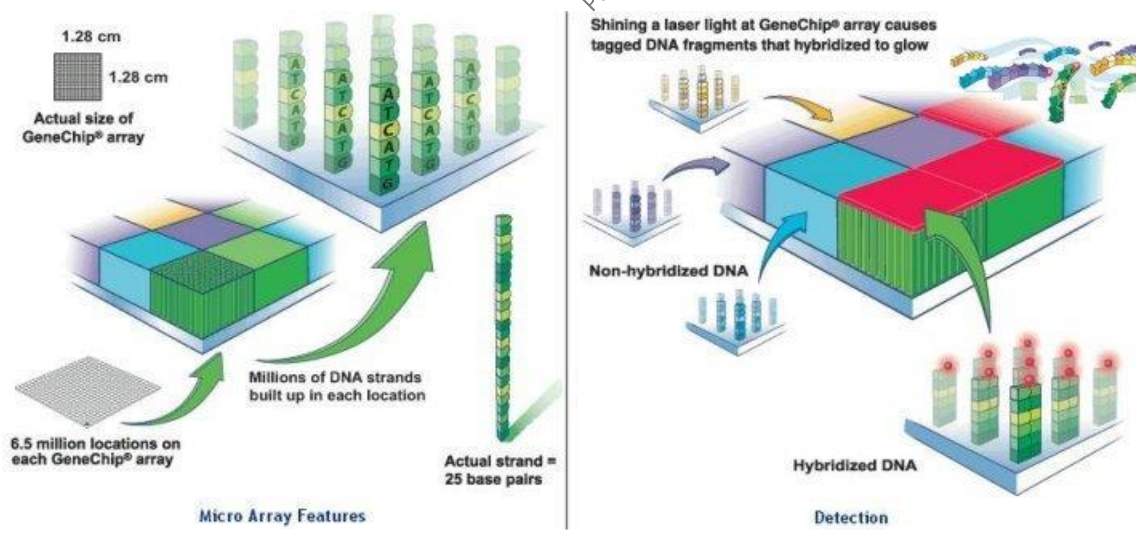
Exemple 447.: Afficher l'image d'un microarray

R 3.1.0

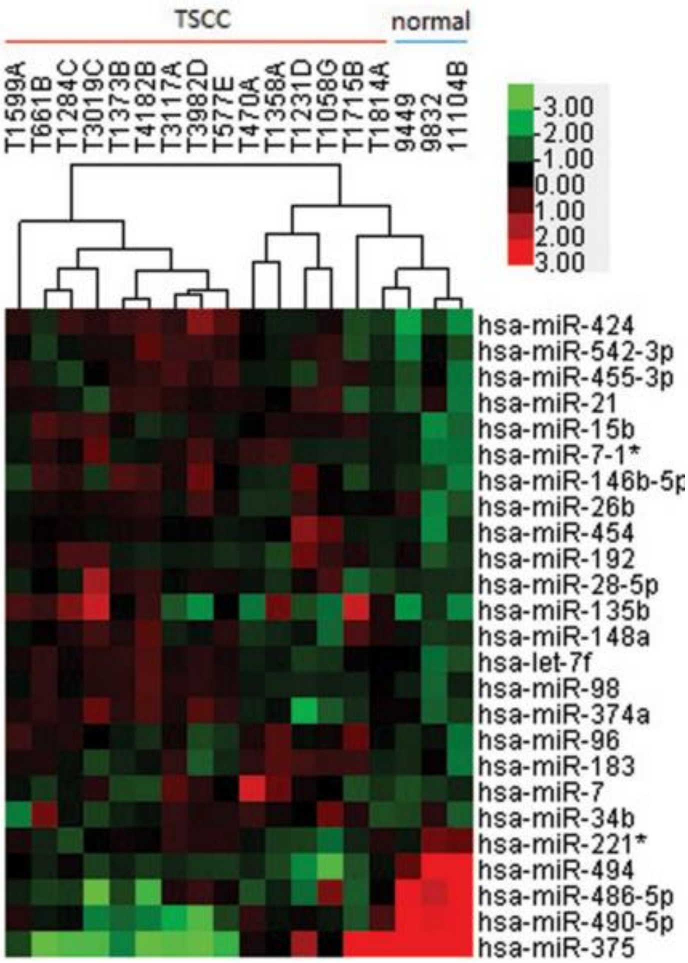
Voici à quoi ressemble un microarray Affymetrix:



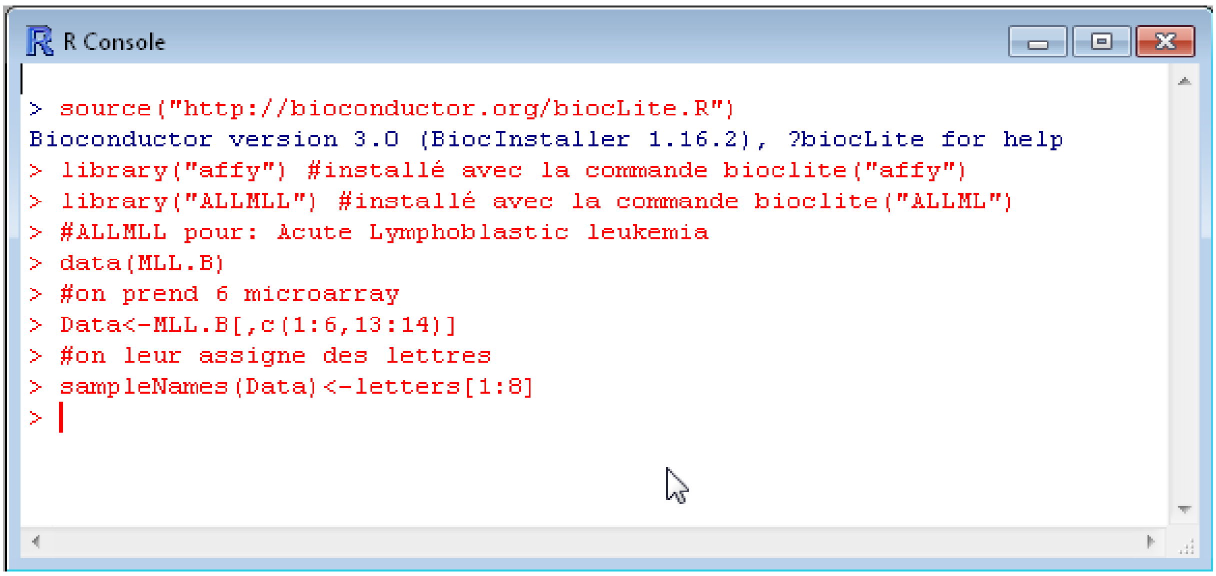
Le principe est schématique le suivant:

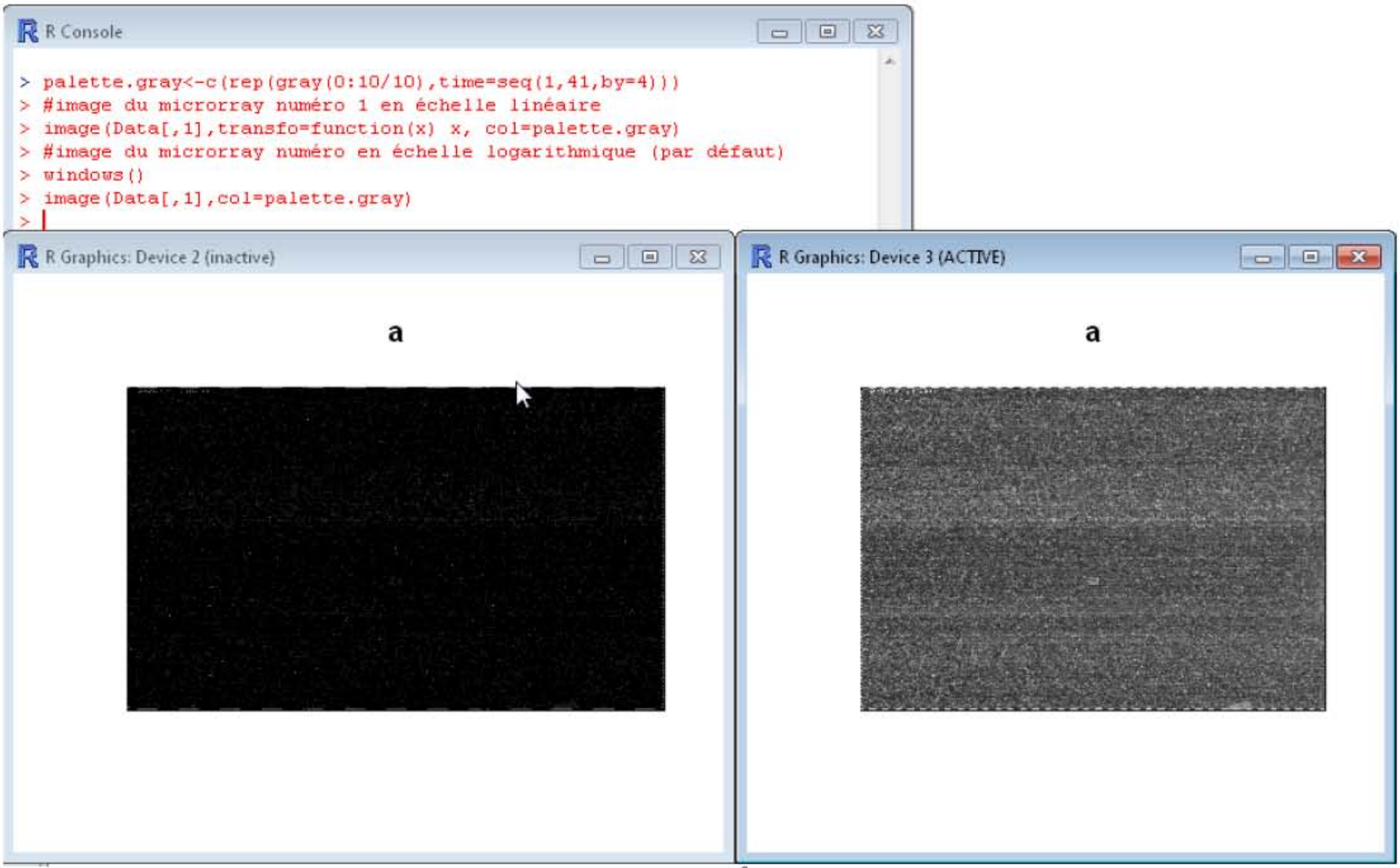


Normalement les colonnes représentent des familles d'organes ou de fonctionnalités biologiques et les lignes des gènes:

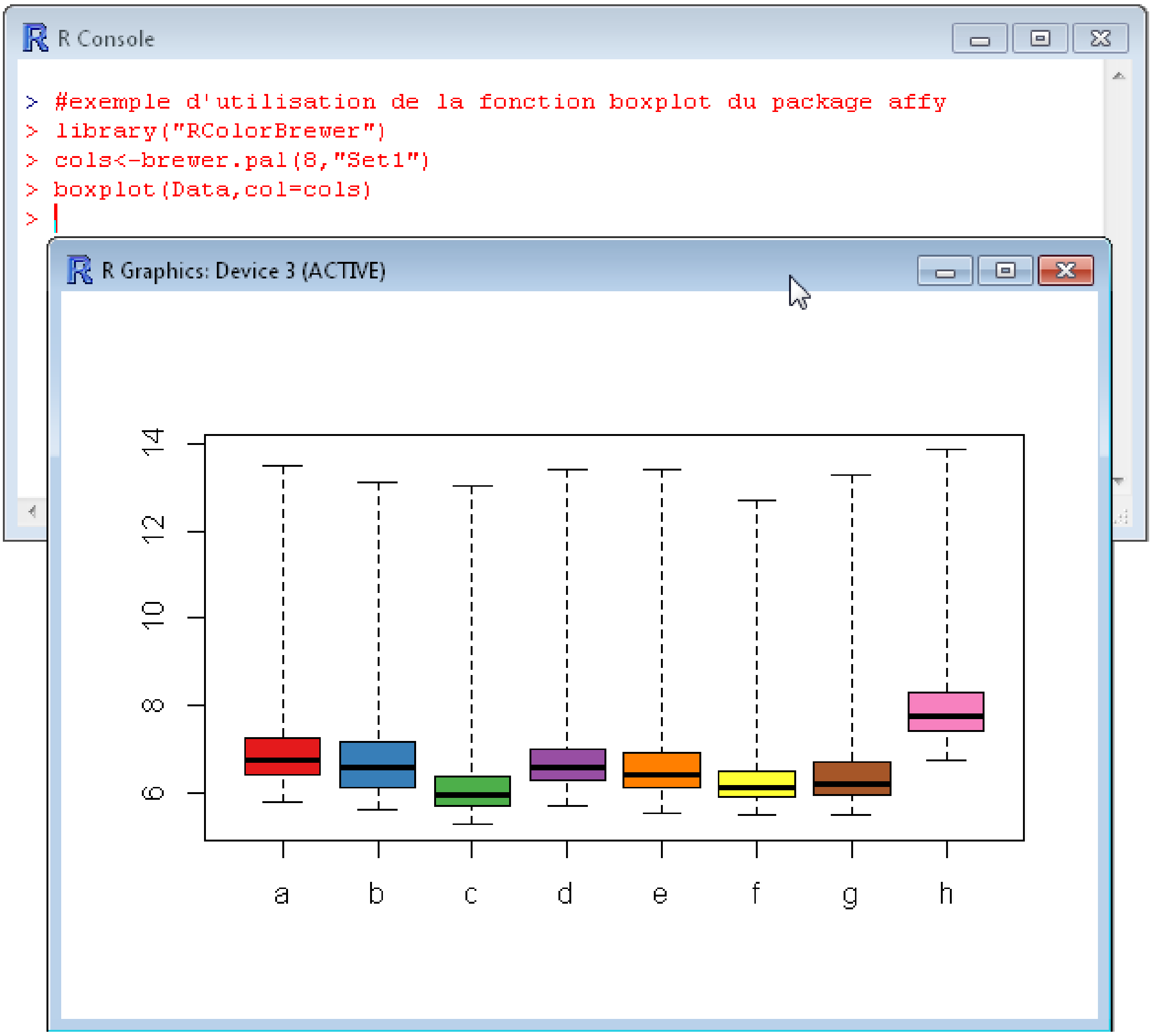


Ce rappel étant fait, passons maintenant à la pratique:

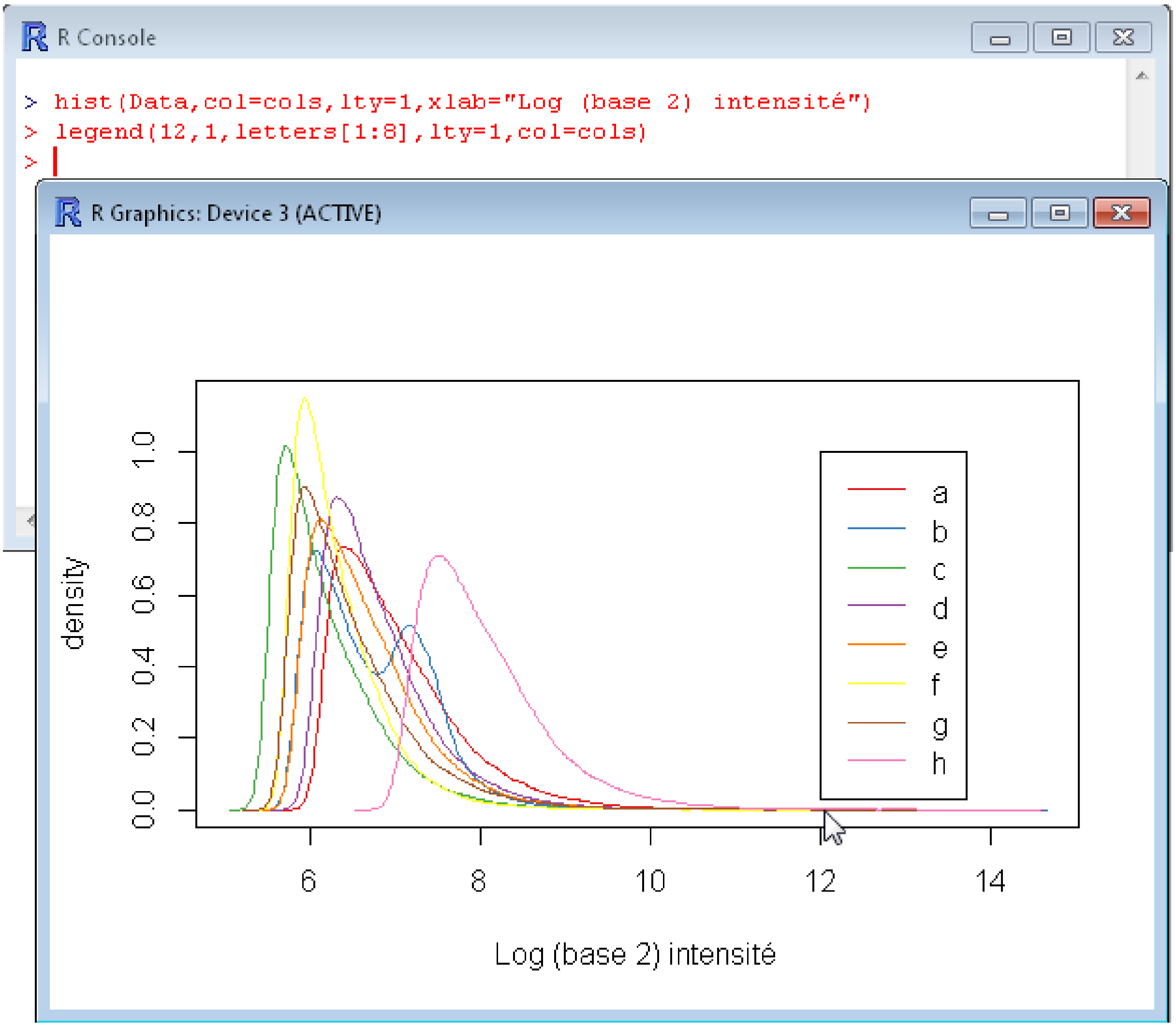



R Console
> palette.gray<-c(rep(gray(0:10/10),time=seq(1,41,by=4)))
> #image du microrray numéro 1 en échelle linéaire
> image(Data[,1],transfo=function(x) x, col=palette.gray)
> #image du microrray numéro en échelle logarithmique (par défaut)
> windows()
> image(Data[,1],col=palette.gray)
R Graphics: Device 2 (inactive)
a
R Graphics: Device 3 (ACTIVE)
a

Nous regardons les distributions d'intensité de nos 8 microarray pour voir s'il y en a qui sont significativement différents:



Il se semble qu'il y ait a priori une anomalie avec le microarray *h*. Nous pouvons investiguer cela un peu mieux avec un histogramme:



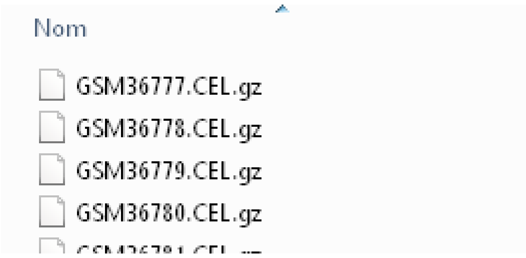
On voit qu'un des microarray a une distribution bimodale ce qui est souvent le signe d'un défaut de conception.

Lire et traiter un fichier Affymetrix *.CEL

Nous allons charger un des fichiers *.CEL de génome humain (homo sapiens). Pour cela le fichier suivante de 1GB:



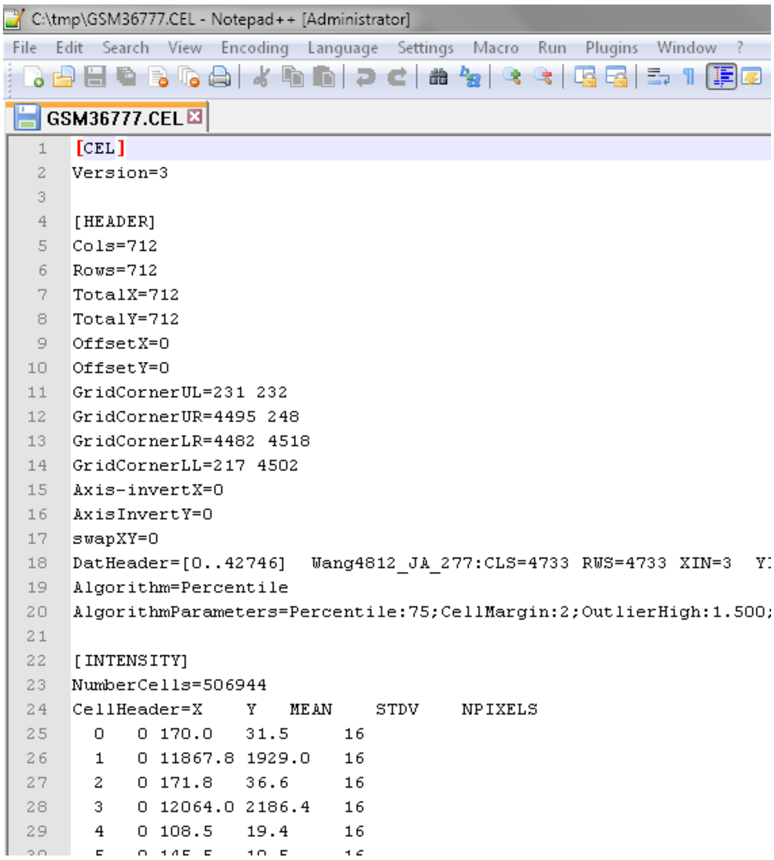
Qui après décompression contient:



Si nous décompressons un de ces fichiers, nous avons alors à chaque fois un fichier d'environ 1MB:



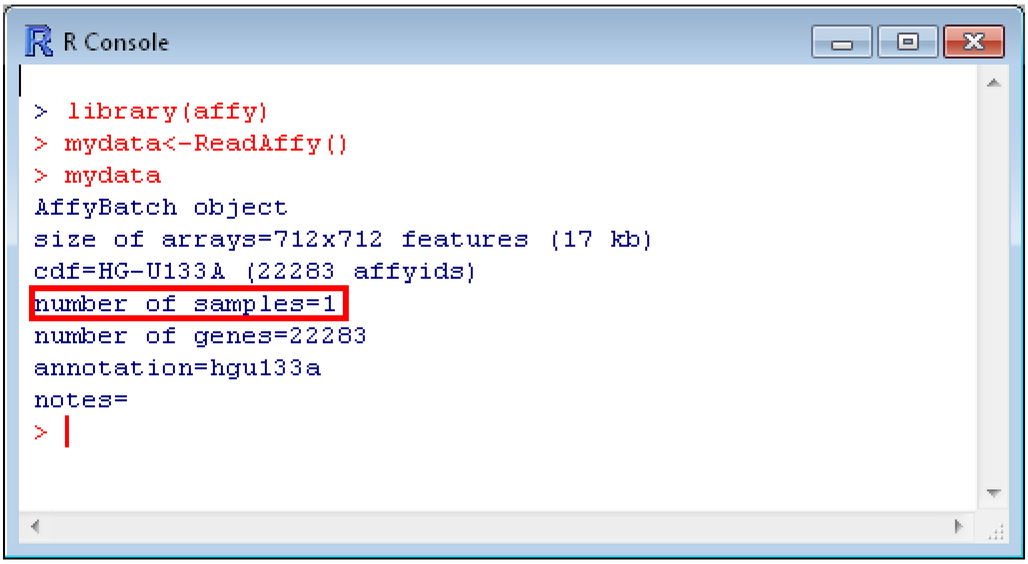
qui contient une fois ouvert avec NotePad++:



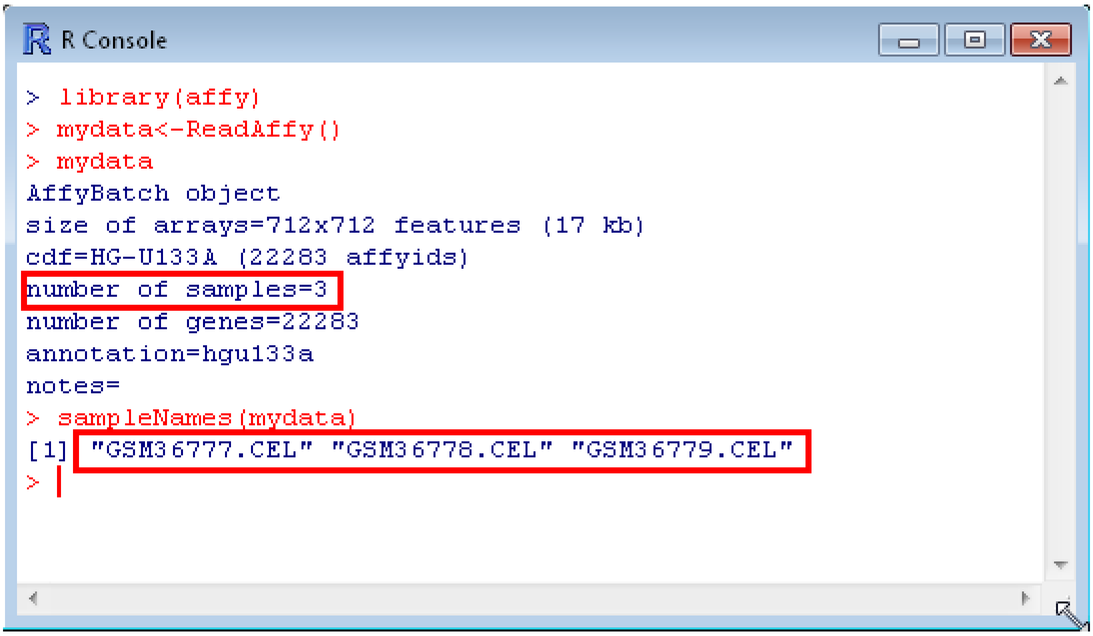
Ensuite, nous utilisons la fonction **ReadAffy( )** :



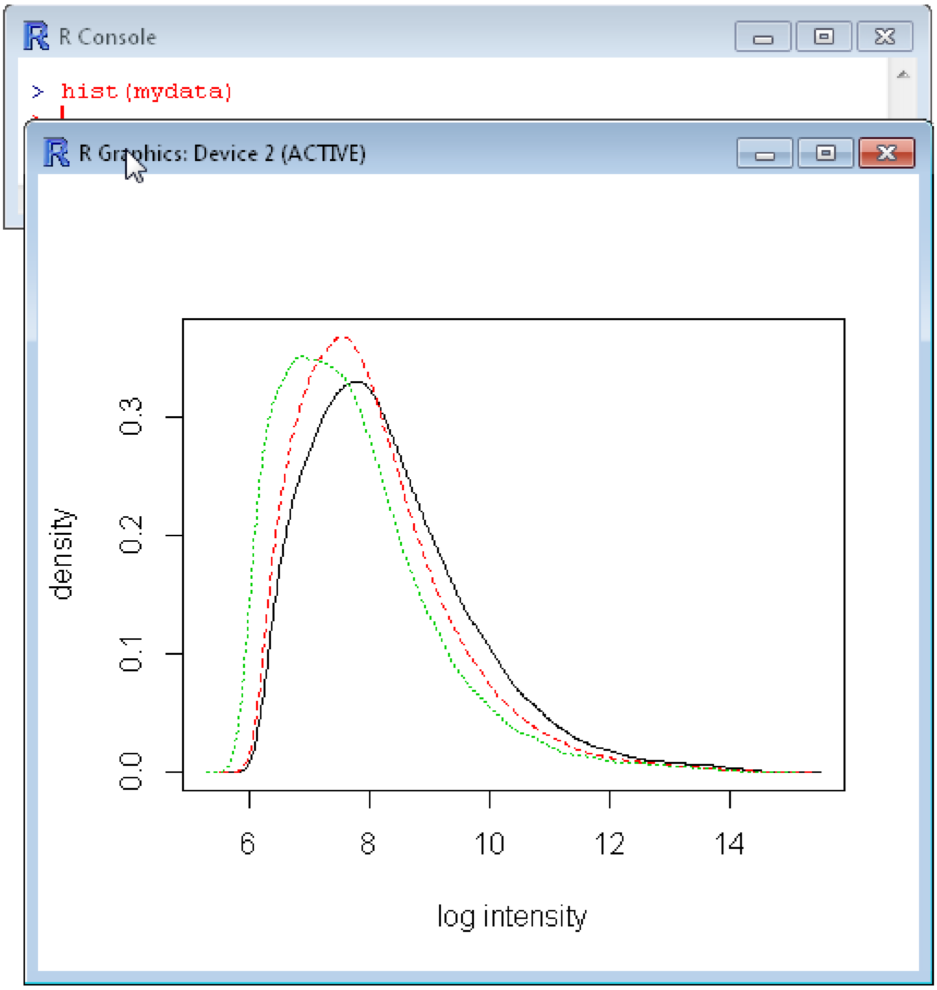
Si il y a un seul fichier *.CEL dans le dossier nous avons:



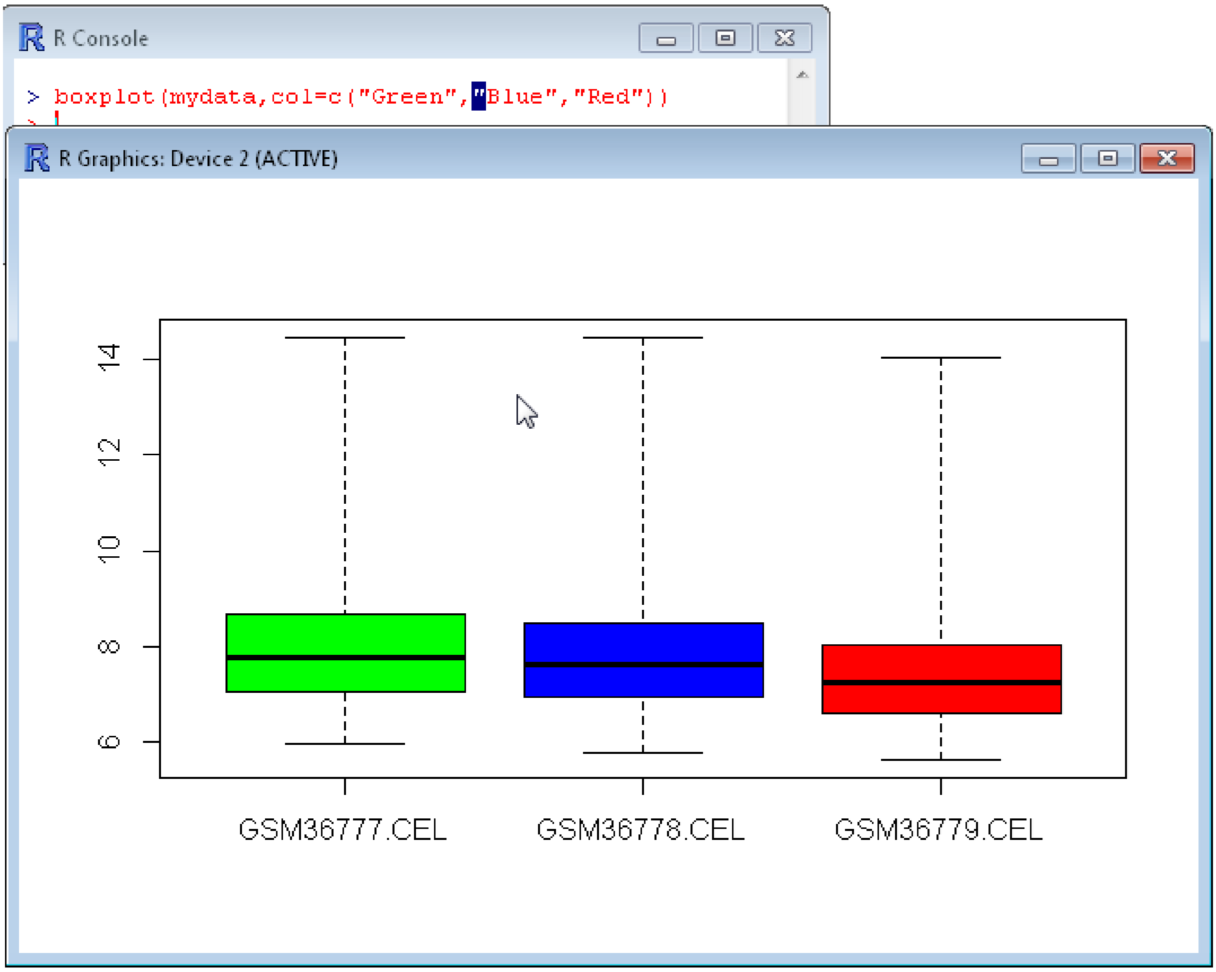
Mais si nous avons 3 fichiers *.CEL alors:



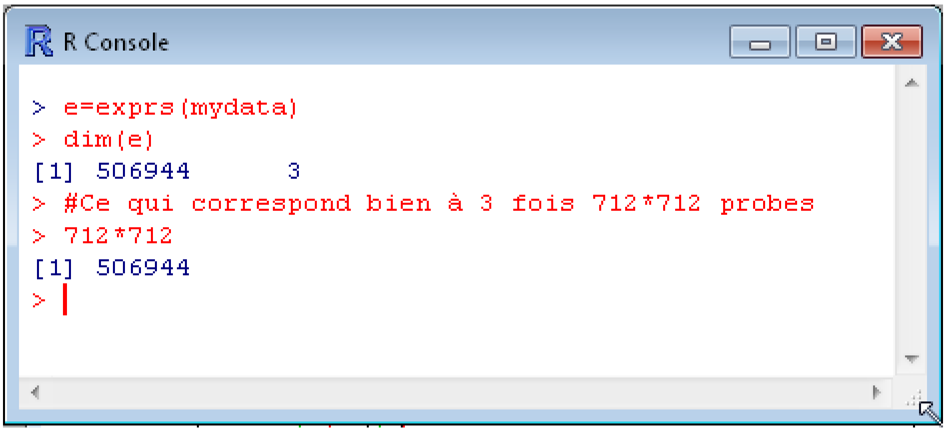
Comme ces fichier *.CEL correspondent respectivement à un unique microarray chacun nous pouvons faire aussi un histogramme de la fluorescence (nous le faisons brut de coffre):



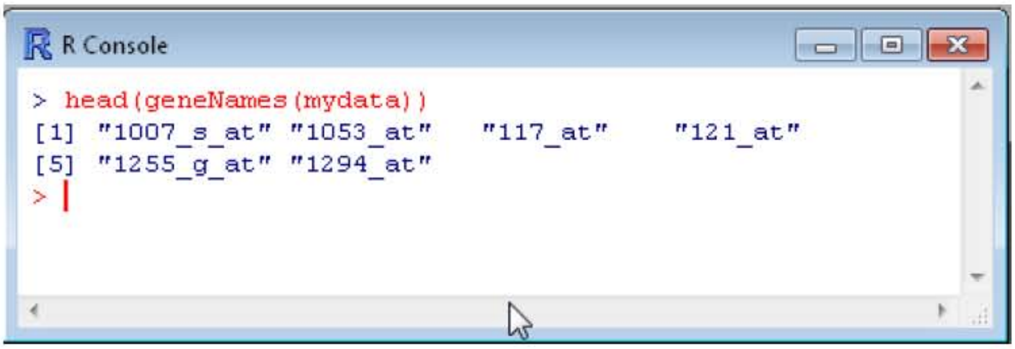
et comme d'habitude nous pouvons faire un boxplot:



Nous pouvons avoir un petit résumé simplifié de notre jeu de fichier *.CEL:



Nous pouvons avoir le nom des gènes avec la commande **geneNames( )**:


R Console
> head(geneNames(mydata))
[1] "1007_s_at" "1053_at" "117_at" "121_at"
[5] "1255_g_at" "1294_at"
>

Exemple 448.: Lire des fichiers *.PDB (Protein Databank)

R 3.2.1

Voyons comment lire des fichiers *.PDB pour différents usages

Représentation 3D d'une protéine avec choix des orbitales (ORTEP)

Nous allons pour représenter cela[48] simplement utiliser les fonctions **read.pdb( )** et **visualize( )** du package **Rpdb**:

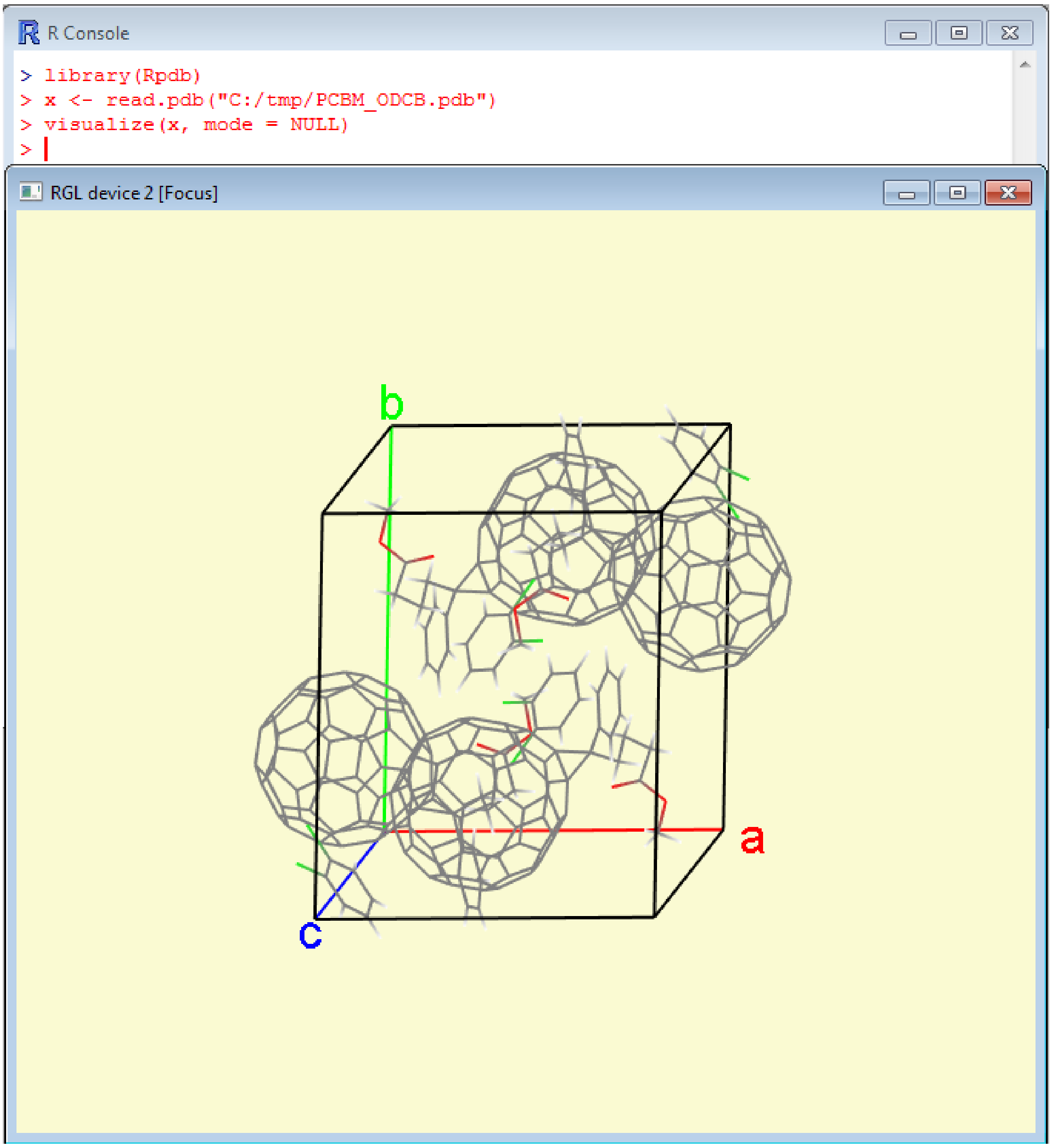


Figure 343 Graph ORTEP (Oak Ridge Thermal Ellipsoid Plot Program) squelette

Ou sous une forme légèrement différente avec le choix des orbitales (s,p,l,d,…):

[48] ORTEP est un acronyme qui signifie *Oak Ridge Thermal Ellipsoid Plot*

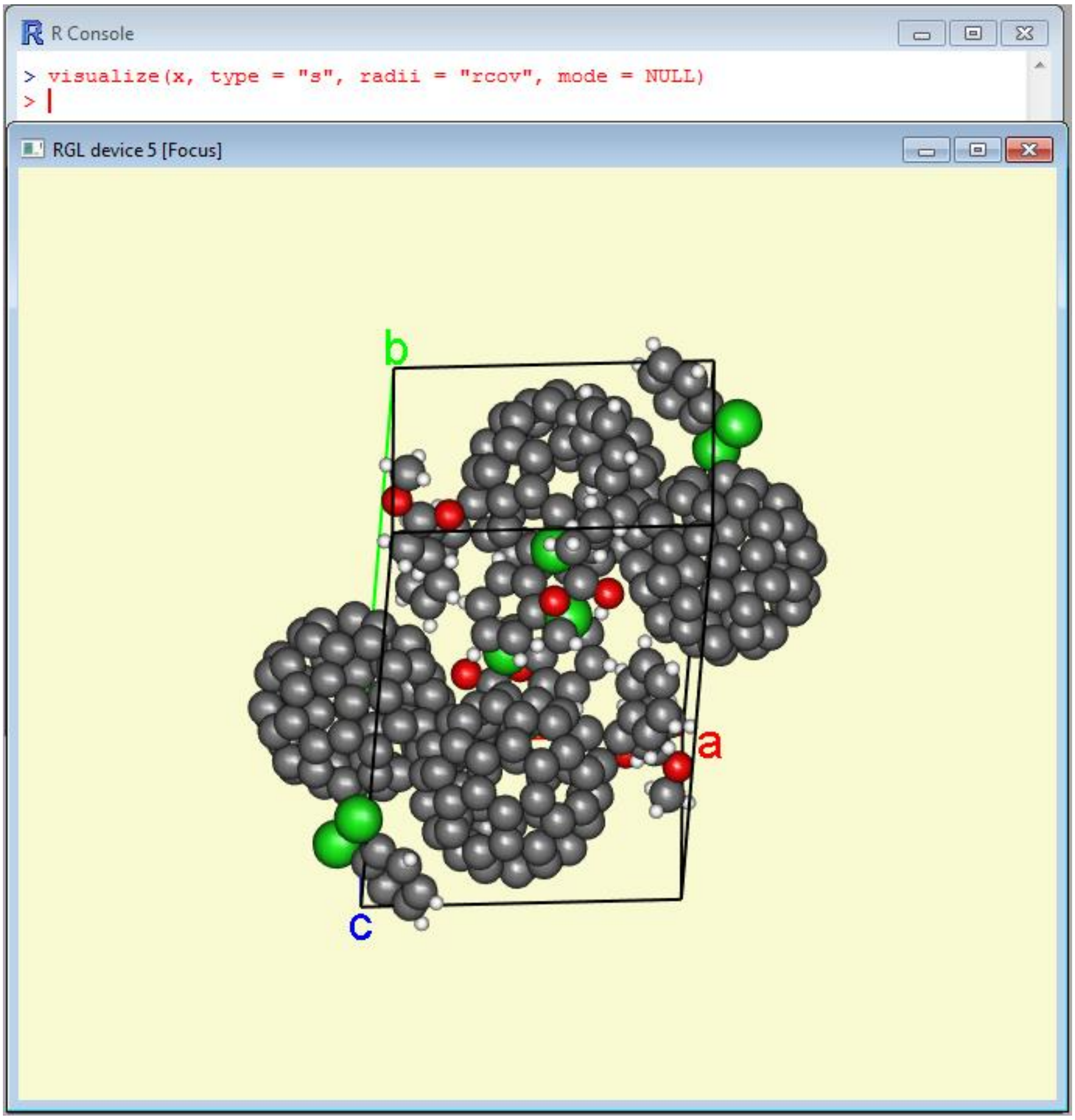


Figure 344 Graph ORTEP (Oak Ridge Thermal Ellipsoid Plot Program) avec orbitales choisies

Diagramme de Ramachandran

Suite à la demande d'une élève (le résultat peut être considérablement amélioré)…

D'abord on charge les packages **Rpdb** et **bio3d** nécessaires pour faire le tout (et **MASS** pour la partie statistique) et à l'aide de la fonction **torsion.pdb( )** on extrait les angles:

```
R Console
> library(Rpdb)
> library(bio3d)
> library(MASS)
> pdb <- read.pdb( "c:/tmp/1ldm.pdb" )
  HEADER    OXIDOREDUCTASE(CHOH(D)-NAD(A))          25-NOV-87   1LDM      1LDM   3
> tor <- torsion.pdb(pdb)
> head(tor$tbl)
           phi       psi        chi1       chi2       chi3       chi4 chi5
[1,]        NA 117.31852          NA         NA         NA         NA   NA
[2,] -159.93011 176.59872 -125.51051         NA         NA         NA   NA
[3,]  -62.86952 -60.95588 -141.52729 -163.21317         NA         NA   NA
[4,]  -51.62781 -42.28400 -142.72965 -139.98539 -153.8801 171.98798   NA
[5,]  -66.29308 -33.59464  -79.31437  -13.53205         NA         NA   NA
[6,]  -75.80475 -28.04824  -41.26972  175.81591  162.1303  66.26194   NA
> |
```

Une fois ceci fait nous combinons un ensemble de techniques variées utilisant la fonction **smoothScatter( )** pour les couleurs de fonds, la fonction **scatter.smooth( )** pour la régression **LOESS** et enfin **kde2d( )** et **contour( )** du package **MASS** pour les courbes de niveau:

```
R Console
> smoothScatter(na.omit(tor$phi), na.omit(tor$psi),xlab="phi",ylab="psi",pch = 1,bandwidth = 30,
+ colramp = colorRampPalette(c("white", "orange", "red")),main="Ramachandran plot")
> par(new=T)
> scatter.smooth(na.omit(tor$phi), na.omit(tor$psi),xlab="",ylab="")
> par(new=T)
> edl<-kde2d(na.omit(tor$phi), na.omit(tor$psi),n=100)
> contour(edl,main="Ramachandran plot", lwd=2)
> |
```

Ce qui donne (voir page suivante):

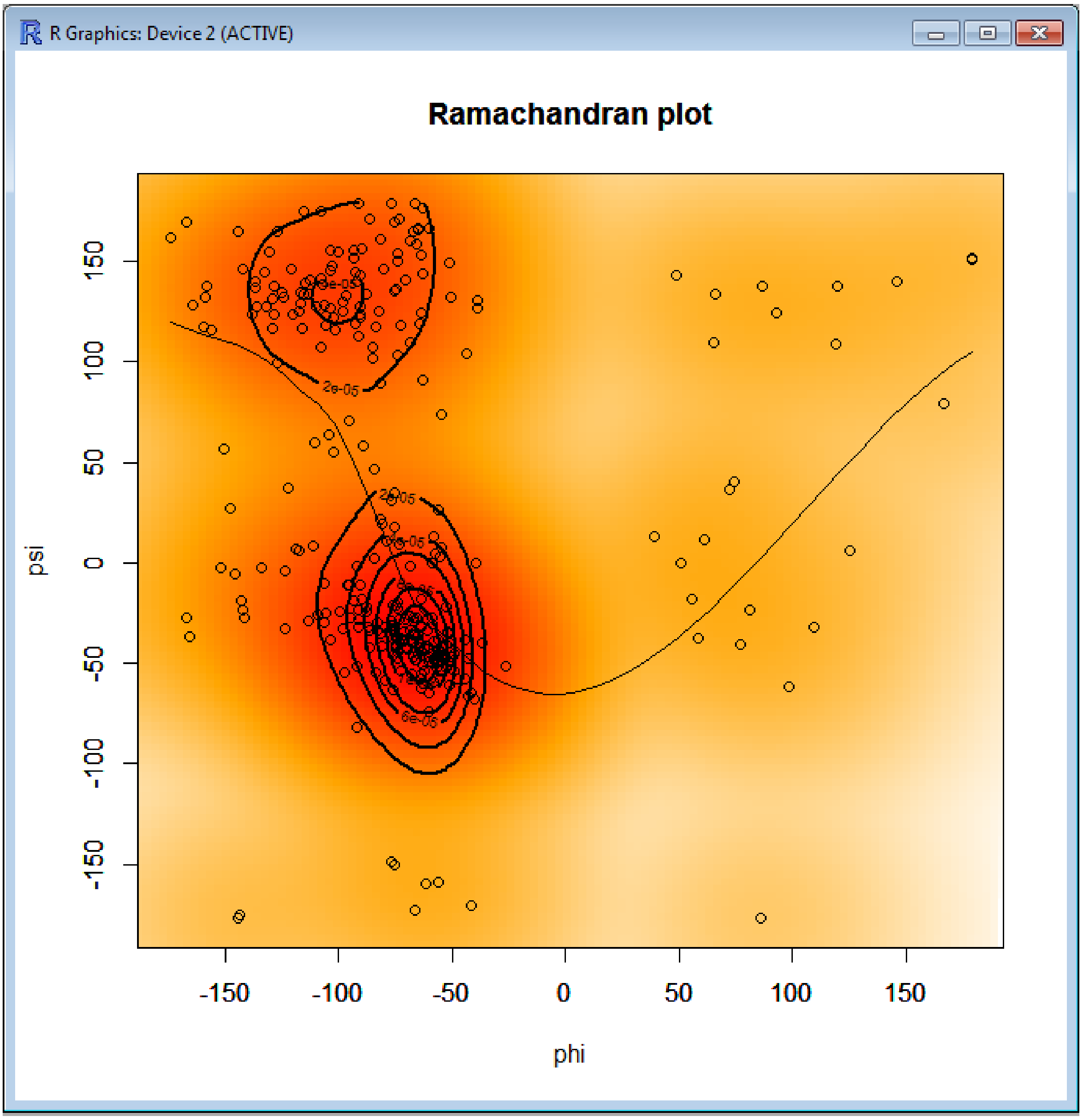


Figure 345 Graph de Ramachandran

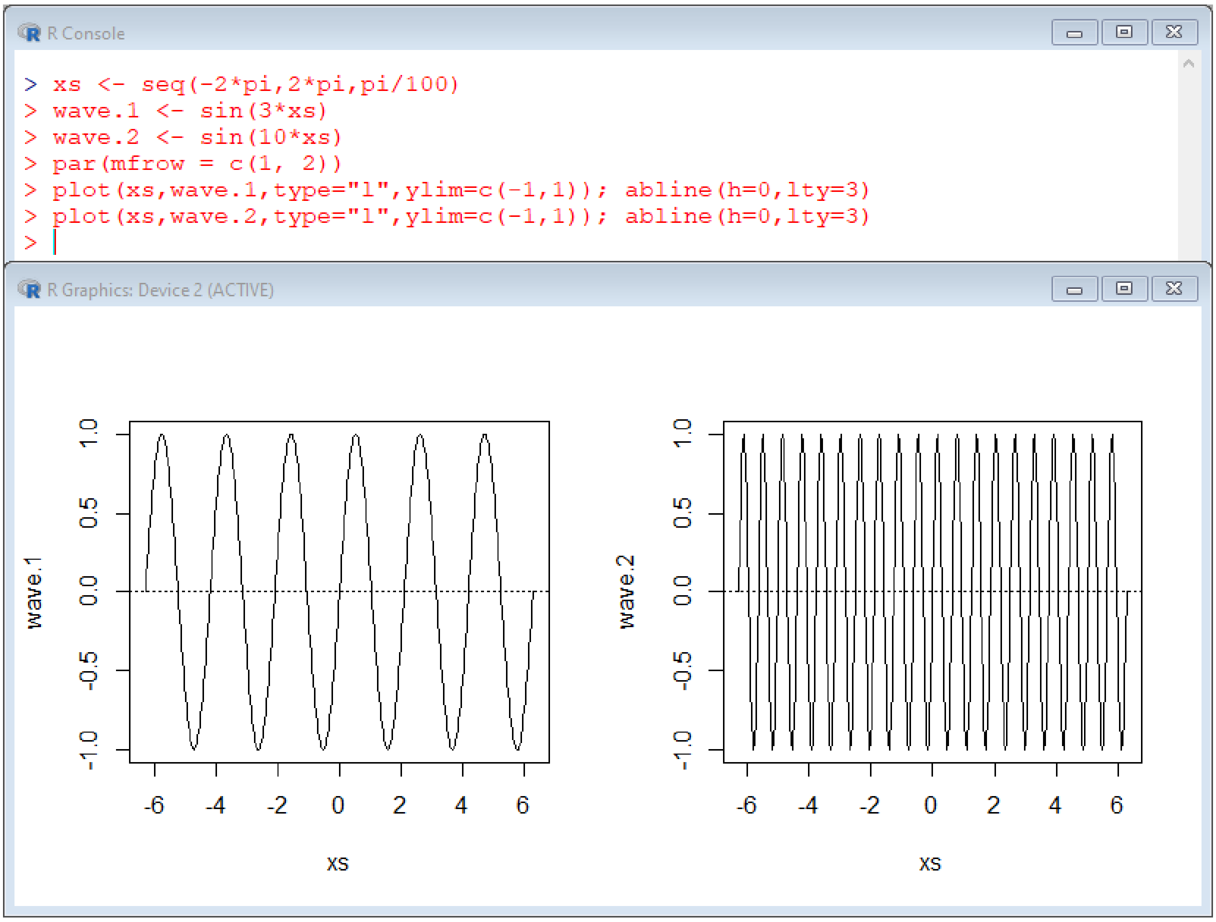
33. Traitement du Signal

Le but de ce chapitre va être d'introduire les packages **signal** et **rwave** de R dans le domaine du traitement du signal en substitut de MATLAB, Octave ou autre…

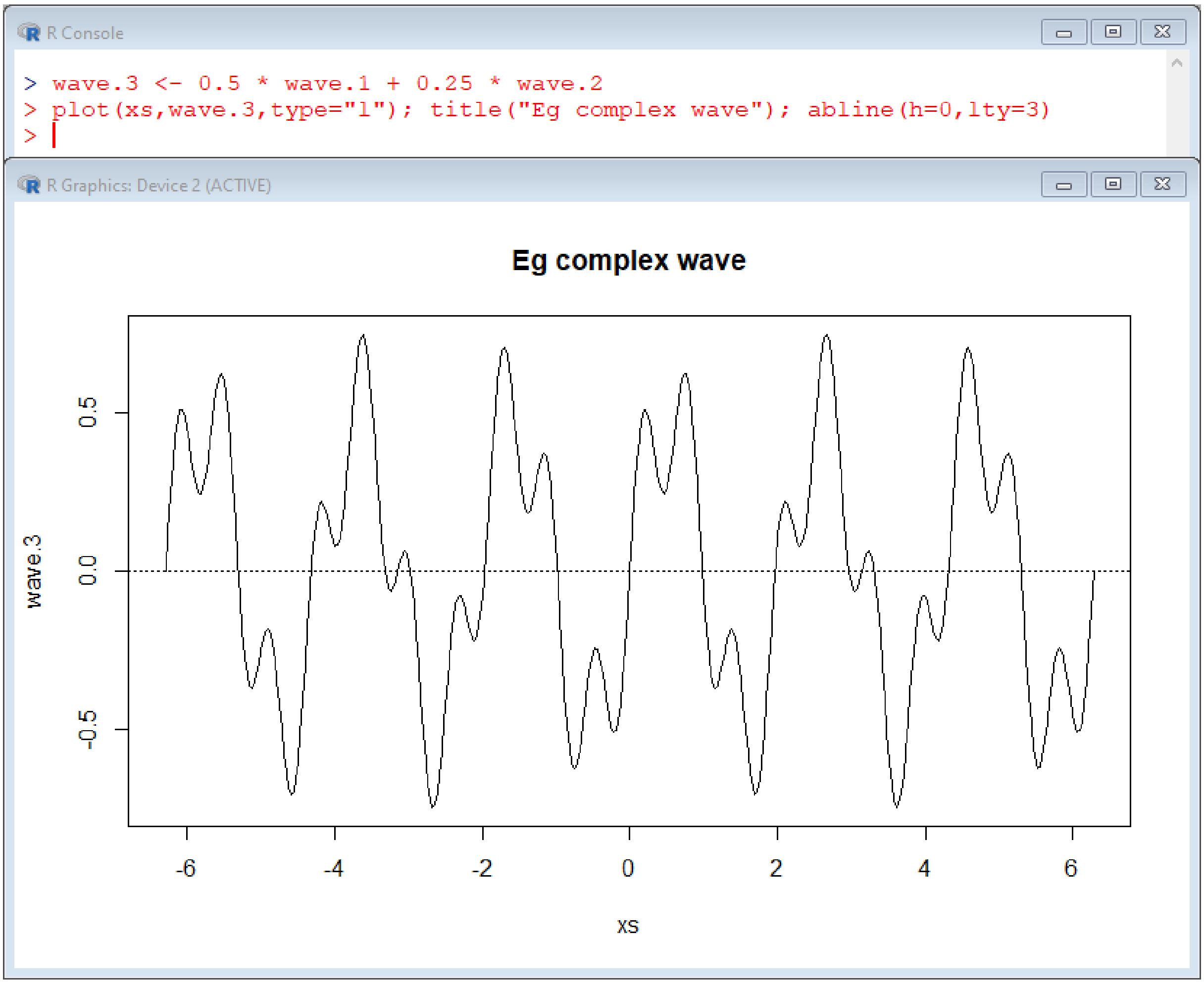
Exemple 449.: Génération de signaux périodiques

R 3.6.3

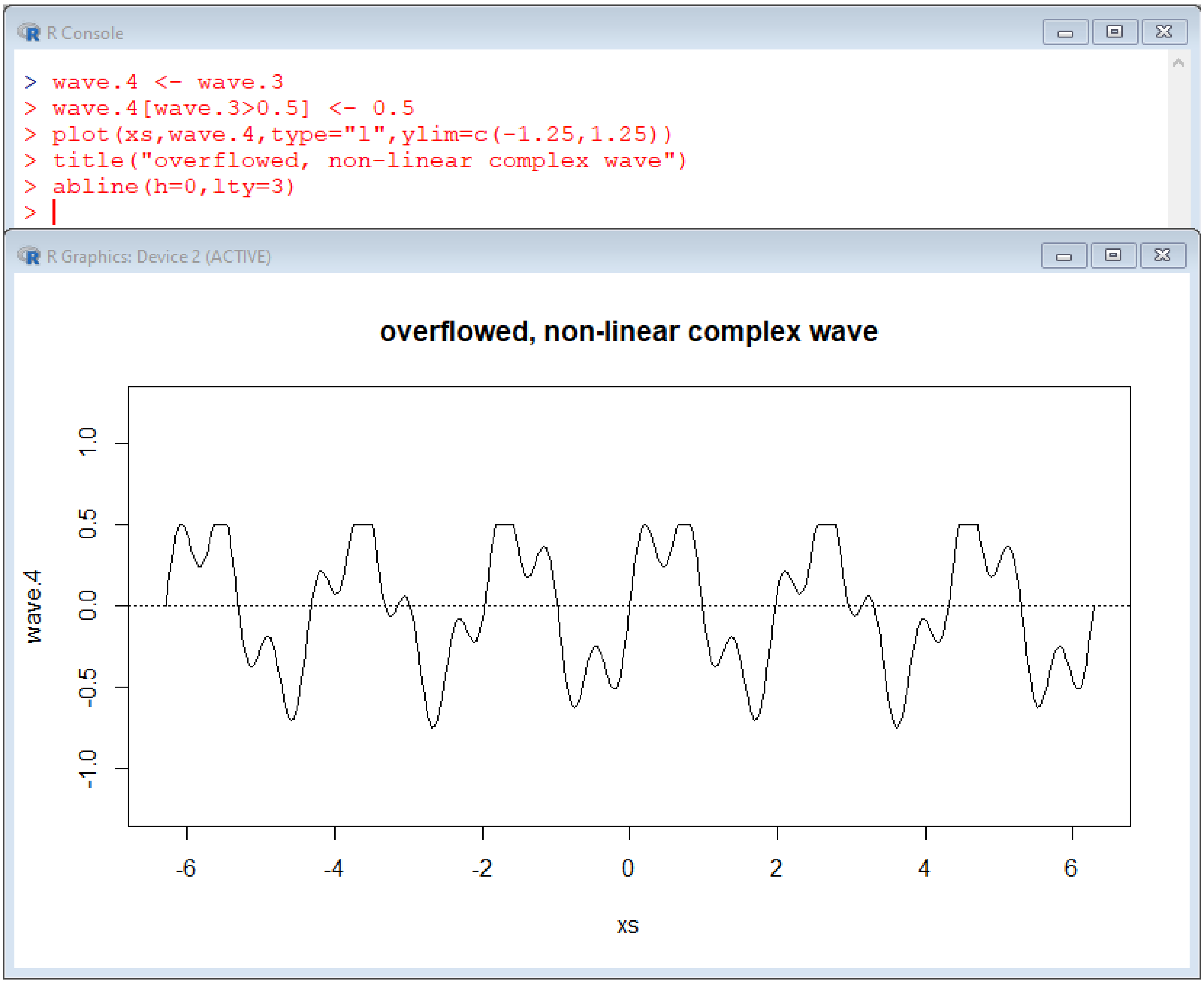
Commençons par générer deux signaux périodiques simples:



qui peut être combinée linéairement en une onde un peu plus complexe:

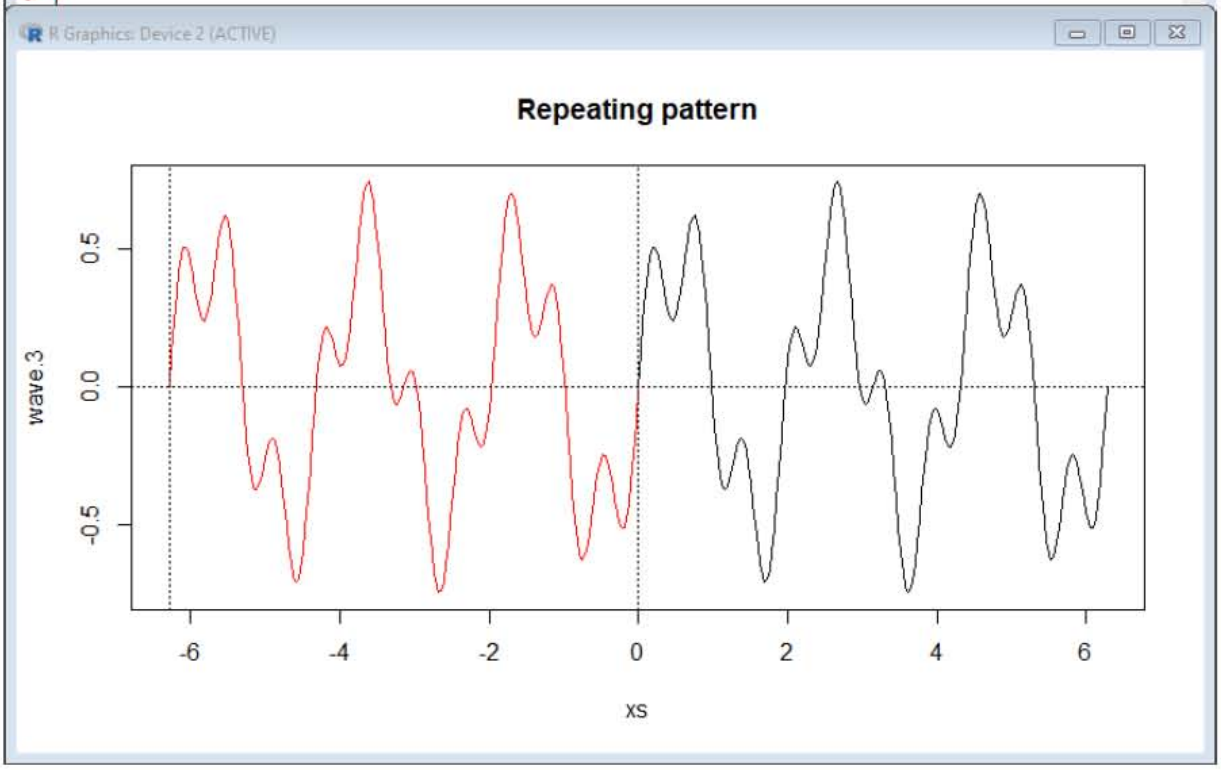


Joseph Fourier a montré que toute onde périodique peut être représentée par une somme d'ondes sinusoïdales simples. Cette somme est appelée la série de Fourier comme nous le savons. La série Fourier ne tient que lorsque le système est linéaire. S'il y a, par exemple, un certain effet de débordement (un seuil où la sortie reste la même quelle que soit la quantité d'entrée donnée), un effet non linéaire entre dans l'image, brisant l'onde sinusoïdale et le principe de superposition:



On peut aussi s'amuser à construire un signal typique d'une somme de Fourier:

```
> repeat.xs<-seq(-2*pi,0,pi/100)
> wave.3.repeat<-0.5*sin(3*repeat.xs) + 0.25*sin(10*repeat.xs)
> plot(xs,wave.3,type="l");
> title("Repeating pattern")
> points(repeat.xs,wave.3.repeat,type="l",col="red");
> abline(h=0,v=c(-2*pi,0),lty=3)
>
```



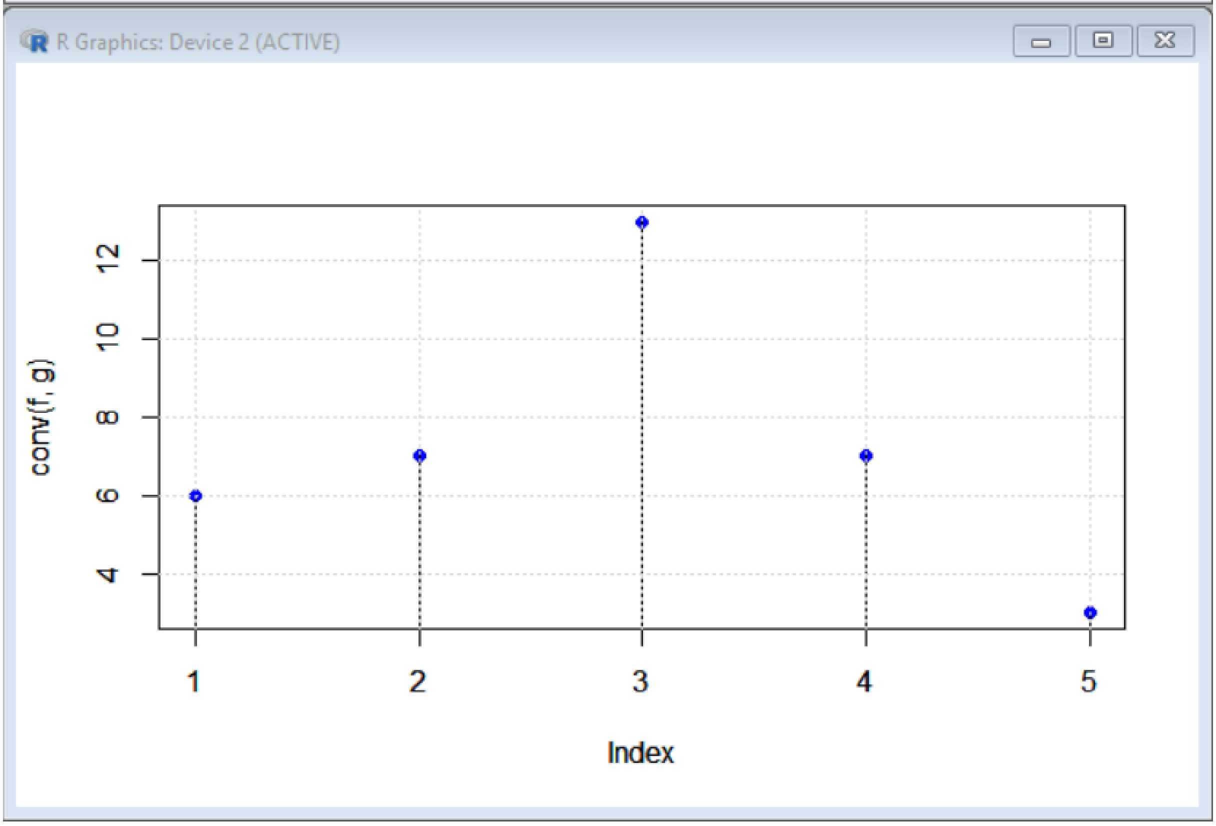
Exemple 450.: Convolution discrète

R 3.4.2

Nous commençons avec une convolution discrète linéaire simple histoire de reproduire celle étudiée dans le cours théorique en utilisant la fonction **conv( )** du package **signal**:

R Console

```
> library("signal")
> f<-c(3,2,1)
> g<-c(2,1,3)
> conv(f,g)
[1]  6  7 13  7  3
> plot(conv(f,g),col = "blue", pch = 16)
> par(new=TRUE)
> plot(conv(f,g), type="h")
> grid()
> 
```



On retombe bien sur le résultat du cours théorique!

Il y a bien évidemment à traiter dans les grands classiques, la convolution discrète de deux gaussiennes en utilisant toujours la fonction **conv( )** du package **signal**:

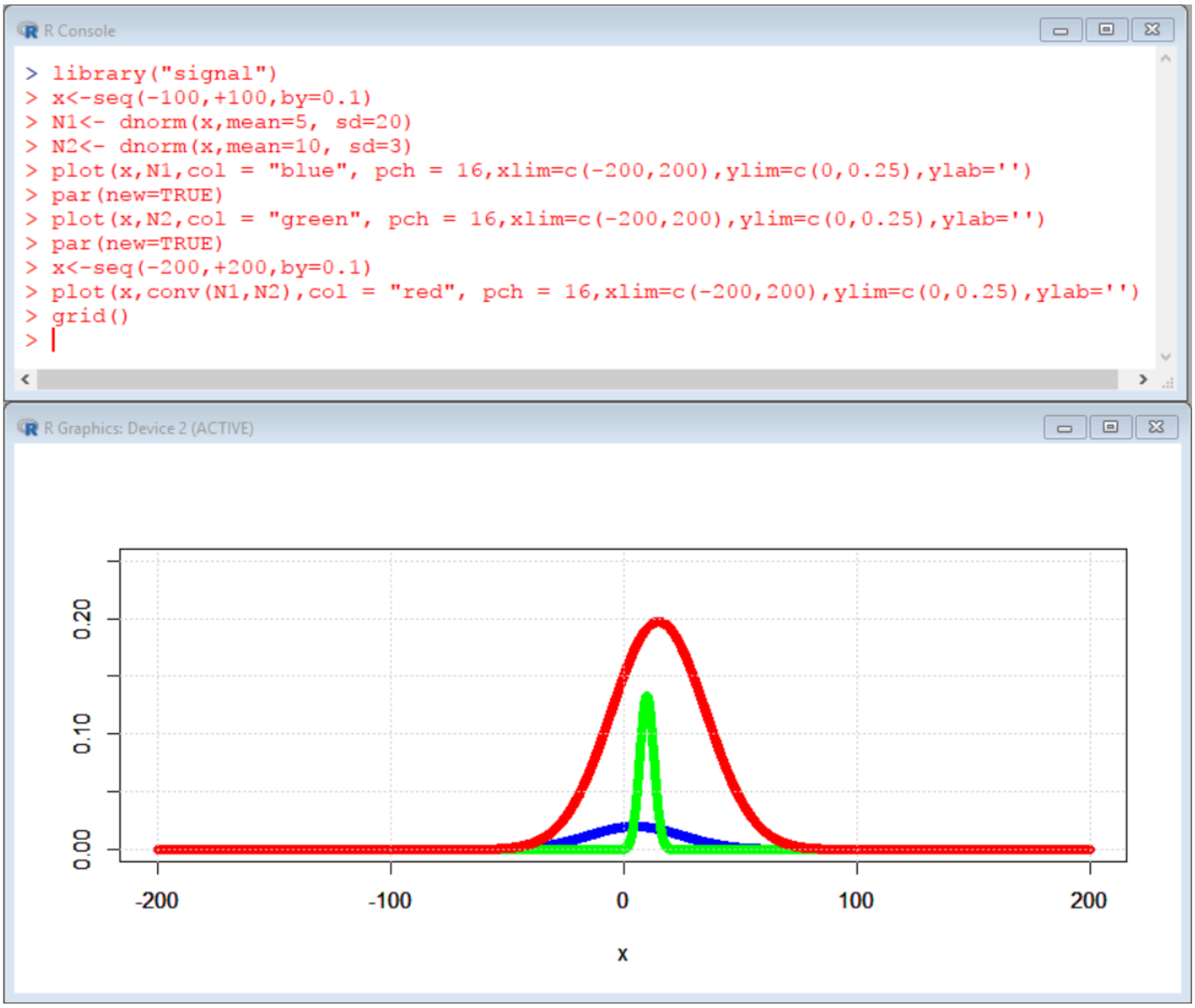
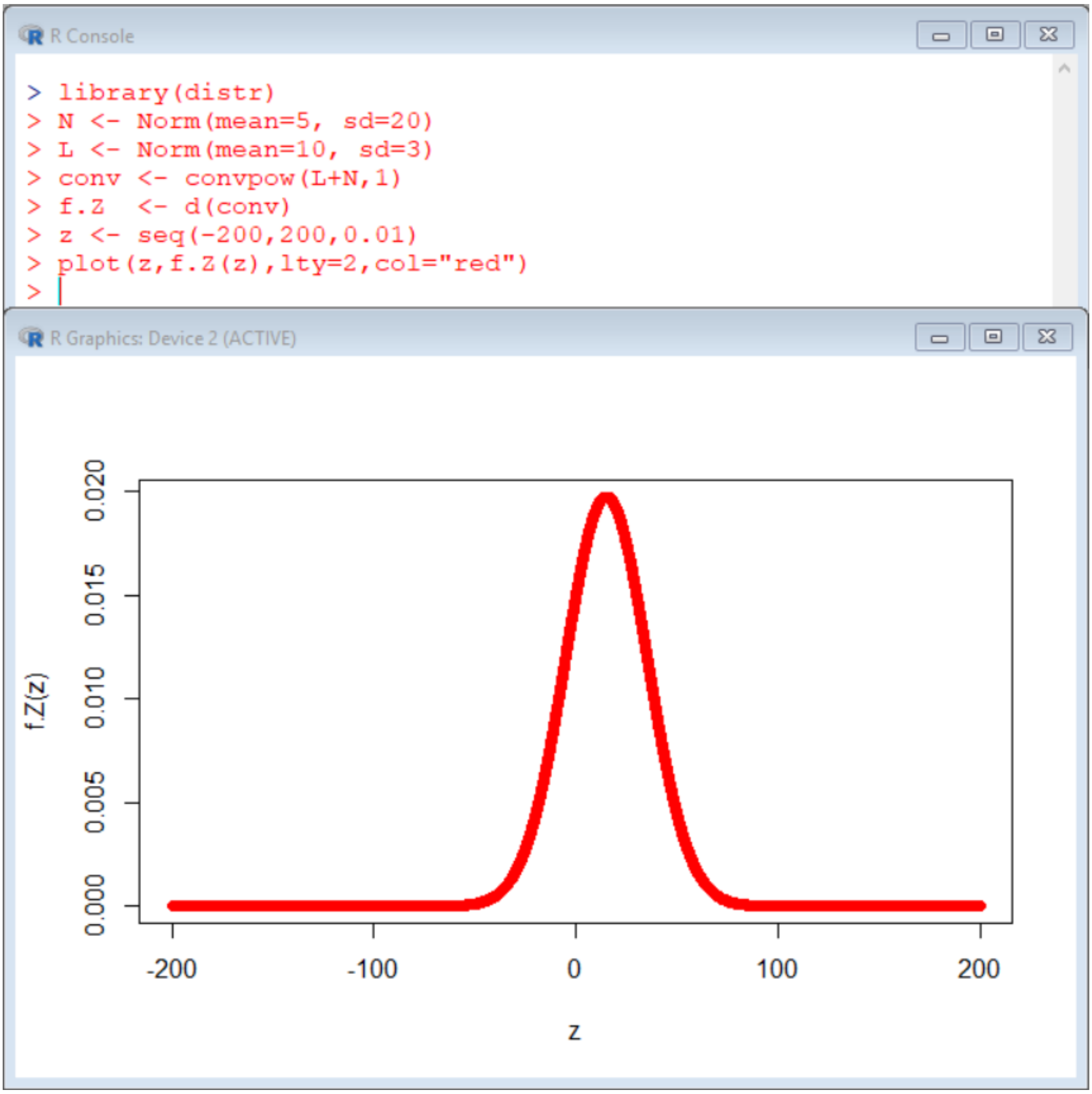


Figure 346 Graph convolution de signaux

Donc là encore, on tombe sur le résultat attendu en conformité avec le cours théorique!

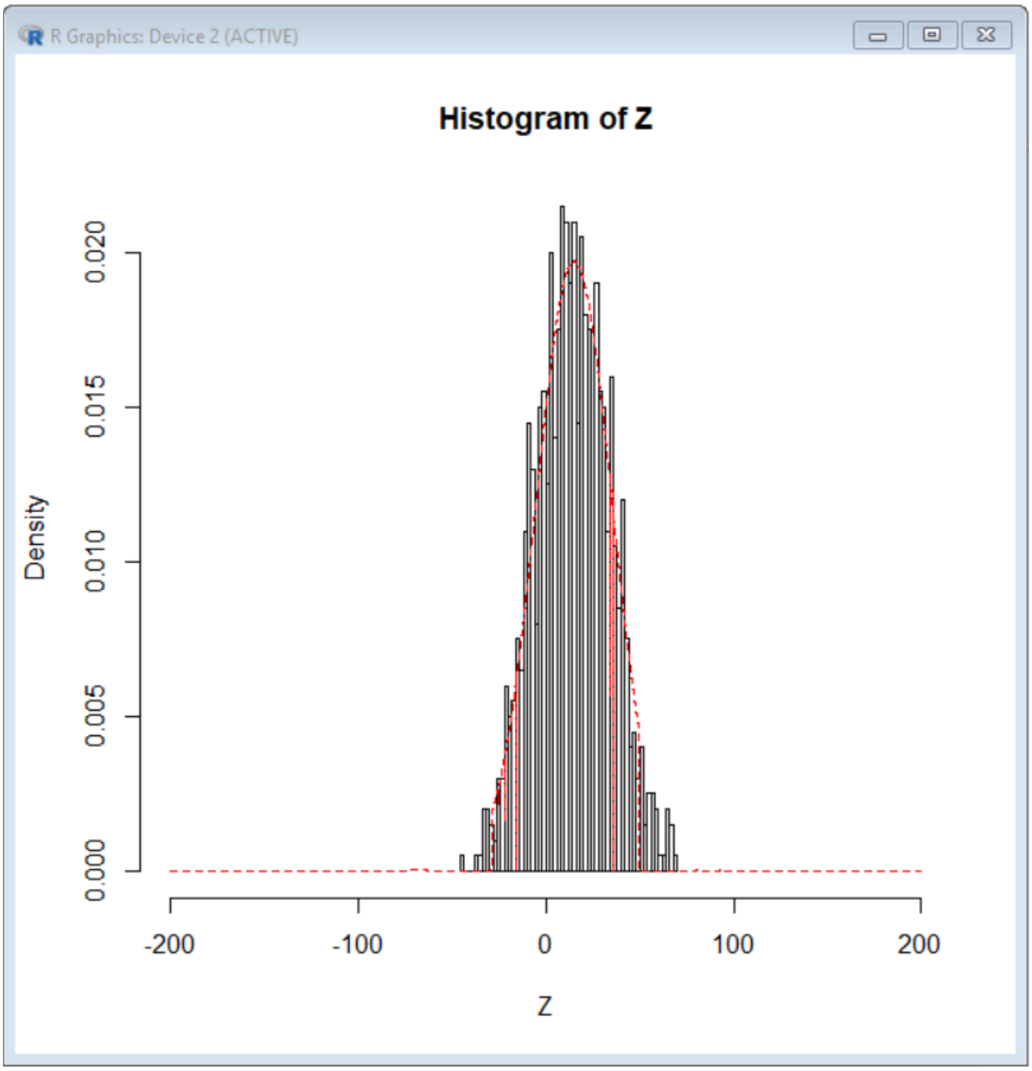
Une autre approche beaucoup plus courte consiste à utiliser les fonctions **convpow( )** et **d( )** du package **distr**:



Ou mon approche préférée car elle met en évidence l'intégrale de convolution et n'utilise aucun package:

```
> #On fait la convolution sous forme d'intégrale numérique
> f.X <- function(x) dnorm(x,5,20)
> f.Y <- function(y) dnorm(y,10,3)
> # intégrale de convolution explicite
> f.Z <- function(z) integrate(function(x,z) f.Y(z-x)*f.X(x),-Inf,Inf,z)$value
> f.Z <- Vectorize(f.Z)
>
> #On le fait sous forme probabiliste
> X <- rnorm(1000,5,20)
> Y <- rnorm(1000,10,3)
> Z <- X + Y
>
> #On compare les méthodes
> hist(Z,freq=F,breaks=50, xlim=c(-200,200))
> z <- seq(-200,200,0.01)
> lines(z,f.Z(z),lty=2,col="red")
>
```

Ce qui donne:



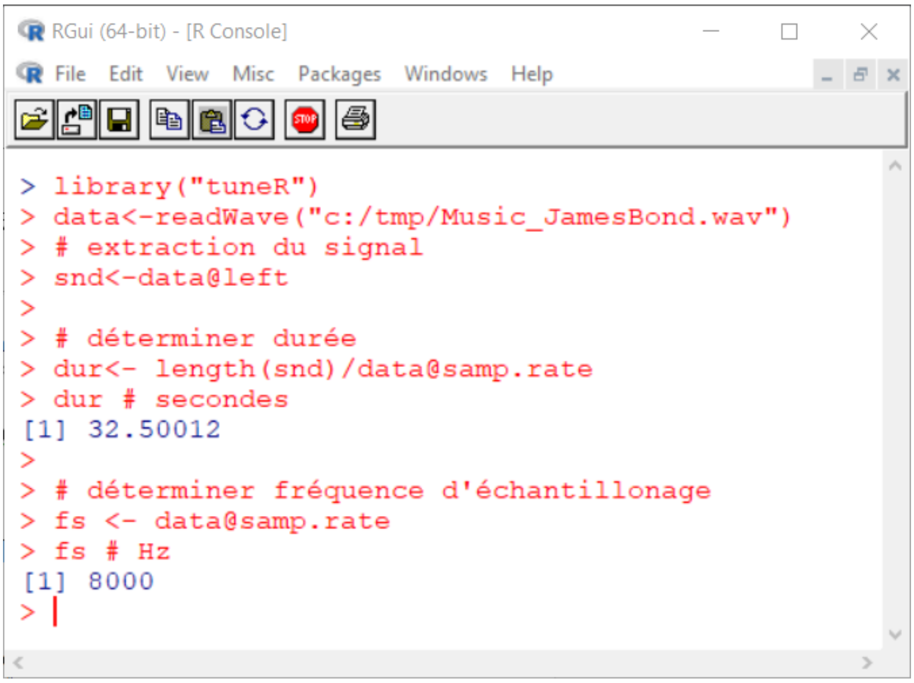
Enfin le dernier cas d'application majeur est la convolution utilisée comme fenêtre de filtrage en finance. Considérons

Exemple 451.: Spectrogrammes

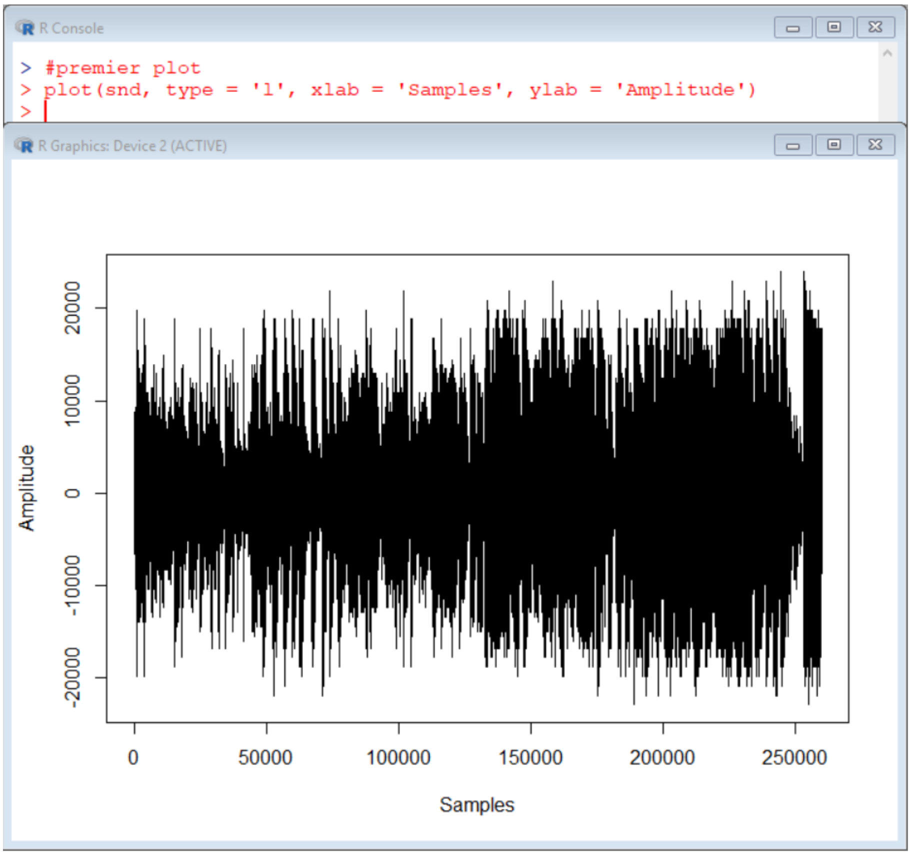
R 3.5.1

Nous allons voici ici comment afficher différents spectrogrammes d'un signal provenant (exemple particulier!) d'un fichier son.

Pour cela, nous utiliser le package **tuneR** et sa fonction **readWave( )** pour extraire quelques informations du fichier wave importé:



Avant de voir les spectrogrammes, regardons à quoi ressemblent les 32 secondes du canal gauche simplement dans un premier temps:



Maintenant commençons par un premier spectrogramme avec la fonction **spectrogram( )** du package **phonTools**:

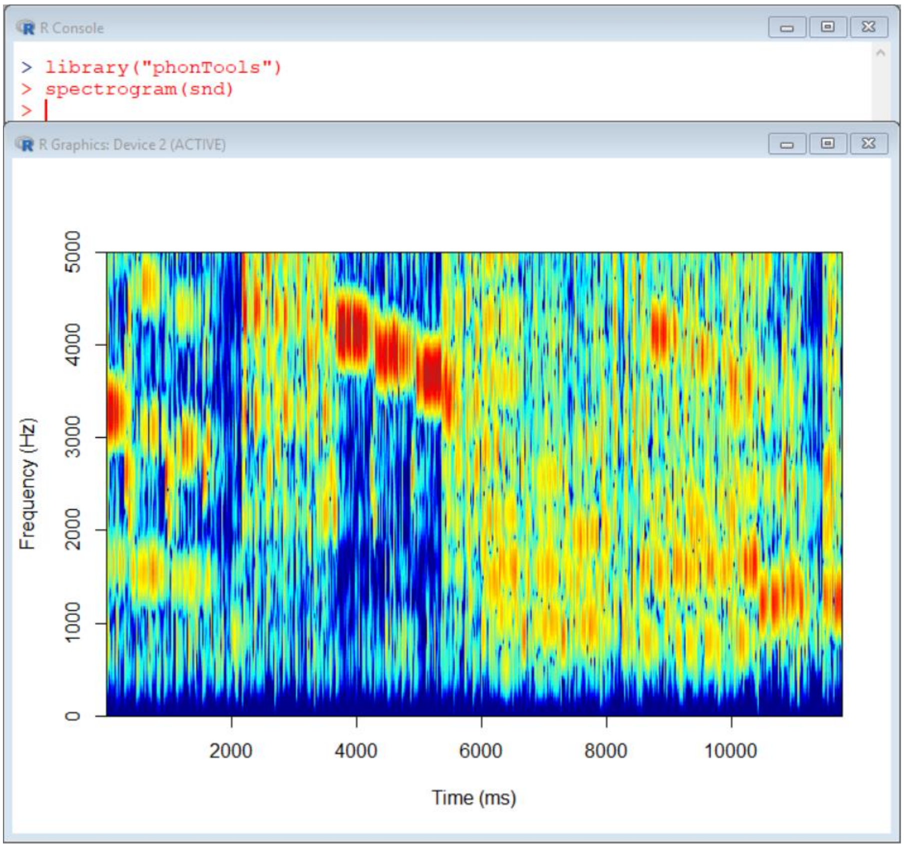


Figure 347 Graph spectrogramme de signal 1/3

Voyons maintenant un autre spectrogramme avec la fonction **spectro( )** du package **seewave**:

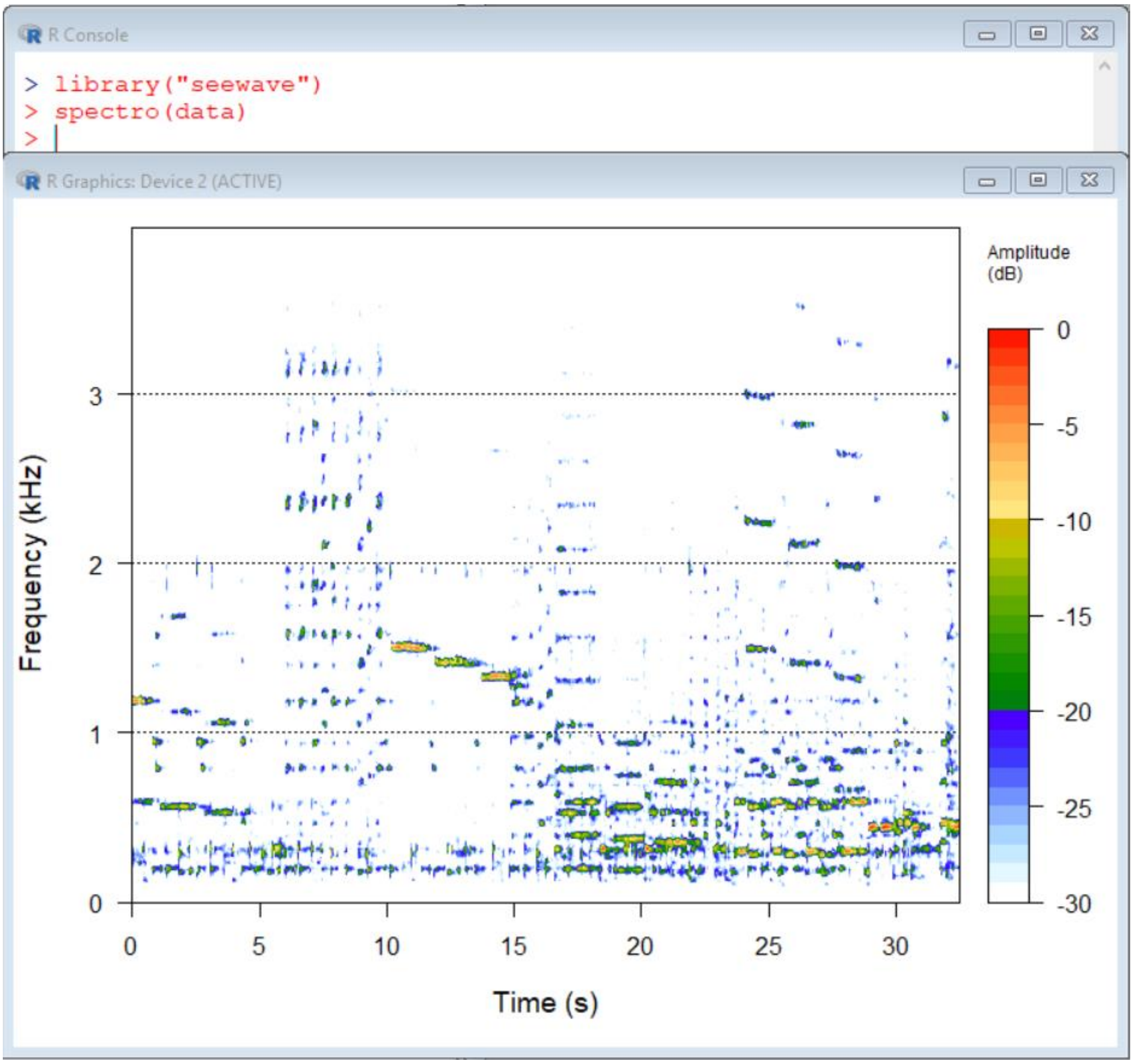


Figure 348 Graph spectrogramme de signal 2/3

et un troisième spectrogramme avec le package **ggplot2** et sa fonction **ggspectro( )** :

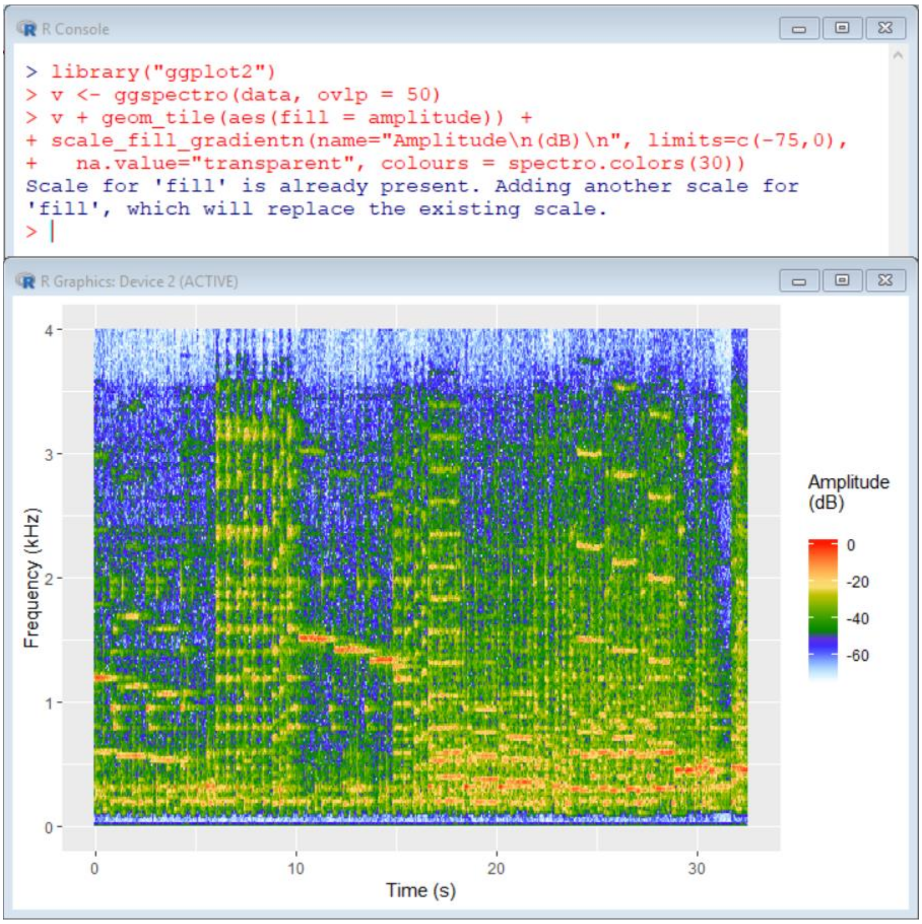
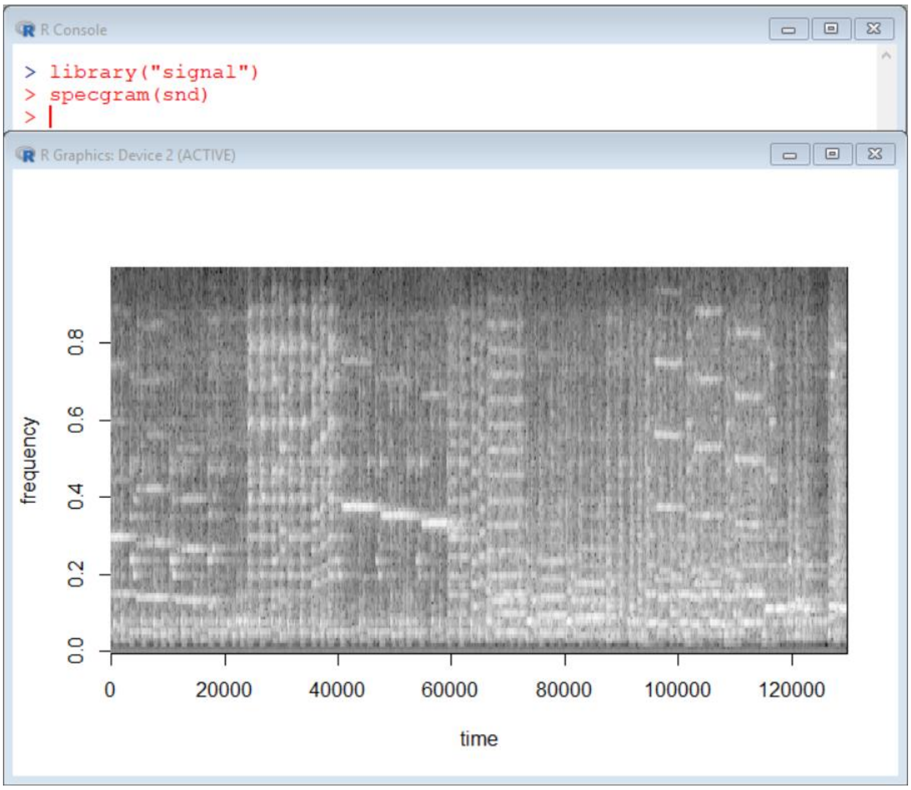


Figure 349 Graph spectrogramme de signal 3/3 (ggplot)

et un quatrième spectrogramme avec la fonction **specgram( )** du package **signal**:



Et voyons maintenant un dernier spectrogramme avec la fonction **FTGramImage( )** du package **hht**:

```
> library("hht")
> dt <- mean(diff(snd))
>
> ft <- list()
> ft$nfft <- 4096
> ft$ns <- 30
> ft$nov <- 29
>
> time.span <- c(0, 30)
> freq.span <- c(0, 30)
> amp.span <- c(0, max(snd))
> FTGramImage(snd, dt, ft, time.span = time.span, freq.span = freq.span,
+     amp.span = amp.span, pretty = TRUE, img.x.format = "%.1f",
+     img.y.format = "%.0f",
+     main = "James Bond - Fourier Spectrogram")
Adjusting Time and Frequency limits to nice looking numbers (the "pretty" option is currently set to TRUE)
Warning message:
In FTGramImage(snd, dt, ft, time.span = time.span, freq.span = freq.span,  :
  Requested maximum frequency is higher than the Nyquist frequency.
> 
```

Ce qui donne:

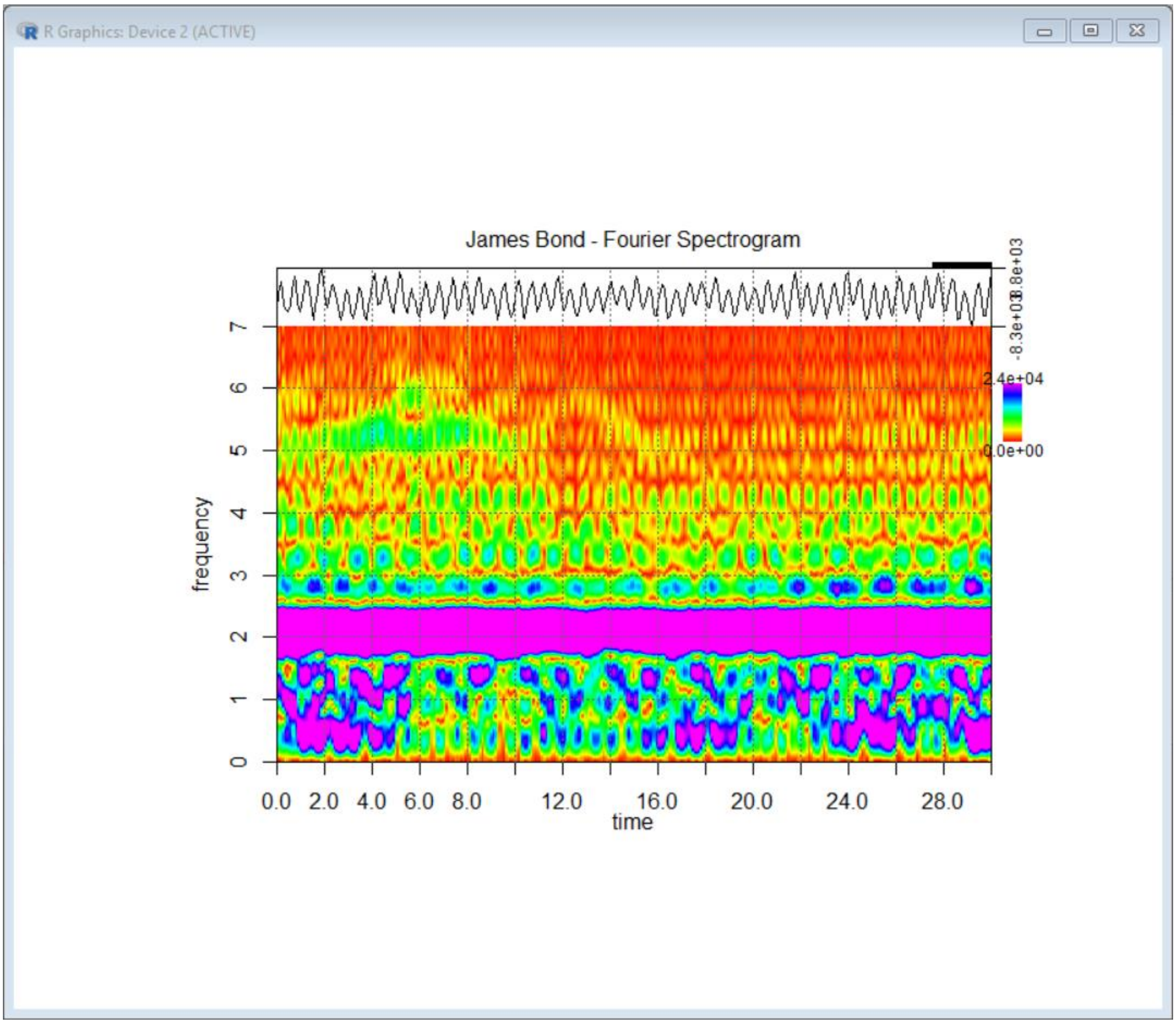


Figure 350 Graph spectrogramme de Fourier

34. Équations différentielles ordinaires ou partielles

Il peut paraître curieux qu'un logiciel initialement dédié aux statistiques pures fasse de la résolution d'ODE et PDE mais bon… comme avec un langage de script il est possible de faire beaucoup de chose alors pourquoi pas (cela vaut probablement mieux que de le faire dans un tableur).

Nous allons ici présenter la syntaxe pour résoudre et représenter les équations différentielles que nous avons démontrées dans le cours théorique. À chaque équation différentielle sera dédiée un exercice.

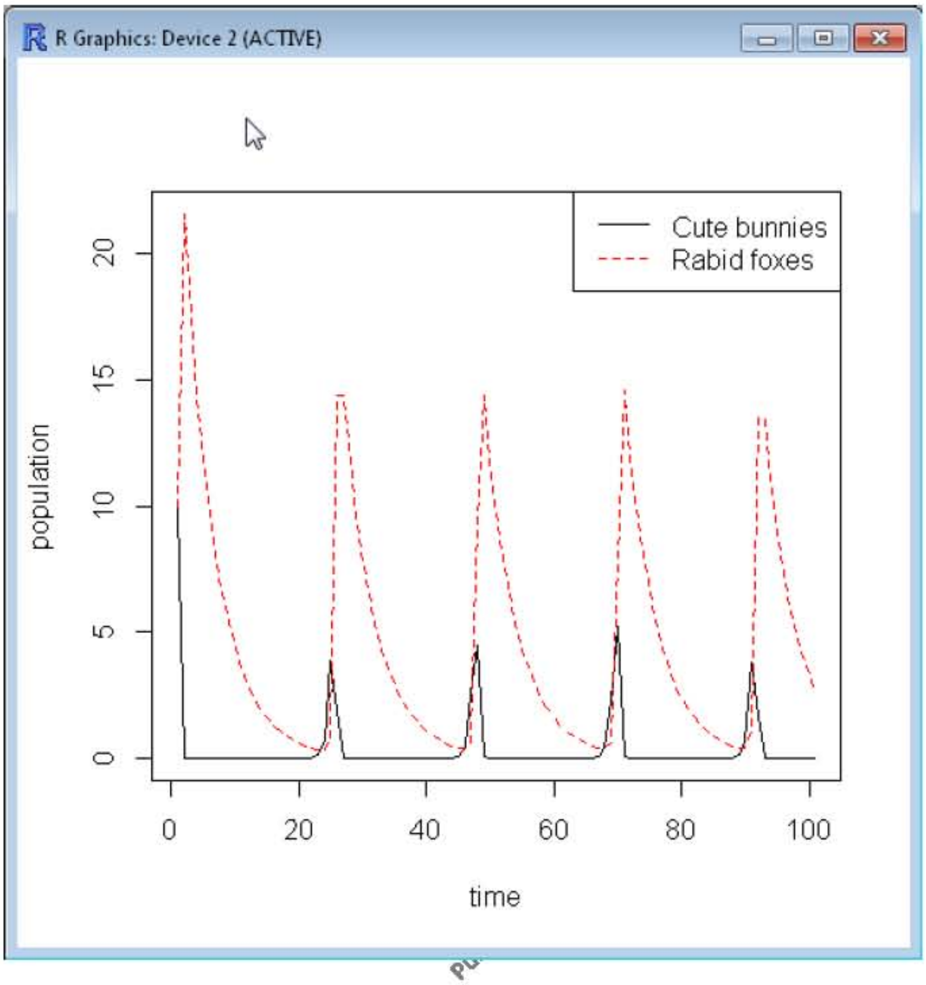
Exemple: Équations de Lotka-Volterra

R 3.1.0

À une variante de notation près au niveau des variables par rapport au cours théorique nous avons pour ce système d'ODE:

```
> library(deSolve)
>
> LotVmod <- function (Time, State, Pars) {
+     with(as.list(c(State, Pars)), {
+         dx = x*alpha - beta*x*y
+         dy = -y*gamma + delta*y*x
+         return(list(c(dx, dy)))
+     })
+ }
>
> Pars<-c(alpha=2,beta=0.5,gamma=0.2,delta=0.6)
> State<-c(x=10,y=10)
> Time<-seq(0,100,by=1)
>
> out<-as.data.frame(ode(func=LotVmod,y=State,parms=Pars,times=Time))
>
> matplot(out[,-1],type="l",xlab="time",ylab="population")
> legend("topright",c("Cute bunnies","Rabid foxes"),lty=c(1,2),col=c(1,2),box.lwd=0)
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Cute bunnies
Rabid foxes
population
time
0
5
10
15
20
0
20
40
60
80
100

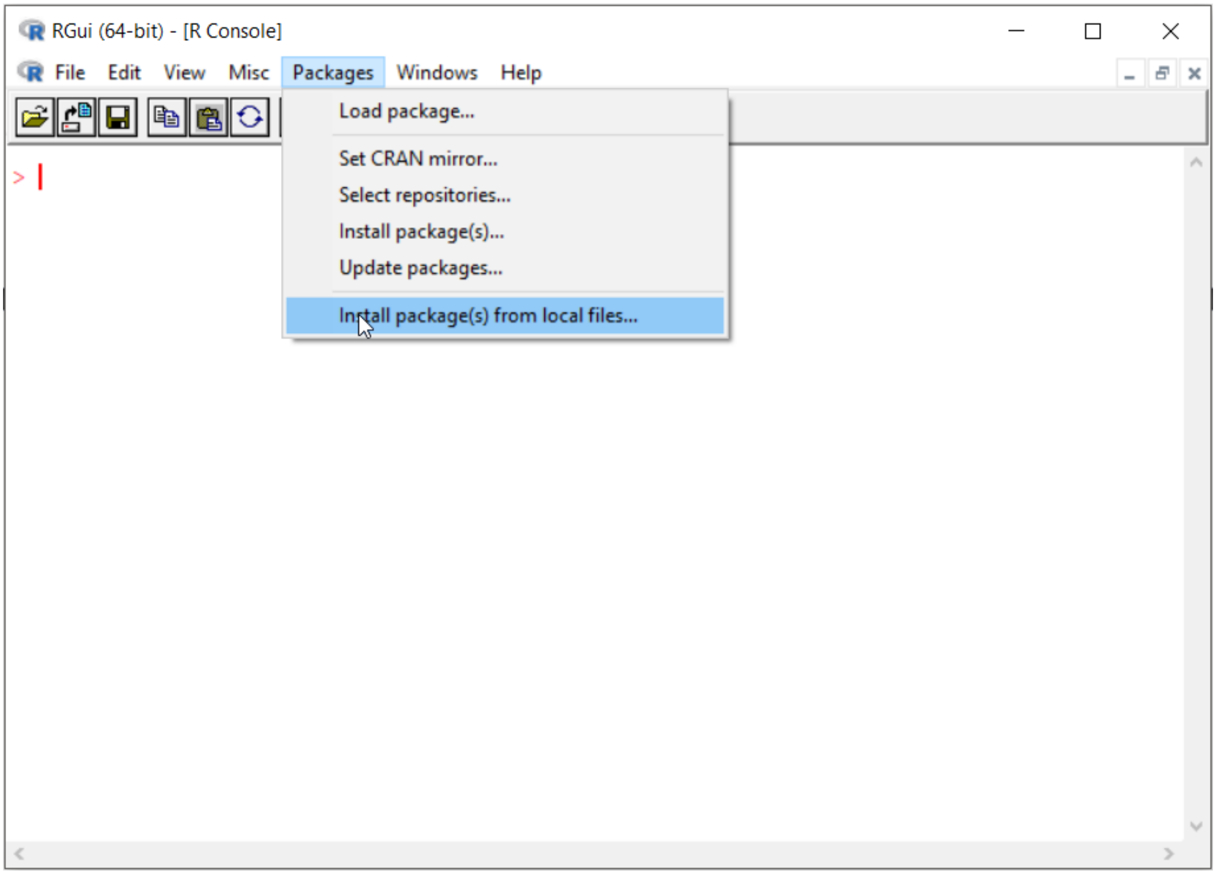
Exemple: Compétition des faucons et colombes

R 3.4.1

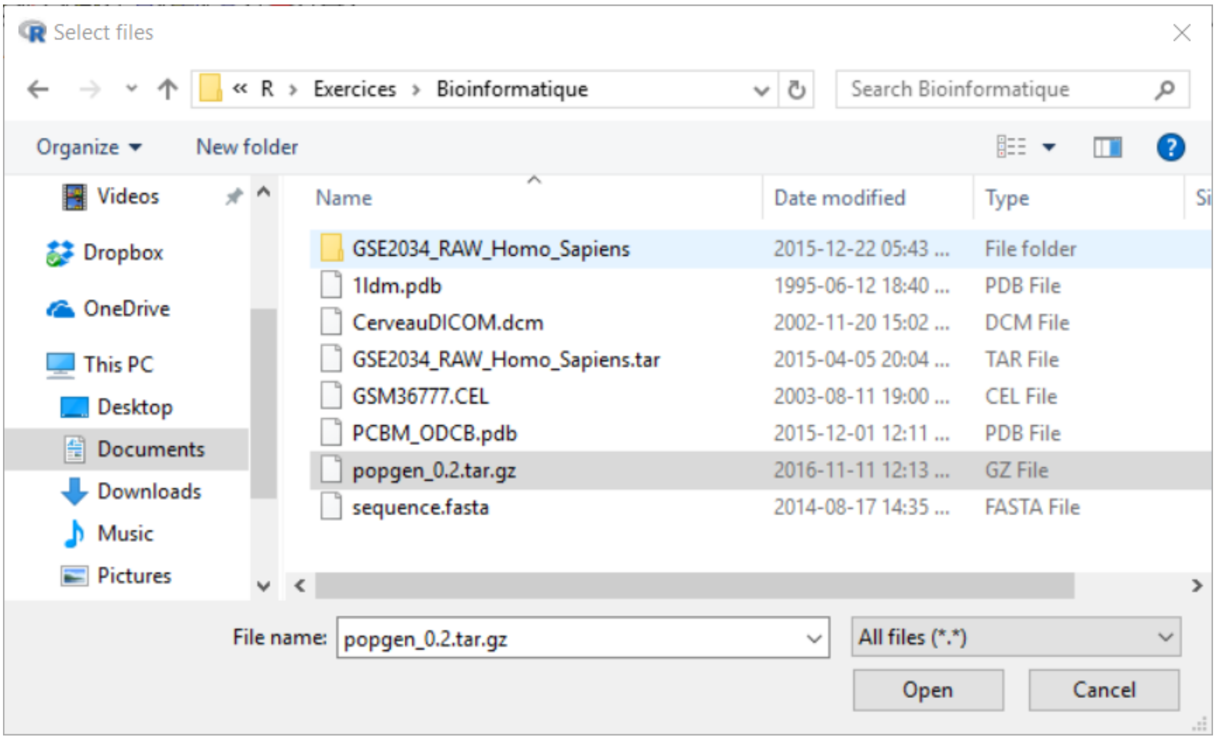
D'abord, nous allons devoir télécharger le package ici manuellement:

http://faculty.umb.edu/liam.revell/programs/

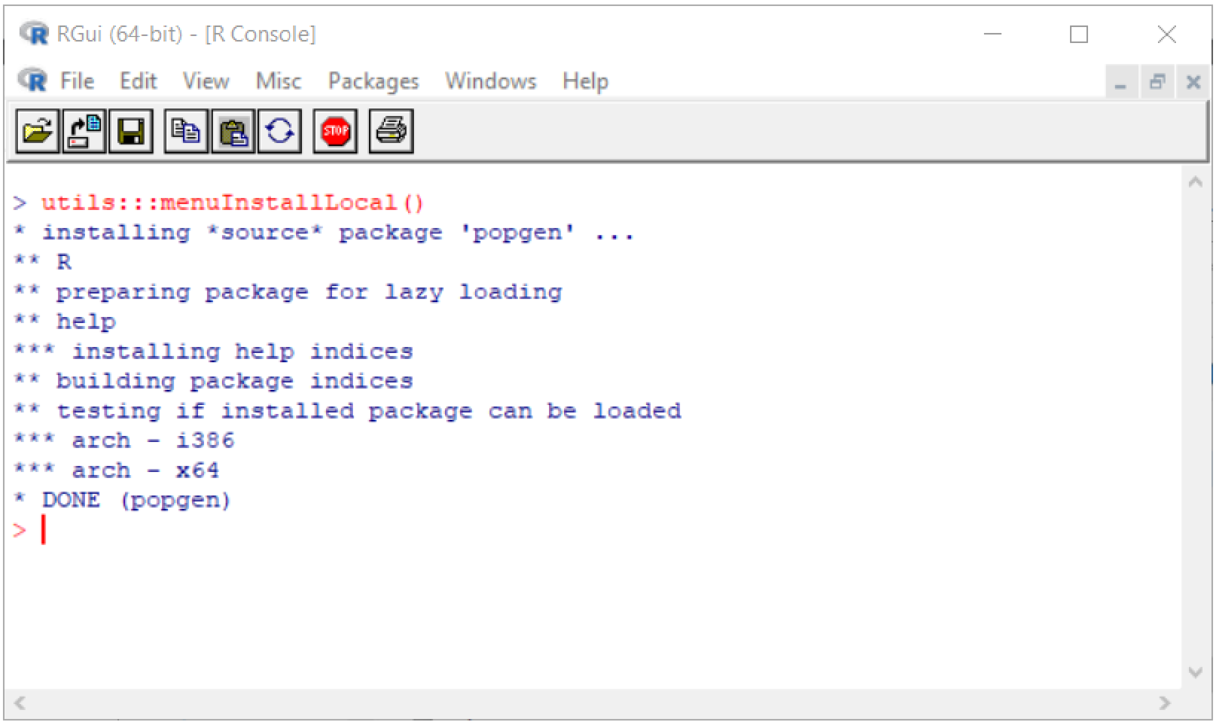
Une fois ceci fait, nous le chargeons manuellement dans R:



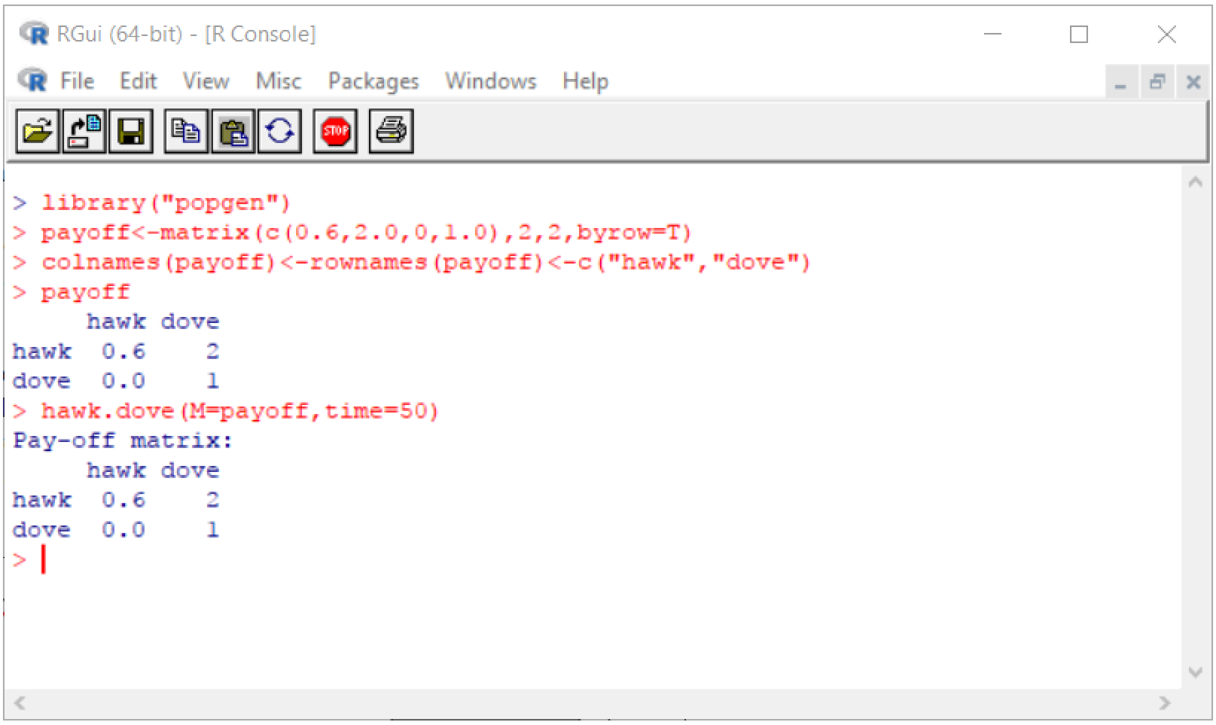
Nous sélectionnons le fichier téléchargé:



Ce qui donnera:

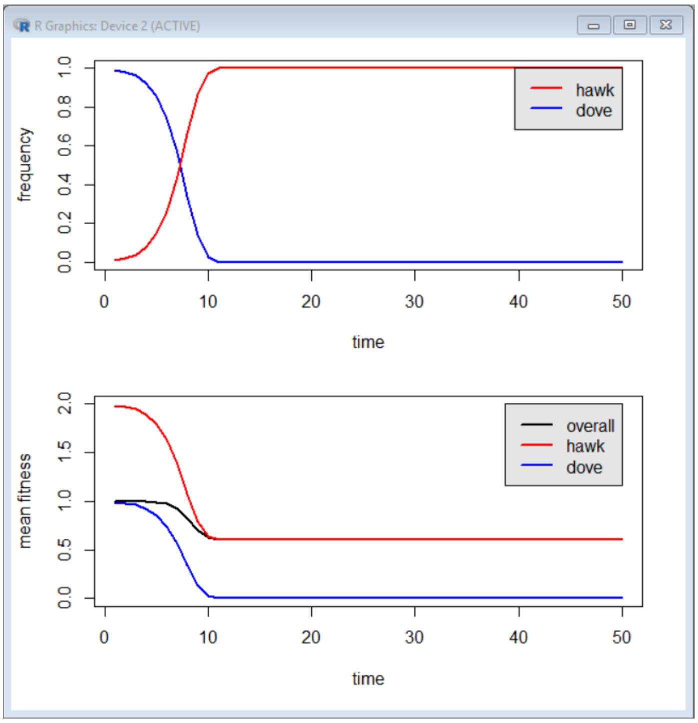


Nous définissons la matrice des gains:



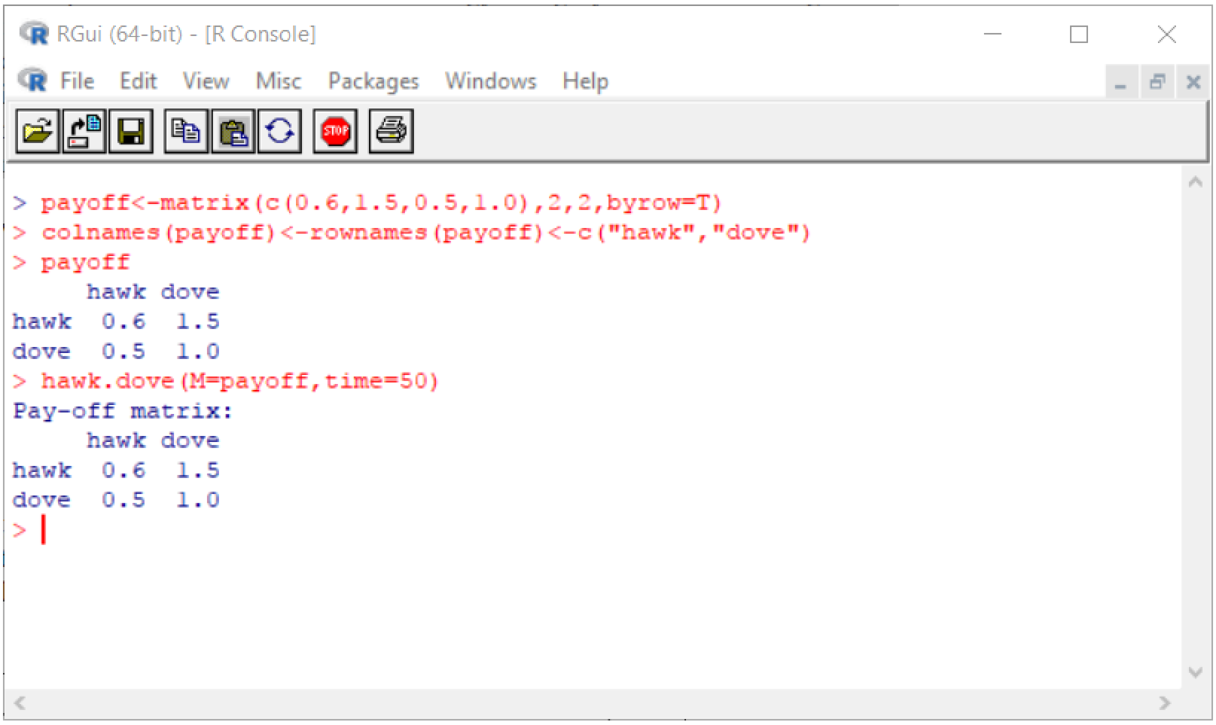
La matrice de gains ci-dessus peut être conceptualisée comme le résultat d'interactions concurrentielles pour les ressources (par exemple, dans un affrontement compétitif entre un faucon et une colombe, le faucon reçoit 2 unités de ressources et la colombe n'obtient rien).

Chaque résultat est donné par un graphique à deux panneaux. Le panneau supérieur montre les changements dans les fréquences relatives de chaque stratégie dans le temps. Le panneau inférieur montre l'état de santé moyen de la population sur la même période. Ainsi, par exemple, voici le résultat en utilisant la matrice de paiement donnée ci-dessus pour 50 générations avec une fréquence initiale de f(faucon) = 0,01 et f(colombe)= 0.99:



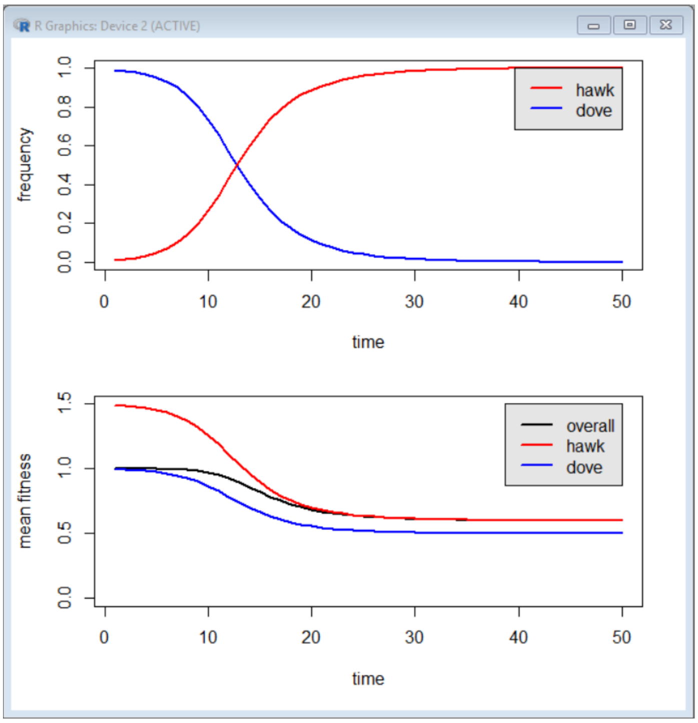
Le résultat ci-dessus peut être interprété comme suit: la stratégie des faucons est très efficace lorsqu'elle est rare. Cela converge vers une valeur statistique, ce qui suggère que la stratégie des faucons est évolutivement stable (un ESS). L'efficience de la stratégie est la plus élevée au début de la simulation, de même que l'efficience moyenne de chaque stratégie (c.-à-d., Les faucons sont plus forts quand ils sont rares); Cependant, la forme moyenne diminue dans le temps jusqu'à un niveau d'équilibre de 0.6. L'aptitude de la stratégie de la colombe approche zéro lorsque les colombes sont rares, ce qui suggère que la stratégie de la colombe n'est pas évolutivement stable dans ce paramétrage de modèle.

Considérons maintenant la matrice de gains suivante:



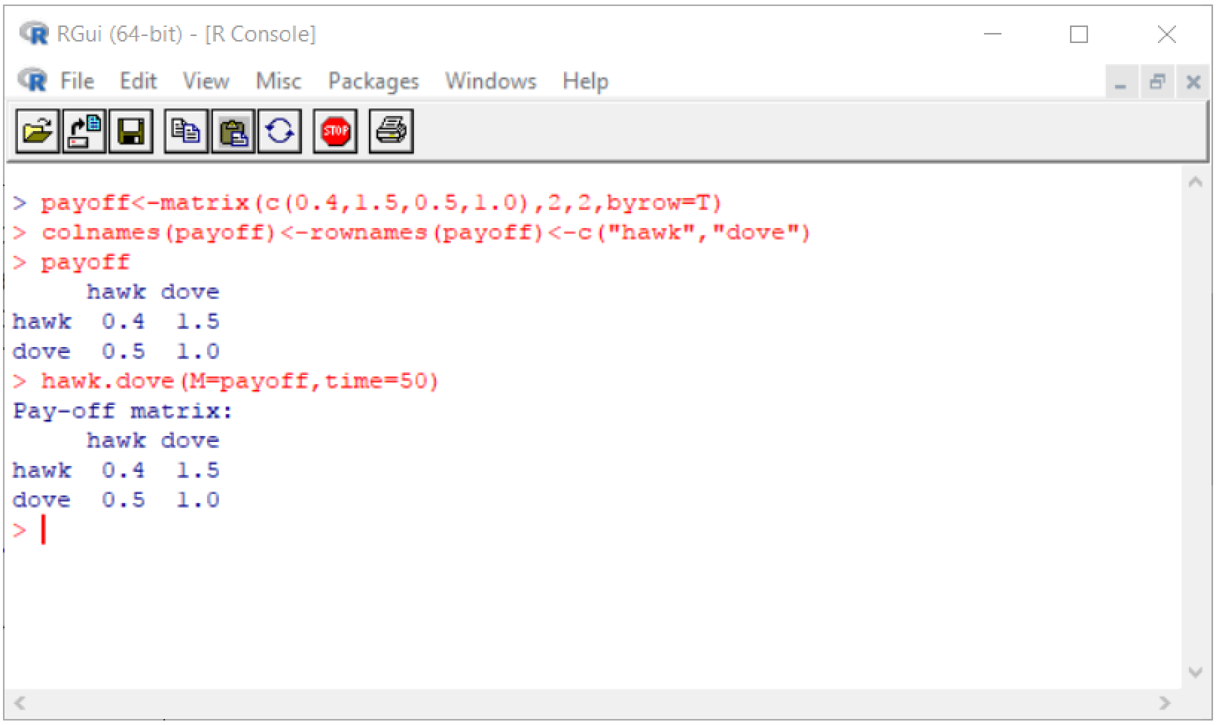
Ici, les colombes reçoivent des gains même lors de rencontres avec des faucons (peut-être, par exemple, l'agression des faucons est échangée contre une capacité à trouver de la nourriture. Les colombes trouvent les ressources alimentaires de manière plus efficace et peuvent manger avant que les faucons arrivent.) Cette matrice de paiement modifiée protège-t-elle les colombes contre l'extinction?

Cela donne:



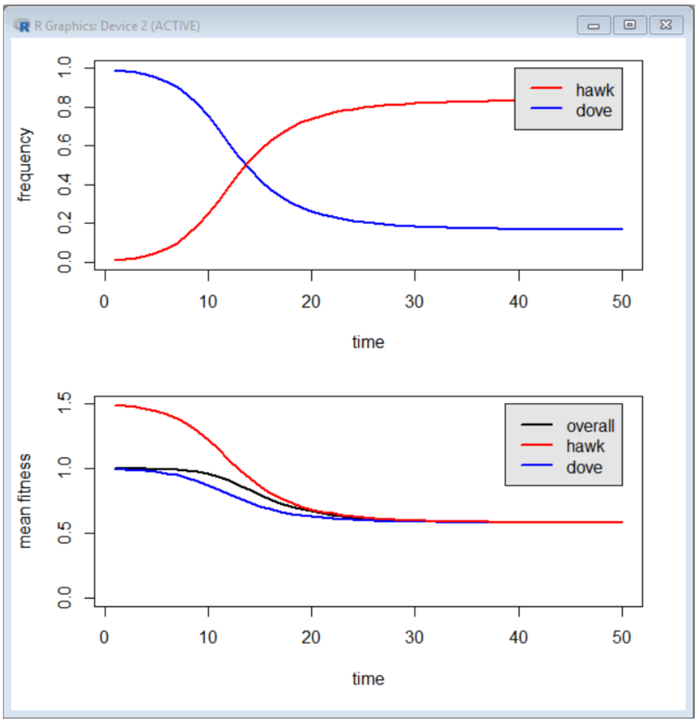
La stratégie de la colombe va encore dans le sens d'une extinction, mais le temps d'extinction est beaucoup plus long.

Essayons une matrice de gains dans laquelle les coûts de l'agression entre les faucons augmentent encore, disent à 60% de la valeur de la ressource (de sorte que l'aptitude physique d'un faucon dans une interaction faucon-faucon est de 0.4). Cela correspond à la matrice de paiement gain suivante:



```
> payoff<-matrix(c(0.4,1.5,0.5,1.0),2,2,byrow=T)
> colnames(payoff)<-rownames(payoff)<-c("hawk","dove")
> payoff
     hawk dove
hawk  0.4  1.5
dove  0.5  1.0
> hawk.dove(M=payoff,time=50)
Pay-off matrix:
     hawk dove
hawk  0.4  1.5
dove  0.5  1.0
>
```

Ce qui donne:



Enfin, nous avons trouvé un paramétrage modèle qui suggère la coexistence.

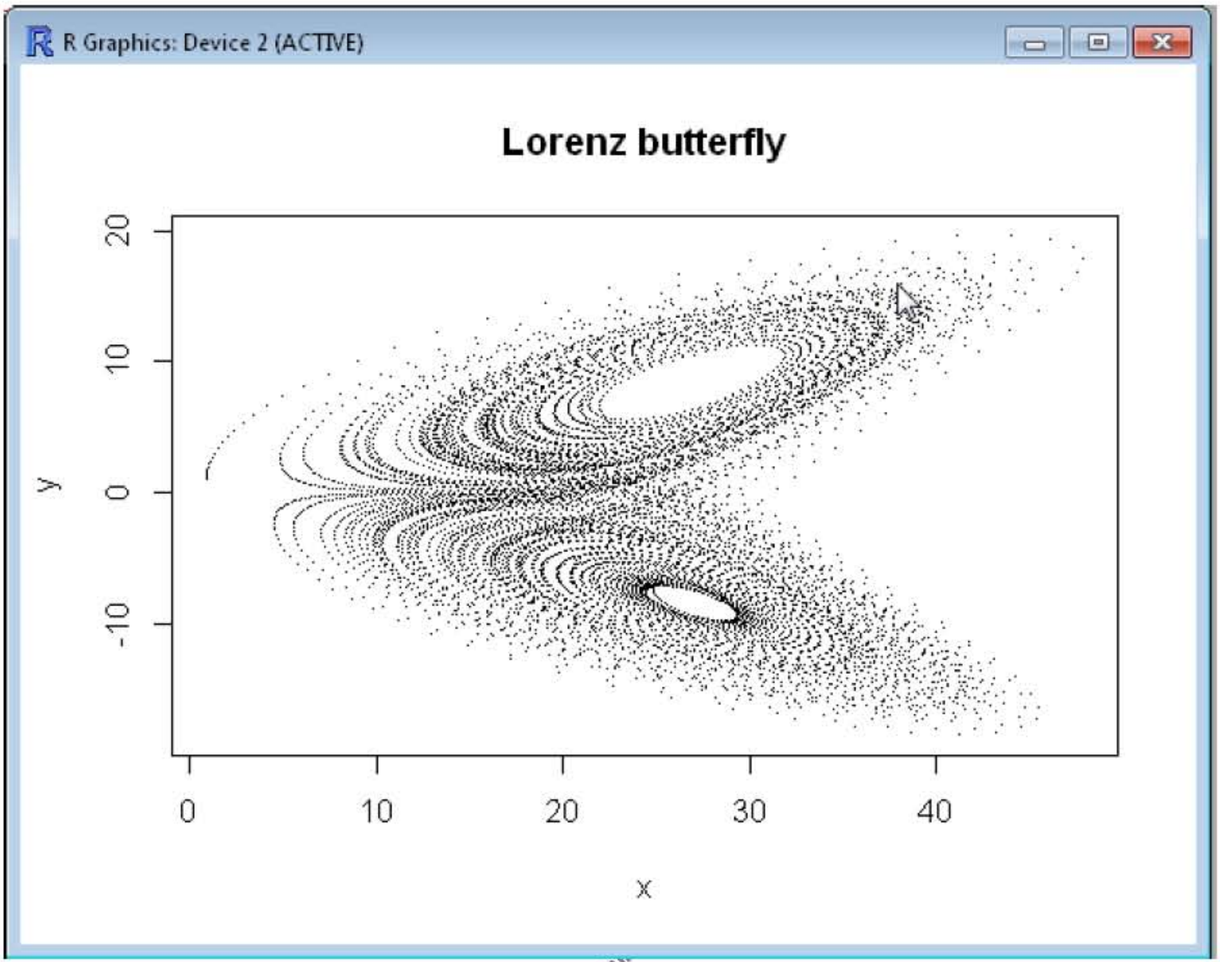
Exemple: Équations de Lorenz

R 3.1.0

À une variante de notation près au niveau des variables par rapport au cours théorique nous avons pour ce système d'ODE:

```
> library(deSolve)
> chaos <- function(t, state, parameters) {
+ with(as.list(c(state)), {
+   dx <- -8/3 * x + y * z
+   dy <- -10 * (y - z)
+   dz <- -x * y + 28 * y - z
+   list(c(dx, dy, dz))
+ })
+ }
> yini <- c(x = 1, y = 1, z = 1)
> times <- seq(0, 100, 0.01)
> out <- ode(y = yini, times = times, func = chaos, parms = 0)
> plot(out[,"x"],out[,"y"],pch = ".",main="Lorenz butterfly",xlab="x",ylab="y")
>
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
Lorenz butterfly
20
10
0
-10
y
0
10
20
30
40
x

35. Scripting (fichiers *.R)

D'abord un petit rappe concernant les raccourcis clavier:

- **Ctrl+L** efface le contenu du terminal
- **Ctrl+R** exécute la ligne de script actif
- **Ctrl+S** sauvegarde le fichier de script actif
- **F5** exécute tout le fichier de script actif

Le raccourci clavier pour exécuter une ligne ou un bloc sélectionné est **Ctrl+R** sinon pour appeler un fichier .R et l'exécuter on utilisera la commande **source( )** .

Ensuite quelques rappels:

- Utilisez la nomenclature de Leszinsky-Reddick pour le nommage de vos variables

- Commentez massivement vos scripts!

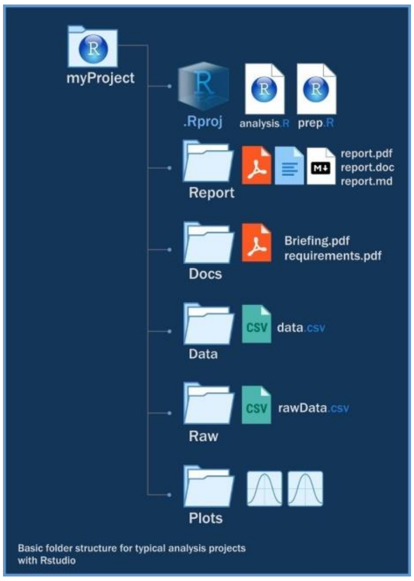
- Utilisez tant que faire ce peut[49] *pck::Nom_Fonction* plutôt que juste *Nom_Fonction* (mais attention ce sera au détriment de la performance[50]!)

- Écrivez avec une orientation la plus fonctionnelle possible afin de pouvoir faire des tests unitaires

- Conservez vos scripts de tests (pour reproduction par des tiers)

- N'oubliez pas la fonction **set.seed( )** pour faire des tests reproductibles

Enfin, une proposition, pour la structure de dossiers et fichiers de vos projets, organisez-vous de la manière suivante:

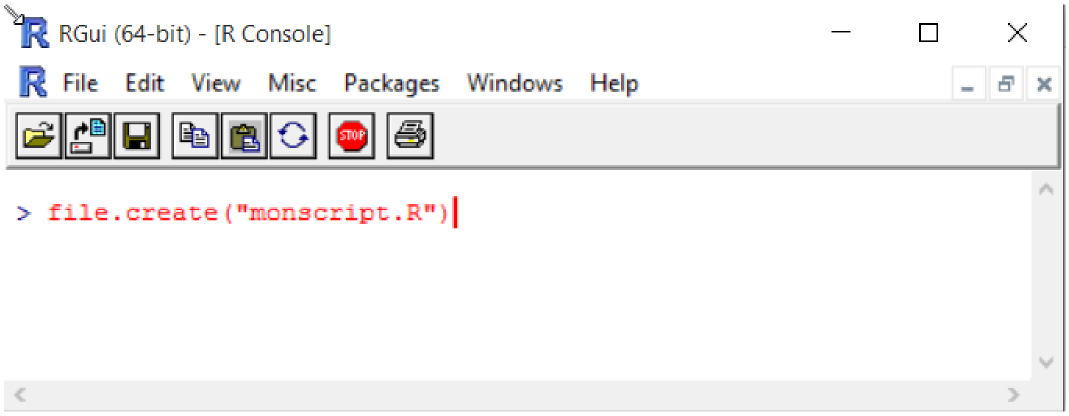


[49] Mais c'est un conseil personnel. Chacun ses habitudes et ses envies!
[50] Effectivement le package étant appelé à chaque fois…

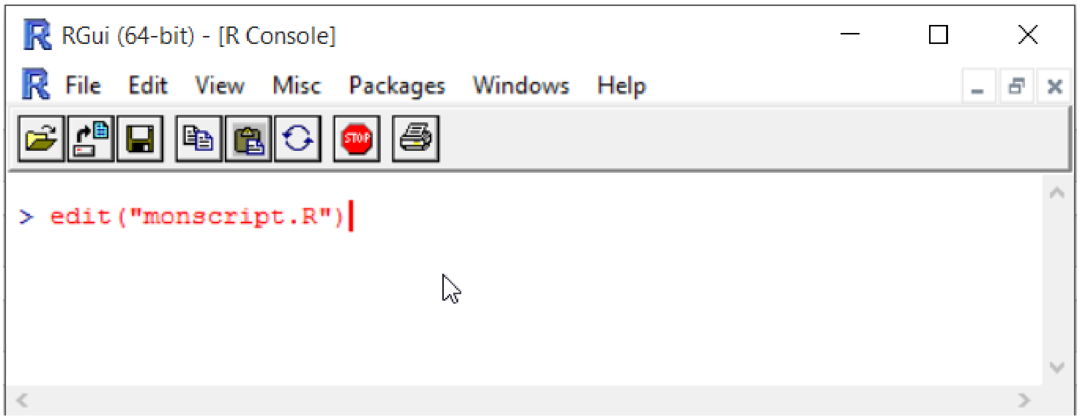
Exemple 452.: Créer/Éditer un fichier R en ligne de commande

R 3.1.3

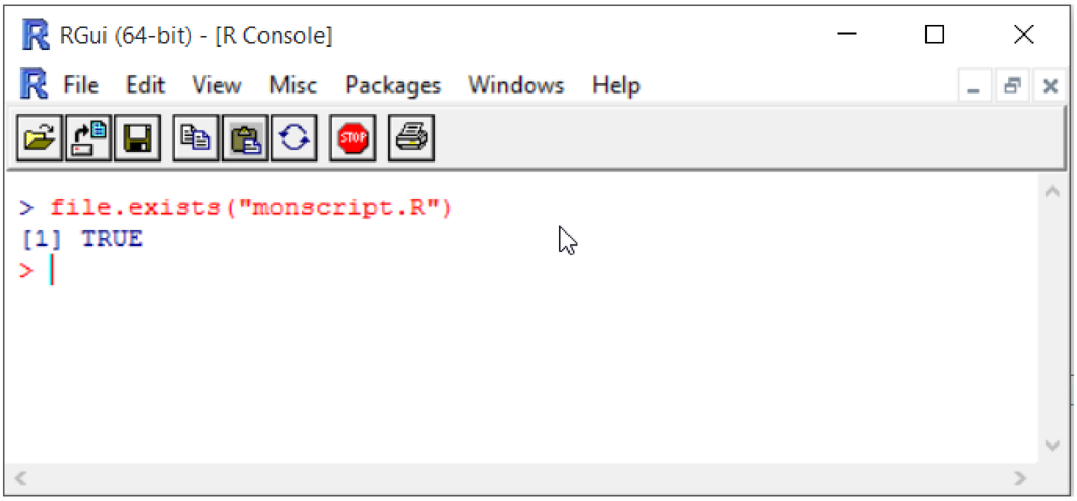
Pour créer un fichier de script en ligne de commande dans le dossier de travail par défaut, on utilisera **file.create( )** :



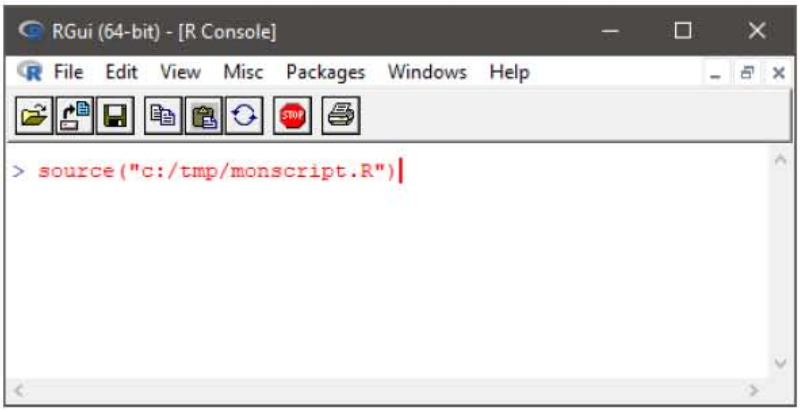
Ensuite pour l'éditer nous écrirons en ligne de commande la fonction **edit( )** ou **file.edit( )** :



Pour vérifier si un fichier existe on utilisera la commande **file.exists( )** :



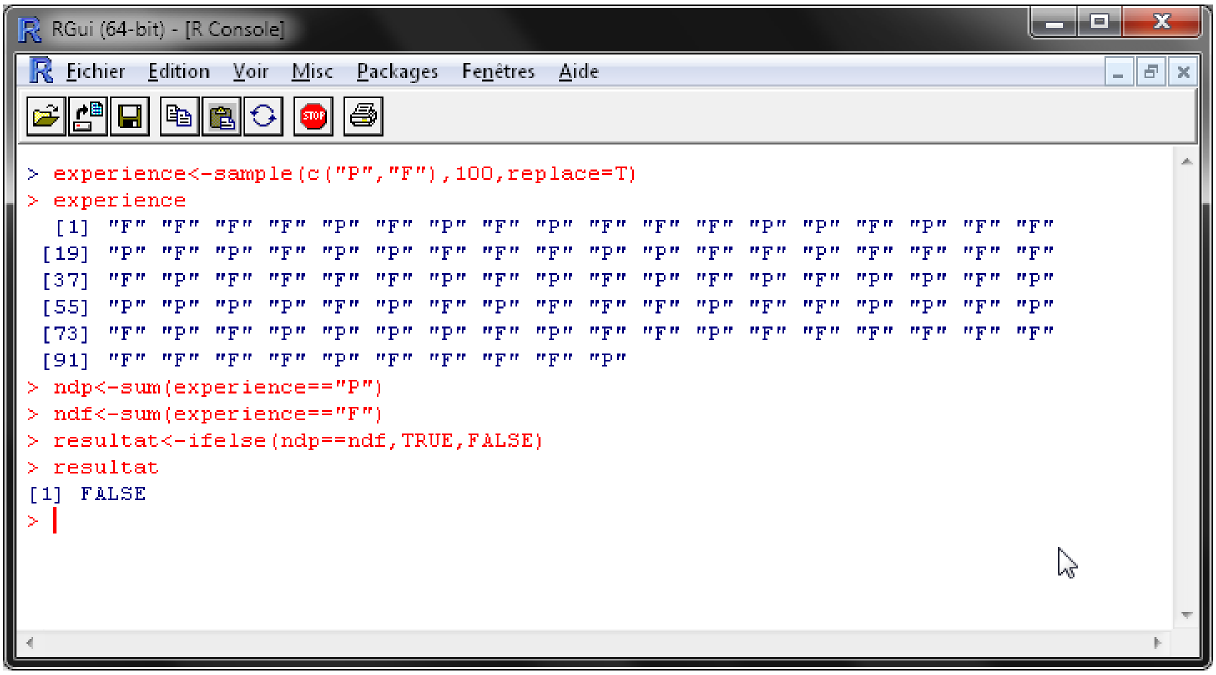
Pour exécuter un script on utilisera la commande **source( )**:



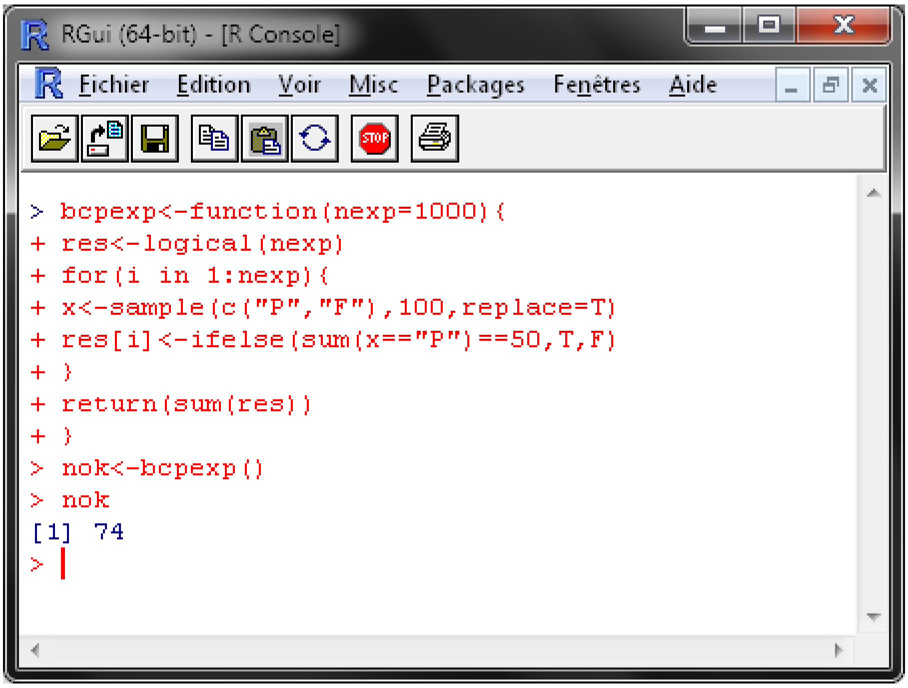
Exemple 453.: Faire une expérience aléatoire binaire

R 3.0.2

Nous tirons à pile ou face et regardons s'il y égalité entre les piles et les faces:

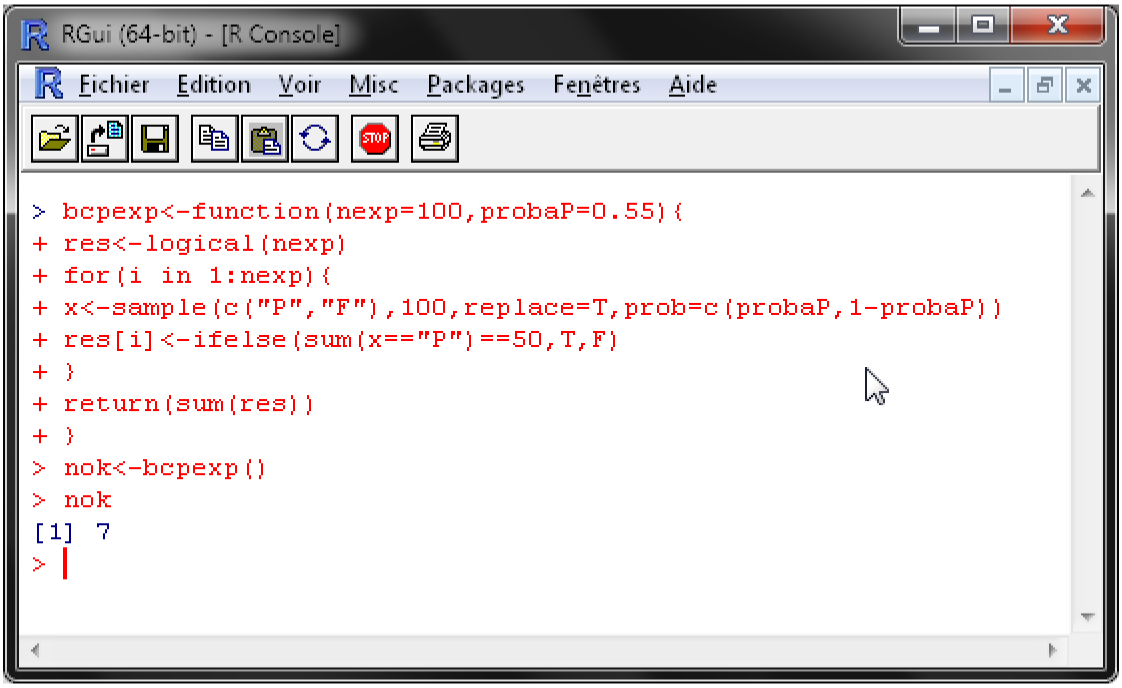


Souvent il n'y aura pas égalité (logique) mais nous savons cependant que la pièce n'est pas truquée. Alors bon combien de fois avons nous la non égalité si nous répétons souvent ce test?



Donc 74 fois sur 1'000 l'hypothèse nulle comme quoi la pièce est non truquée est juste, le reste du temps elle est fausse. Donc nous avons un risque de première espèce d'environ $\alpha \cong 93\%$ de rejeter à tort l'hypothèse nulle.

Supposons que maintenant la pièce soit légèrement truquée:



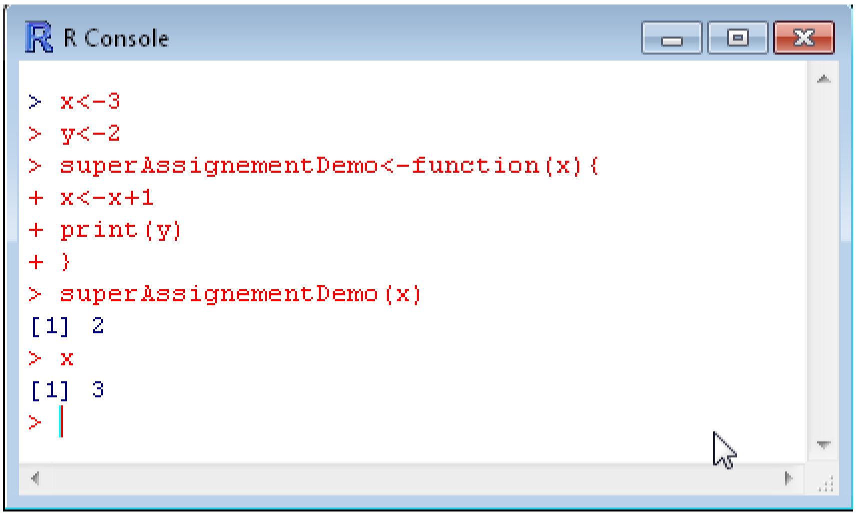
Donc sur 1'000 expériences d'une pièce truquée, notre test à décidé 7 fois à tort que l'hypothèse nulle était vraie à tort et 993 fois qu'elle fausse à raison. Nous avons donc un risque de deuxième espèce de $\beta \cong 0.007\%$

Exemple 454.: Portée des variables

R 3.0.2

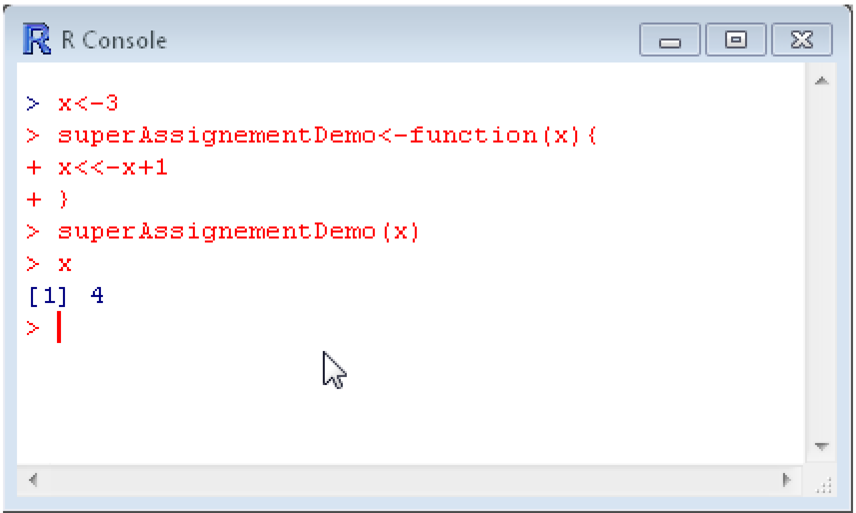
Dans tout langage de script ou de programmation la notion de portée des variables est importante à comprendre et dans certains cas très importante à utiliser.

Considérons d'abord l'exemple classique ci-dessous le plus simple:

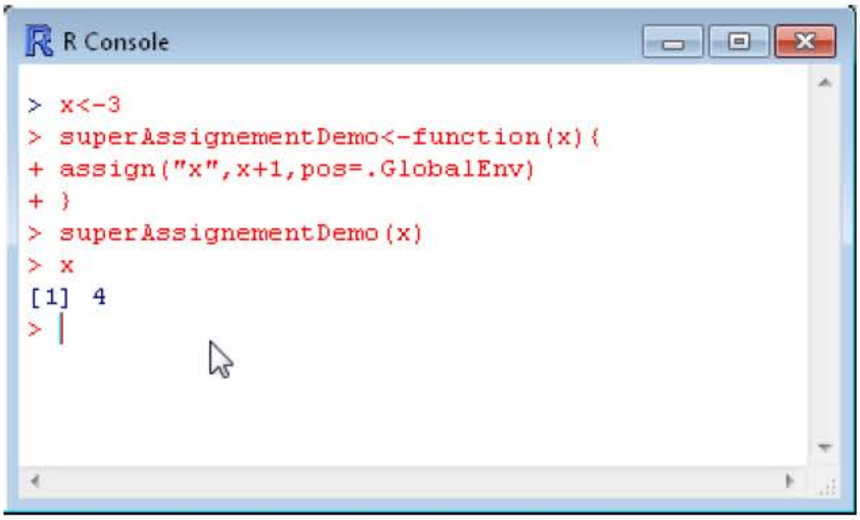


Nous remarquons que la fonction est capable de lire la variable *y* qui est à l'extérieur de son domaine de définition bien que cette même variable ne soit pas passée en tant qu'argument. Ensuite, nous remarquons aussi que la fonction n'est pas capable de modifier la variable *x* qui se trouve à l'extérieur de son domaine de définition puisqu'avant et après *x* vaut toujours 3.

Il faut voir maintenant comment nous pouvons écraser la variable *x* au-delà de la fonction elle-même. Pour cela, la première méthode consiste à utiliser l'opérateur de superaffectation <<-:



ou en utilisant la fonction **assign( )** :


R Console
> x<-3
> superAssignementDemo<-function(x){
+ assign("x",x+1,pos=.GlobalEnv)
+ }
> superAssignementDemo(x)
> x
[1] 4
>

Exemple 455.: Comprendre les structures conditionnelles

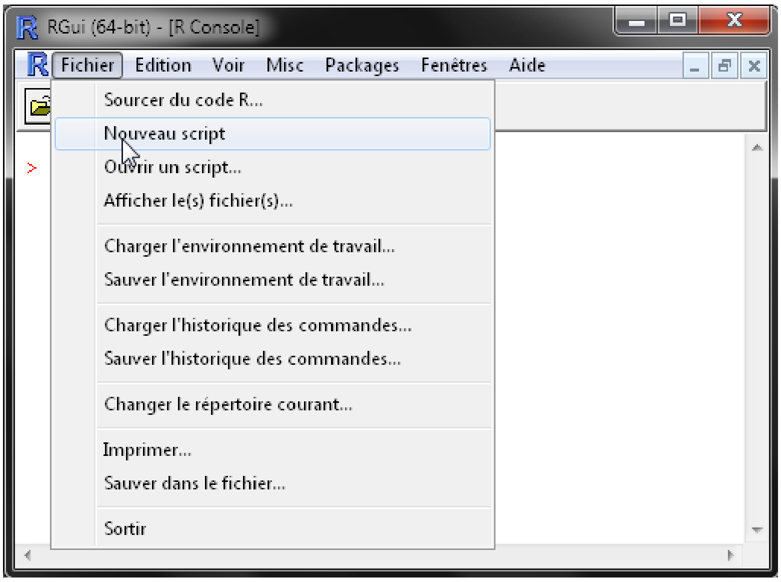
R 3.0.2

Structure conditionnelle *If...else*

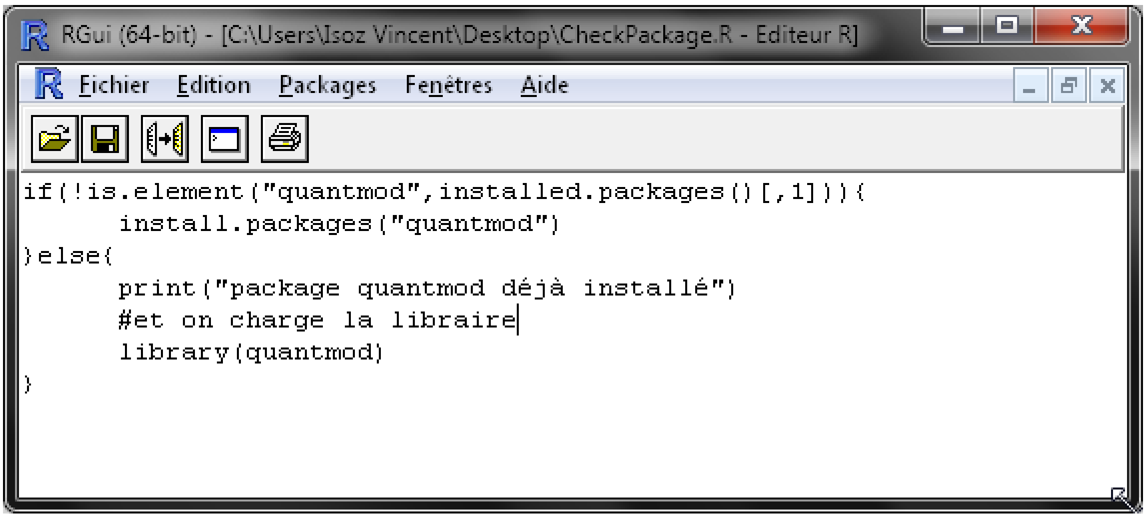
La structure conditionne **if**... **else**... ne peut opérer que sur un unique attribut (donc pas applicable sur l'ensemble des attributs d'un vecteur).

Un cas classique dans R consiste dans les scripts à vérifier si un package est installé ou non:

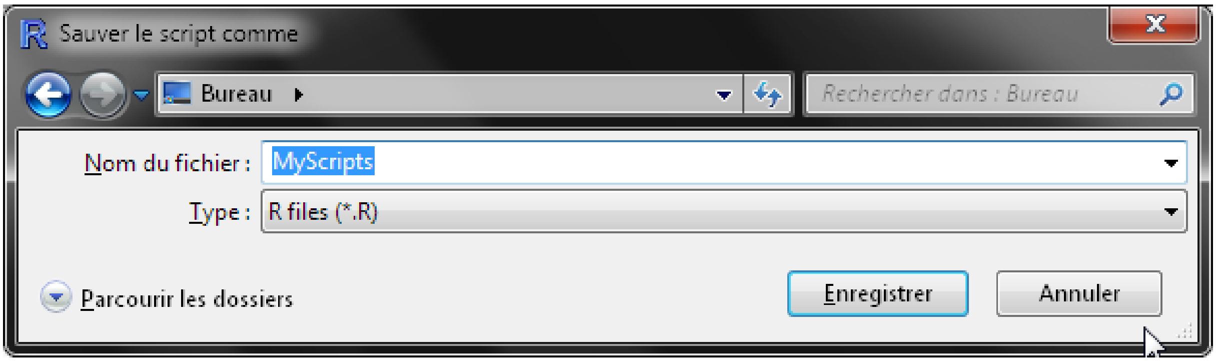
Nous allons donc créer un nouveau script:



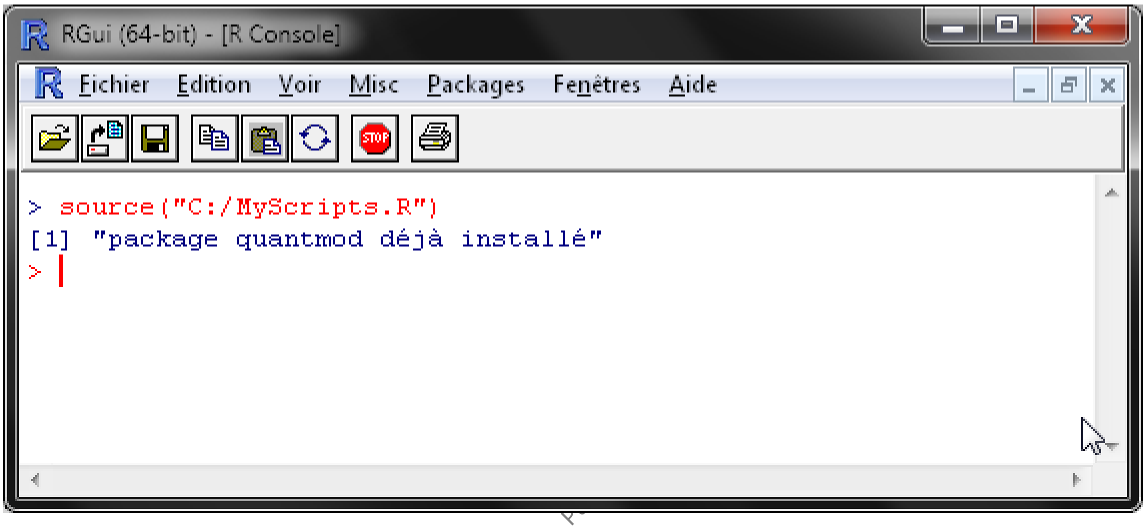
Et nous écrivons:



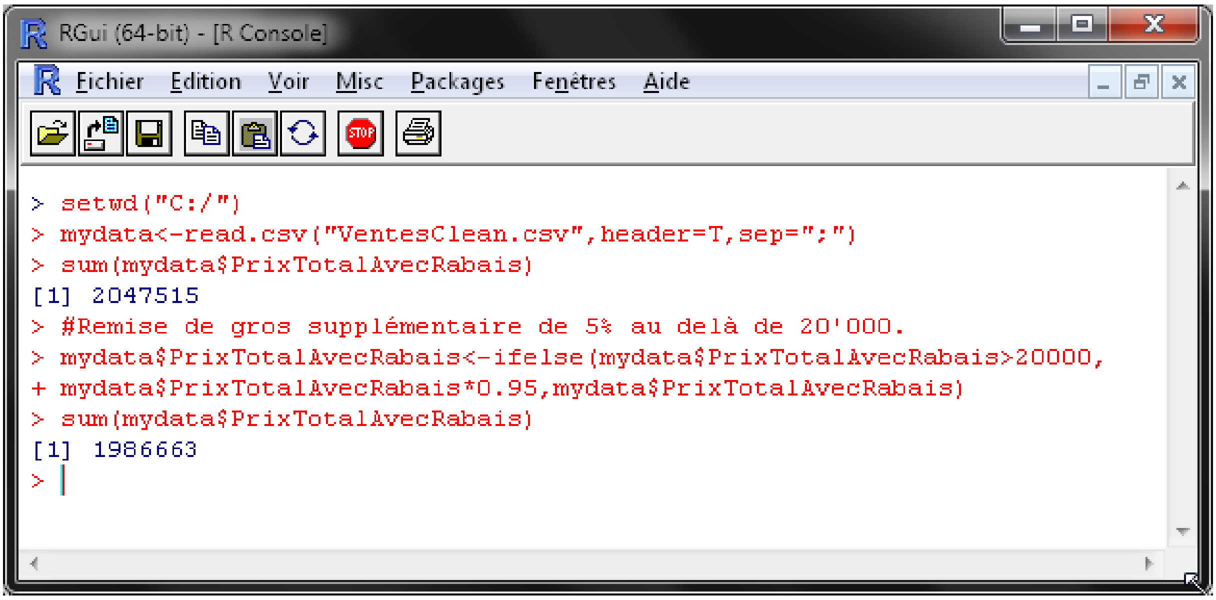
Ensuite, nous enregistrons notre fichier en tant que fichier script (pouvant contenir plus qu'une fonction):



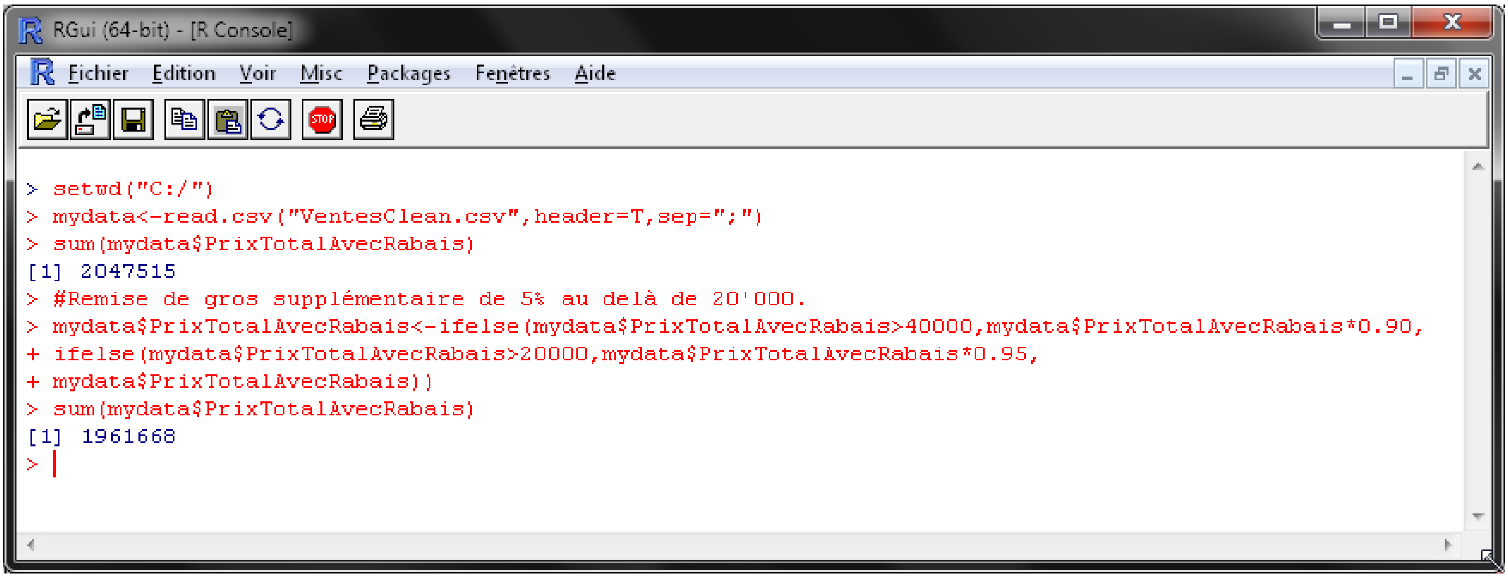
Une fois ceci fait nous pouvons charger et utiliser le script:



Nous un premier exemple avec la structure conditionnelle **ifelse( )** qui est très souvent utilisé pour opérer directement sur l'ensemble des attributs d'un vecteur:

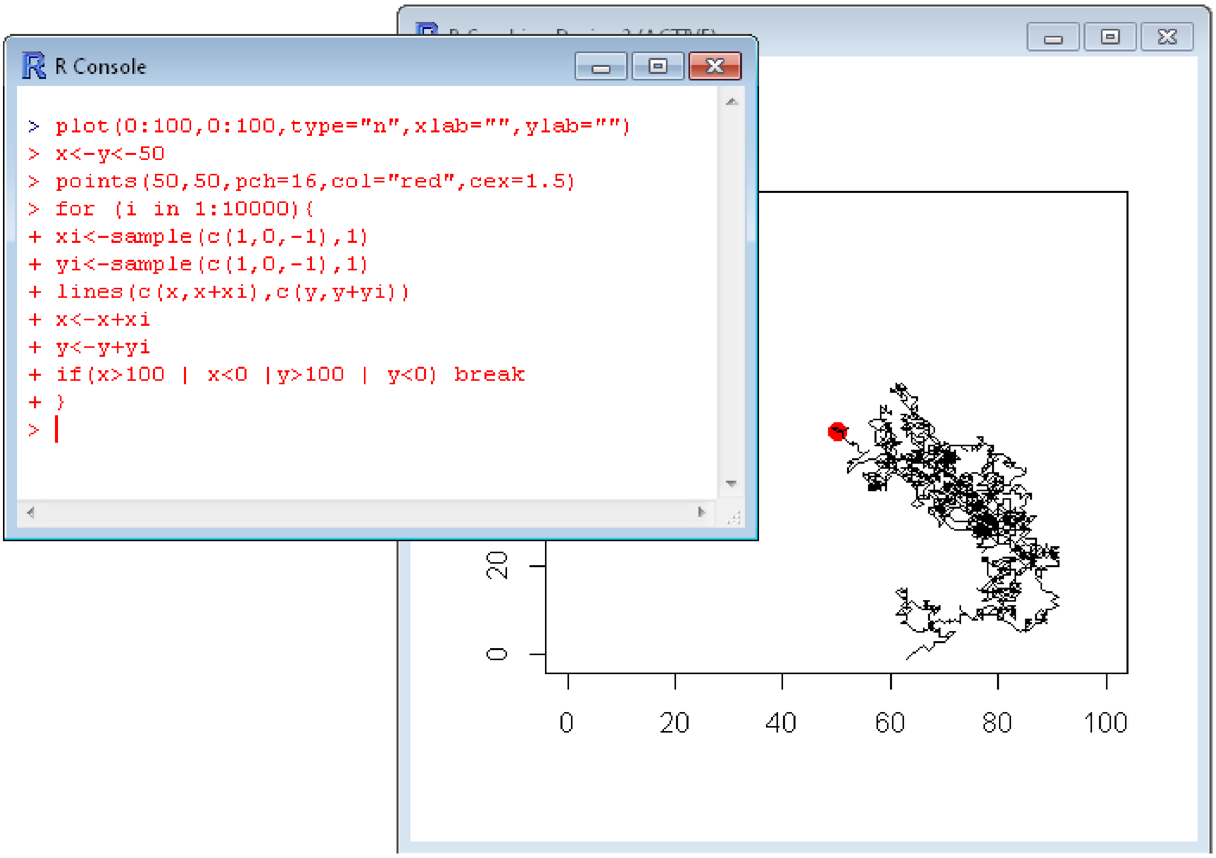


Et évidemment il est possible d'imbriquer des **ifelse**:



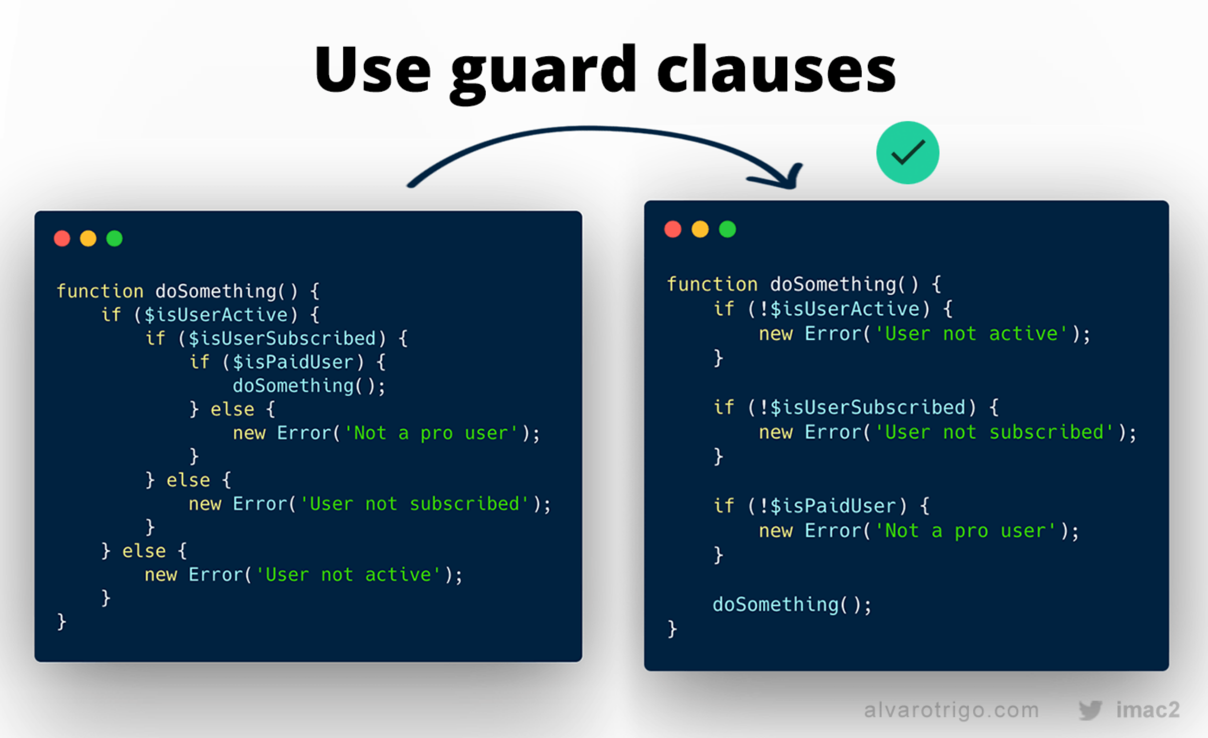
Le **ifelse** peut aussi être utilisé pour autre chose que de faire des calculs bien évidemment (par exemple renommer le contenu d'un vecteur-colonne) et ainsi de suite...

Un autre exemple académique ultra connu et typique d'application des structures conditionnelles est de représenter graphiquement une marche aléatoire (mouvement brownien). Ce qui donne (il existe plusieurs manières d'écrire un tel script!):



Attention! Certains professionnels de la programmation utilisent la technique de la "Guard Clause" qui permet de rendre le code beaucoup plus lisible plutôt que d'imbriquer des IF…

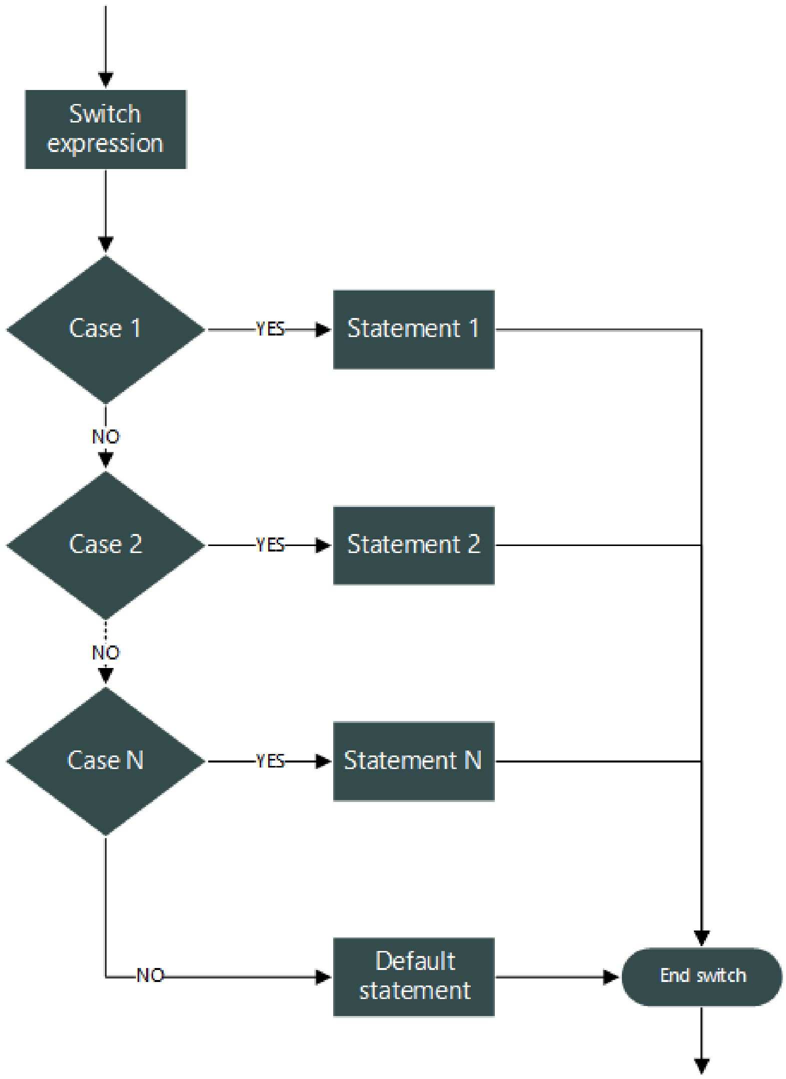
ELSE les uns dans les autres! La technique peut être illustrée par la figure suivante (qui n'est pas en R):



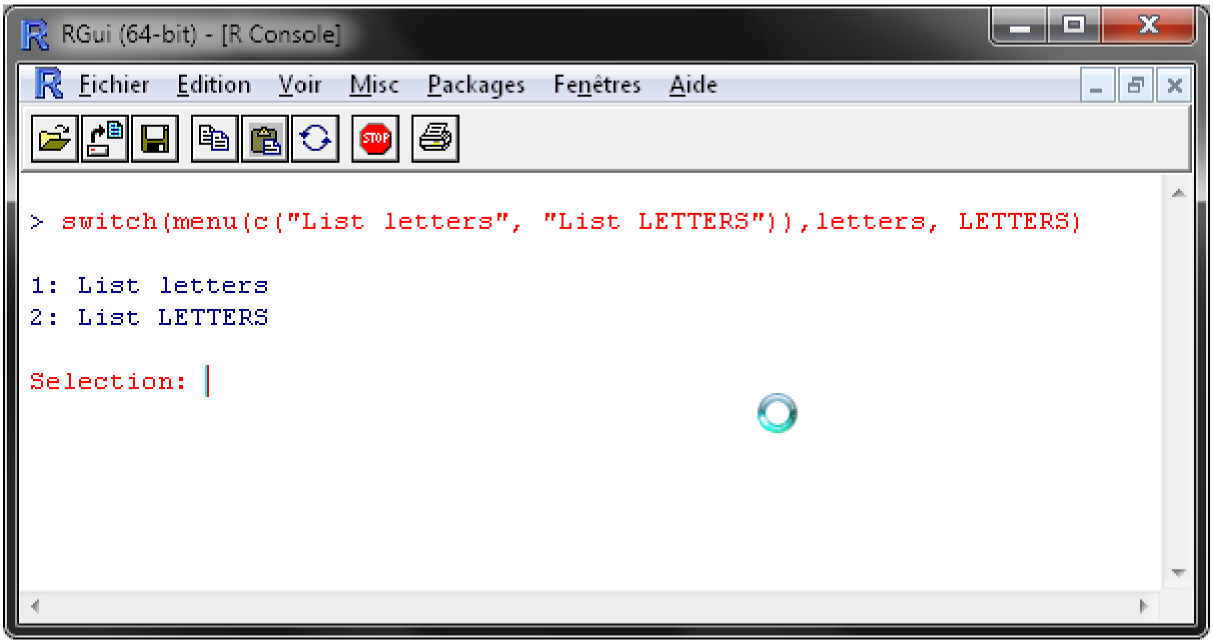
Structure conditionnelle *Switch*

La structure conditionnelle est à peu près équivalente à la structure *if* à ce près que nous ne pouvons pas toujours faire de tests logiques complexes, que l'écriture est à résultat égal plus court qu'une structure *if* et que normalement à résultat égal le temps d'exécution d'une structure switch est toujours plus rapide en temps processus.

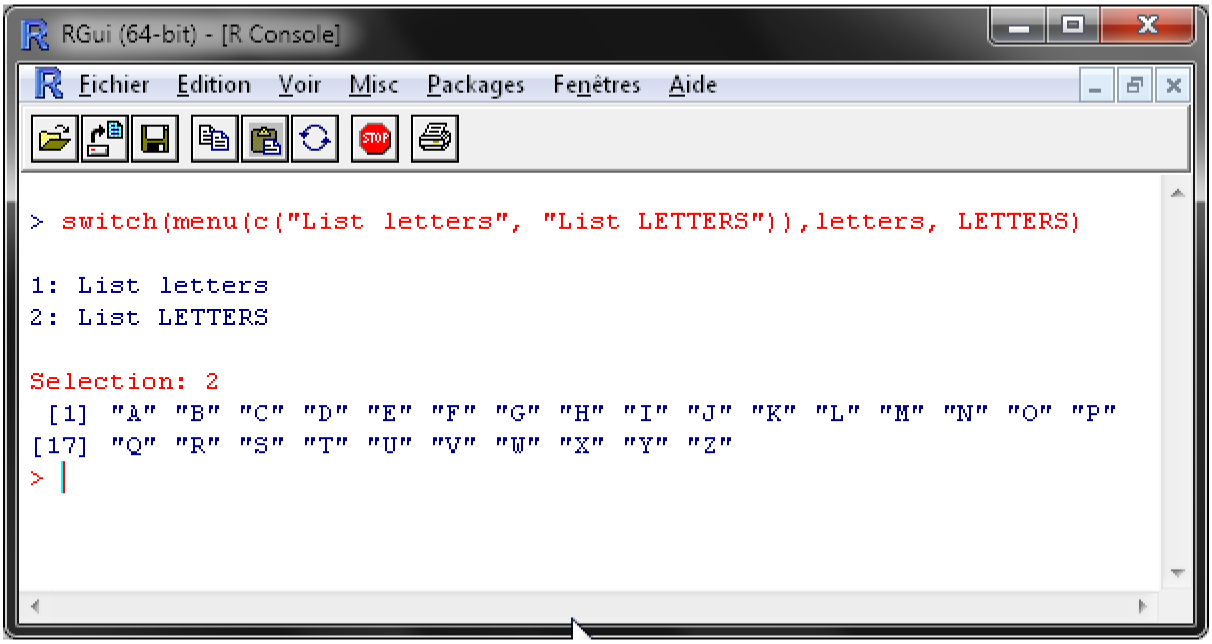
Rappelons le principe:



Voyons un petit cas classique qui consiste à faire un petit menu à choix exécutant certaines commandes en fonction du choix de l'utilisateur:



et si l'utilisateur fait un choix:

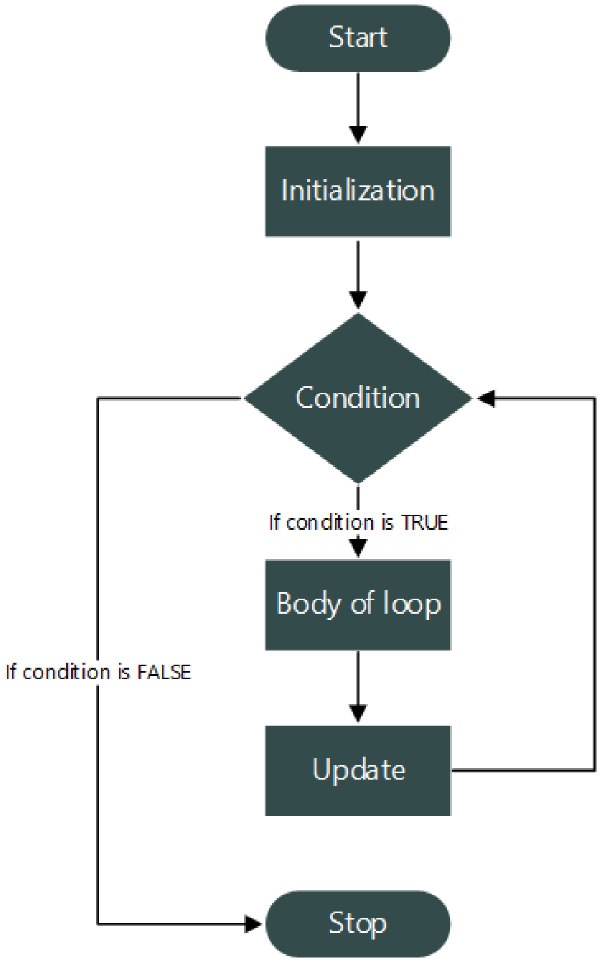


Exemple 456.: Comprendre les structures itératives

R 3.0.2

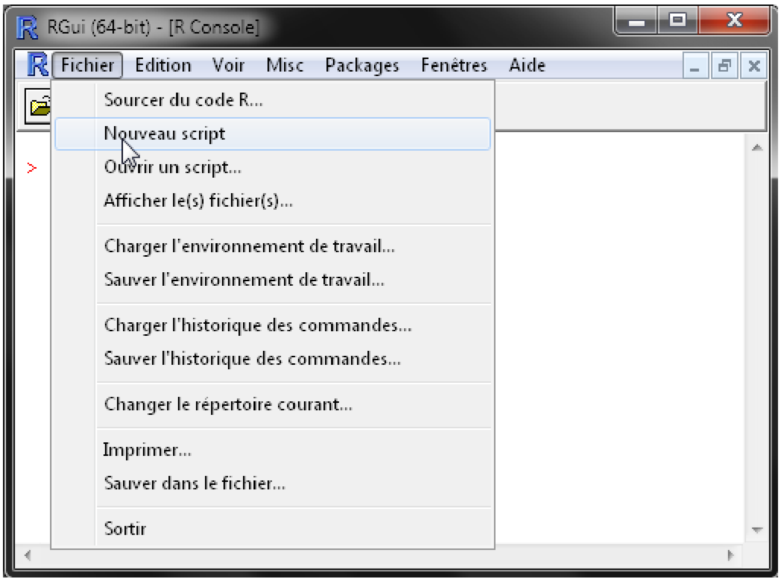
Les structures itératives ont elles aussi une place très importante dans R (comme dans tous les langages de toute façon!).

Rappelosn le principe:

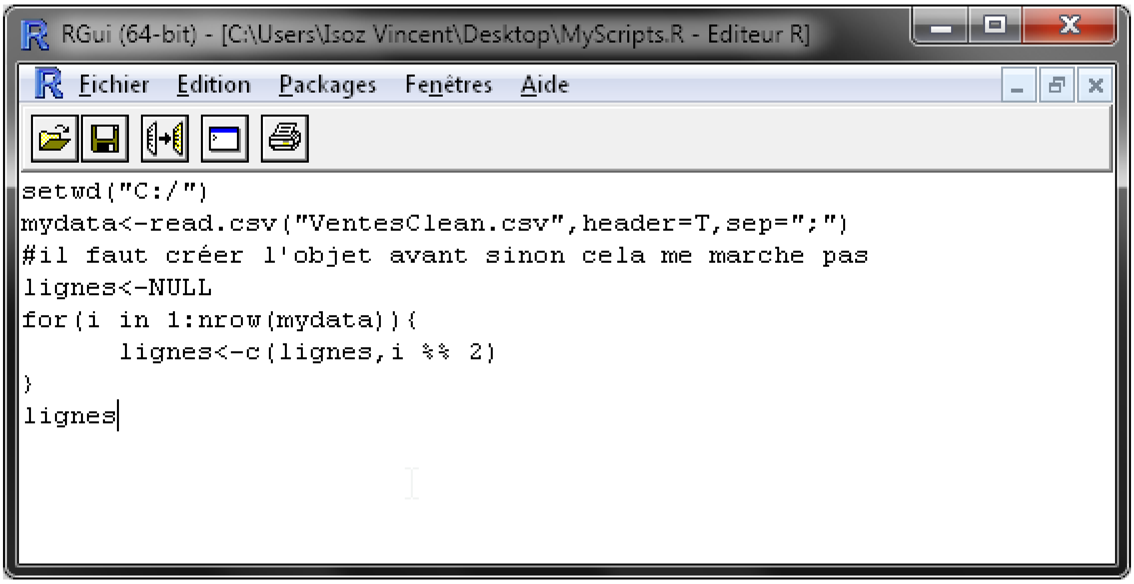


Considérons le cas où nous souhaiterions avec une identification rapide des lignes paires ou impaires d'un data frame:

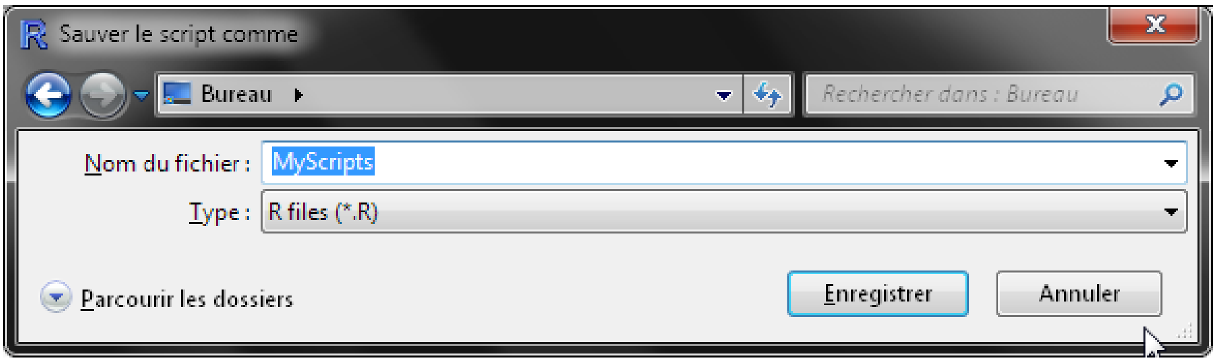
Nous allons donc créer un nouveau script:



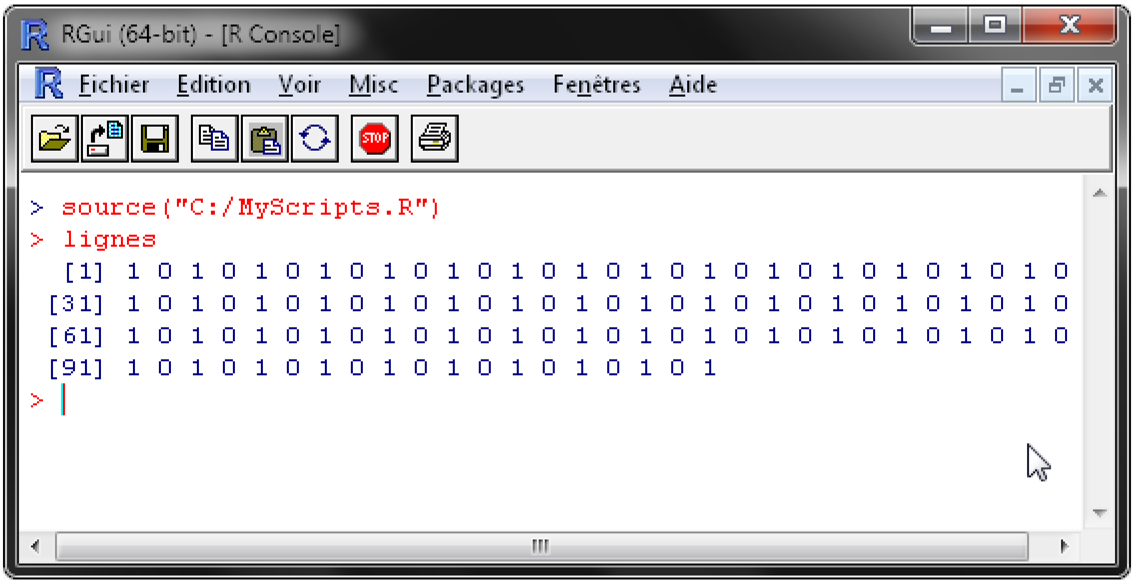
Et nous écrivons notre premier exemple avec une boucle **for( )** :



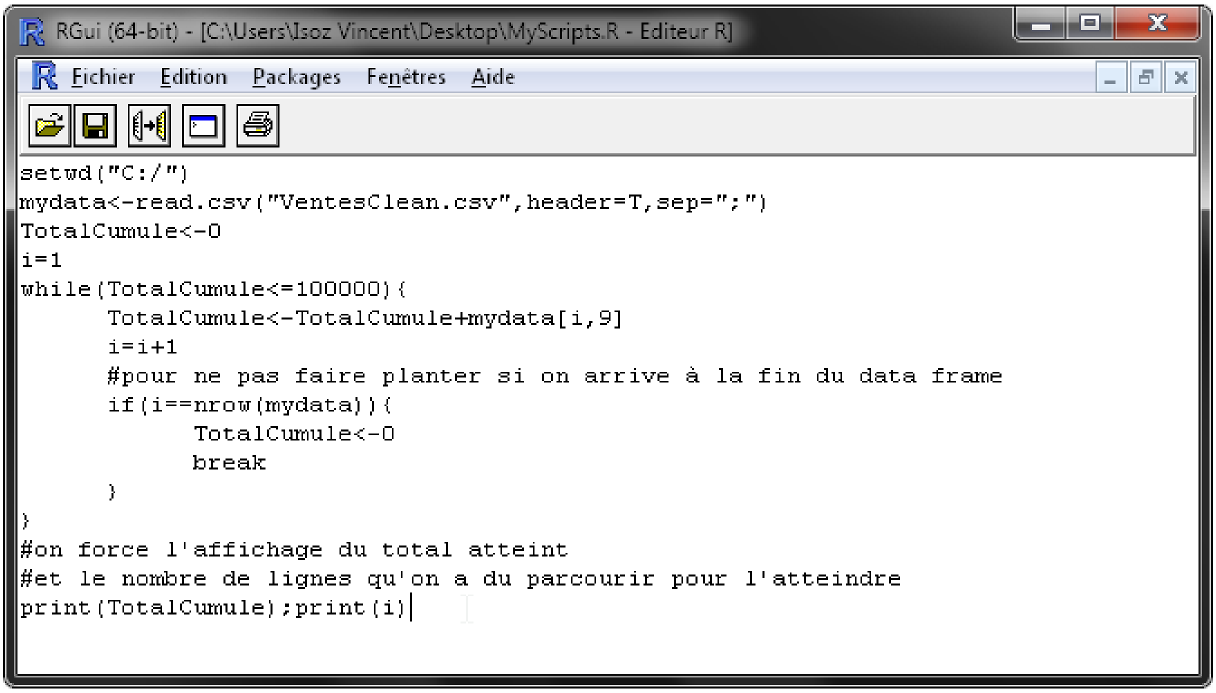
Ensuite, nous enregistrons notre fichier en tant que fichier script (pouvant contenir plus qu'une fonction):



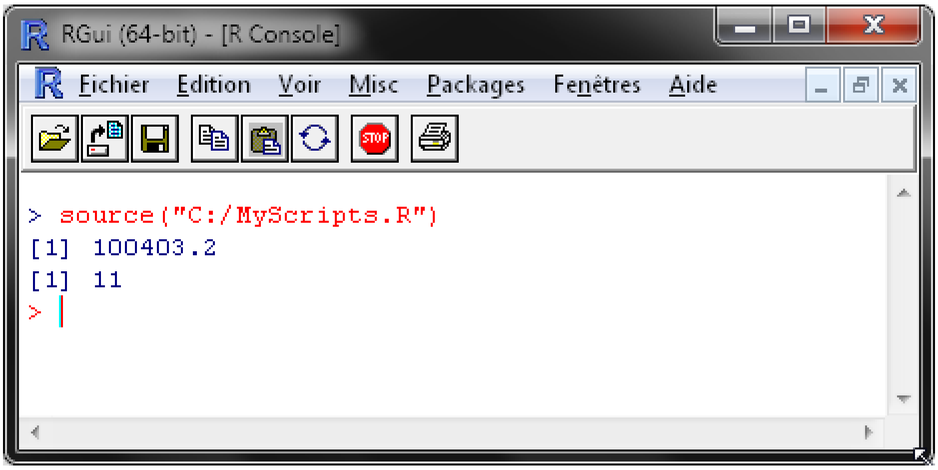
Une fois ceci fait nous pouvons charger et utiliser le script:



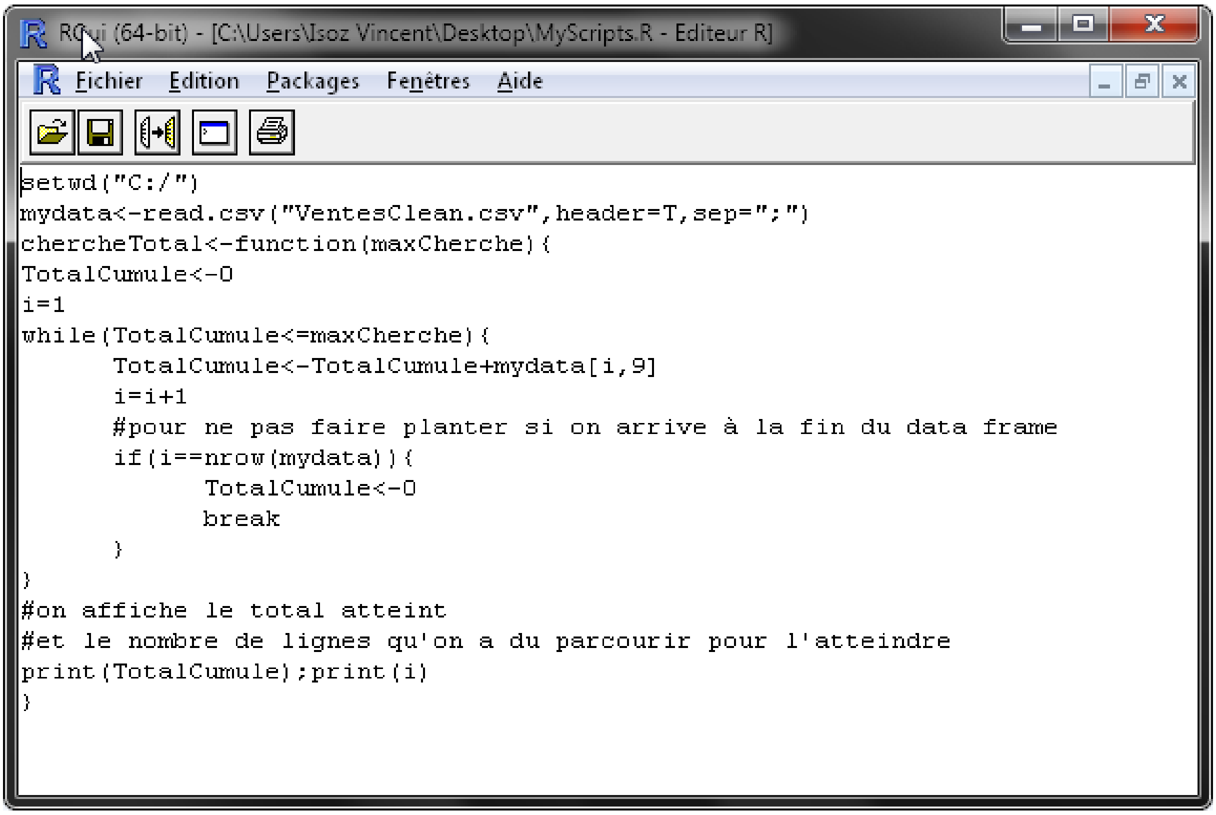
Ou encore l'instruction **while( )** qui permet d'identifier par exemple à partir de quel moment une certaine somme est atteinte:



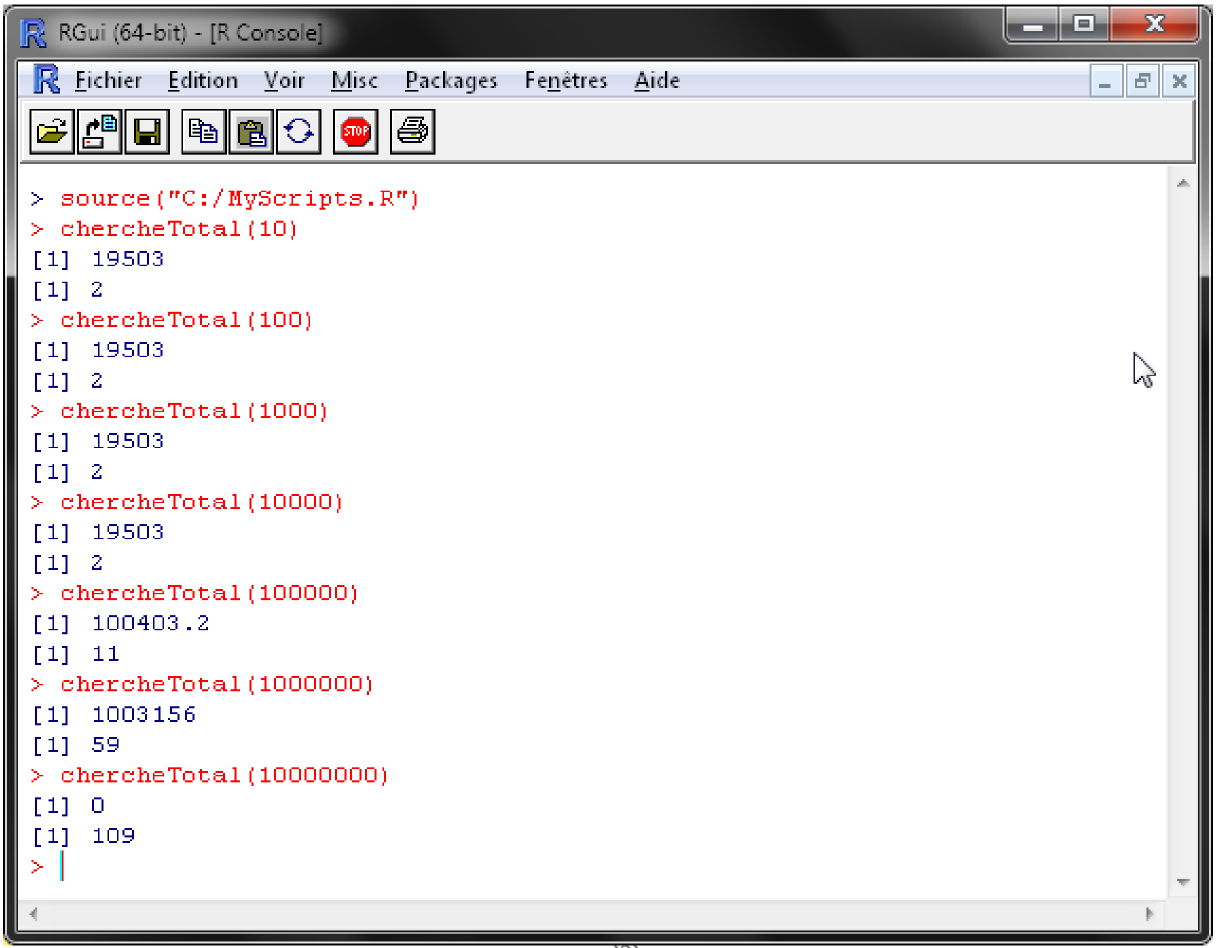
et quand on exécute le script, nous obtenons:



Nous pouvons nous amuser à en faire une fonction avec un paramètre d'entrée:



Ce qui donnera:



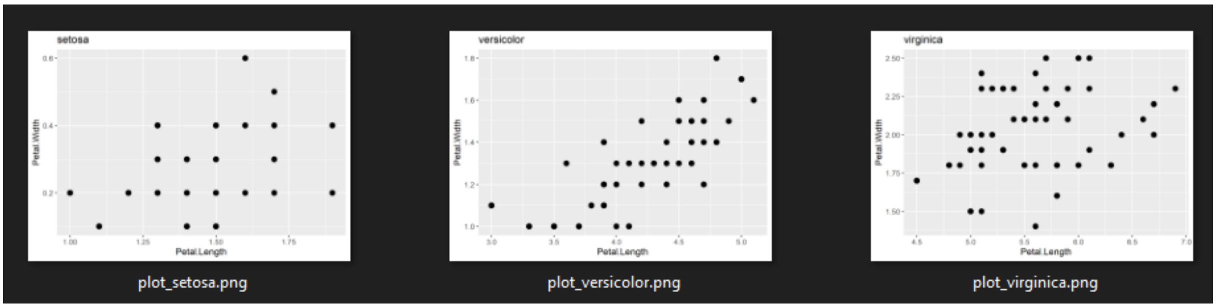
R se base principalement sur la manipulation de vecteurs et matrices donc plutôt que de faire des boucles parfois inutilement pensez vectoriel!!! Un exemple typique consistant à calculer la somme des carrés des erreurs d'un vecteur:

```
RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> #inutilement compliqué avec une boucle
> set.seed(1)
> x<-runif(10, min=1000, max=2000)
> x
 [1] 1265.509 1372.124 1572.853 1908.208 1201.682 1898.390 1944.675 1660.798
 [9] 1629.114 1061.786
> moyenne<-mean(x)
> for (i in 1:10) x[i]<-(x[i]-moyenne)^2
> SSE<-x
> sum(SSE)
[1] 896744.7
>
> #pensé "vectoriel"
> set.seed(1)
> x<-runif(10, min=1000, max=2000)
> x
 [1] 1265.509 1372.124 1572.853 1908.208 1201.682 1898.390 1944.675 1660.798
 [9] 1629.114 1061.786
> moyenne<-mean(x)
> SSE<-(x-moyenne)^2
> sum(SSE)
[1] 896744.7
> |
```


Enfin gardez à l'exprit que les boucles peuvent être utilisées pour générer des graphiques comme l'illustre l'exemple suivant qui utilise **ggsave( )**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
> library("ggplot2")
> data("iris")
>
> # list of values to loop over
>   uniq_species = unique(iris$Species)
>
>
> # Loop
>
> for (i in uniq_species) {
+
+   temp_plot = ggplot(data= subset(iris, Species == i)) +
+                   geom_point(size=3, aes(x=Petal.Length, y=Petal.Width )) +
+                   ggtitle(i)
+
+   ggsave(temp_plot, file=paste0("plot_", i,".png"), width = 14, height = 10, units = "cm")
+ }
> |
```

Ce qui donnera:



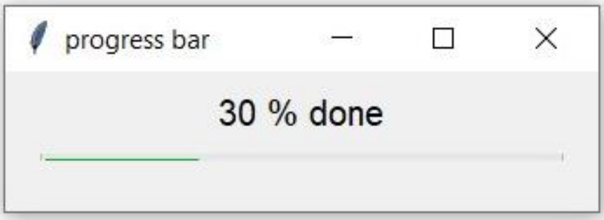
On peut avec les structures itératives créér les très demandées barres de progression comme l'illustrent les trois différentes appproches ci-dessous:

```
> total <- 20
> pb <- txtProgressBar(min = 0, max = total, style = 3)
  |                                                                    |   0%>
> for(i in 1:total){
+    Sys.sleep(0.1)
+    setTxtProgressBar(pb, i)
+ }
  |=========================================                           |  30%
```

```
> pb <- winProgressBar(title = "progress bar", min = 0,
+                      max = total, width = 300)
>
> for(i in 1:total){
+    Sys.sleep(0.3)
+    setWinProgressBar(pb, i, title=paste( round(i/total*100, 0),
+                                          "% done"))
+ }
```




```
> total <- 20
> pb <- tkProgressBar(title = "progress bar", min = 0,
+                     max = total, width = 300)
>
> for(i in 1:total){
+    Sys.sleep(0.3)
+    setTkProgressBar(pb, i, label=paste( round(i/total*100, 0),
+                                         "% done"))
+ }
```



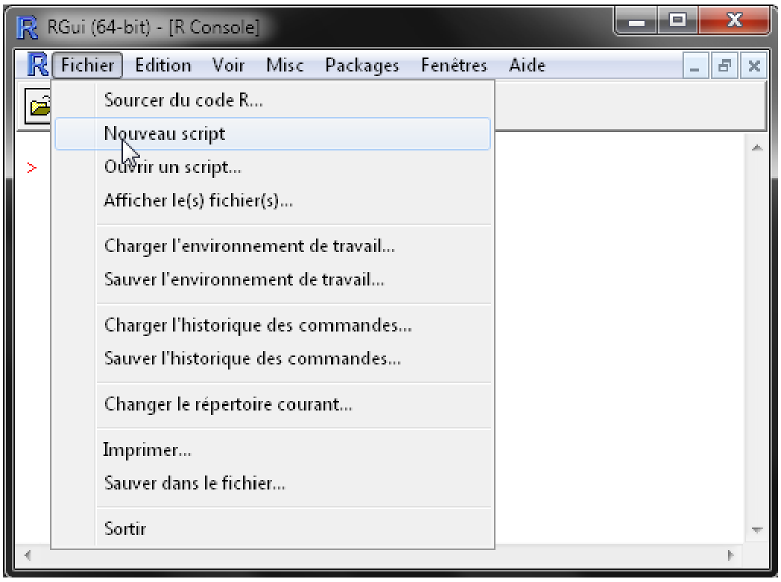
requires tcltk

Exemple 457.: Programmer une bascule à moyennes

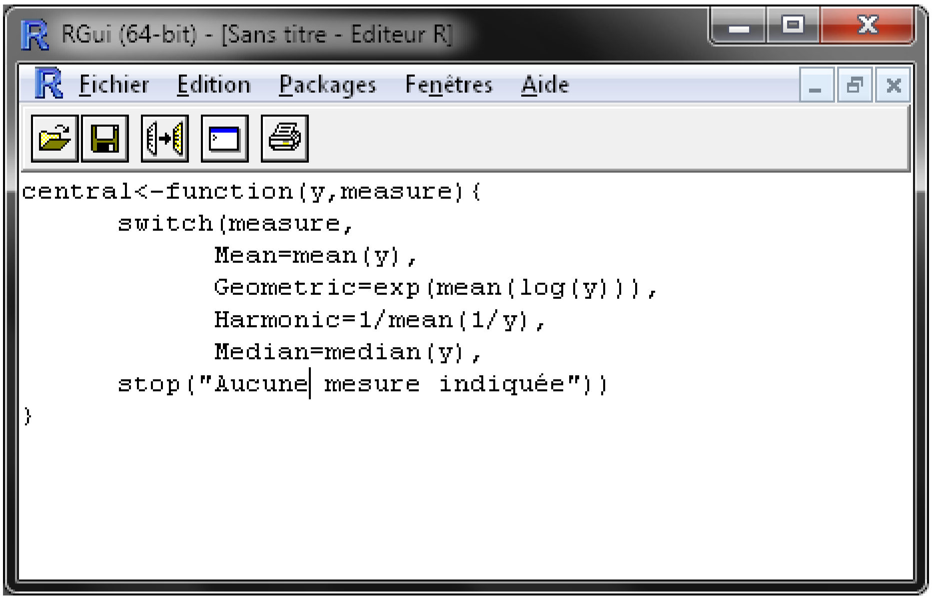
R 3.0.2

Comme premier script, nous allons écrire une fonction qui permet de calculer au choix la moyenne arithmétique, géométrique (ce qui nous évitera de faire appel au package **psych** qui contient une commande **geometric.mean( )** pour la moyenne géométrique), harmonique et la médiane.

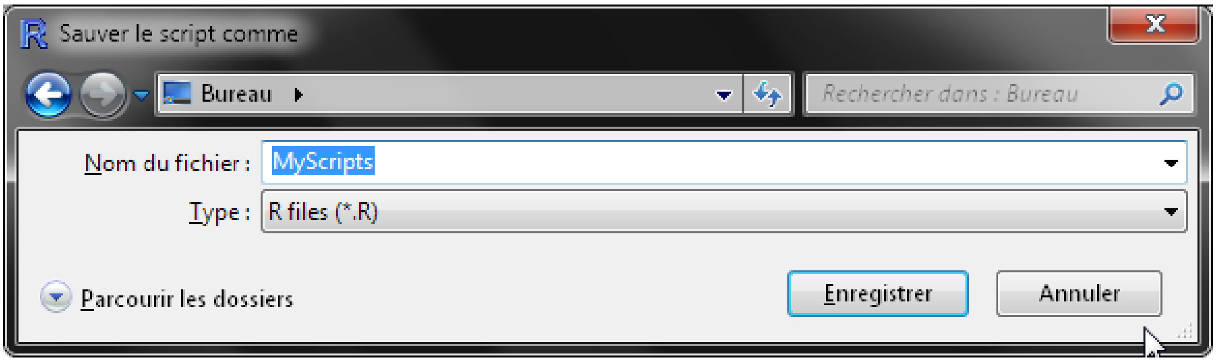
Nous allons donc créer un nouveau script:



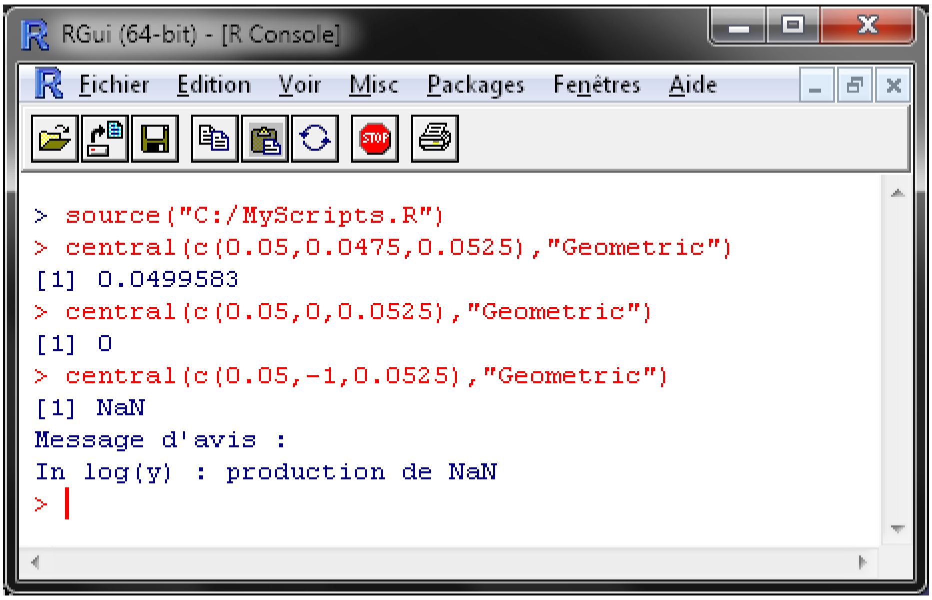
et nous écrivons notre script pour découvrir par la même occasion la structure conditionnelle **switch**:



Ensuite, nous enregistrons notre fichier en tant que fichier script (pouvant contenir plus qu'une fonction):



Une fois ceci fait nous pouvons charger et utiliser le script:



Remarque: Certes ce bout de code est discutable car s'il y a une valeur négative ou nulle dans le vecteur, la moyenne géométrique va renvoyer une erreur (voir le cours théorique pour traiter de ces cas là).

Exemple 458.: Fonctions

Programmer une fonction avec un paramètre en entrée renvoyant un tableau

R 3.0.2

Nous écrivons (en même temps c'est une jolie application de la commande **sapply( )**):

```
r.stat.basics<- function(x){
       m <- mean(x)
       n <- length(x)
       s <- sd(x)
       skew <- sum((x-m)^3/s^3)/n
       kurt <- sum((x-m)^4/s^4)/n - 3
       return(c(n=n, mean=m, stdev=s, skew=skew, kurtosis=kurt))
}
```

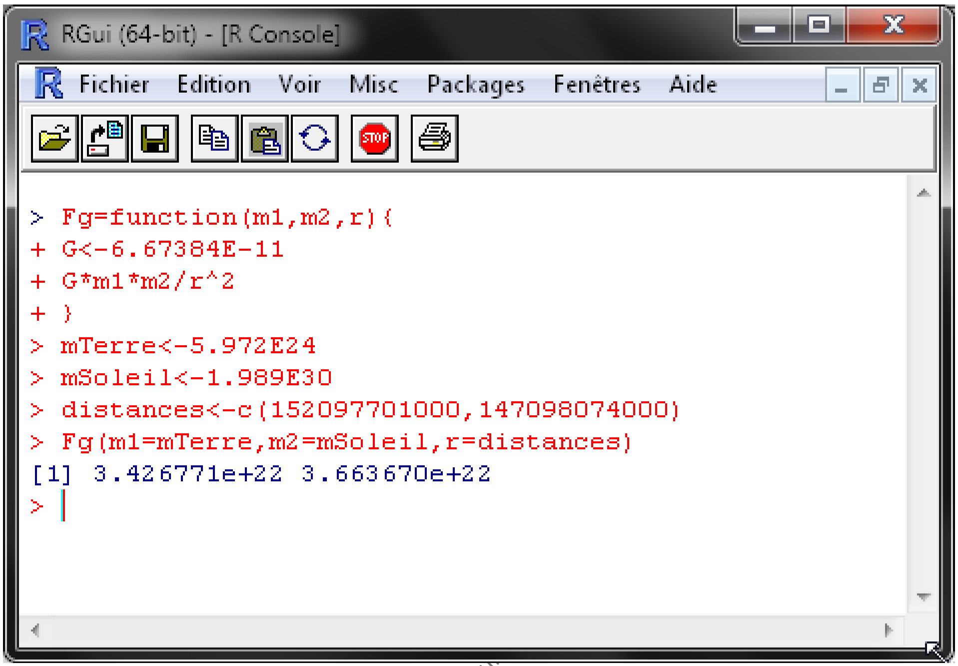
et nous testons:

```
> source("C:/myScripts.R")
> setwd("C:/")
> mydata<-read.csv("VentesClean.csv",header=T,sep=";")
> vars<-c("Quantite","PrixTotalAvecRabais")
> sapply(mydata[vars],r.stat.basics)
            Quantite PrixTotalAvecRabais
n         109.000000          109.000000
mean        8.110092        18784.544771
stdev       6.314797        15122.273003
skew        1.649741            2.143203
kurtosis    2.839111            5.155884
> 
```

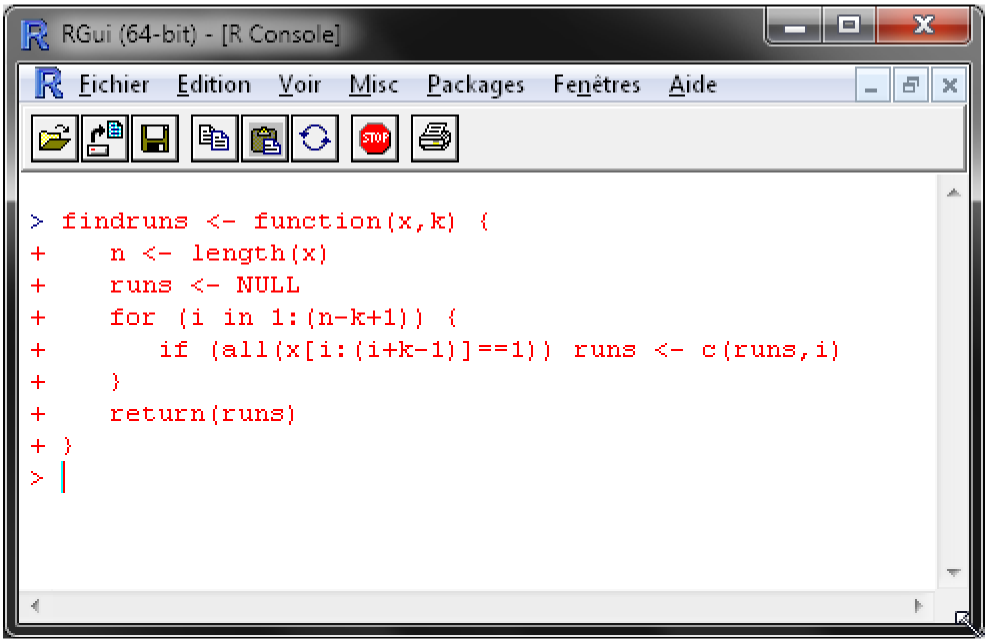
Programmer une fonction avec plus d'un paramètre en entrée

R 3.0.2

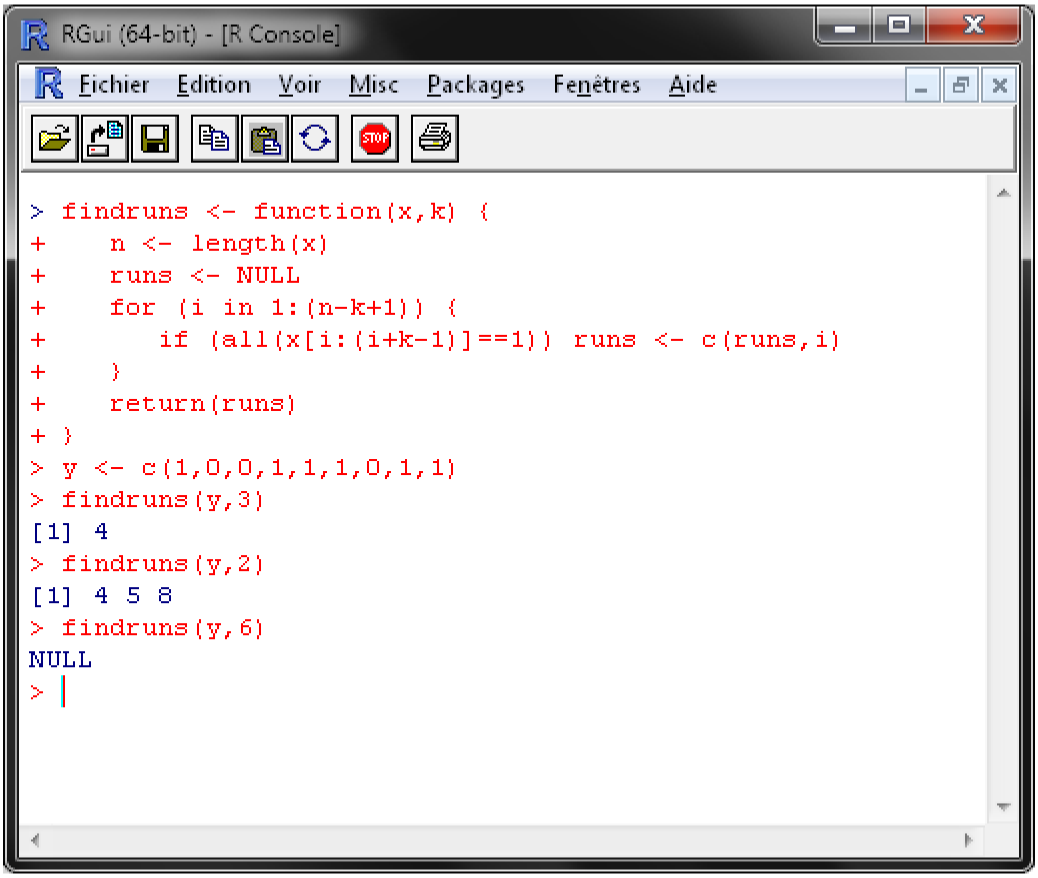
Programmer une fonction R avec deux paramètres en entrée est très similaire à n'importe quel autre langage. Ce qui est drôle c'est de passer un vecteur à un des deux paramètres. Considérons l'exemple physique avec le couple Terre/Soleil et l'apogée et au périgée:



Ou une fonction qui peut être utile dans la pratique en plus d'être instructive (à personnaliser selon les besoins) qui va authentifier dans un vecteur les positions d'une séquence continue de *k* éléments identiques (dans le cas présent il s'agit de "1"):



Avec une entrée cela donnera:


RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> findruns <- function(x,k) {
+ n <- length(x)
+ runs <- NULL
+ for (i in 1:(n-k+1)) {
+ if (all(x[i:(i+k-1)]==1)) runs <- c(runs,i)
+ }
+ return(runs)
+ }
> y <- c(1,0,0,1,1,1,0,1,1)
> findruns(y,3)
[1] 4
> findruns(y,2)
[1] 4 5 8
> findruns(y,6)
NULL
>

Programmer une fonction avec plus d'un paramètres en entrée et utilisant une liste

R 3.0.2

Il s'agit ici probablement du cas le plus courant dans les complications pratiques du logiciel.

Nous créons le script suivant:

```
C:\myScripts.R - Editeur R
basic.stats <- function(x,more=F) {
 stats <- list()

 clean.x <- x[!is.na(x)]

 stats$n <- length(x)
 stats$nNAs <- stats$n-length(clean.x)

 stats$mean <- mean(clean.x)
 stats$std <- sd(clean.x)
 stats$med <- median(clean.x)
 if (more) {
      stats$skew <- sum(((clean.x-stats$mean)/stats$std)^3)/length(clean.x)
      stats$kurt <- sum(((clean.x-stats$mean)/stats$std)^4)/length(clean.x)-3
 }
 unlist(stats)
}
```

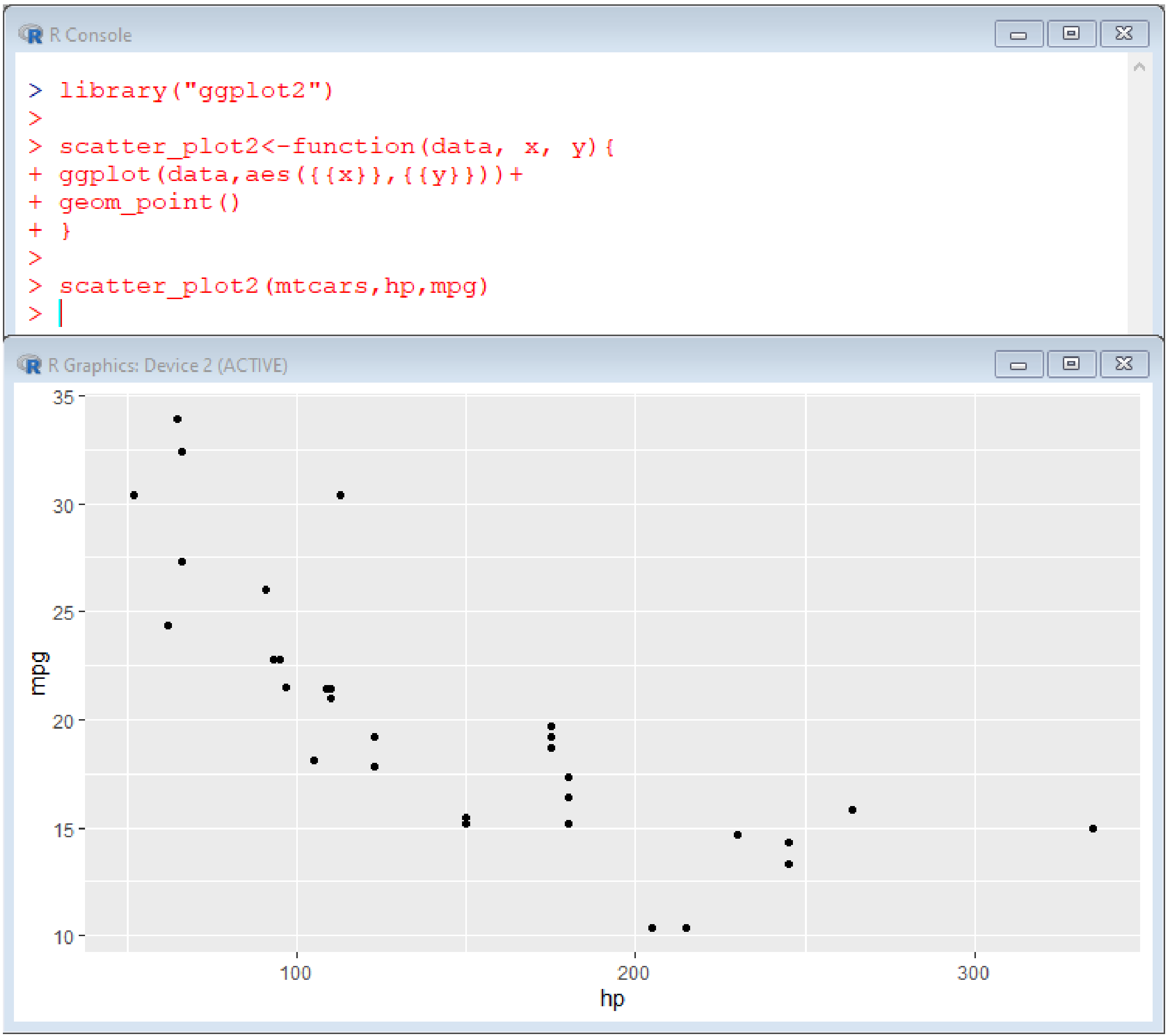
Et nous le testons sans options dans un premier temps:

```
R Console
> source("C:/MyScripts.R")
> basic.stats(c(45,2,4,46,44,65,NA,6,-213,-3,-45))
       n     nNAs     mean      std      med
11.00000  1.00000 -4.90000 79.94505  5.00000
>
```

et avec options dans un deuxième temps:

```
R Console
> basic.stats(c(45,2,4,46,44,65,NA,6,-213,-3,-45),TRUE)
        n      nNAs      mean       std       med      skew      kurt
11.000000  1.000000 -4.900000 79.945051  5.000000 -1.636075  1.701595
>
```

Attention les paramètres peuvent parfois nécessiter une syntaxe spéciale pour pouvoir fonctionner comme l'illustre cet exemple avec **ggplot2** (notez les accolades autour des variables dans la fonction!):



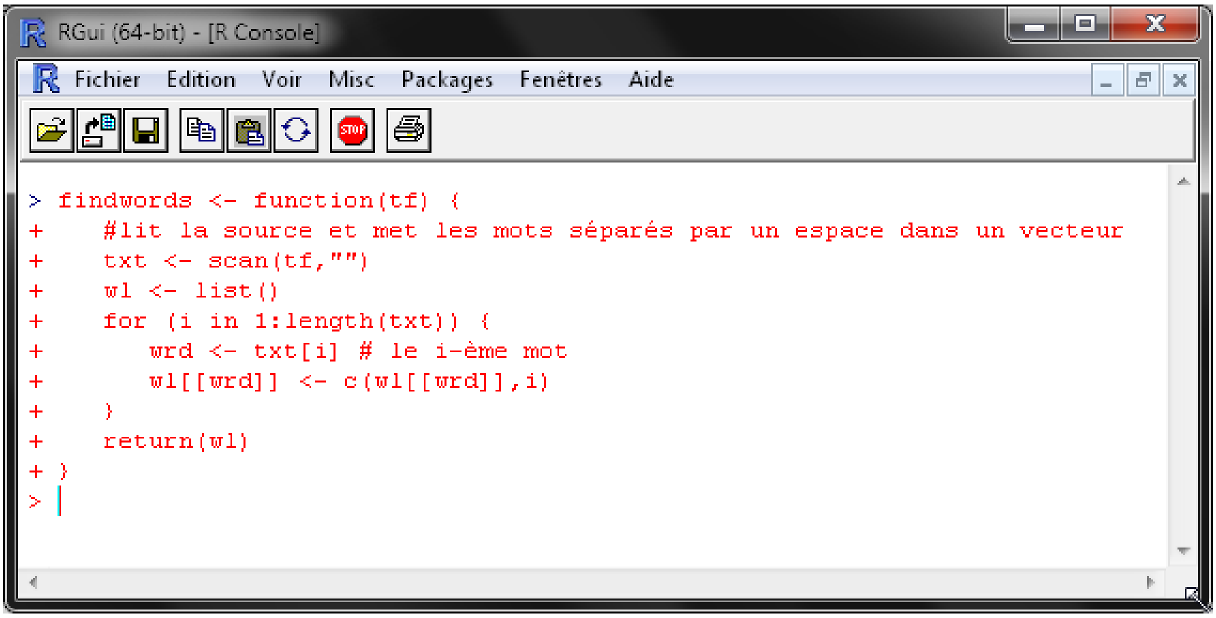
Programmer une fonction avec un fichier texte en entrée

R 3.0.2

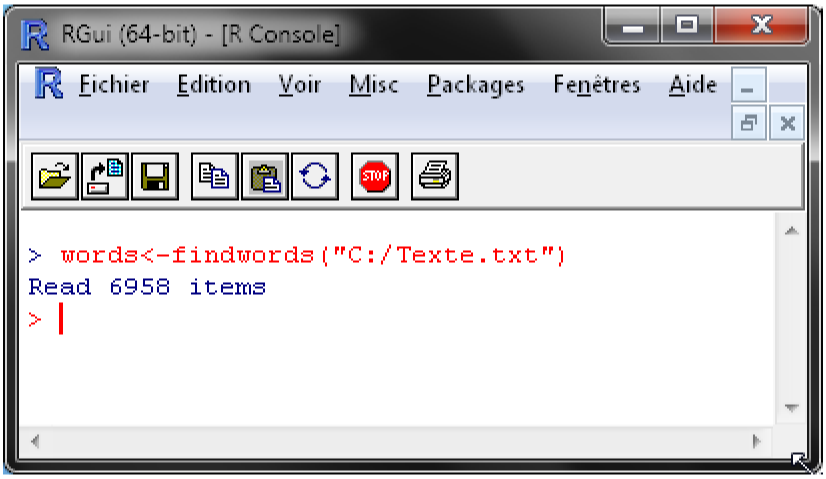
Écrire une fonction qui prend un fichier texte en entrée est identique à écrire une fonction qui prend en entrée un scalaire ou un vecteur.

Pour cet exemple, nous allons prendre un fichier texte simple et générer un objet de type liste dont chaque entrée correspondra à un mot se trouvant dans le fichier texte et nous associerons à chacune de ses entrées, un vecteur contenant la position des endroits où ce même mot a été trouvé.

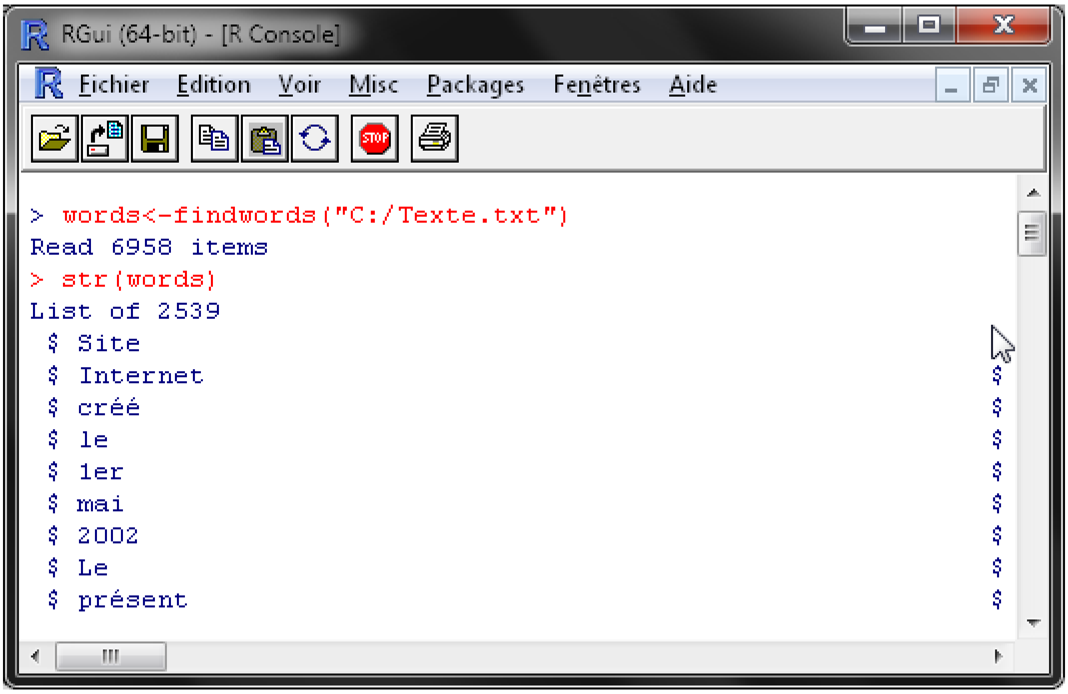
La fonction dans un premier temps est la suivante (prenez le temps de lire car c'est instructif!):



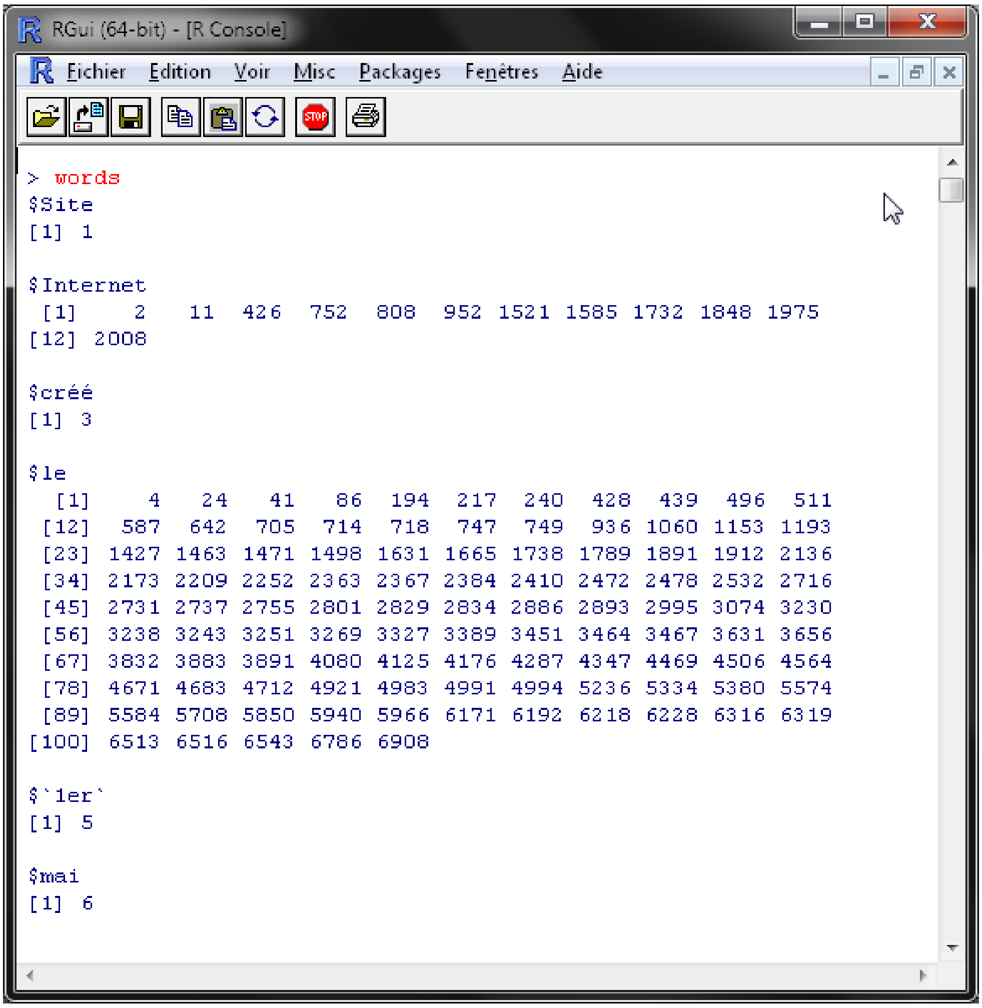
Ensuite, nous utilisons la fonction:



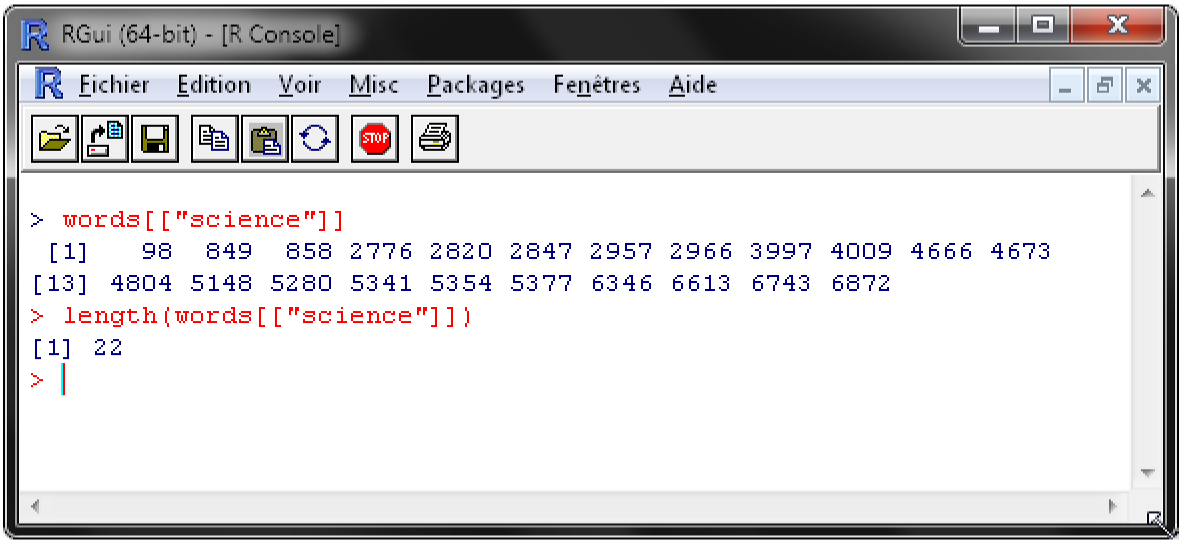
Un résumé de la liste nous donnera:



et un affichage complet (mais dont la capture d'écran est tronquée):



Ou pour un mot précis:

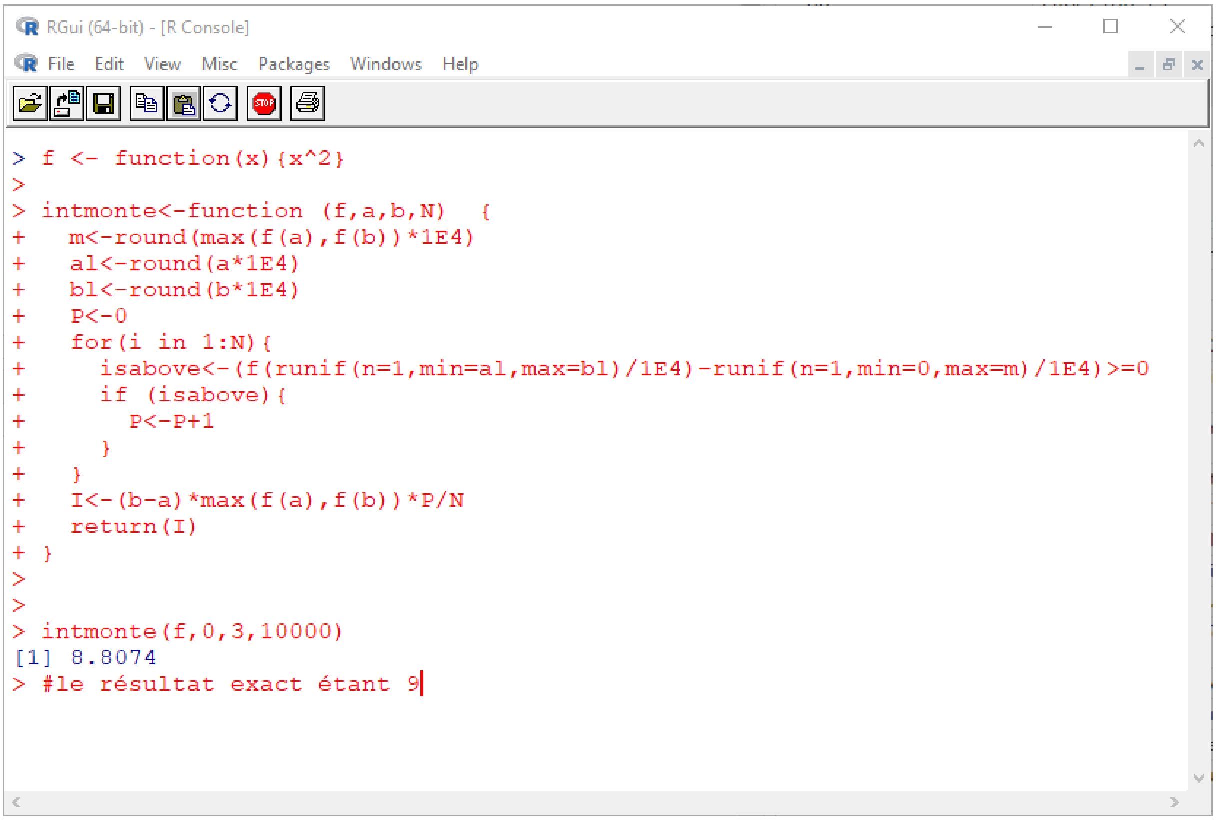


et ainsi de suite avec toutes les techniques déjà étudiées pour les listes au début de cet e-book.

Programmer une fonction avec une fonction quelconque en entrée

R 3.61

Nous allons ici simplement reprendre l'exemple rencontré lors de notre étude des méthodes de Monte Carlo:



Programmer une fonction avec des arguments indéfinis

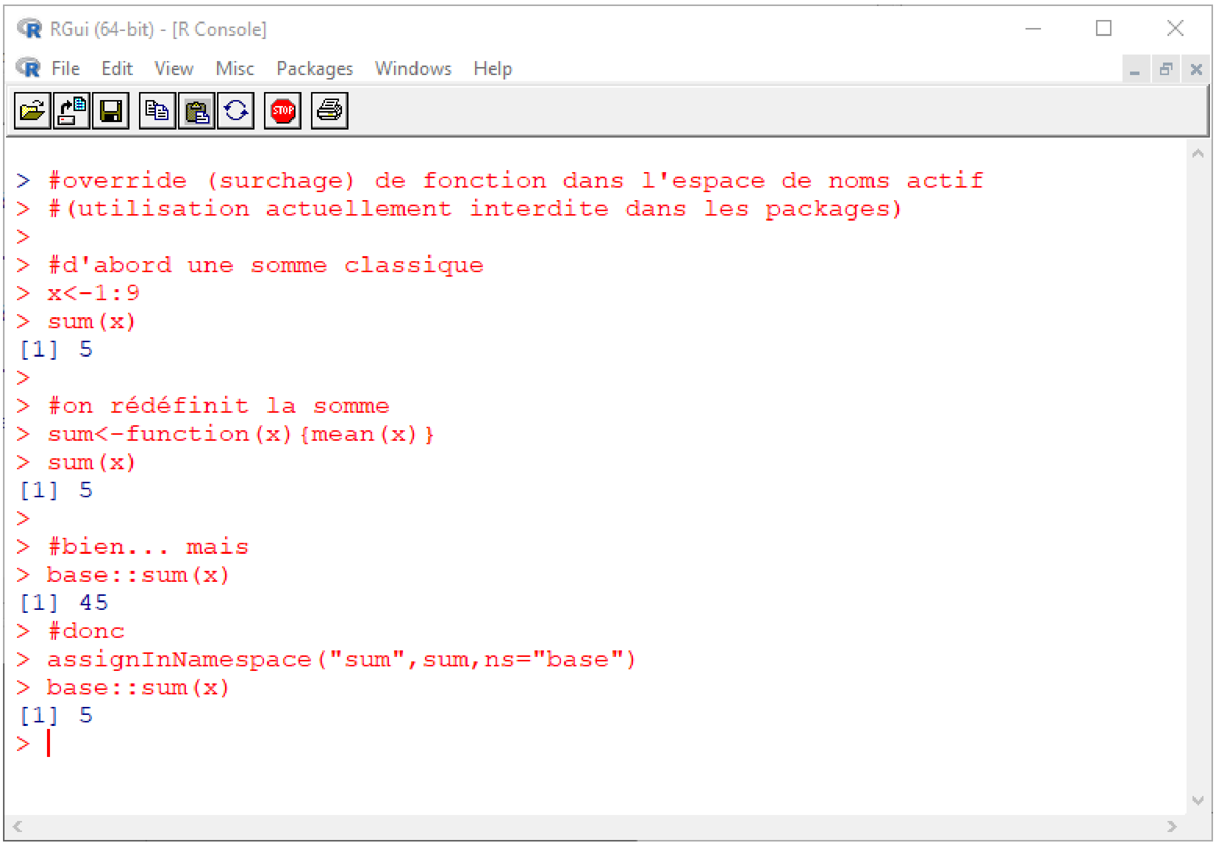
R 3.2.1

ToDo

Surcharge de fonction dans l'espace de noms par défaut

R 3.6.2

L'exemple suffit à lui-même:



Calculer Pi avec la méthode de Monte-Carlo

R 3.0.2

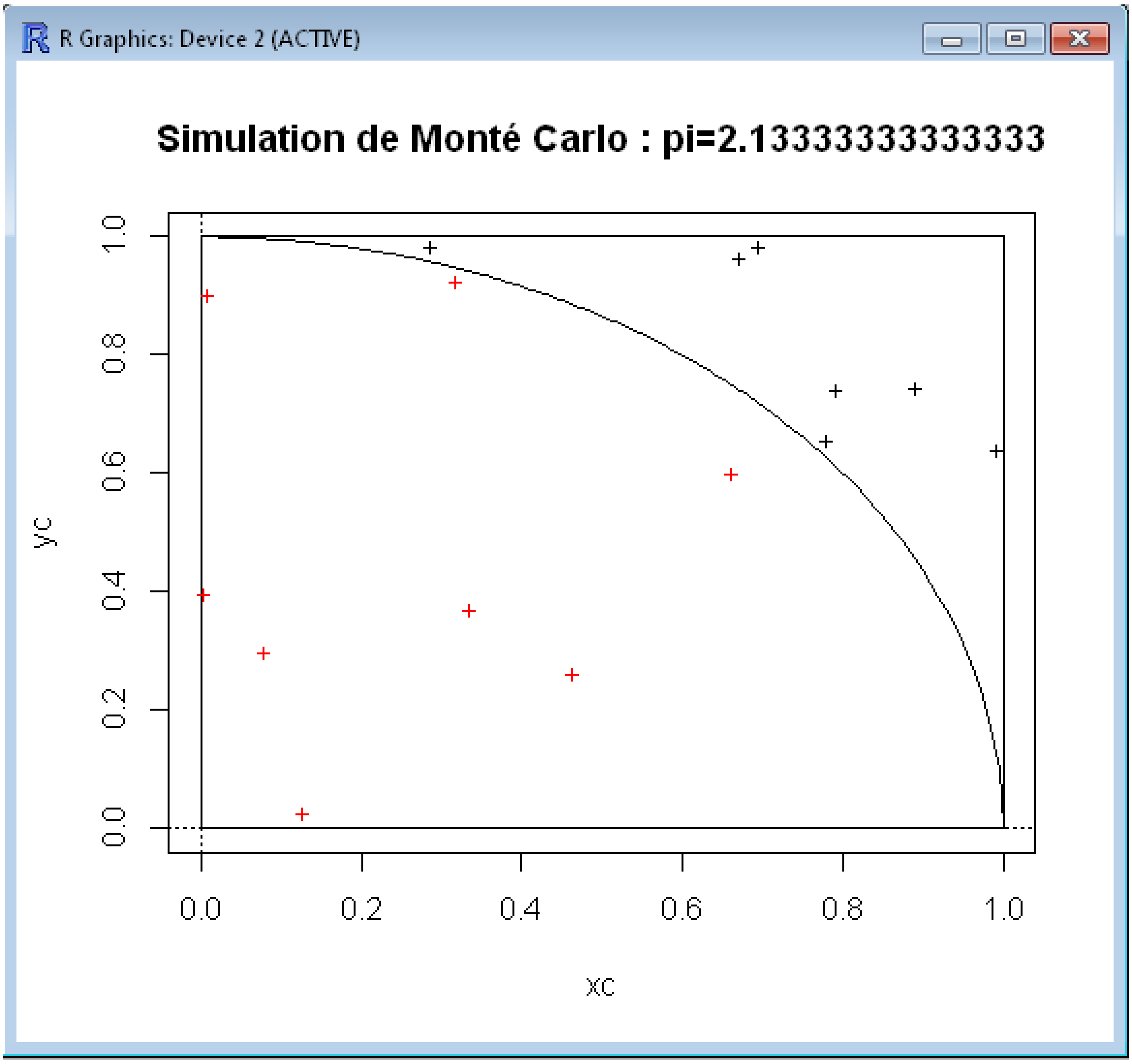
Un grand classique qui a son utilité pour être étendu à de nombreux autres domaines dans la pratique (nous avons déjà vue un autre version de ce script lors de notre étude des méthodes de Monte Carlo bien plus haut!):

```
> pi.monte.carlo.plot <- function (n=10000, pch='.', ...) {
+   #on genere des points que dans le premier quadrant du cercle
+   x <- runif(n)
+   y <- runif(n)
+   #on regarde les points qui sont à l'intérieur
+   interieur <- x^2 + y^2 <= 1
+   p <- 4*sum(interieur)/n
+   #on génère un quart de cercle
+   xc <- seq(0,1,length=200)
+   yc <- sqrt(1-xc^2)
+   plot( xc, yc, type='l' )
+   #un peu d'esthétique
+   lines( c(0,1,1,0,0), c(0,0,1,1,0) )
+   abline(h=0, lty=3)
+   abline(v=0, lty=3)
+   #on trace les points
+   points(x[interieur], y[interieur], col='red', pch=pch)
+   points(x[!interieur], y[!interieur], pch=pch)
+   title(main=paste("Simulation de Monté Carlo : pi=",p,sep=''))
+ }
> 
```

Avec la boucle pour faire tourner le script *N* fois:

```
> N<-100
> for(i in 1:100){
+   pi.monte.carlo.plot(i, pch='+', cex=3)
+   #une petite pause pour que le lecteur puisse voir le résultat
+   readline("Press <return to continue")
+ }
Press <return to continue
```

Ce qui donne à chaque étape successive:



Ou une façon plus amusante avec un indicateur d'avancement de calcul (très utilisé dans la pratique) combinant les commandes **winProgressBar( )** et **Sys.sleep( )** :

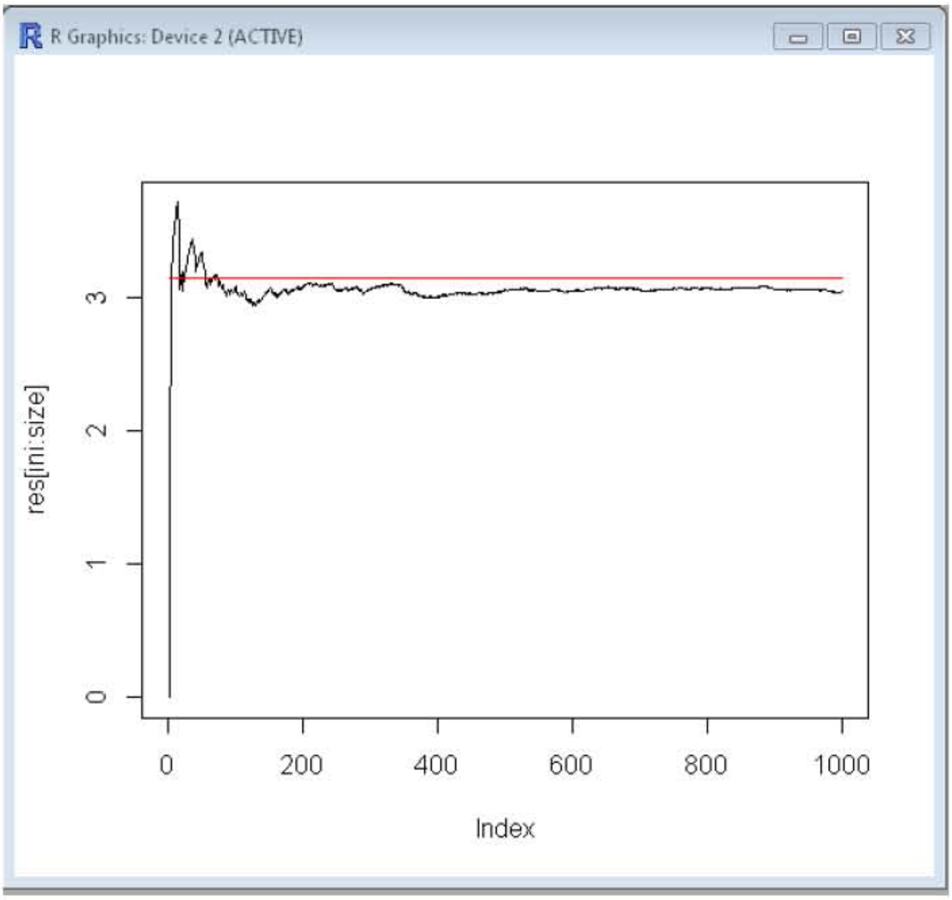
```
R Console
> N<-100
> for(i in 1:100){
+   total<-N
+   #on regenère la barre de progression à chaque fois
+   #car il n'y pas à ma connaissance de commande pour la forcer à rester
+   #devant le graphique
+   pb <- winProgressBar(title="progress bar",min=0,max=total,width=300)
+   setWinProgressBar(pb, i, title=paste( round(i/total*100, 0),"% done"))
+   Sys.sleep(2)
+   close(pb)
+   pi.monte.carlo.plot(i, pch='+', cex=3)
+   #une petite pause pour que le lecteur puisse voir le résultat
+ }

> 
```

Signalons encore une autre approche qui montre la convergence du calcul mais utilisant cette fois-ci le disque en entier (et pas que le premier quadrant):

```
R Console
> simulation = function(long){
+   c = rep(0,long)
+   numberIn = 0
+   for(i in 1:long){
+     x = runif(2,-1,1)
+     if(sqrt(x[1]*x[1] + x[2]*x[2]) <= 1){
+       numberIn = numberIn + 1
+     }
+     prop = numberIn / i
+     piHat = prop *4
+     c[i] = piHat
+   }
+   return(c)
+ }
>
> size=1000
> res=simulation(size)
> ini=1
> plot(res[ini:size],type="l")
> lines(rep(pi, size)[ini:size],col="red")
> 
```

Ce qui donne:


R Graphics: Device 2 (ACTIVE)
res[ini:size]
0
1
2
3
0
200
400
600
800
1000
Index

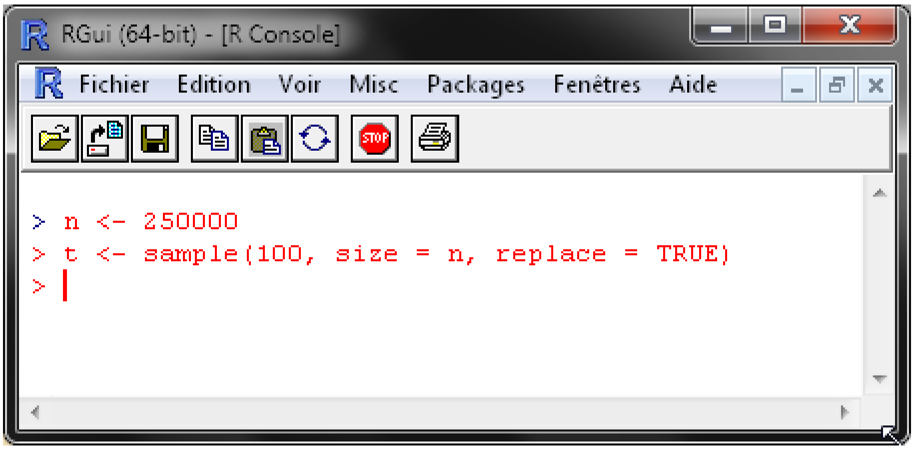
Exemple 459.: Optimisation de scripts

R 3.0.2

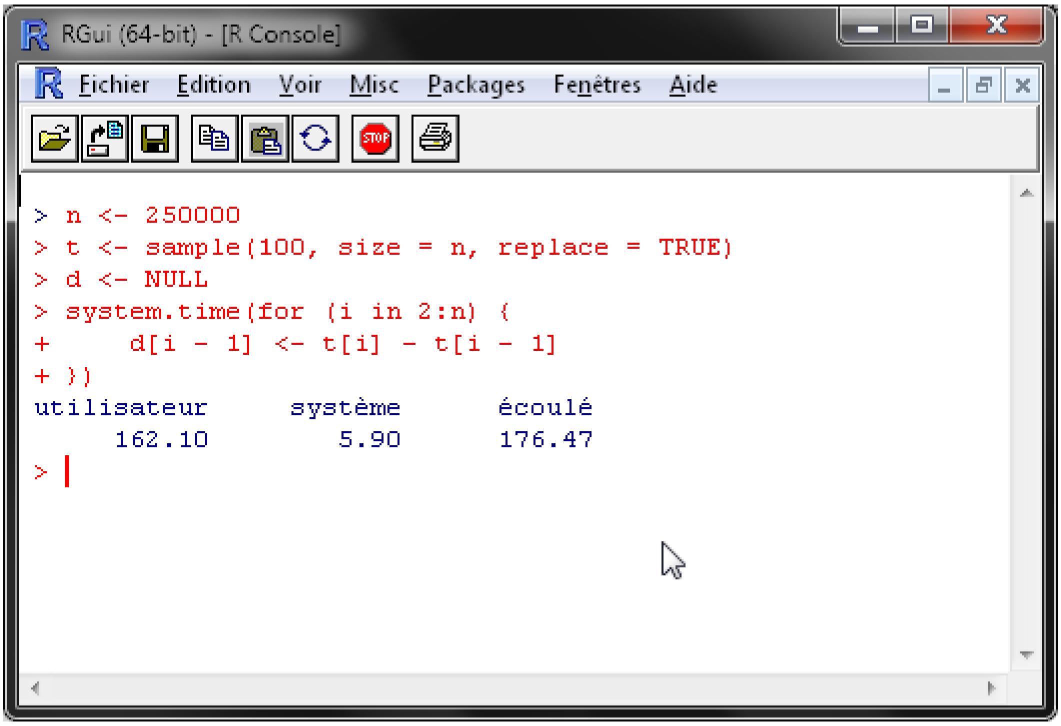
Dans les grosses applications, l'optimisation de scripts est un élément crucial (c'est un métier en soit par ailleurs). Voyons un joli petit exemple avec R qui consiste à calculer les différences (variations) éléments par éléments d'un vecteur par ordre d'apparition (certes il existe un package qui contient déjà cela mais bon c'est instructif quand même!).

Vectorisation

Nous créons d'abord un vecteur de taille relativement conséquente:

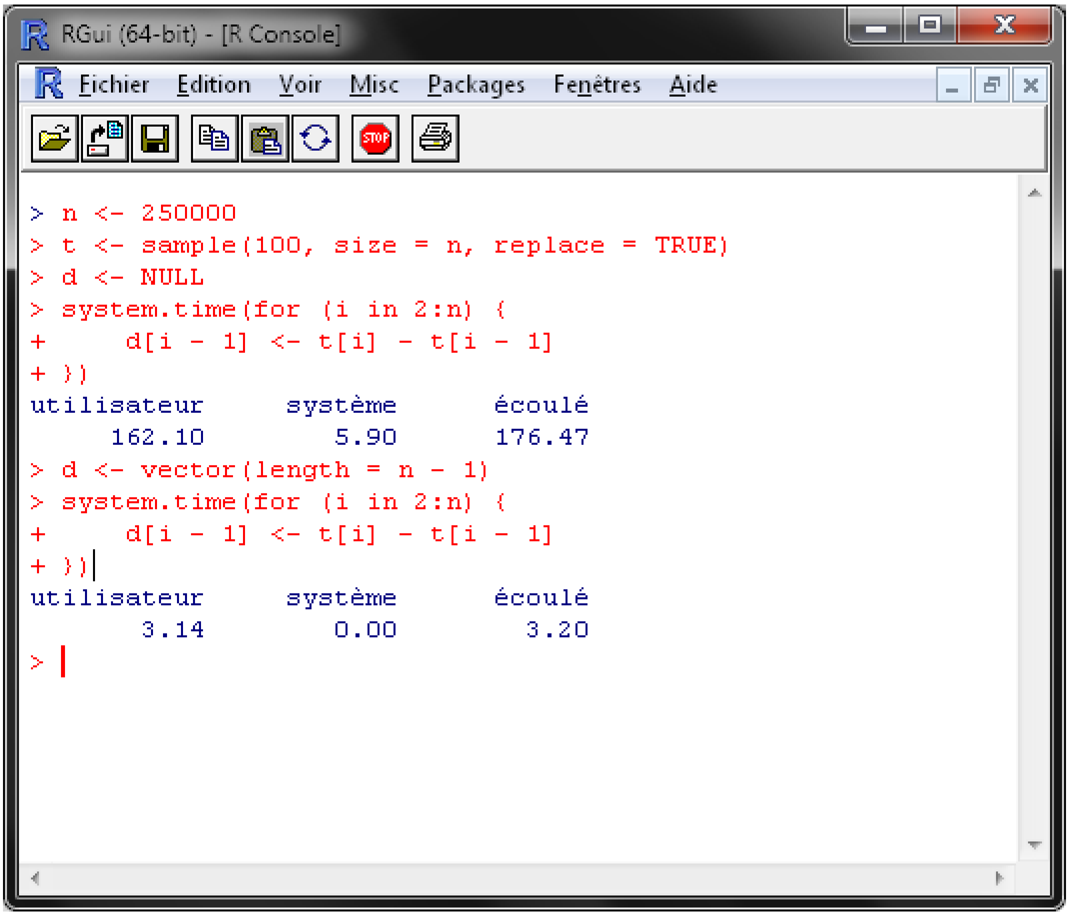


Maintenant faisons le code "au pire" qui consiste à générer un vecteur des différences avec une simple boucle:



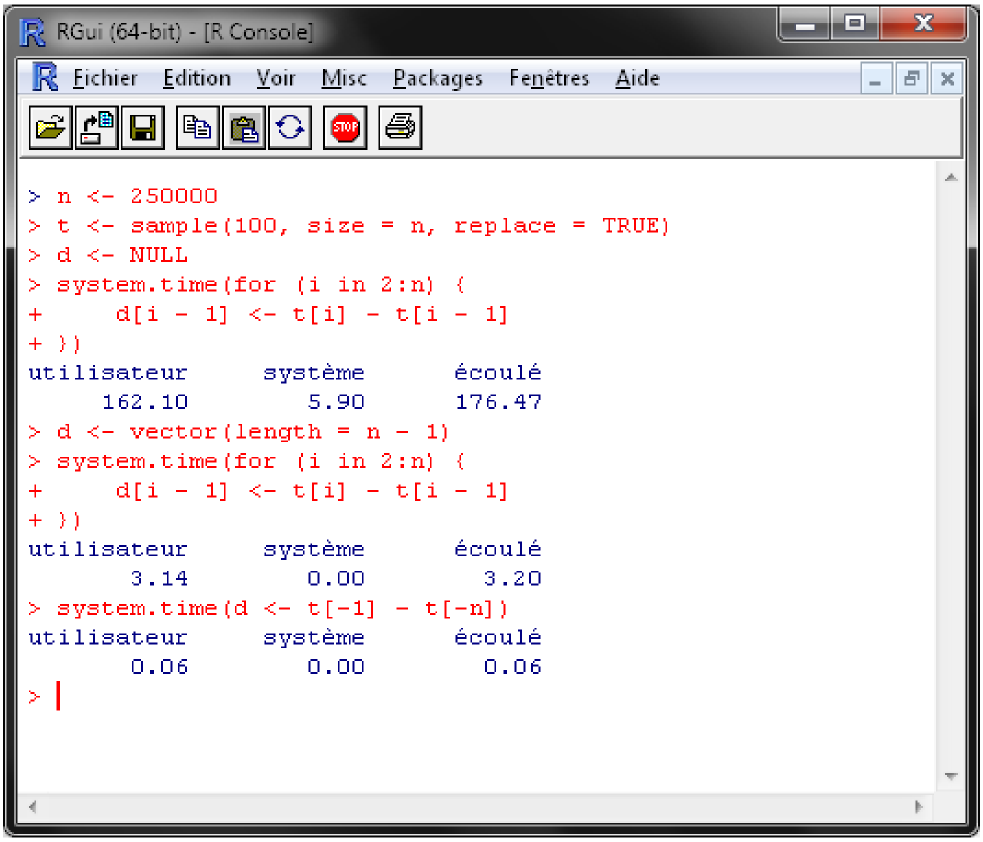
Comme nous pouvons le voir... c'est même plus lent qu'un tableur...

Mais nous pouvons faire mieux ne serait-ce qu'en créant le vecteur des différences avant la boucle:



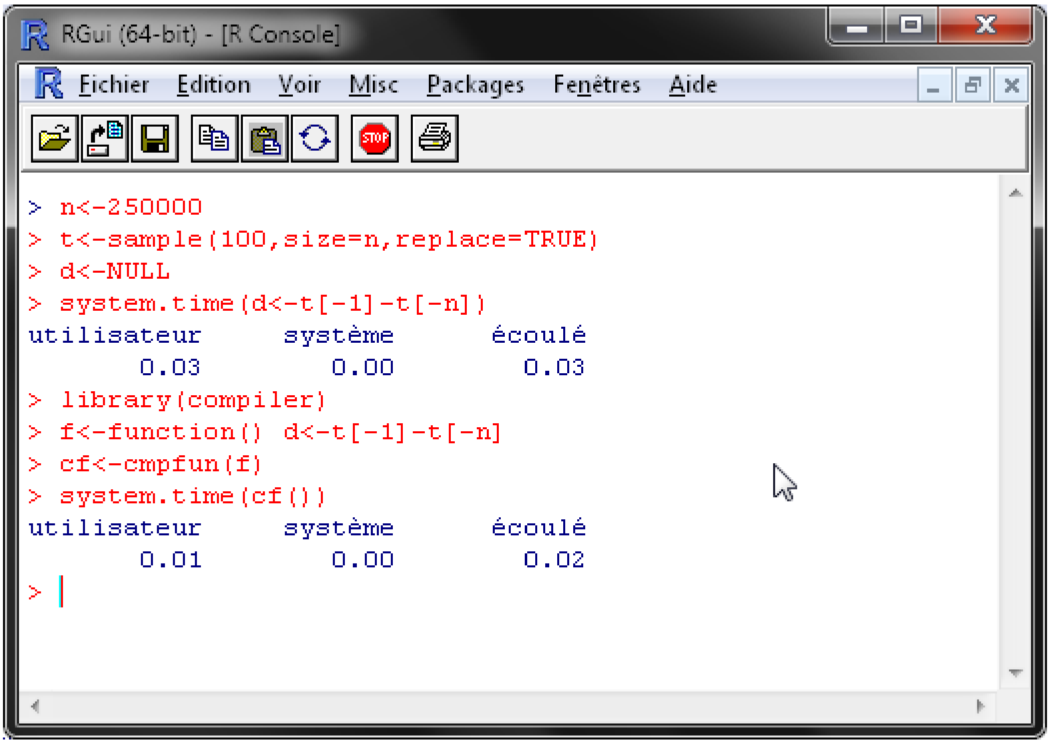
La différence est déjà phénoménale!

Mais nous pouvons faire encore mieux en utilisant les capacités vectorielles de R:

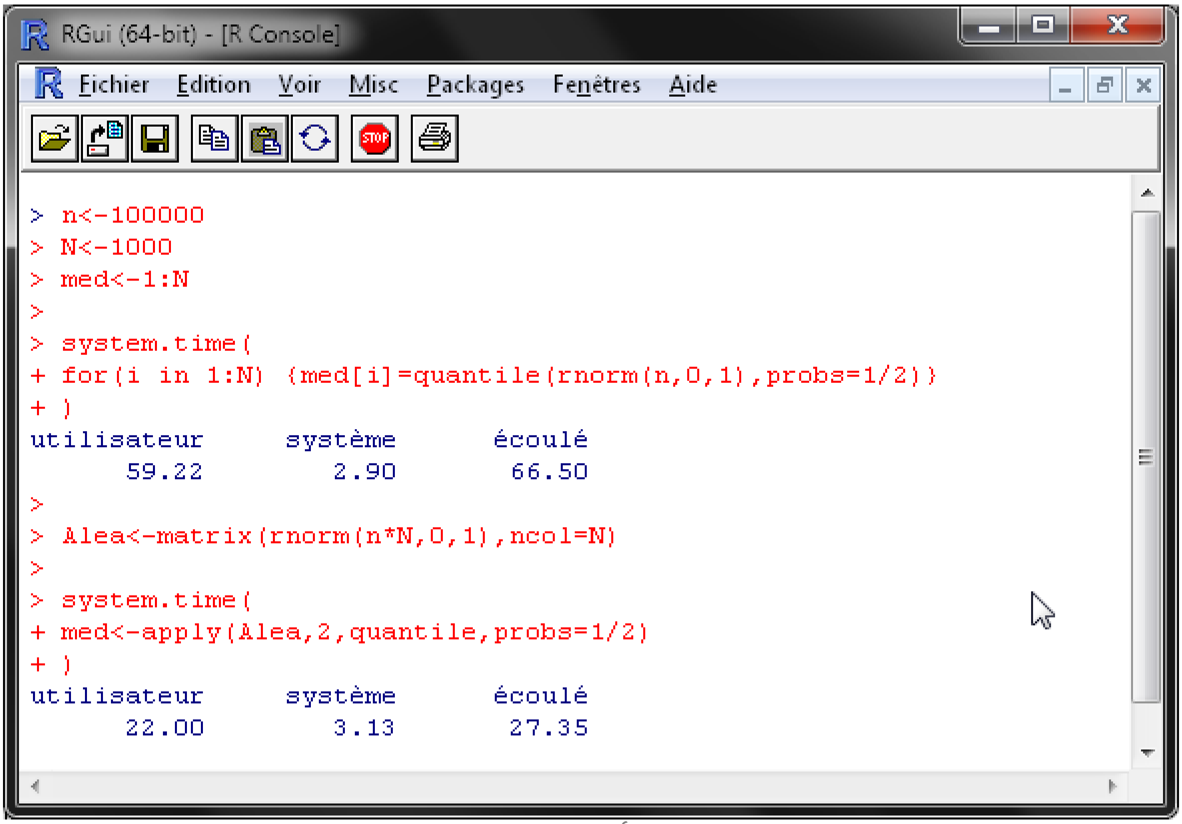


Comme quoi si la force de l'astuce est avec toi... maître de R tu seras...

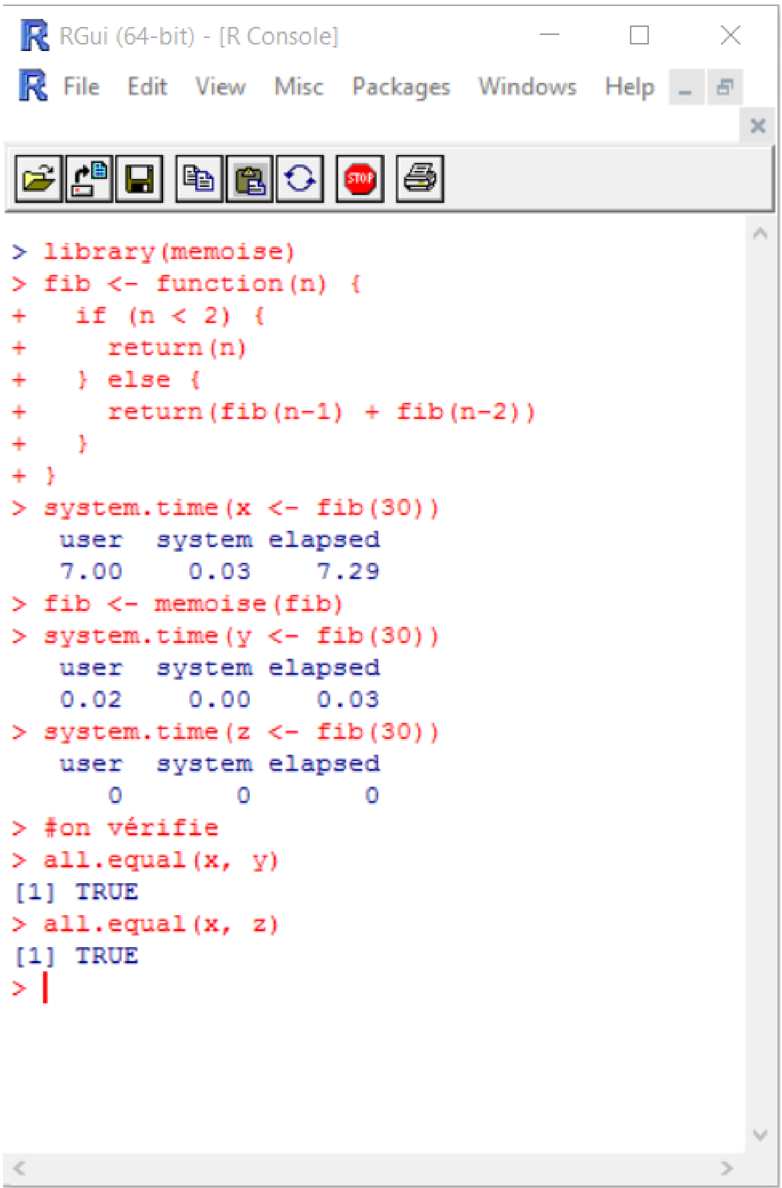
Statistiquement nous pouvons aller encore plus vite en utilisant le package natif compilé de R et sa fonction **cmpfun**:



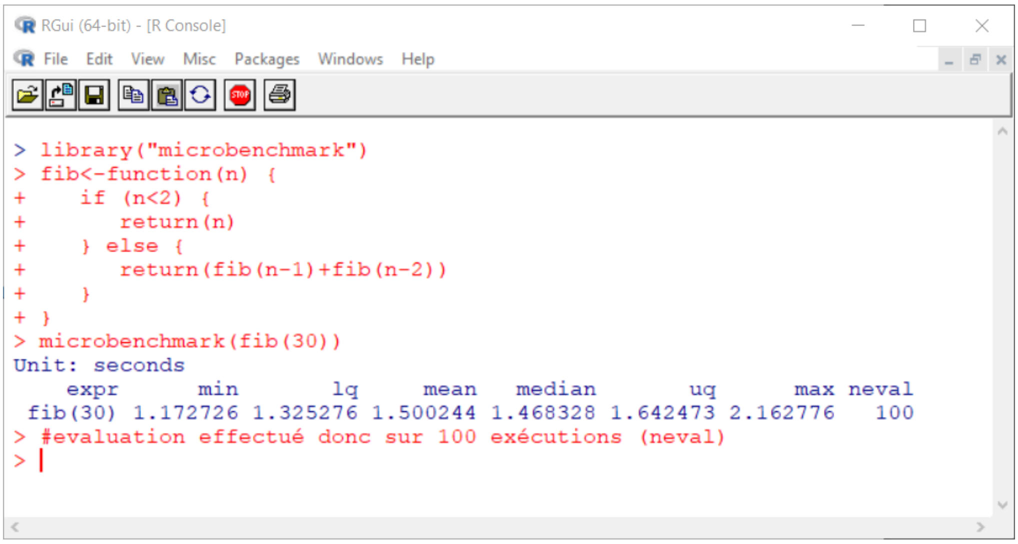
Dans le même genre sinon pour calculer la médiane de *N* échantillons i.i.d de taille *n* dans un vecteur *med* avec une magnifique application de la commande **apply( )** :



Enfin signalons le package **memoise** et sa fonction correspondante **memoise( )** aussi applicable lors de la lecture de bases de données ou l'appel à des API:



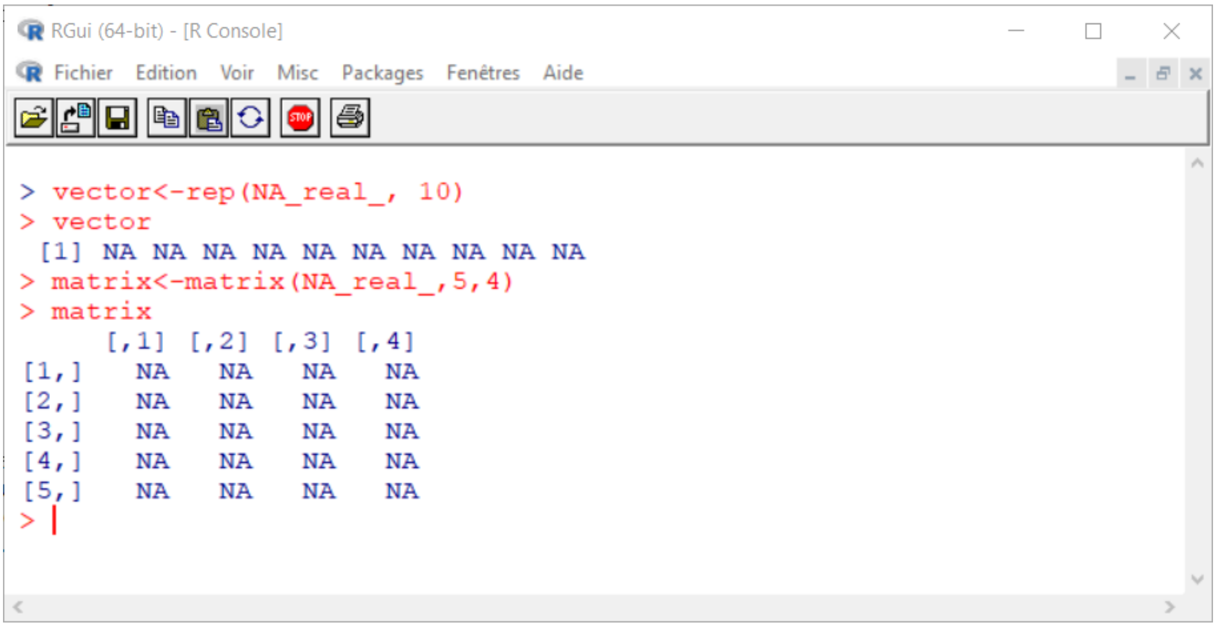
Nous pouvons aussi évaluer avec la fonction **microbenchmark( )** du package… **microbenchmark**:



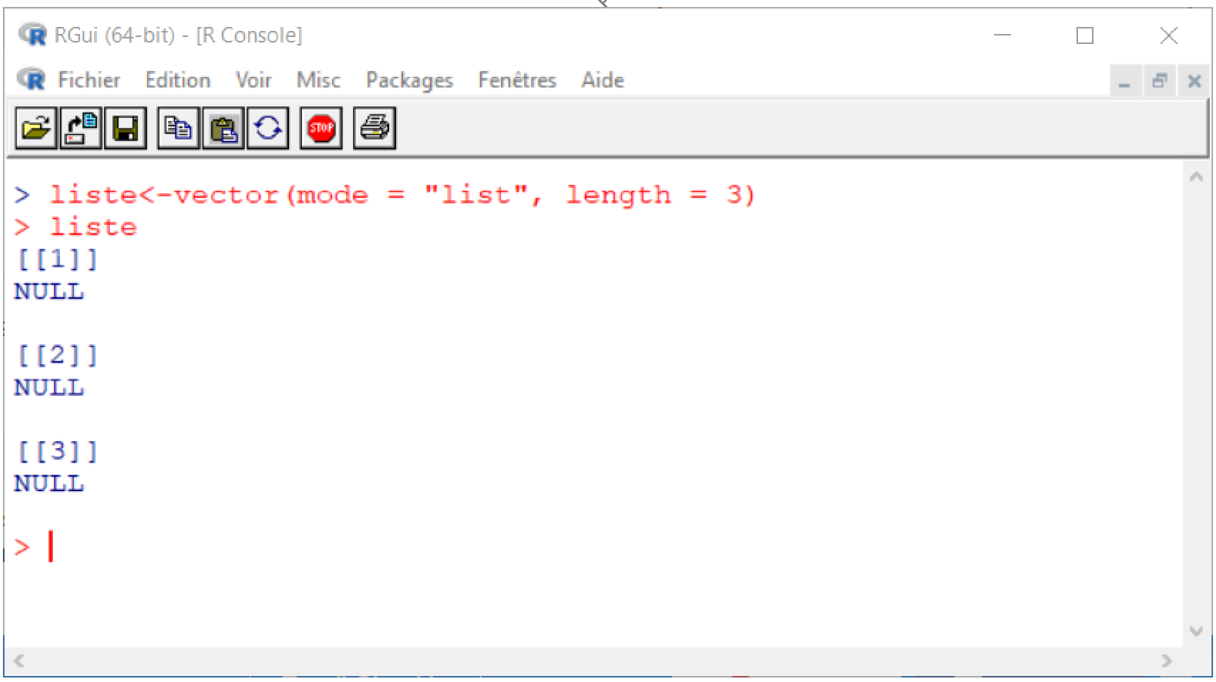
Pré-allocations

Parfois pour gagner en rapidité, cela vaut la peine de pré-allouer un vecteur (in extenso une matrice) ou une liste dans R.

Voyons comment faire pour un vecteur et une matrice:



Et pour une liste c'est un peu plus subtil…:



Exemple 460.: Gestion des erreurs et des exceptions

R 3.1.1

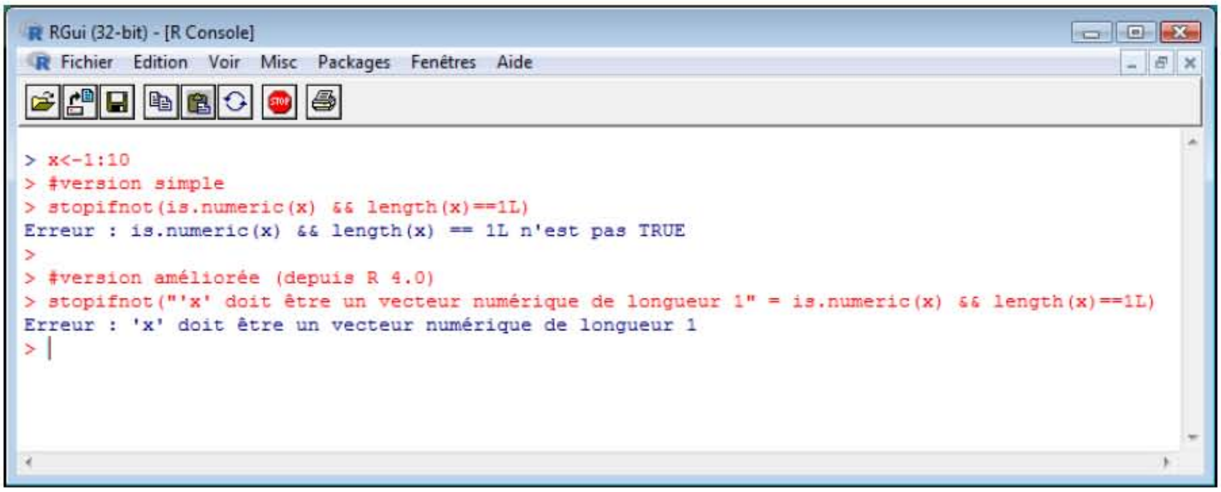
Il est normalement inutile à ce niveau d'utilisation d'un logiciel informatique de préciser ce dont à quoi sert la gestion des erreurs et des exceptions puisque cela est au programme de première année. Nous allons donc voir simplement comment mettre en ouvre cela avec R en utilisant un exemple concret et fréquent faisant usage de la fonction **tryCatch( )**:

```
#on reset les variables en mémoires pour ne pas biaiser l'exemple
rm(list=ls(all=TRUE))

#il faut mettre le trycatch dans une fonction
readUrl <- function(url) {
tryCatch(
        {
            # this is the try part
            message("The 'try' part is run")
            bolError<-0
            mydata<-read.csv(url,header=T,sep=";")
        },
        error=function(cond) {
            message(paste("URL does not seem to exist:", url))
            message("Here's the original error message:")
            message(cond)
            # Choose a return value in case of error
            bolError<-1
        },
        warning=function(cond) {
            message(paste("URL caused a warning:", url))
            message("Here's the original warning message:")
            message(cond)
            bolError<-1
        },
        finally={
            # Here goes everything that should be executed at the end,
            # regardless of success or error.
            message(paste("Processed URL:", url))
            if(bolError==0){return(mydata)}
        }
)
}
url <- "C:/tmp/VentesClean.csv" #fait exprès ensuite de faire un faux pointage
readUrl(url)
```

Ensuite il n'y a qu'à jouer avec en exécutant la fonction et ensuite en l'appelant avec les deux dernières lignes du script visible ci-dessus.

On peut aussi dans des cas plus simples utiliser la fonction **stopifnot( )** comme l'illustre le petit exemple ci-dessous:

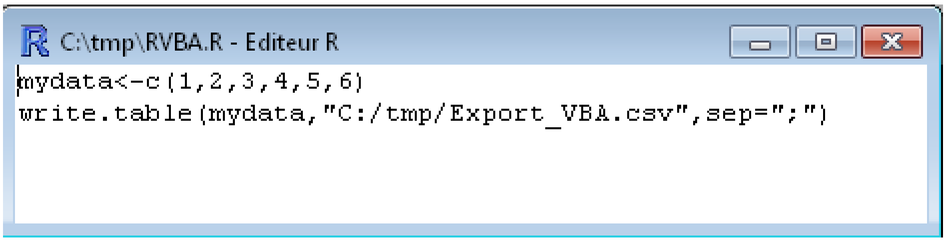

RGui (32-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> x<-1:10
> #version simple
> stopifnot(is.numeric(x) && length(x)==1L)
Erreur : is.numeric(x) && length(x) == 1L n'est pas TRUE
>
> #version améliorée (depuis R 4.0)
> stopifnot("'x' doit être un vecteur numérique de longueur 1" = is.numeric(x) && length(x)==1L)
Erreur : 'x' doit être un vecteur numérique de longueur 1
>

Exemple 461.: Exécuter des scripts *.R avec VBA (depuis MS Excel/Word ou autre)

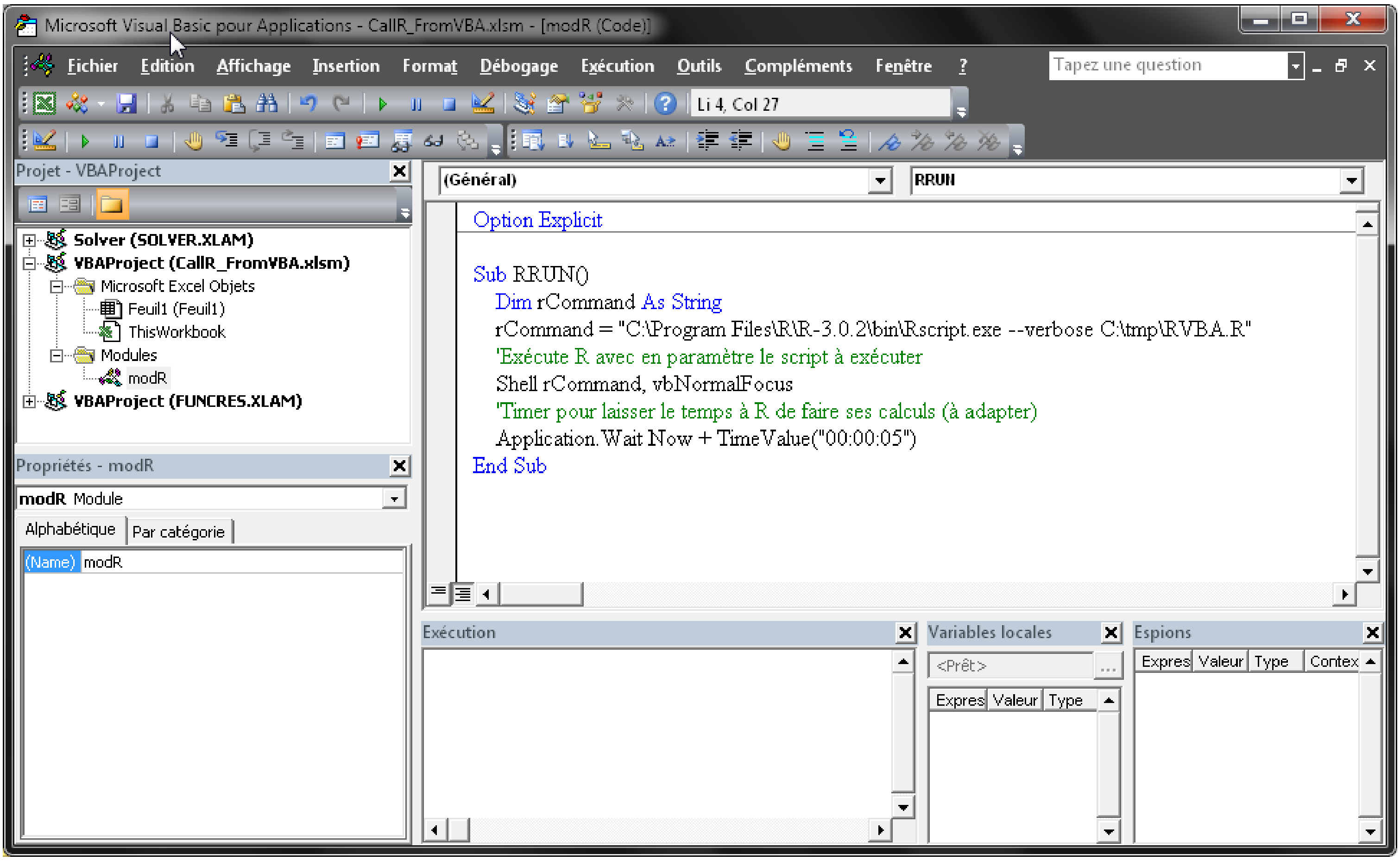
R 3.0.2

Un grand nombre d'entreprises utilisent Microsoft Excel pour faire des "statistiques". L'outil étant très limité, une possibilité très intéressante est de faire des routines VBA qui exportent les données en *.csv, laisser R faire les analyses et ensuite exporter les résultats ou graphiques sous forme d'image pour les importer enfin dans MS Excel. Nous allons ici juste montrer le principe qui se généralise ensuite à des cas beaucoup plus complexes. Pour plus d'informations, je renvoie le lecteur à mon e-book sur le VBA dans MS Excel.

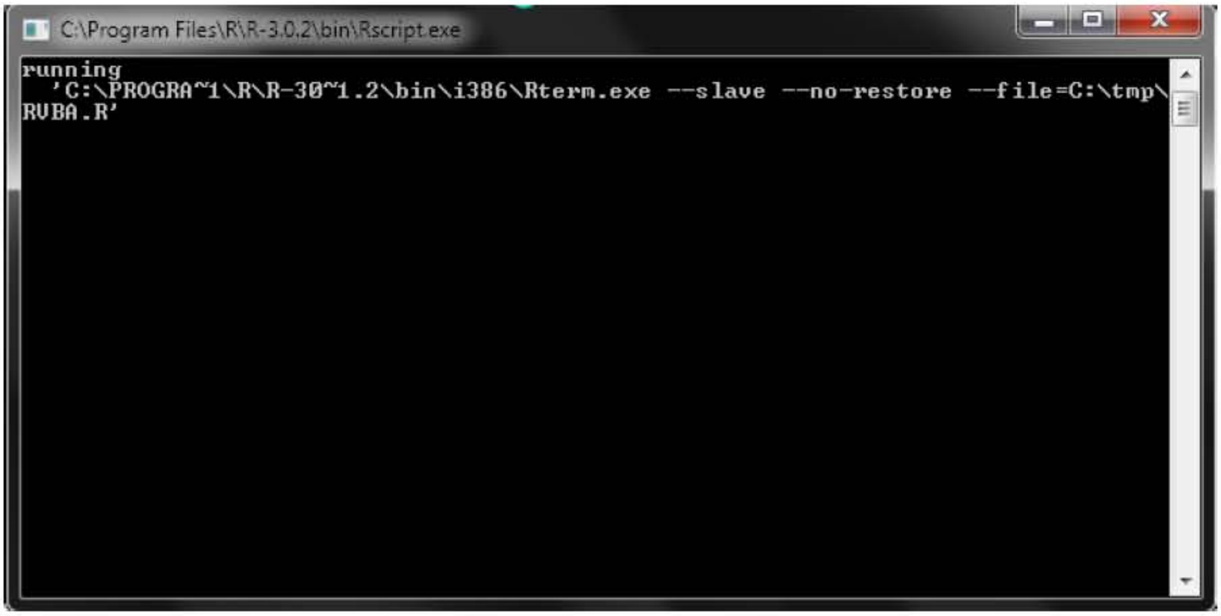
D'abord, nous considérons le script suivant RVBA.R enregistré dans le dossier C:/tmp/:



et ensuite dans un module VBA d'un des logiciels de la suite Microsoft Office, nous écrivons:



Ce qui s'exécute correctement! On voit par ailleurs pendant un petit moment la fenêtre de commande s'ouvrir:



Exemple 462.: Exécuter du code C depuis R

R 3.0.2

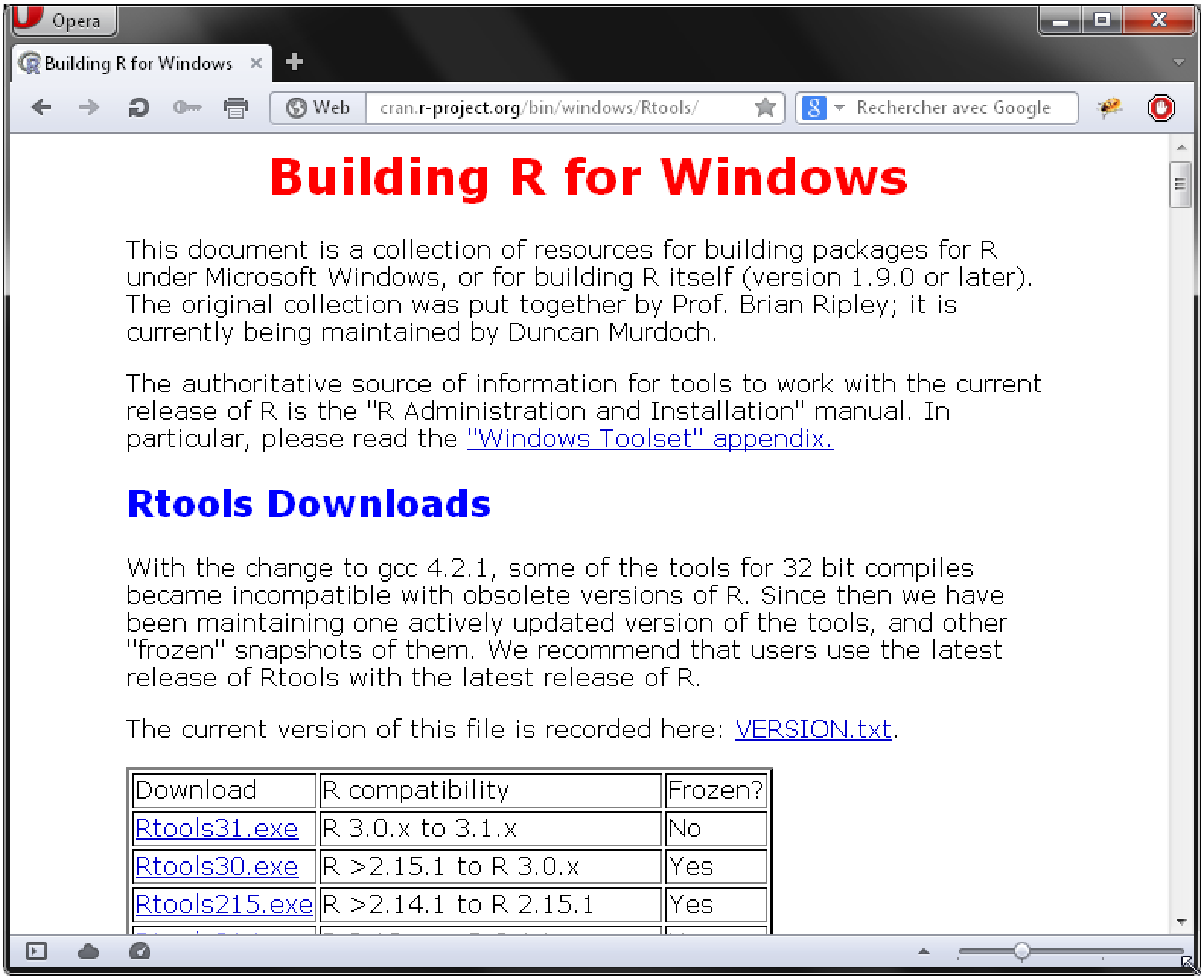
Lorsque l'on traite plus d'une dizaine de millions de lignes de données, travailler avec les outils standard de R devient chose impossible et surtout si aucun package ne correspond à nos besoins. Dès lors, soit on part sur du Matlab pour compiler un script en C, soit on écrit directement du C dans R.

Pour voir cela avec un exemple simple et générique, nous allons reprendre l'exemple générique donné initialement ici:

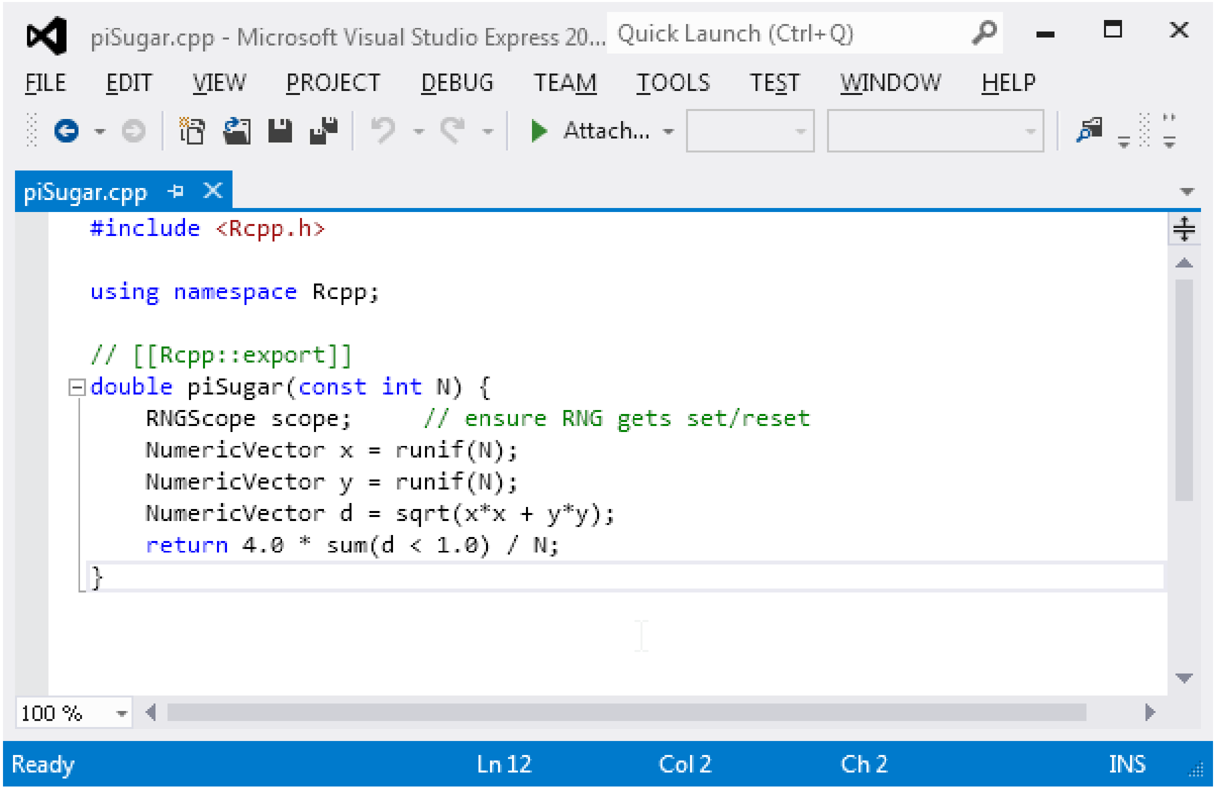
http://dirk.eddelbuettel.com/blog/2012/11/20/

Vous devrez d'abord télécharger et installer les R tools a à l'adresse suivante:

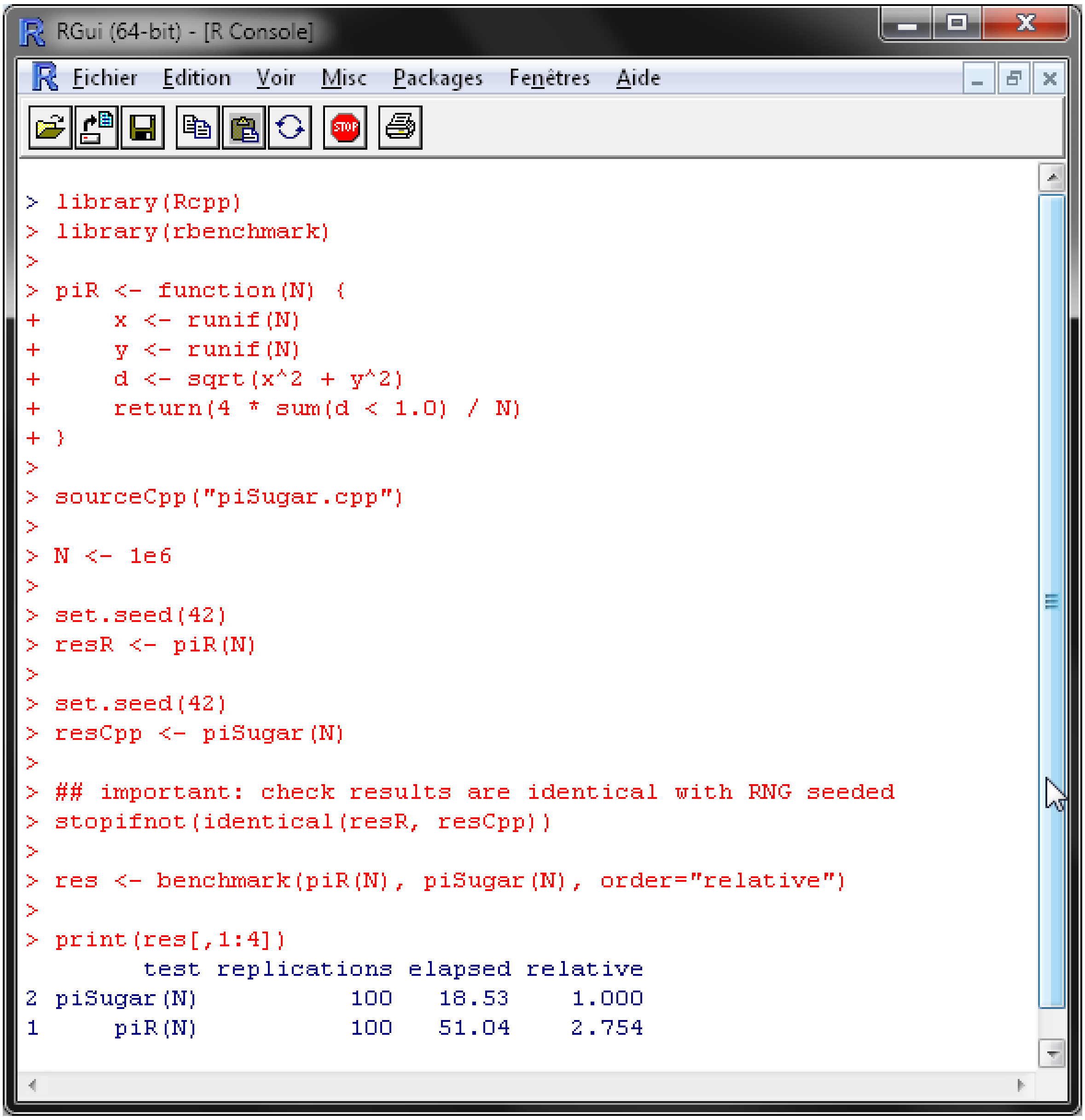
http://cran.r-project.org/bin/windows/Rtools/



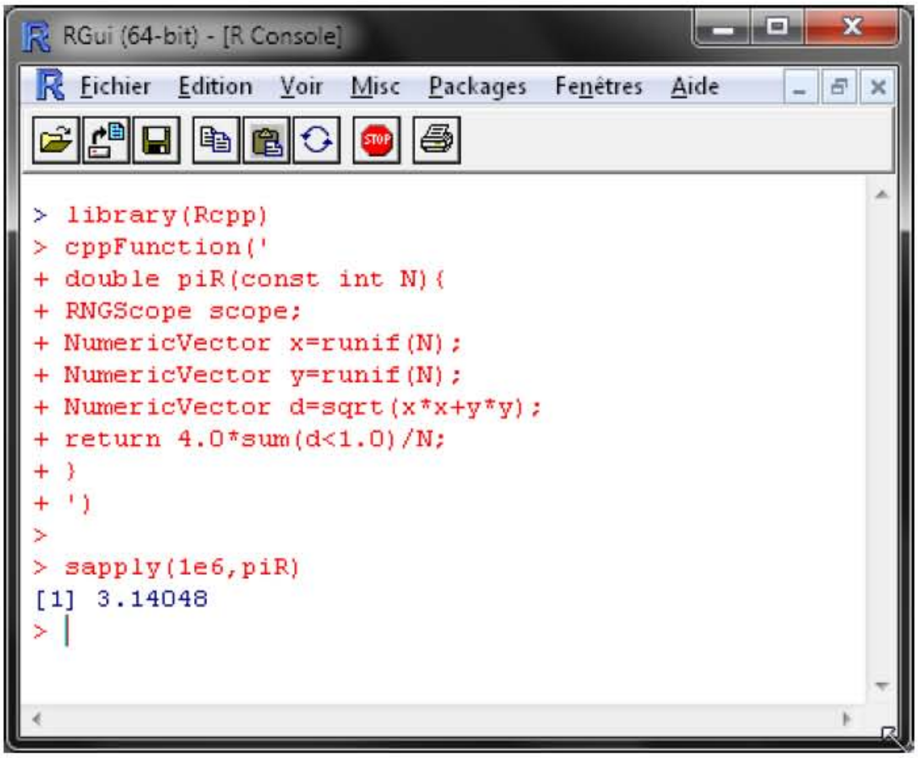
Une fois l'installation effectuée nous créons le fichier C++ suivant *piSugar.cpp* dans le dossier travail de la session courant de notre R:



et nous pouvons exécuter notre petit script qui va donc estimer la valeur de Pi par Monte-Carlo en utilisant le package **Rcpp** qui découle de l'installation des R Tools ainsi que le package **rbenchmark** qu'il faudra avoir installé après coup et qui est spécialisé dans la comparaison de scripts:



Nous pouvons également écrire en C directement dans R et transformer à la volée avec la commande **cppFunction( )** :

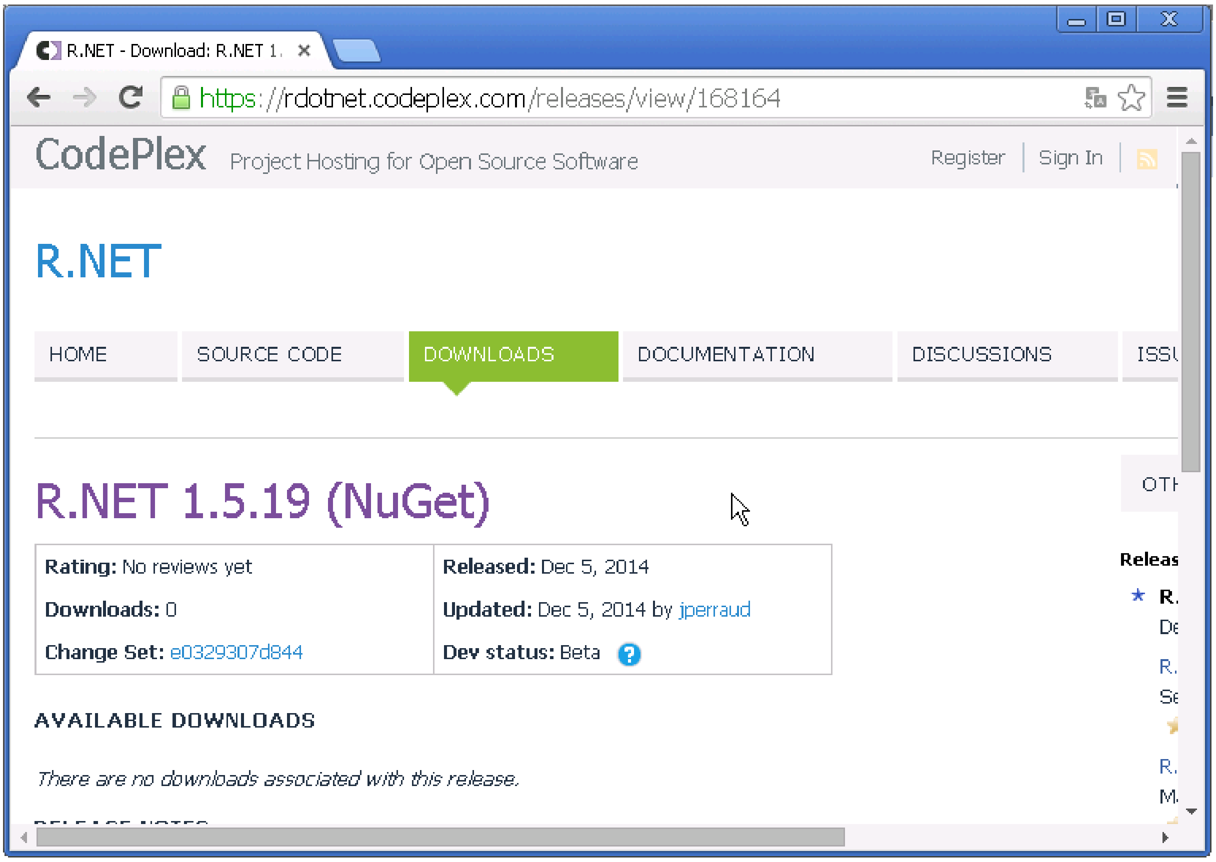

RGui (64-bit) - [R Console]
Fichier Edition Voir Misc Packages Fenêtres Aide
> library(Rcpp)
> cppFunction('
+ double piR(const int N){
+ RNGScope scope;
+ NumericVector x=runif(N);
+ NumericVector y=runif(N);
+ NumericVector d=sqrt(x*x+y*y);
+ return 4.0*sum(d<1.0)/N;
+ }
+ ')
>
> sapply(1e6,piR)
[1] 3.14048
>

Exemple 463.: Exécuter R depuis C# avec R.Net

R 3.1.2

Le C# est un langage très demandé dans la finance dans certains pays. Nous allons donc voir ici une méthode simple pour envoyer à R des valeurs et récupérer des résultats.

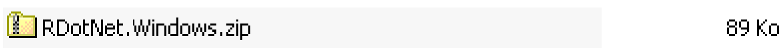
D'abord se rendre sur le site http://rdotnet.codeplex.com:



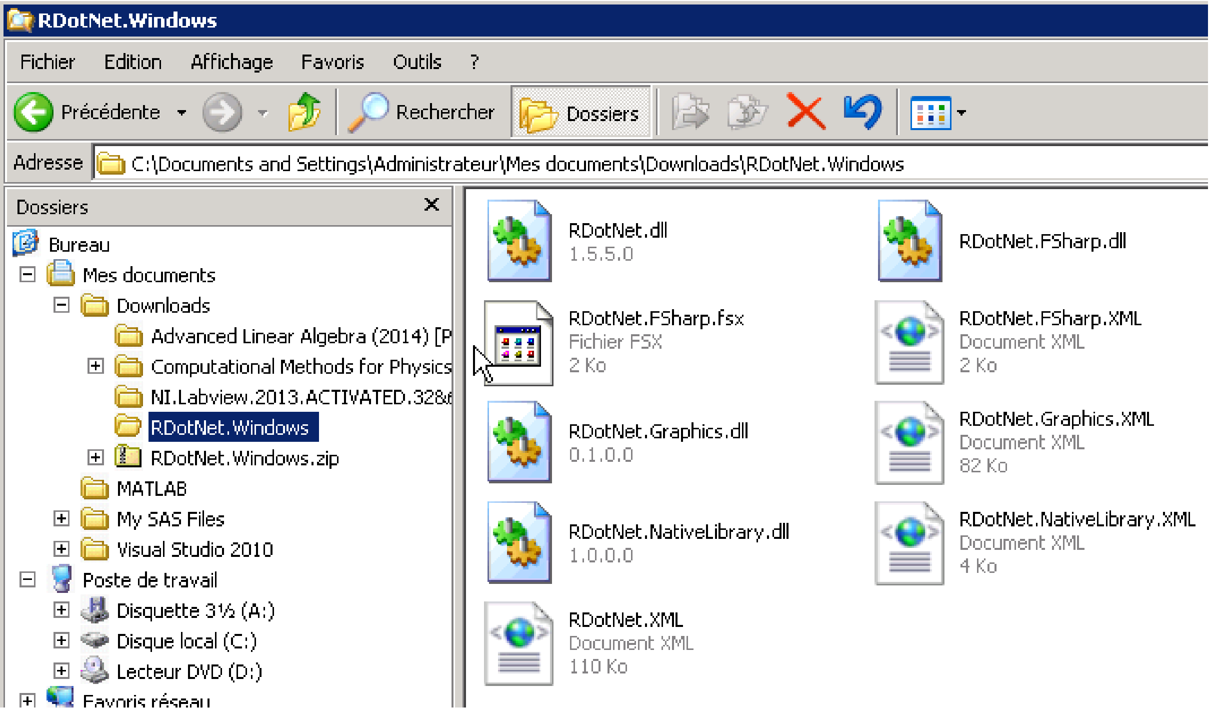
et à droite de la page visible, télécharger la dernière version stable:



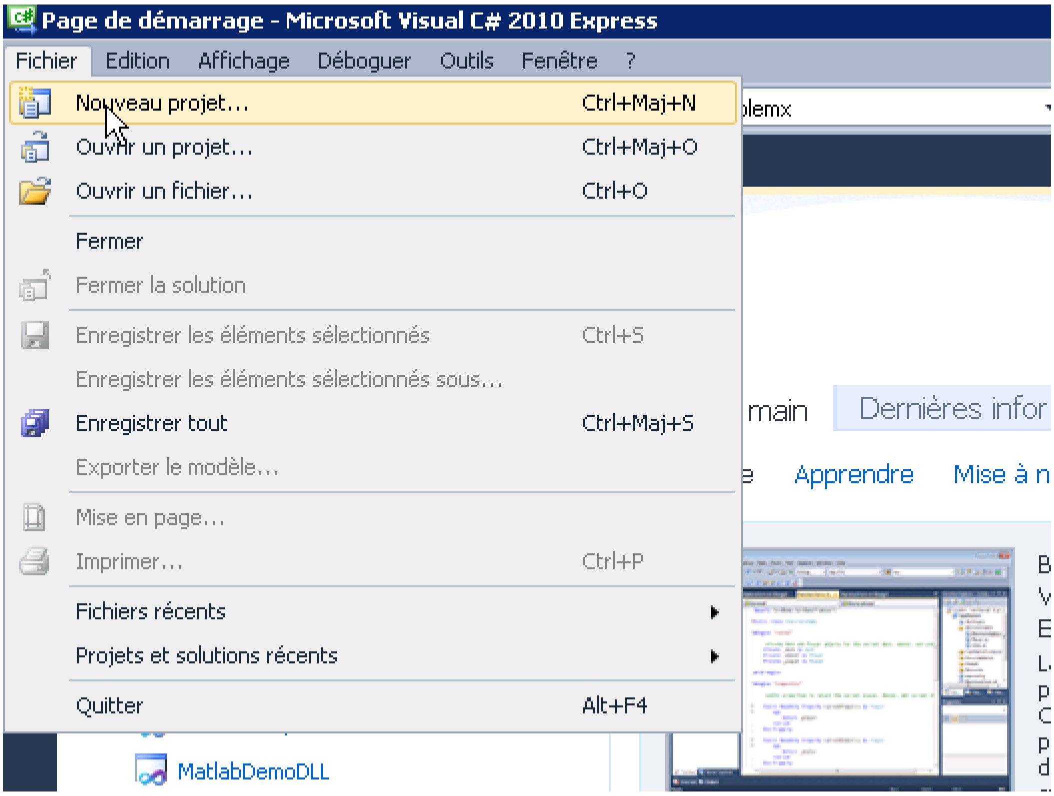
Cela télécharger un très petit fichier zip du type:



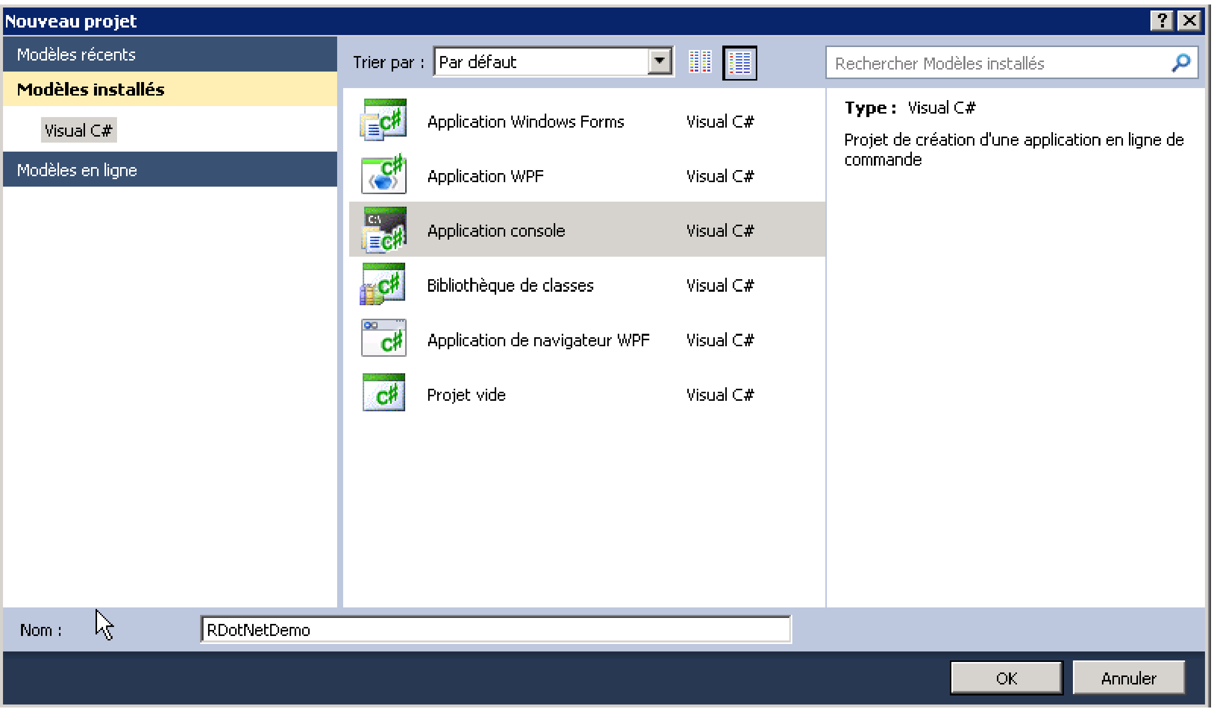
qui après décompression dans le dossier de votre choix donnera:



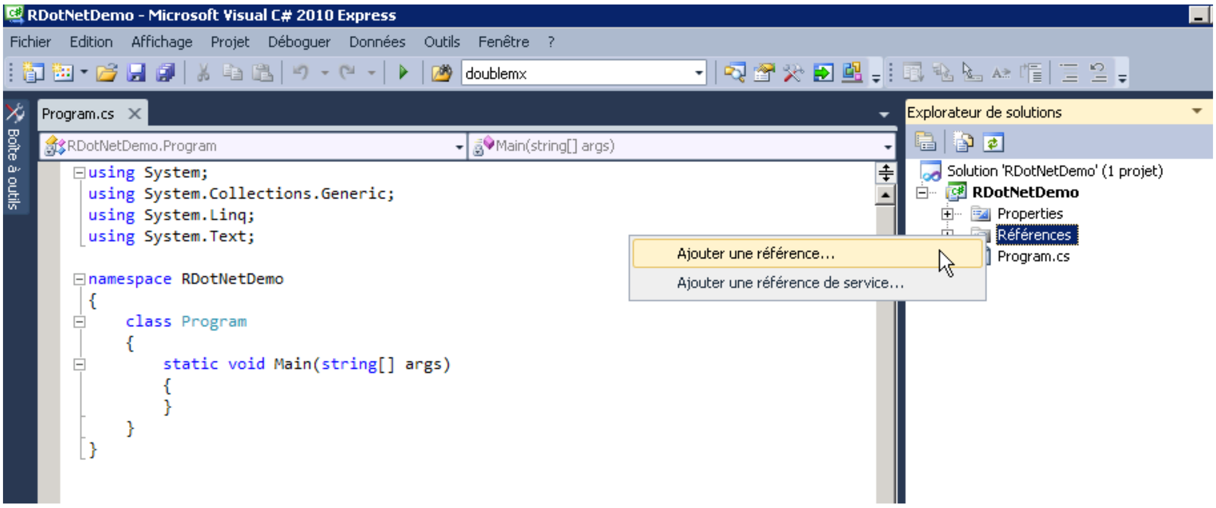
Ensuite ouvrez Visual Studio C# (ici la version 2010 dans un environnement XP x86) et créez un **Nouveau projet…**:



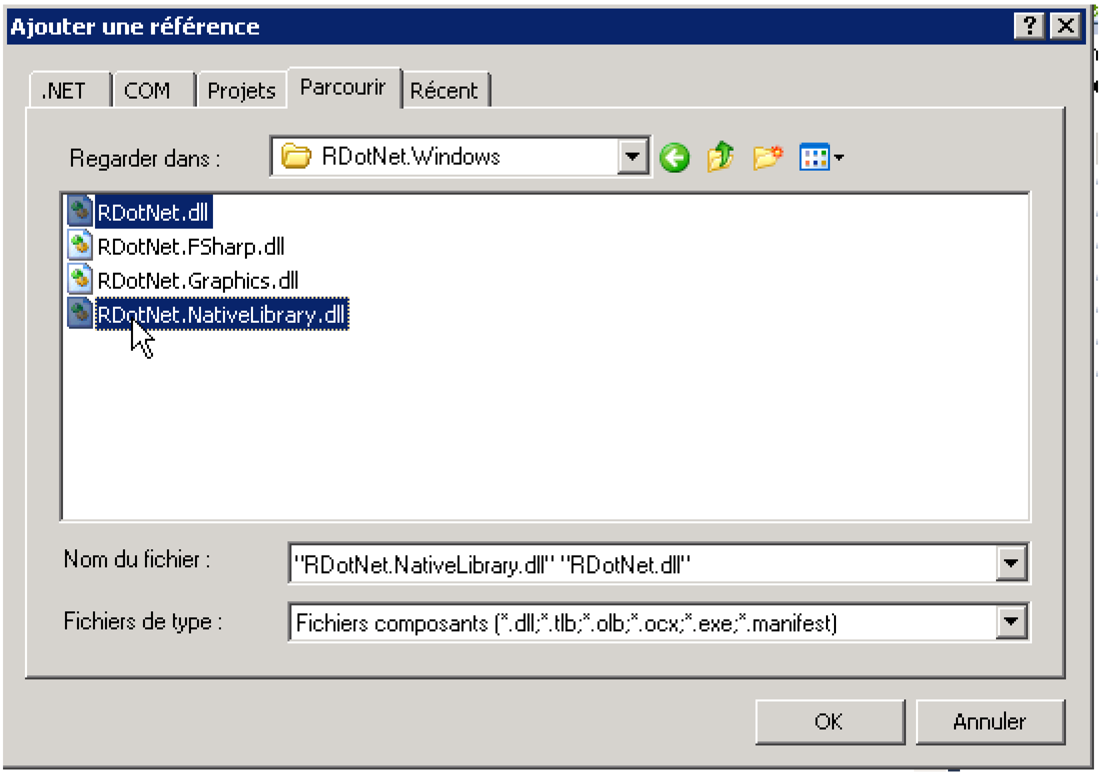
Prenez **Application Console** et donnez un nom au projet (ici *RDotNetDemo*):



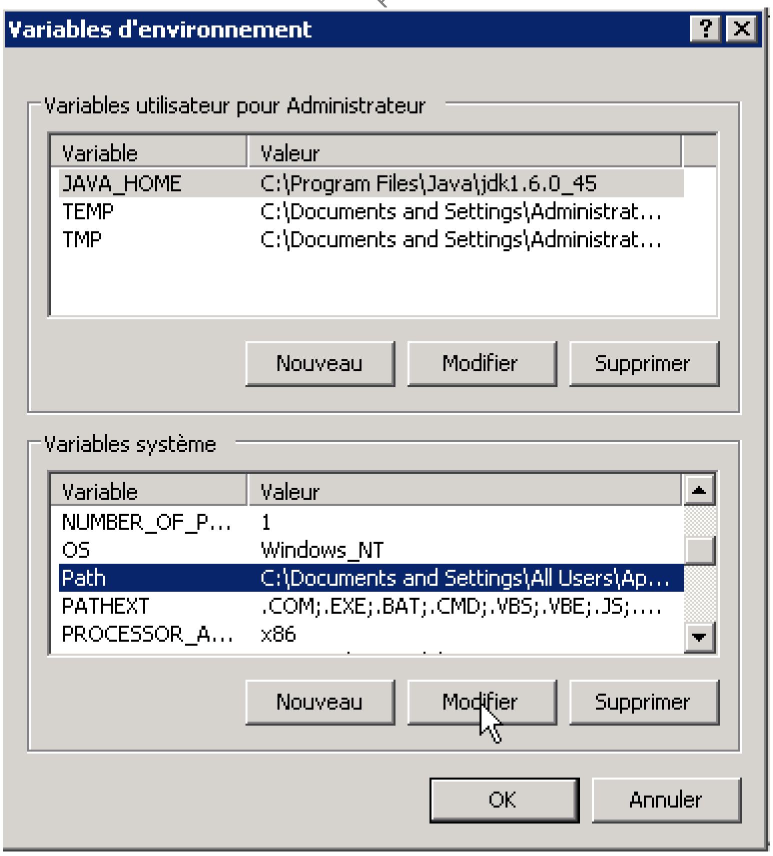
Une fois le projet créé, cliquer sur **Ajouter une référence** dans menu contextuel du volet droit sur le dossier **Références**:

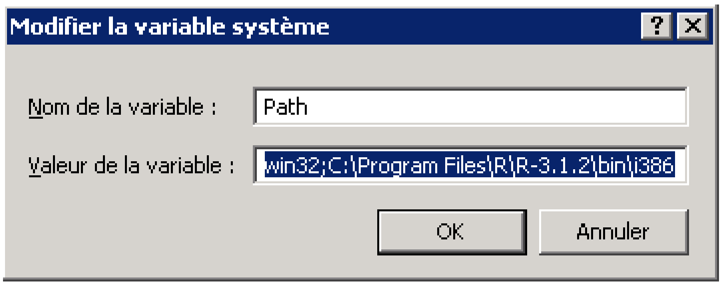


Prenez les deux dll indiquées qui suffiront pour la démonstration:



Ensuite il est conseillé dans certains environnements Microsoft Windows d'ajouter le chemin de R dans la variable d'environnement **PATH**:





Une fois ceci fait, nous écrivons le code suivant pour faire un exemple d'appel de R:

```
using Microsoft.Win32;
using System;
using System.Linq;
using RDotNet;
using System.IO;

namespace Sample1
{
    class Program
    {
        static void Main(string[] args)
        {
            using (RegistryKey registryKey =
Registry.LocalMachine.OpenSubKey(@"SOFTWARE\R-core\R"))
            {
                var envPath = Environment.GetEnvironmentVariable("PATH");
                string rBinPath = (string)registryKey.GetValue("InstallPath");
                string rVersion = (string)registryKey.GetValue("Current Version");
                rBinPath = System.Environment.Is64BitProcess ? rBinPath + "\\bin\\x64"
:
                                                                 rBinPath +
"\\bin\\i386";
                Environment.SetEnvironmentVariable("PATH",
                                           envPath + Path.PathSeparator + rBinPath);
            }
            using (REngine engine = REngine.CreateInstance("RDotNet"))
            {
                // Initializes settings.
                engine.Initialize(); // After Executing this line its crashing.

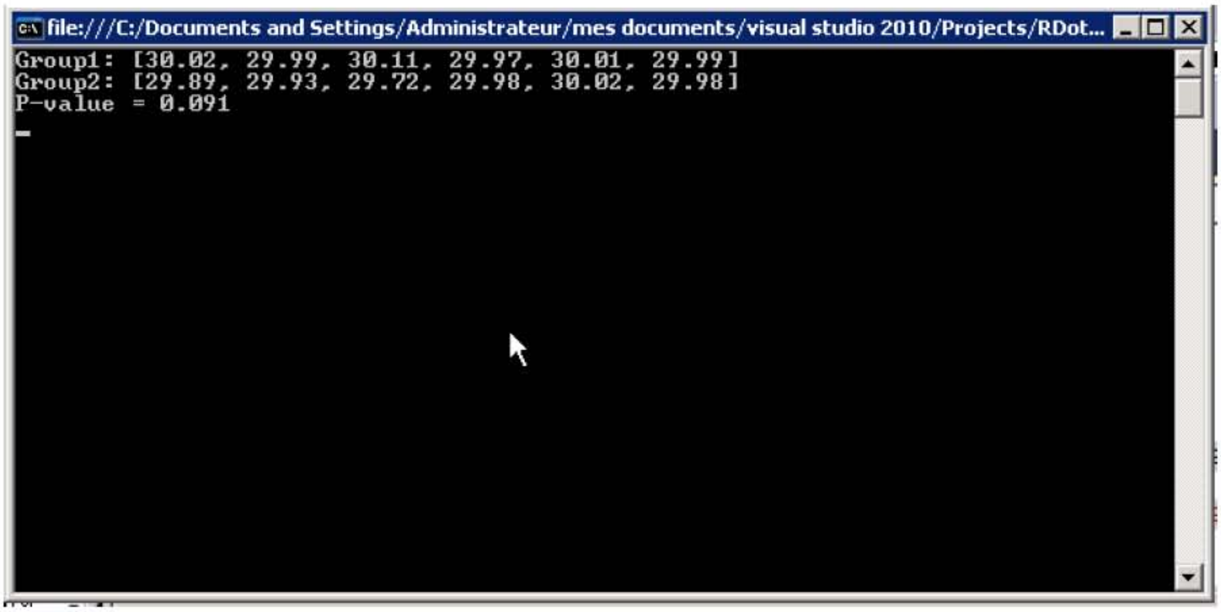
                NumericVector group1 = engine.CreateNumericVector(new double[] {
30.02, 29.99, 30.11, 29.97, 30.01, 29.99 });
                engine.SetSymbol("group1", group1);
                NumericVector group2 = engine.Evaluate("group2 <- c(29.89, 29.93,
29.72, 29.98, 30.02, 29.98)").AsNumeric();

                // Test difference of mean and get the P-value.
                GenericVector testResult = engine.Evaluate("t.test(group1,
group2)").AsList();
                double p = testResult["p.value"].AsNumeric().First();

                Console.WriteLine("Group1: [{0}]", string.Join(", ", group1));
                Console.WriteLine("Group2: [{0}]", string.Join(", ", group2));
                Console.WriteLine("P-value = {0:0.000}", p);
                Console.ReadLine();
            }

        }
    }
}
```

Ce qui donne:


file:///C:/Documents and Settings/Administrateur/mes documents/visual studio 2010/Projects/RDot...
Group1: [30.02, 29.99, 30.11, 29.97, 30.01, 29.99]
Group2: [29.89, 29.93, 29.72, 29.98, 30.02, 29.98]
P-value = 0.091

Exemple 464.: Débugger

R 3.0.2

Voilà un sujet plus qu'important...

Le but ici va être de découvrir des commandes qui sont similaires dans l'idée aux techniques de débogage de MATLAB:

traceback, debug, browser, trace, recover

Traceback

La commande traceback affiche la liste des fonctions appelées avant de lister enfin ce qui a fait bugger le traitement.

Voyons un exemple simple en considérant le script suivant:

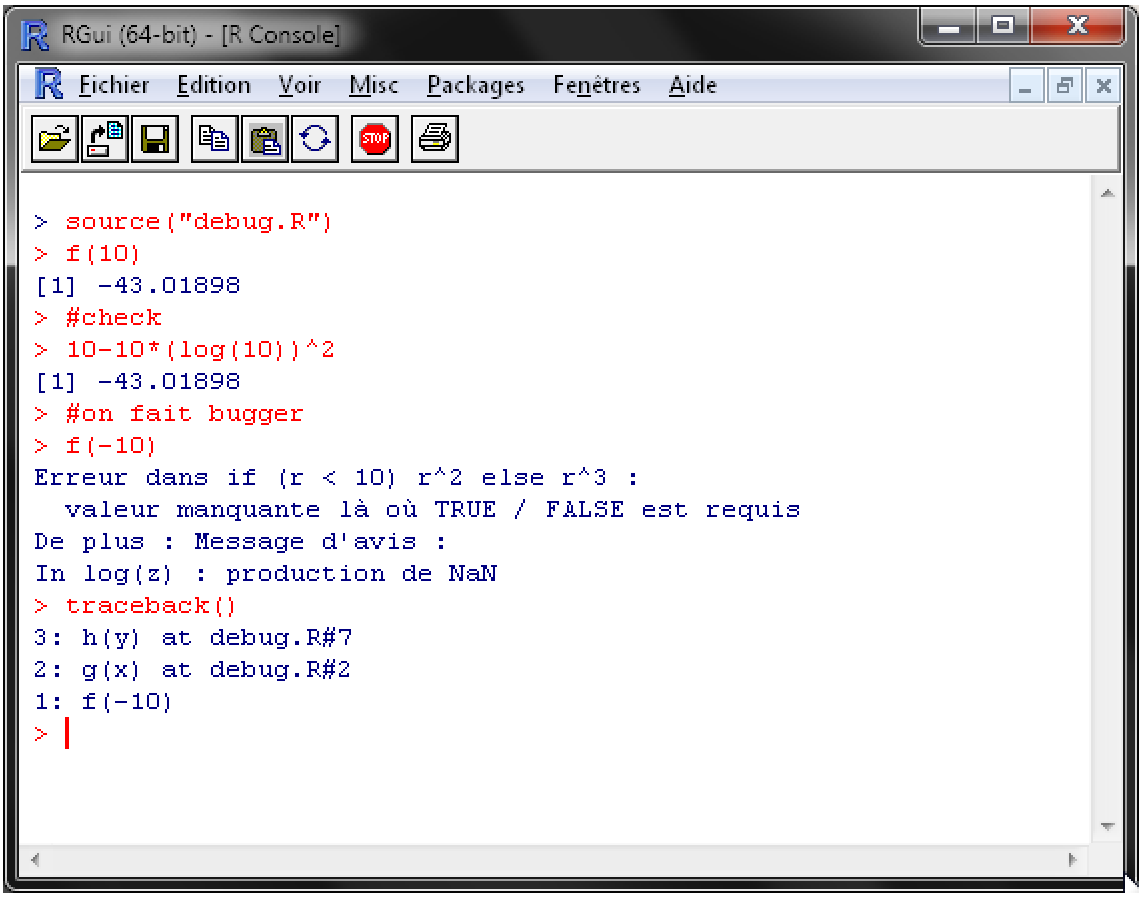
C:\tmp\debug.R - Editeur R

```
f<-function(x){
	r<-x-g(x)
	r
}

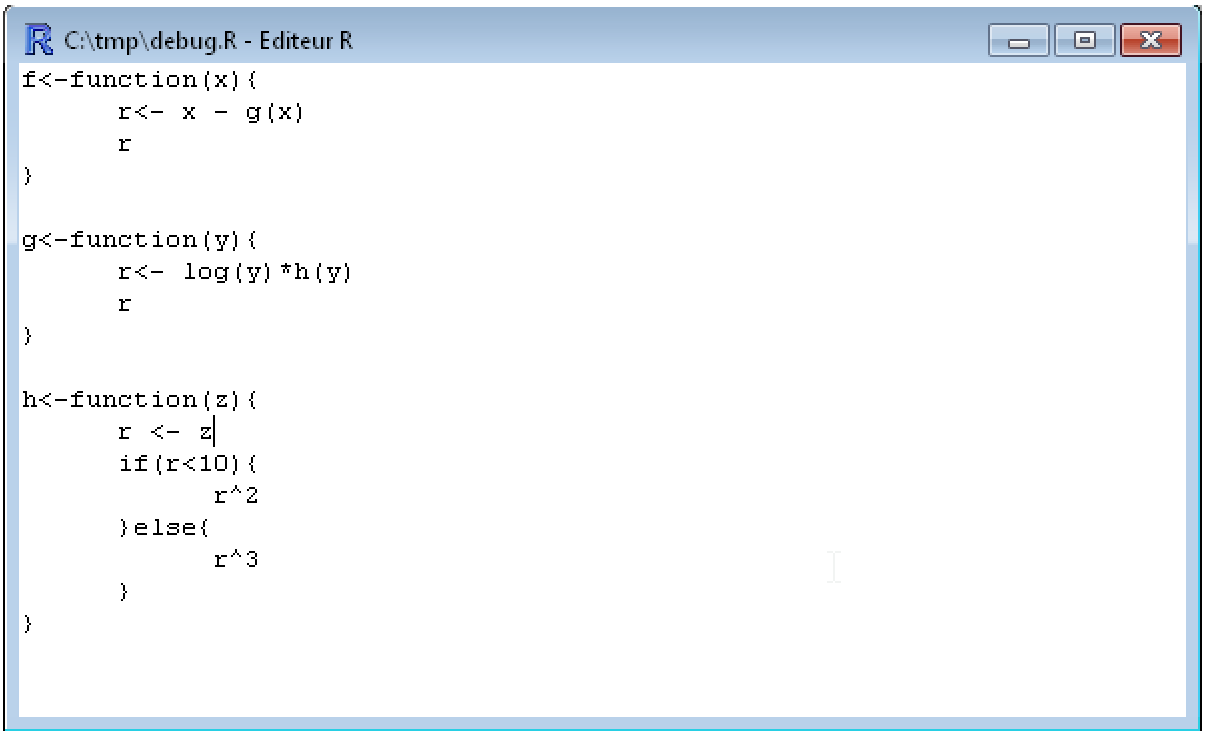
g<-function(y){
	r<-y*h(y)
	r
}

h<-function(z){
	r<-log(z)
	if(r<10)
		r^2
	else
		r^3
}
```

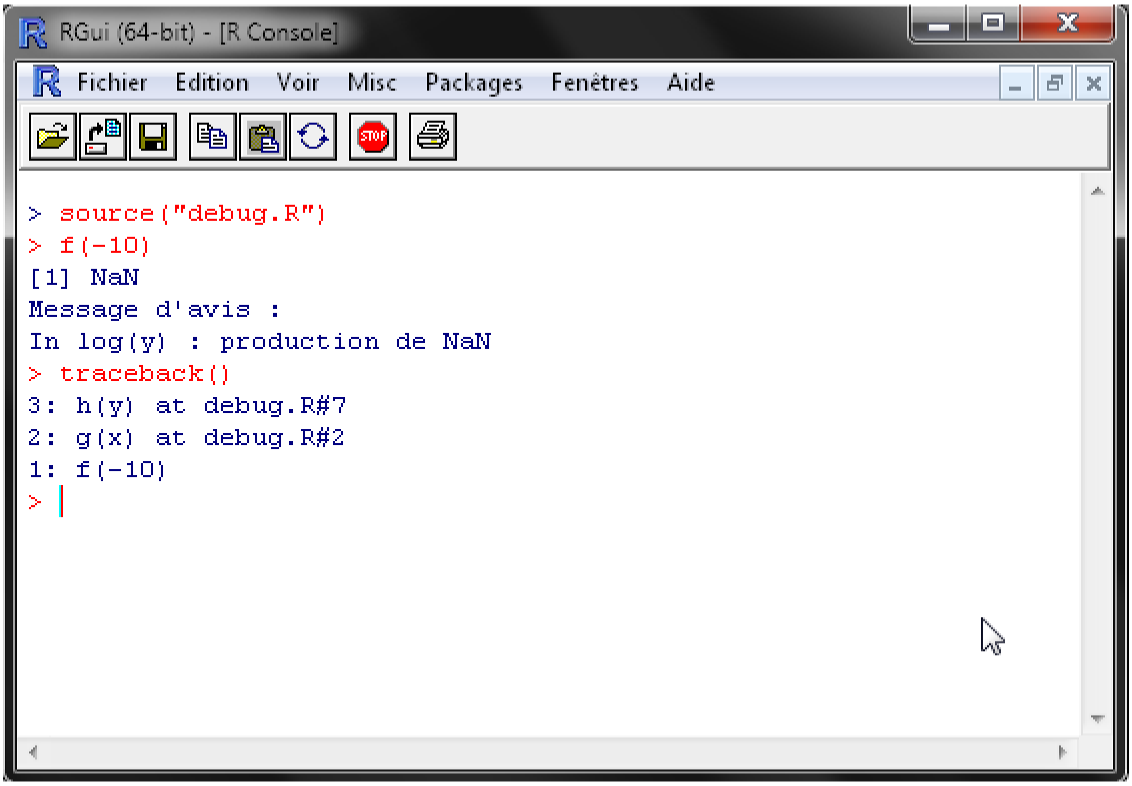
Si maintenant nous lançons ce script en provoquant une erreur volontairement:



Nous voyons que la commande **traceback( )** est finalement a priori peu utile car renvoie des informations trop vagues (du moins sur l'environnement MS Windows). Effectivement, si nous changeons l'emplacement de l'erreur, nous avons...:



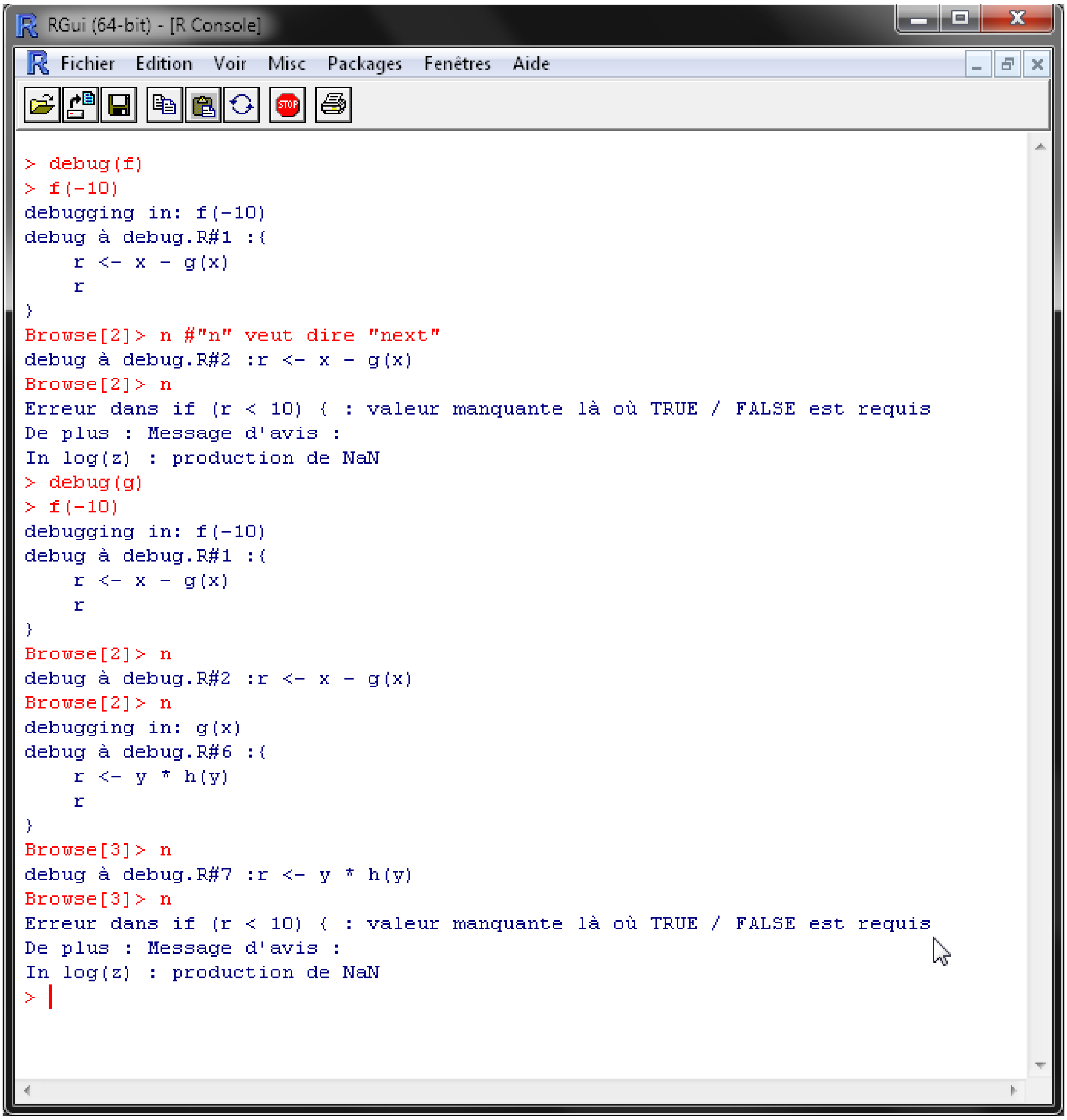
Ce qui nous donnera:



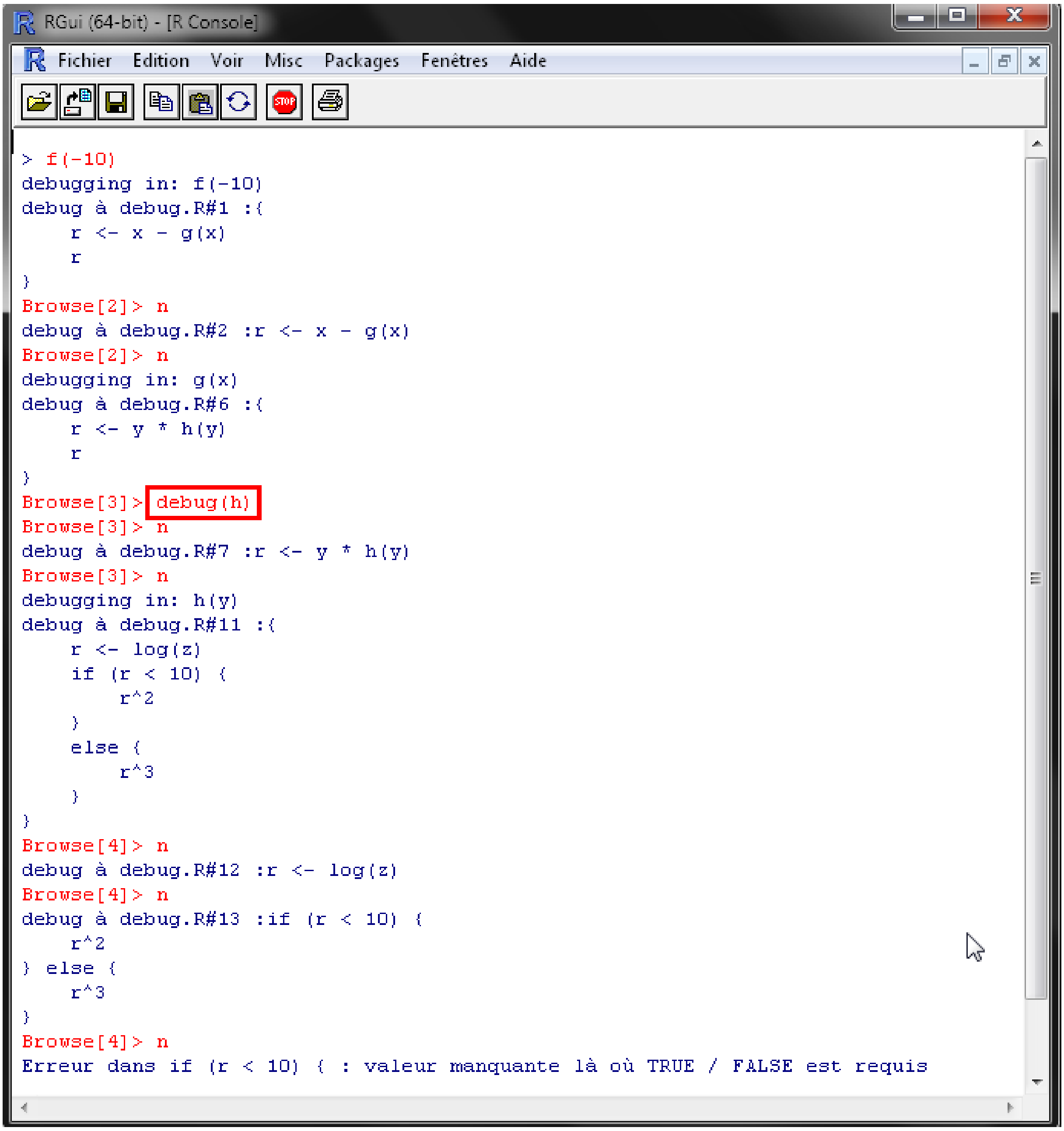
Donc la fonction h(y) apparaît aussi alors qu'elle ne génère aucun bug directement!

Debug

Bon nous remettons le code buggé dans l'état initial et nous allons voir maintenant la fonction **debug( )** dont l'utilité est aussi discutable. Effectivement voyons un exemple:



Comme nous pouvons le voir il est donc important de connaître (identifier) le nom des fonctions bugées au fur et à mesure que l'on avance. Ainsi, dans l'exemple suivant, nous décidons de débugger la fonction h(z) pendant le débogage de g(y):



Liste des commandes:

- **n**: poursuivre le débogage (sans que cela rente nécessairement dans les sous fonctions!)

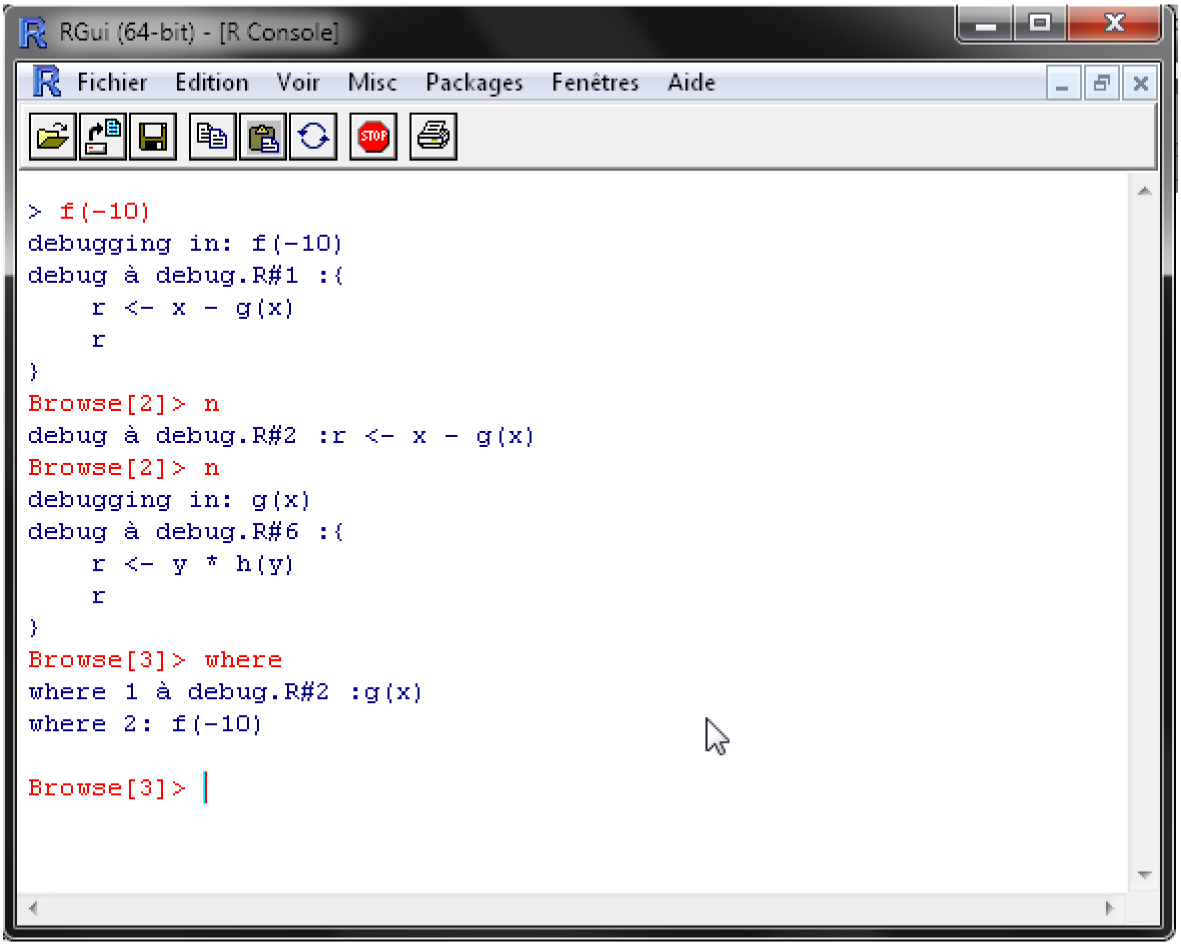
- **c**: exécute l'ensemble restant de la fonction (ou de l'ensemble des fonctions) jusqu'à la fin

- **Q**: arrête le débogage et l'exécution de la fonction

- **print(nom_variable)**: pour pouvoir à tout moment voir le contenu (la valeur) d'une variable pendant un débogage

Remarque: Il suffit de recharger le fichier *.R ou le package pour que le mode débogage qui a été activé en écrivant **debug(***nom_de_la_fonction***)** s'arrête

Voici un exemple de la commande **where**:



Browse

Il existe une autre manière beaucoup plus agréable de débugger qui consiste à écrire la commande **browse( )** à un endroit désiré du code source:

C:\tmp\debug.R - Editeur R

```
f<-function(x){
	r<- x - g(x)
	r
}

g<-function(y){
	r<- y*h(y)
	r
}

h<-function(z){
	browser()
	r <- log(z)
	if(r<10){
		r^2
	}else{
		r^3
	}
}
```

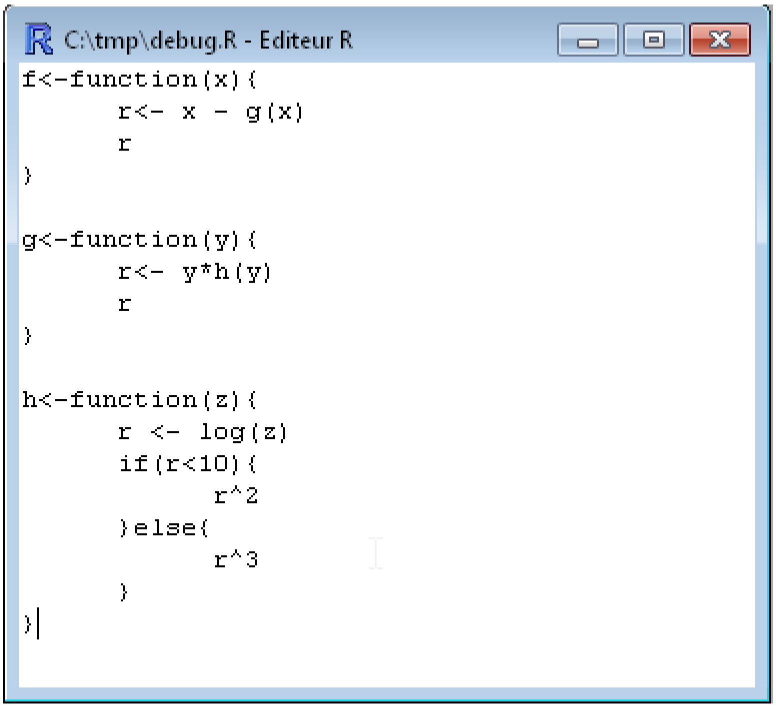
RGui (64-bit) - [R Console]

```
> source("debug.R")
> f(-10)
Called from: h(y)
Browse[1]> n
debug à debug.R#13 :r <- log(z)
Browse[2]> n
debug à debug.R#14 :if (r < 10) {
    r^2
} else {
    r^3
}
Browse[2]> n
Erreur dans if (r < 10) { : valeur manquante là où TRUE / FALSE est requis
De plus : Message d'avis :
In log(z) : production de NaN
> 
```

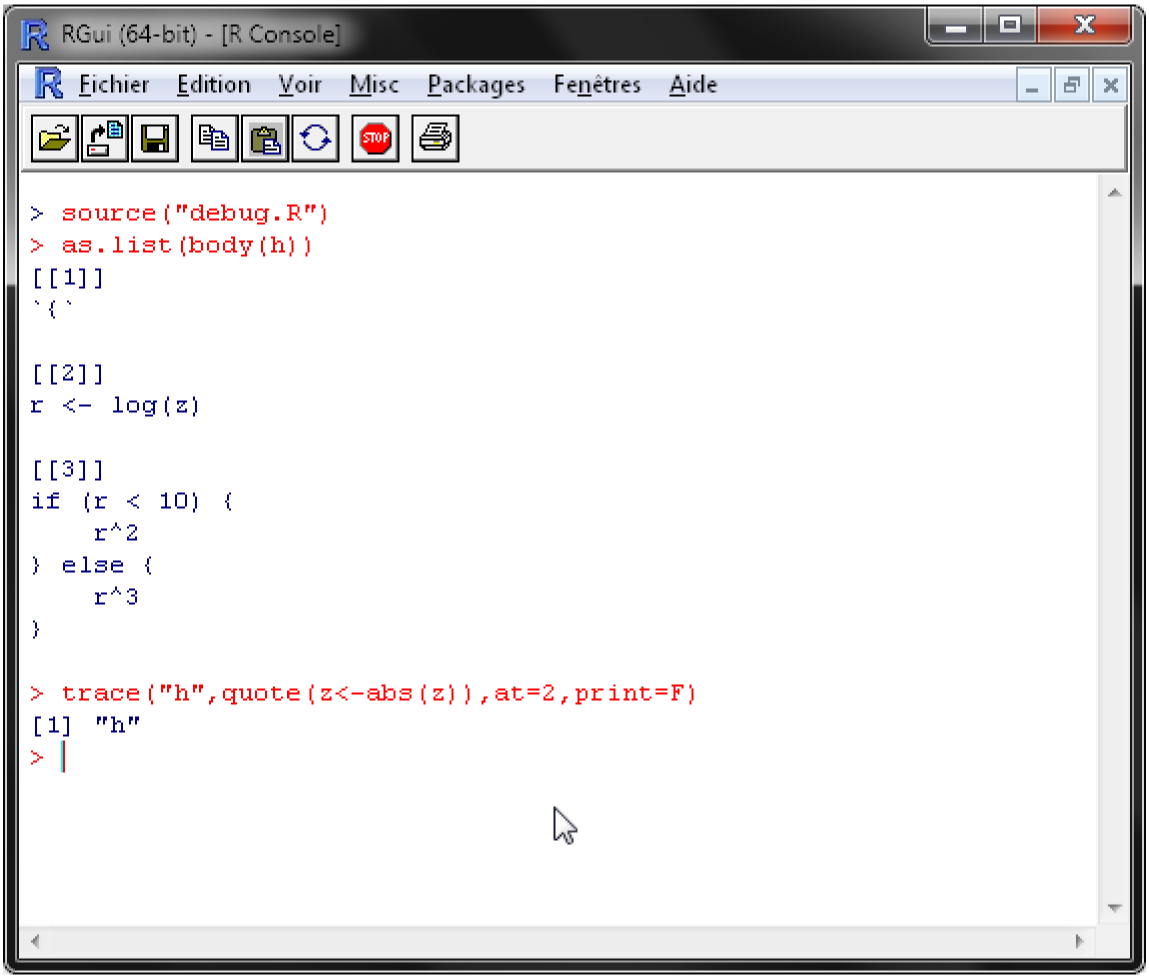
Comme vous pouvez le voir le débogage commence donc à l'endroit où nous avons positionné la function **browse( )** .

Trace

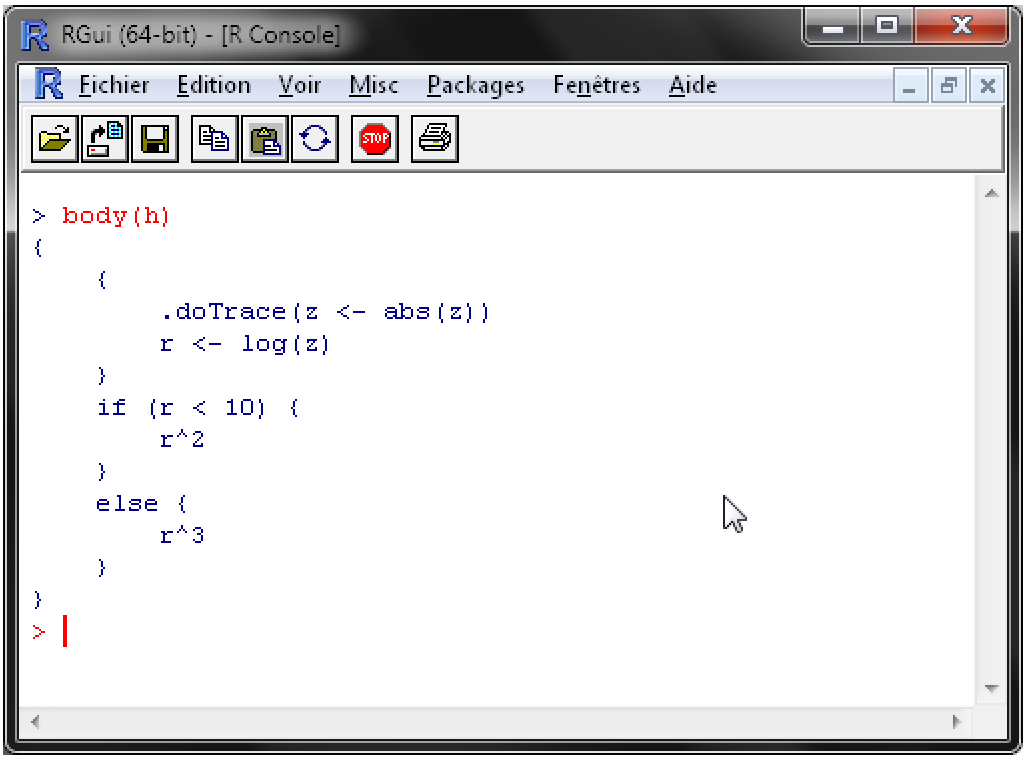
Il est possible lorsque les fonctions sont petites de les modifier "temporairement" pour faire des tests à la volée pendant un débogage. Pour voir cela considérons toujours notre petit script:



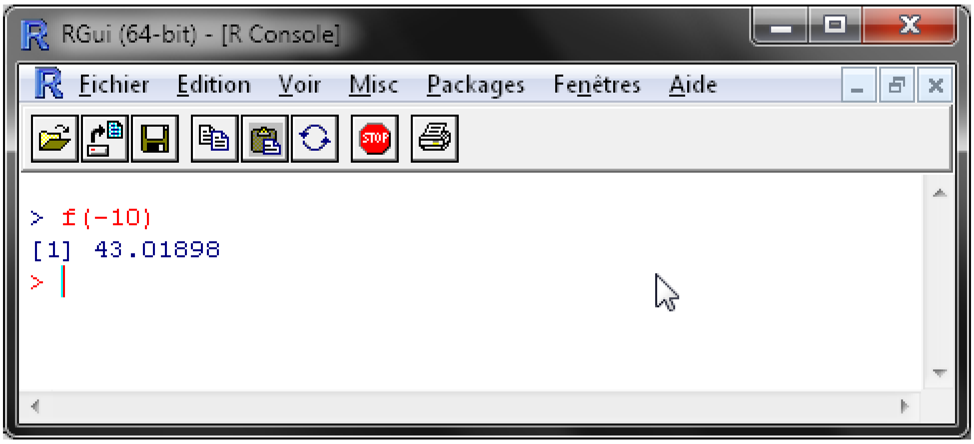
Avant d'utiliser la fonction **trace( )** pour insérer une ligne il faut savoir quel est le numéro d'insertion (et non de remplacement!) et pour cela nous pouvons faire usage de la fonction **as.list(body(...))**:



Nous pouvons vérifier que l'insertion se soit faite:



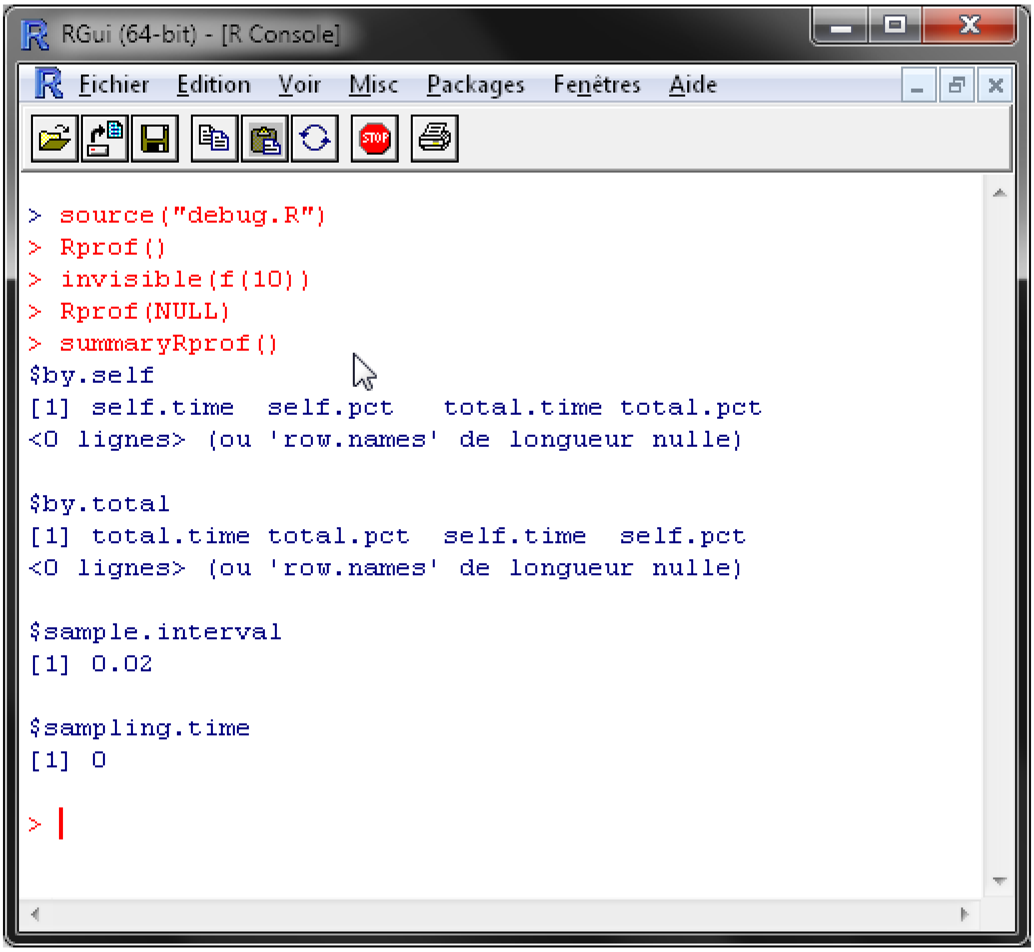
Si maintenant nous exécutons les valeurs négatives passent:



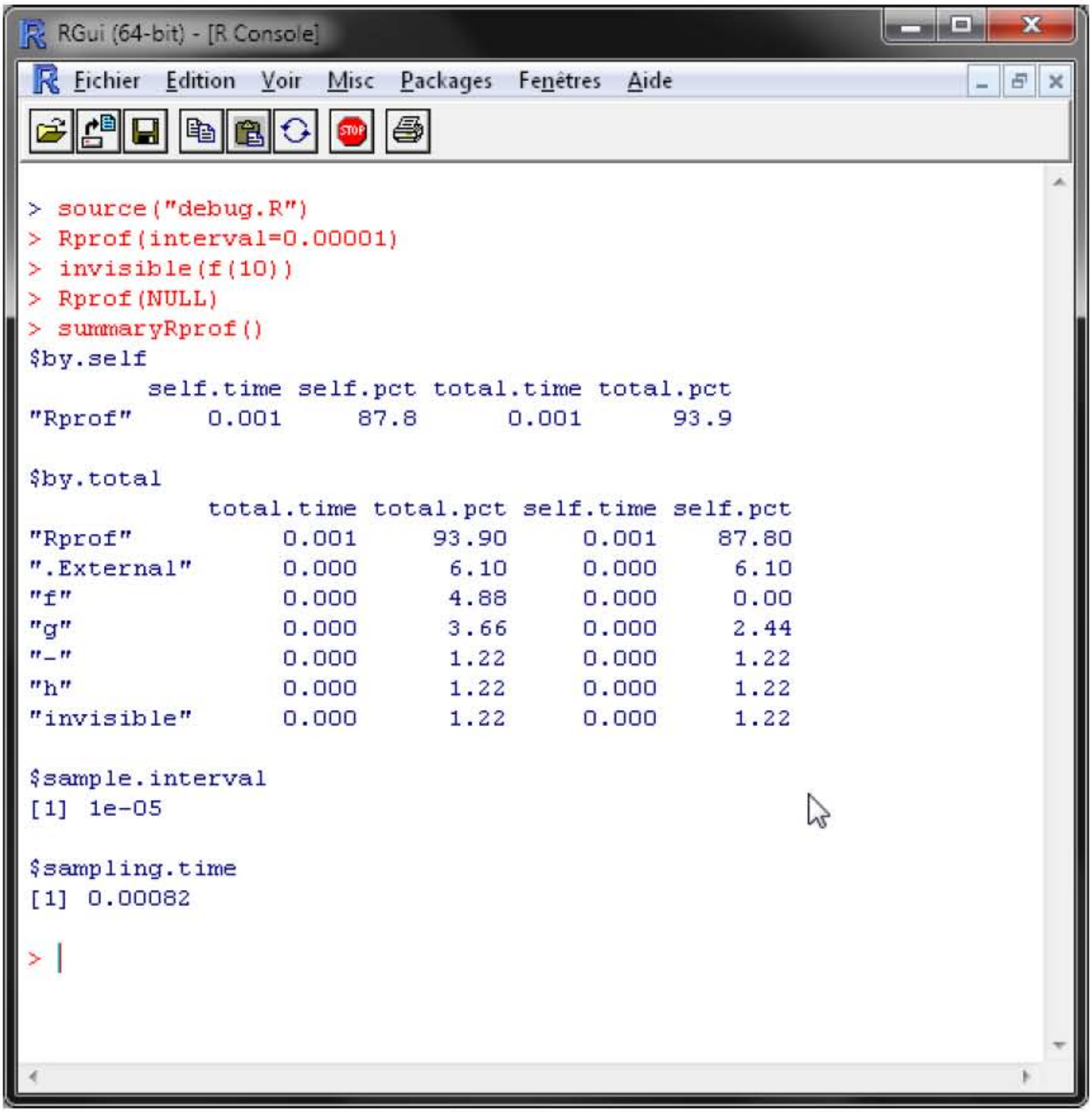
Profilage

Le profilage de script est un sujet important pour ceux cherchant à trouver les goulets d'étranglement de leurs codes. Pour l'exemple, nous utiliserons le même script qu'avant avec les fonctions natives **Rprof( )** , **invisible( )** dont le but est de ne pas imprimer les calculs de la fonction testée, et **summaryRprof( )** .

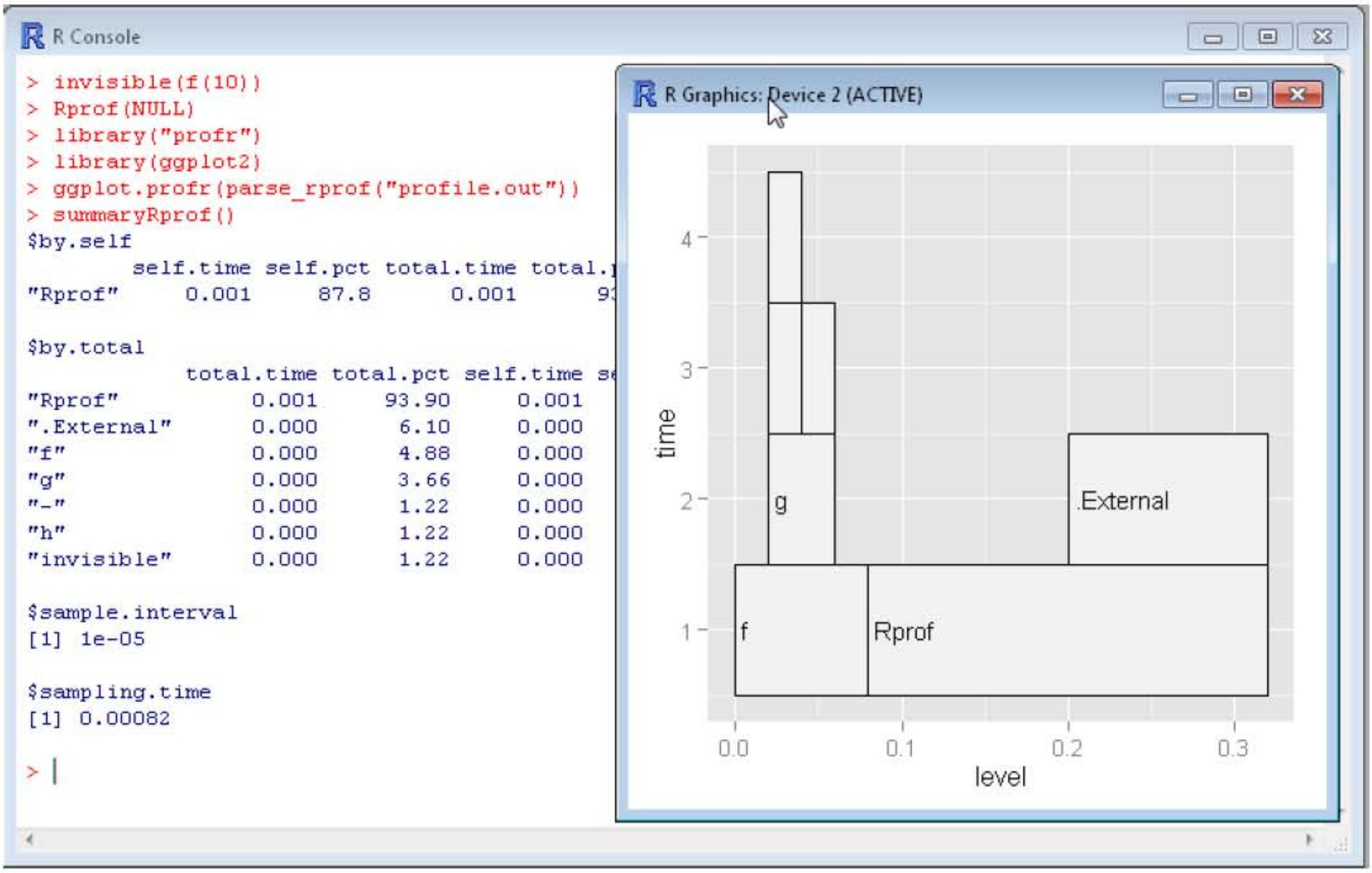
Avec un tout petit script comme celui que nous avons, nous voyons que le profiler a de la peine à profiler quoi que ce soit:



Nous pouvons alors spécifier le temps d'échantillonnage pour avoir quelque d'un peu mieux:



Ou pour ceux qui préfèrent les rapports visuels discutables avec le package **profr**:

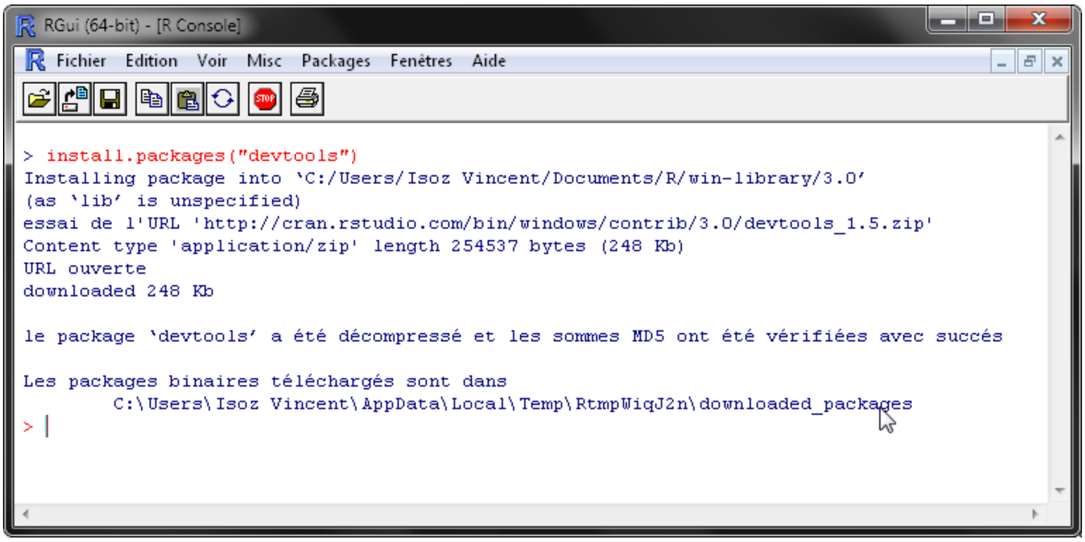


Exemple 465.: Créer rapidement un package R d'un script R

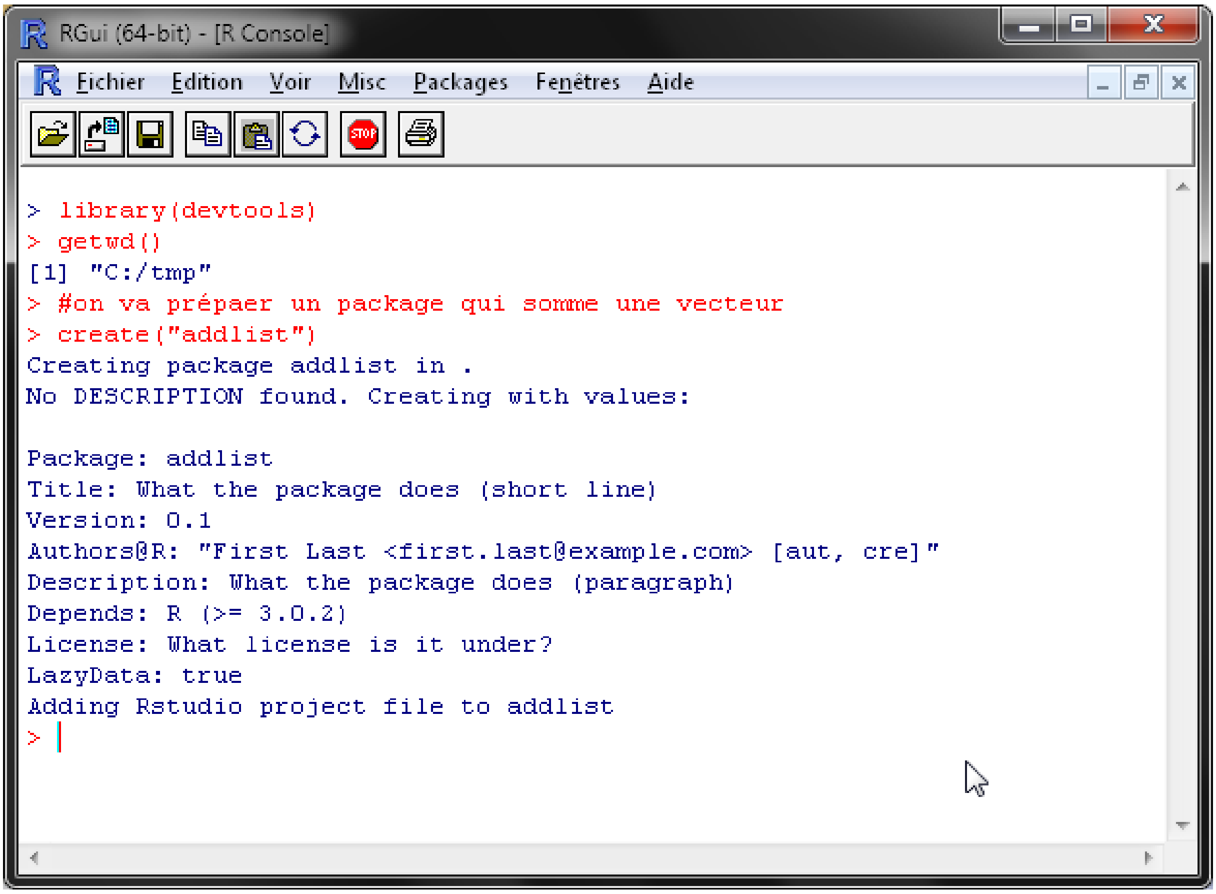
R 3.0.2

Sans R Studio

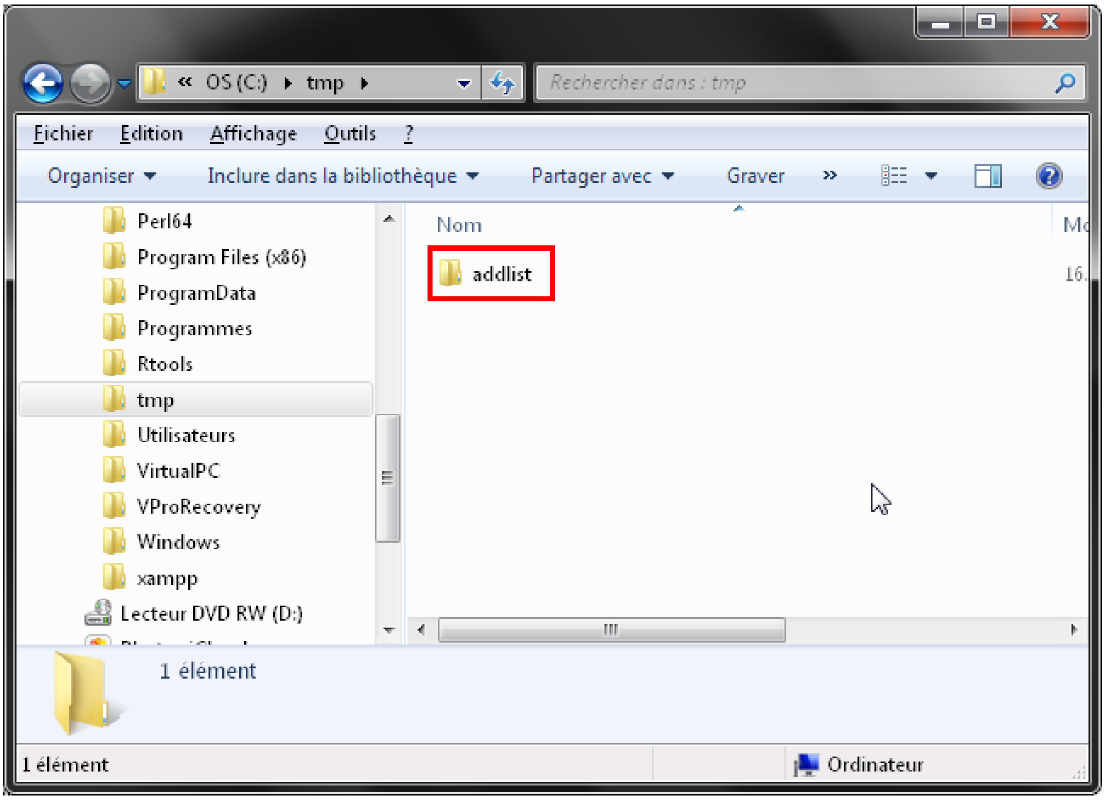
D'abord pour faire rapidement (et mal) un petite package R d'un script R, l'utilisation du package **devtools** sera obligatoire:



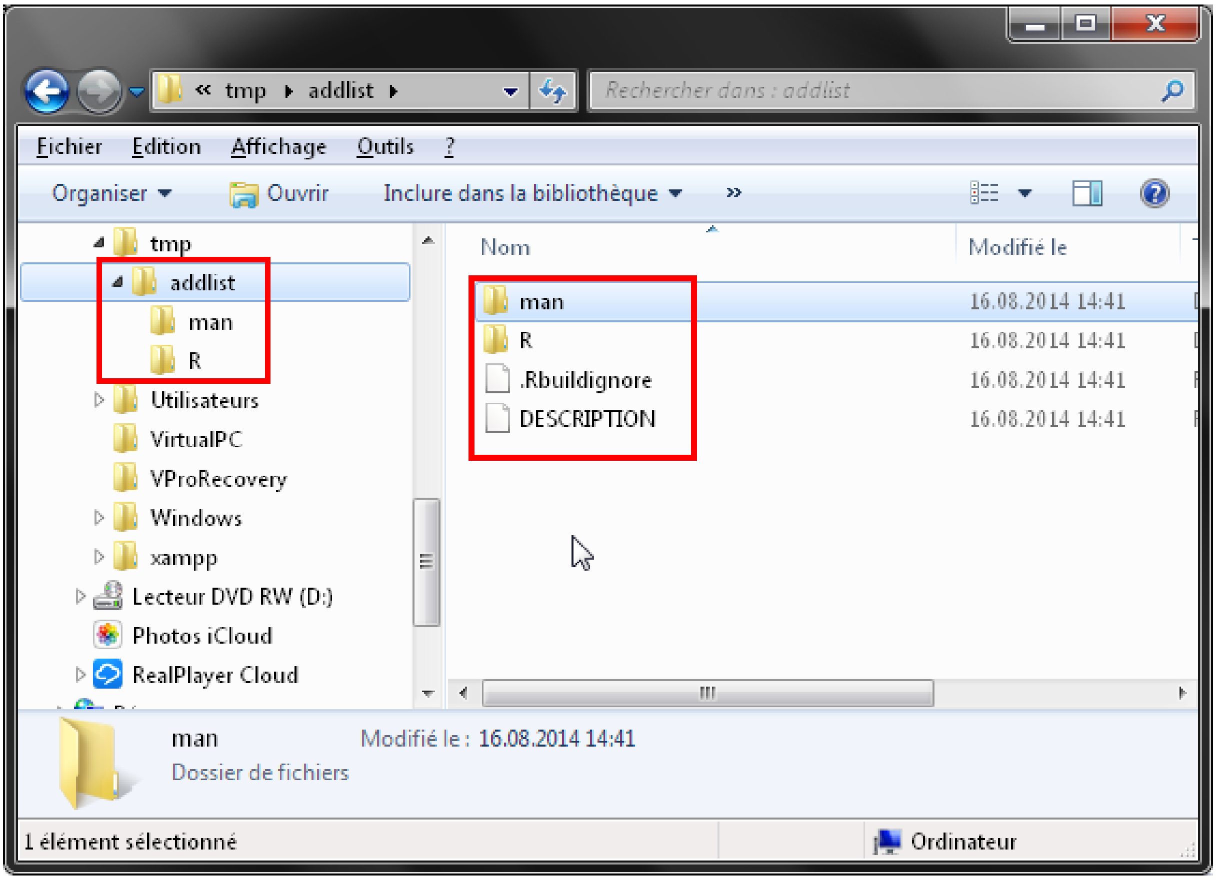
Une fois ce package installé, vérifiez votre dossier de travail par défaut et créez un dossier pour un futur package avec la commande **create( )** du package **devtools**:



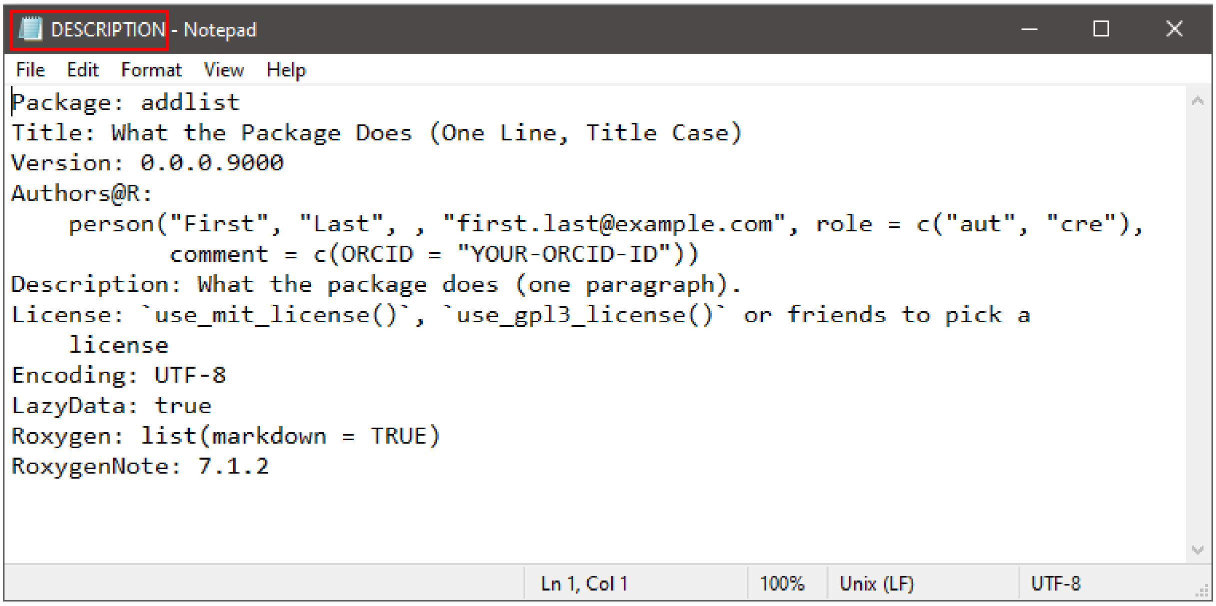
Ce qui créera un dossier avec le nom du nouveau package souhaité:



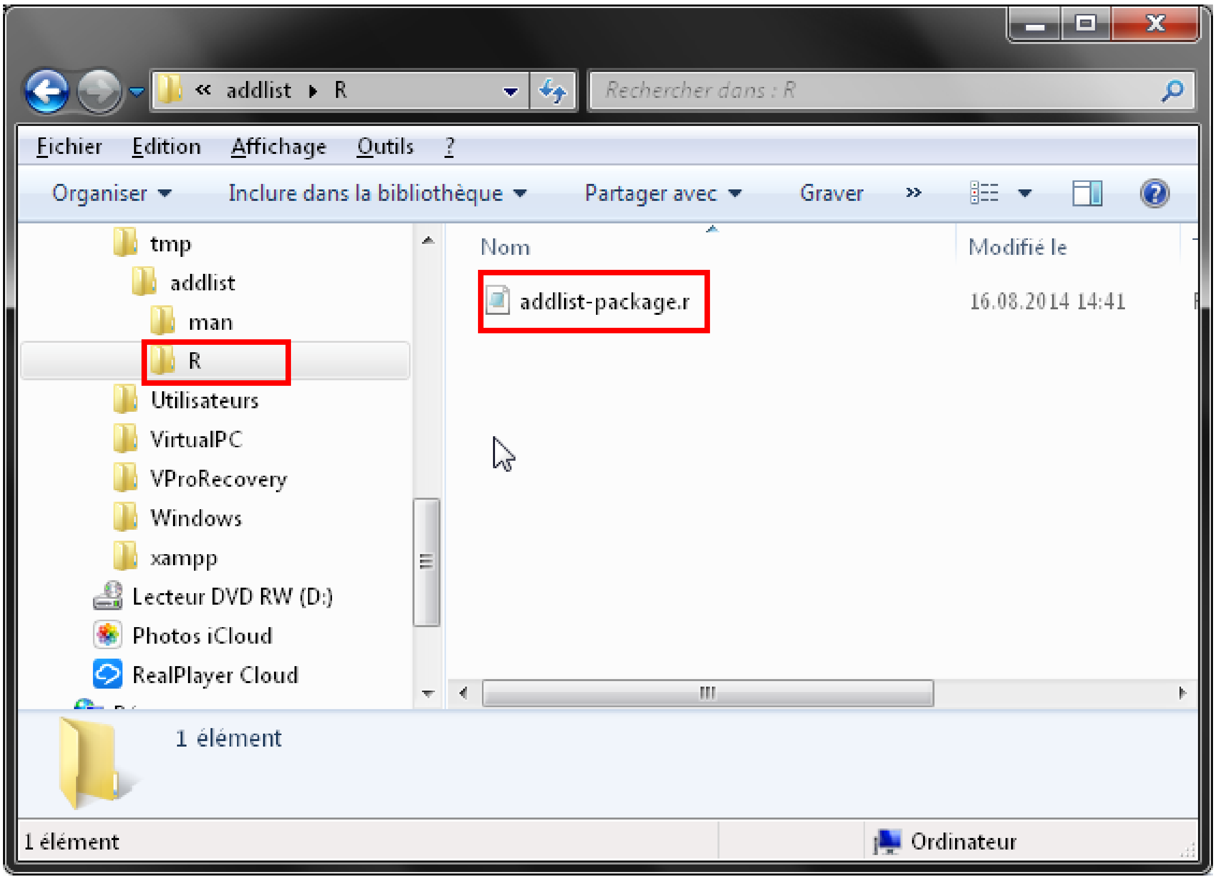
Dont le contenu est:



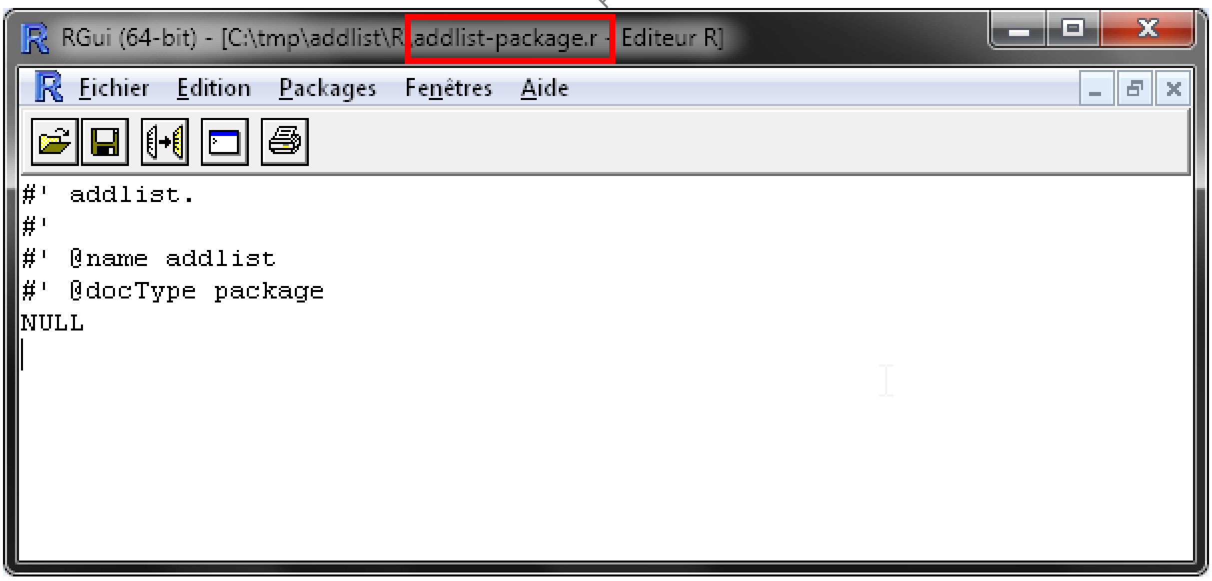
D'abord changeons le contenu du fichier DESCRIPTION pour y mettre les informations adéquates:



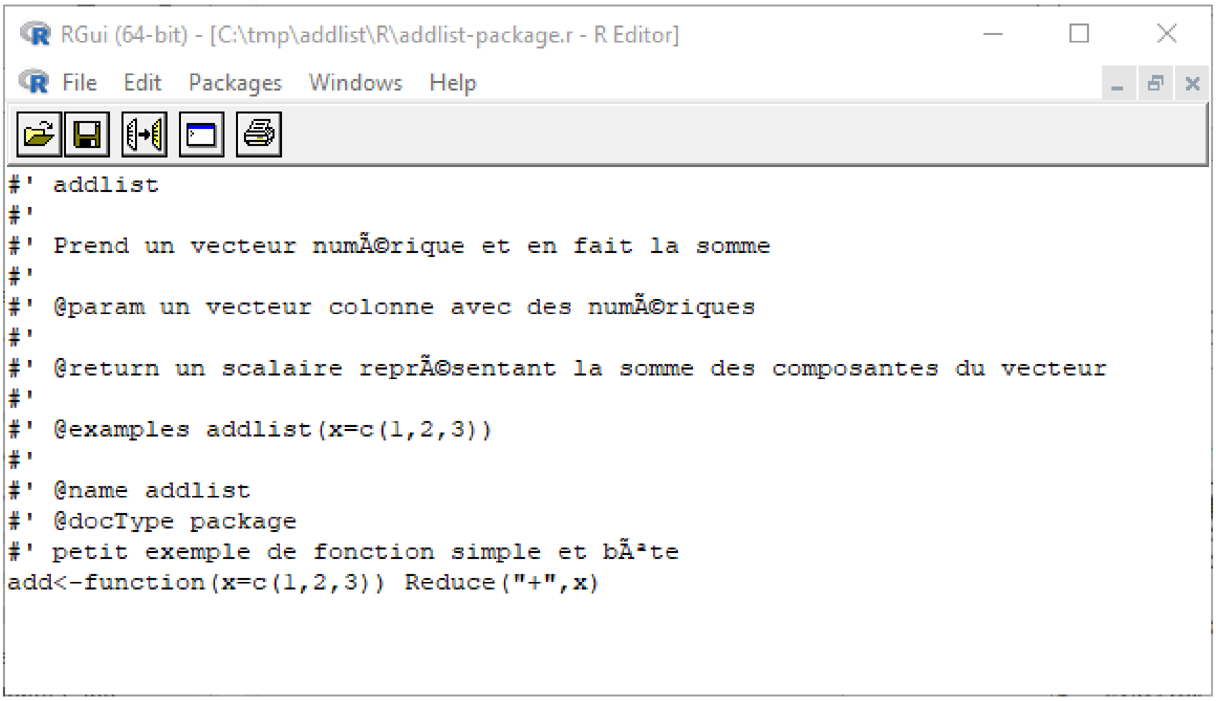
Dans le dossier R du package, il y a notre fichier de script principal:



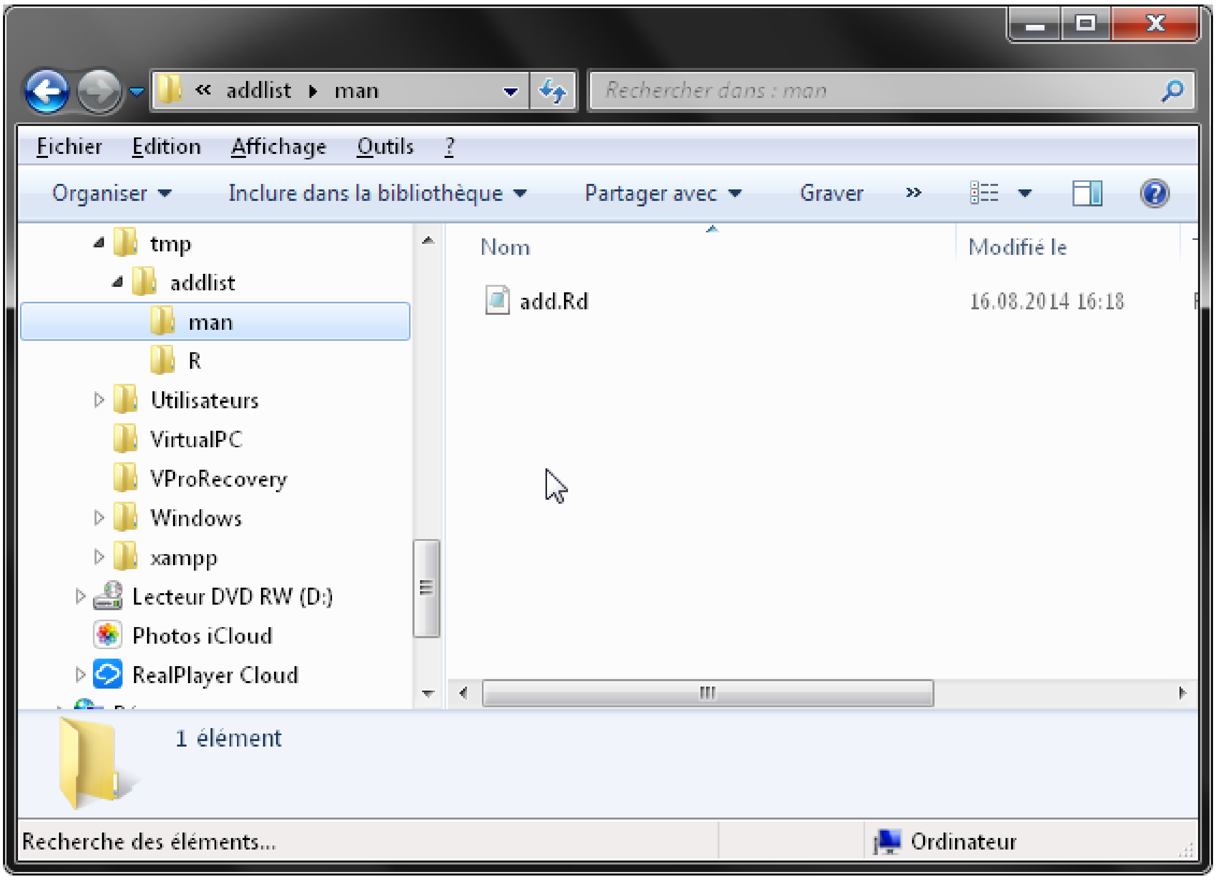
Nous éditons ce fichier qui contiendra la fonction et sa description:



Pour le changer comme ci-dessous:



Dans le dossier **man** (abréviation de *manual*), pour définir l'aide de la fonction, nous rajoutons le fichier *.Rd suivant:



Et nous y mettons le minimum syndical:

add.Rd - Notepad

```
\name{add}
\alias{add}
\title{
Cette fonction utilise Reduce pour faire une somme
}
\usage{
add(x=c(3,4,7))
}
\arguments{
\item{x}{Un vecteur de valeurs numériques}
}
\value{
Un scalaire représentant la somme de toutes les valeurs numériques
}
\description{
Somme toutes les composantes d'un vecteur
}
\examples{
add(x=c(1,2,3))
}
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Ensuite, nous "compilons" le package à l'aide de la commande **build( )** du package **devtools**:

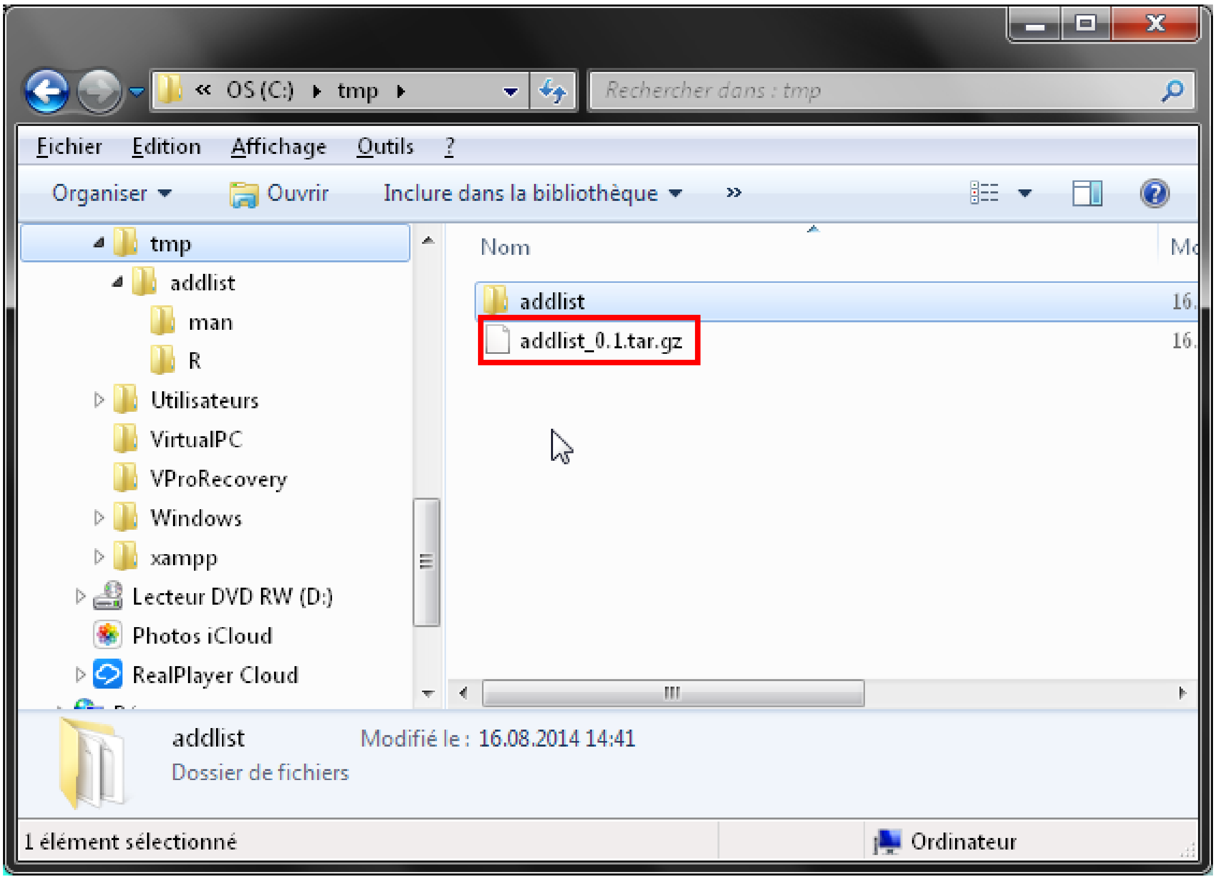
RGui (64-bit) - [R Console]

```
> build("addlist")
"C:/PROGRA~1/R/R-30~1.2/bin/x64/R" --vanilla CMD build "C:\tmp\addlist"  \
  --no-manual --no-resave-data

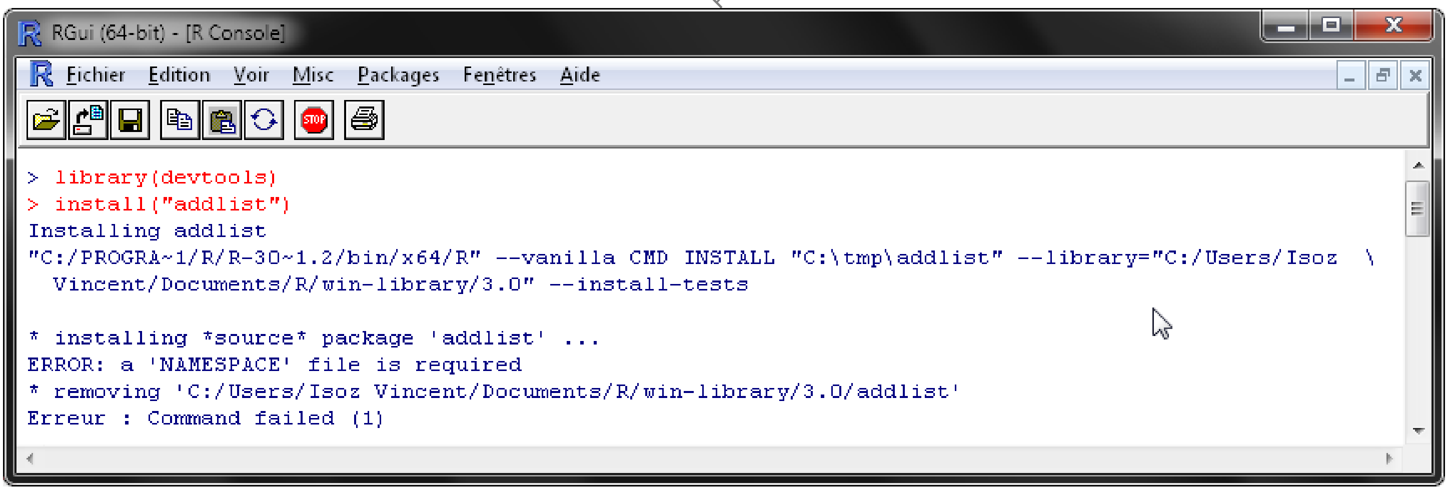
* checking for file 'C:\tmp\addlist/DESCRIPTION' ... OK
* preparing 'addlist':
* checking DESCRIPTION meta-information ... OK
* checking for LF line-endings in source and make files
* checking for empty or unneeded directories
* building 'addlist_0.1.tar.gz'

[1] "C:/tmp/addlist_0.1.tar.gz"
>
```

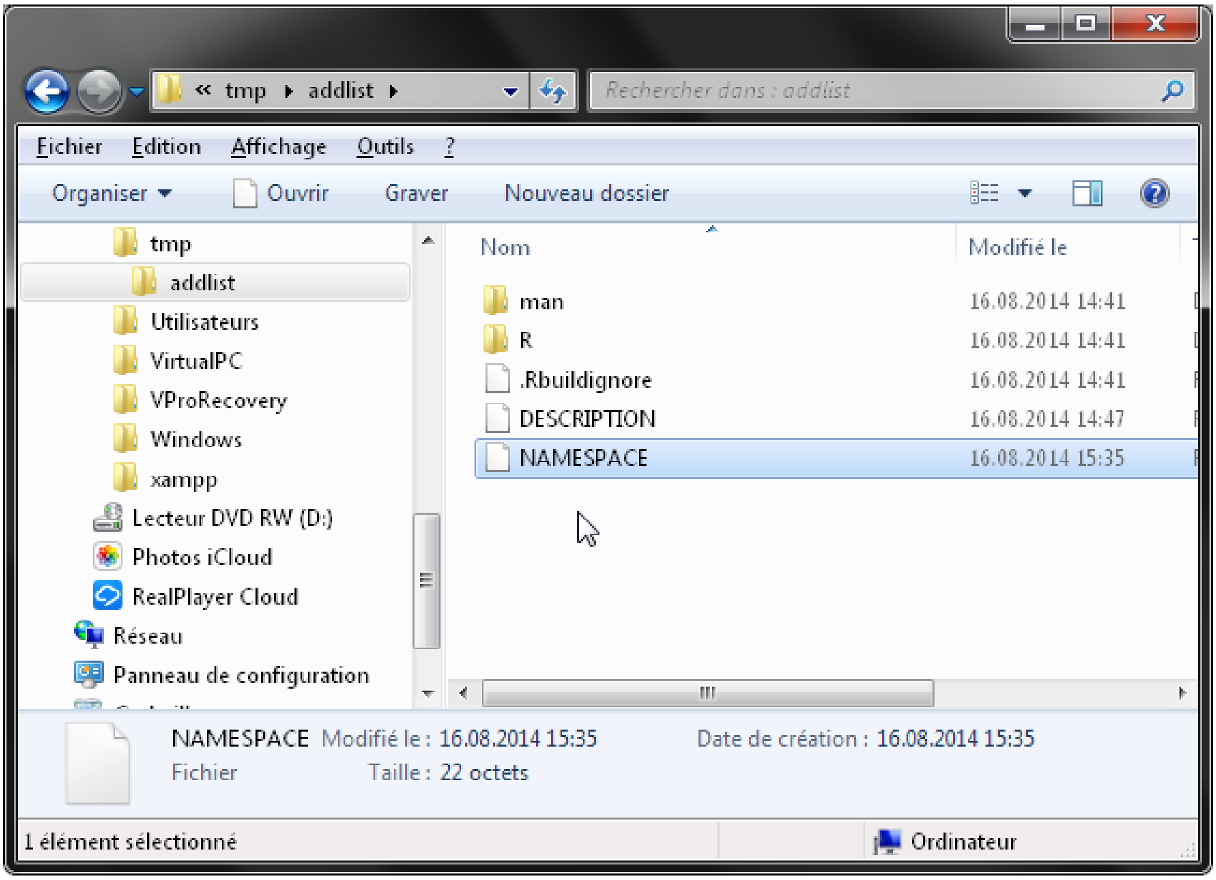
Ce qui donnera:



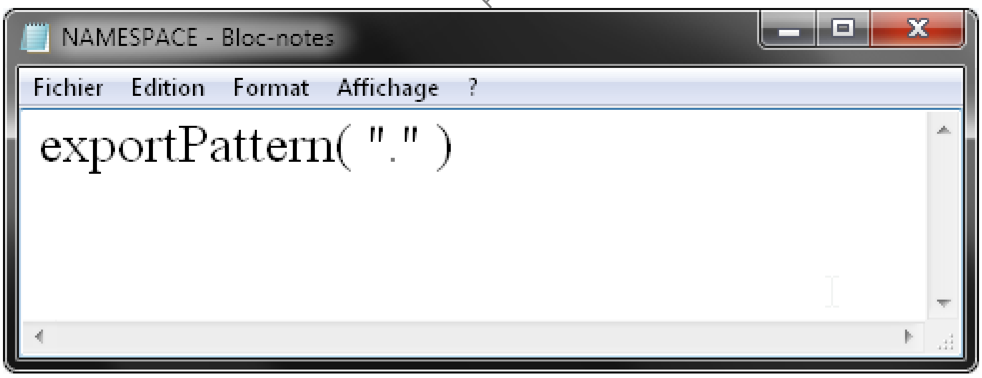
Une fois ceci fait, nous pouvons utiliser la commande **install( )** du package **devtools**:



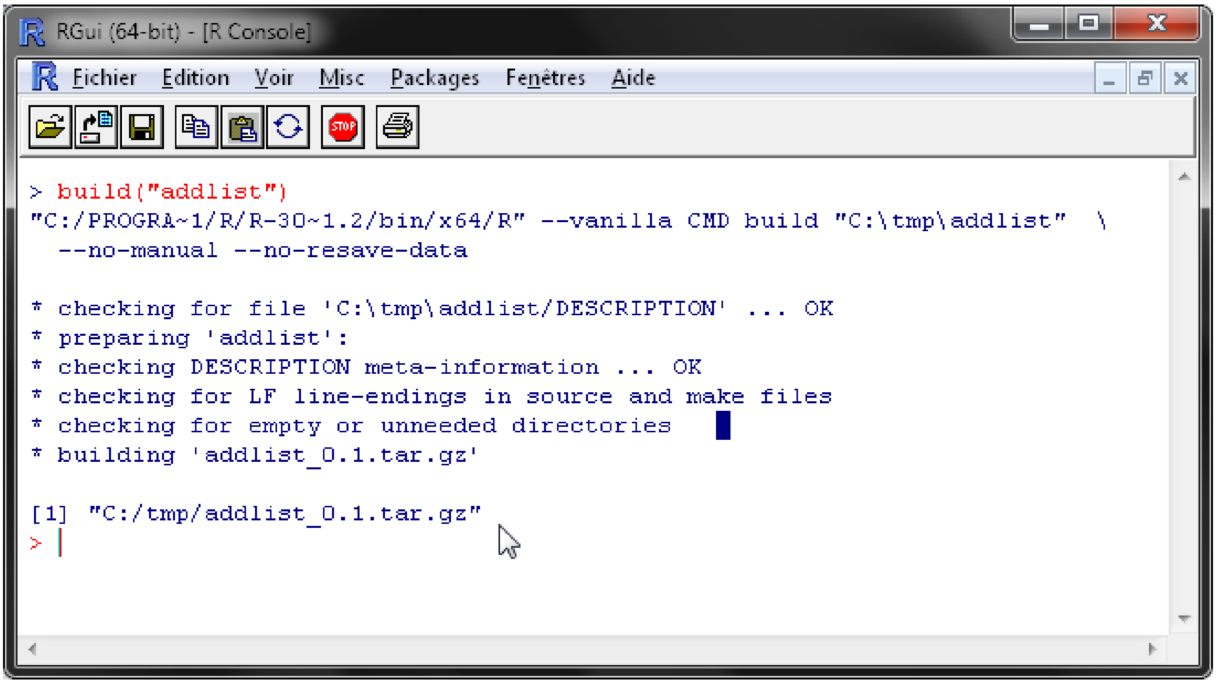
Mais une erreur va empêcher l'installation. C'est parce qu'il fallait ajouter un fichier NAMESPACE au préalable avant de compiler:



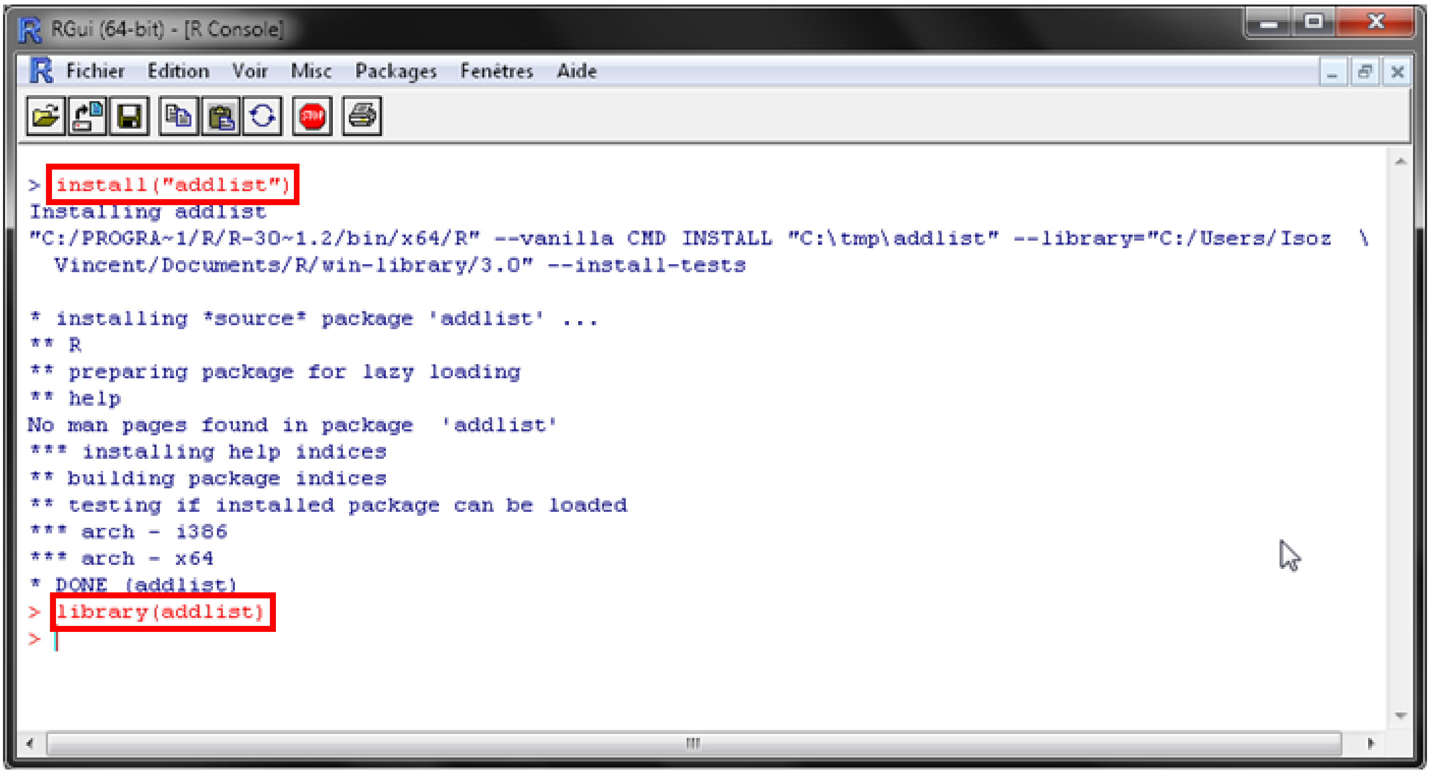
avec le contenu suivant dans le cas particulier qui nous concerne:



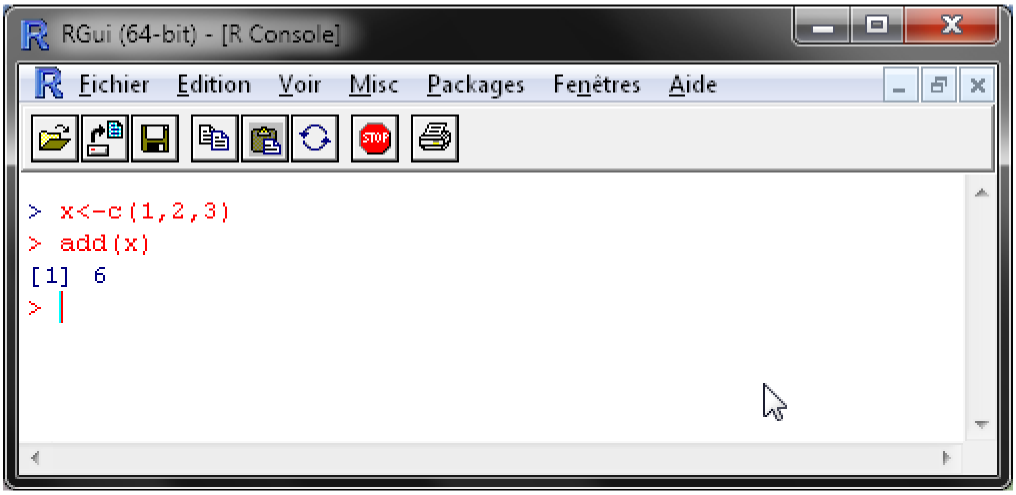
Nous pouvons alors recompiler:



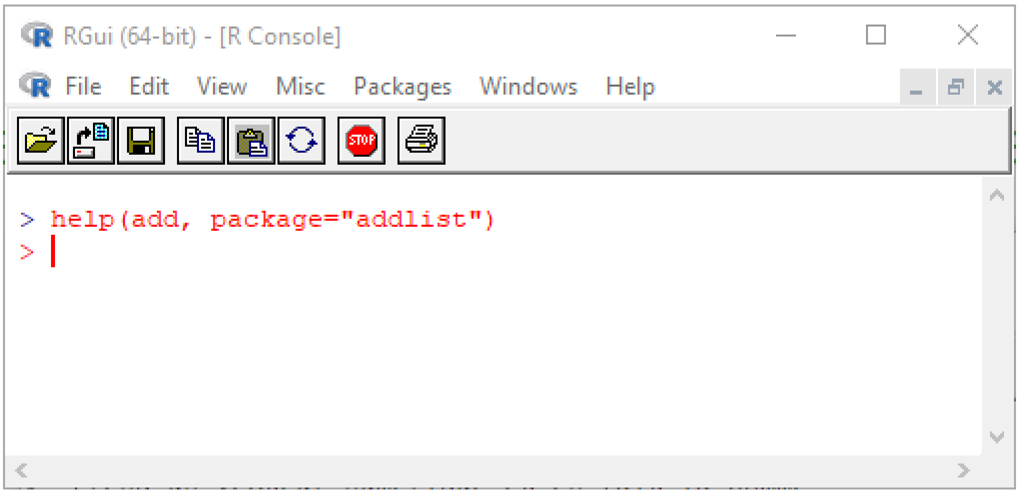
et nous l'installons cette fois-ci sans erreurs et il ne faut évidemment pas oublier le charger aussi!:



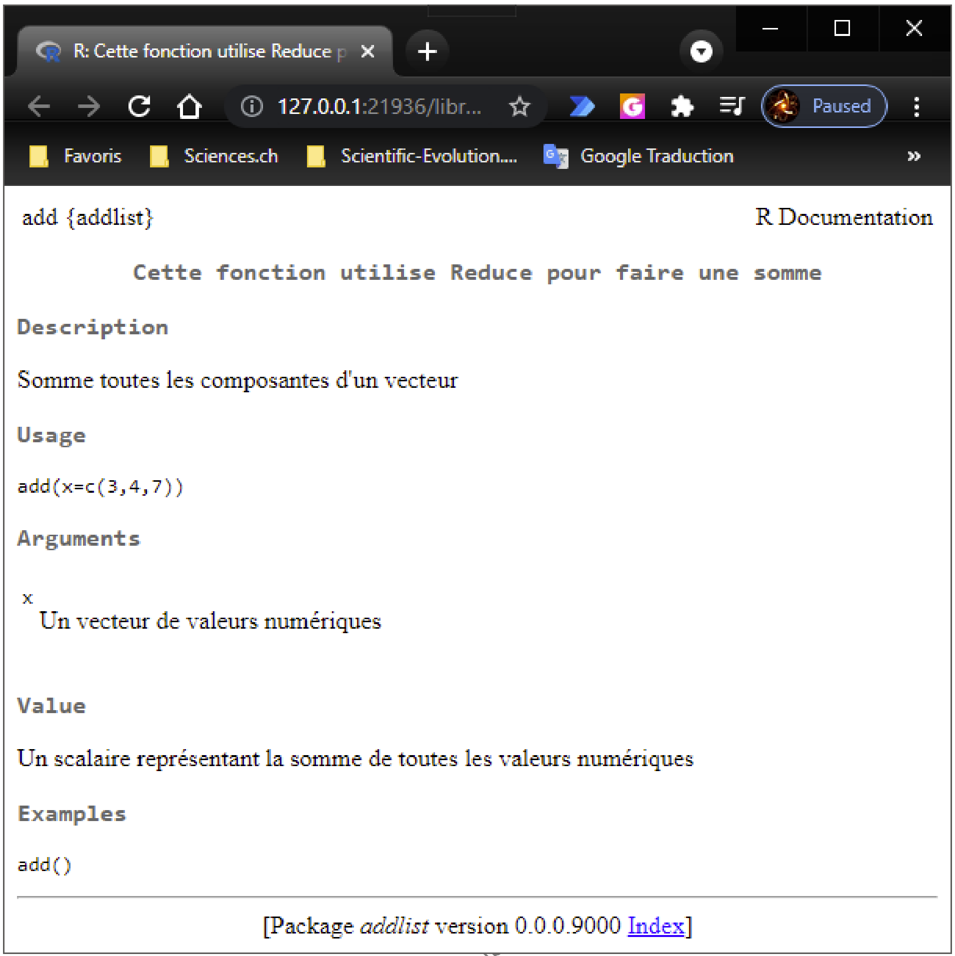
Nous pouvons ensuite utiliser notre fonction comme toute autre fonction:



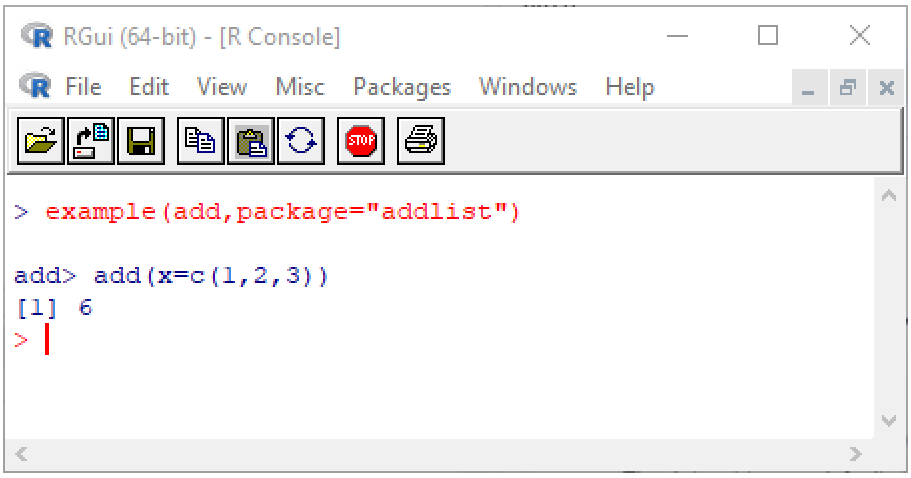
et si nous faisons appel à l'aide:



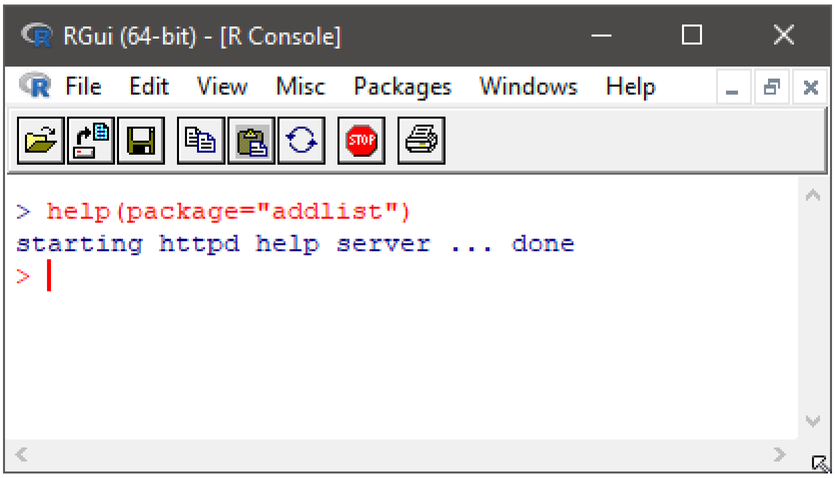
Ce qui donnera:



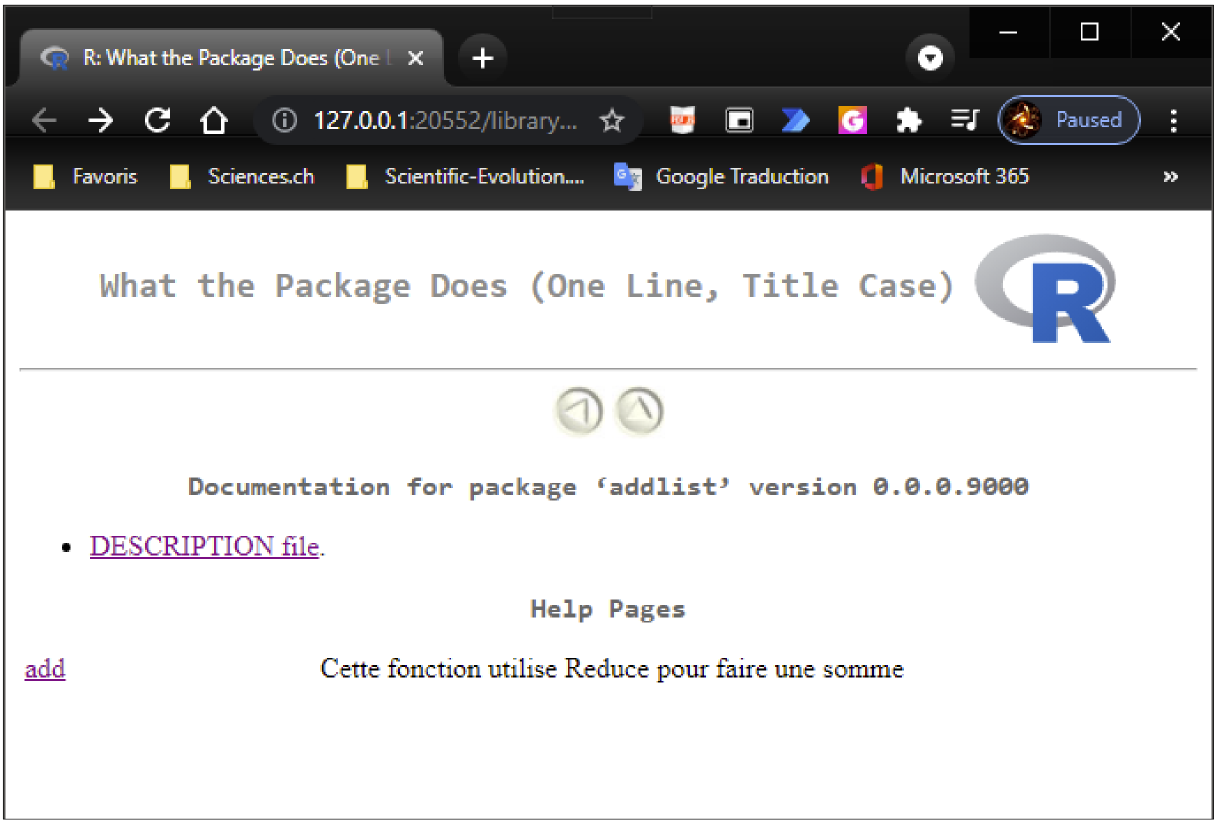
On peut aussi écrire pour avoir un exemple:



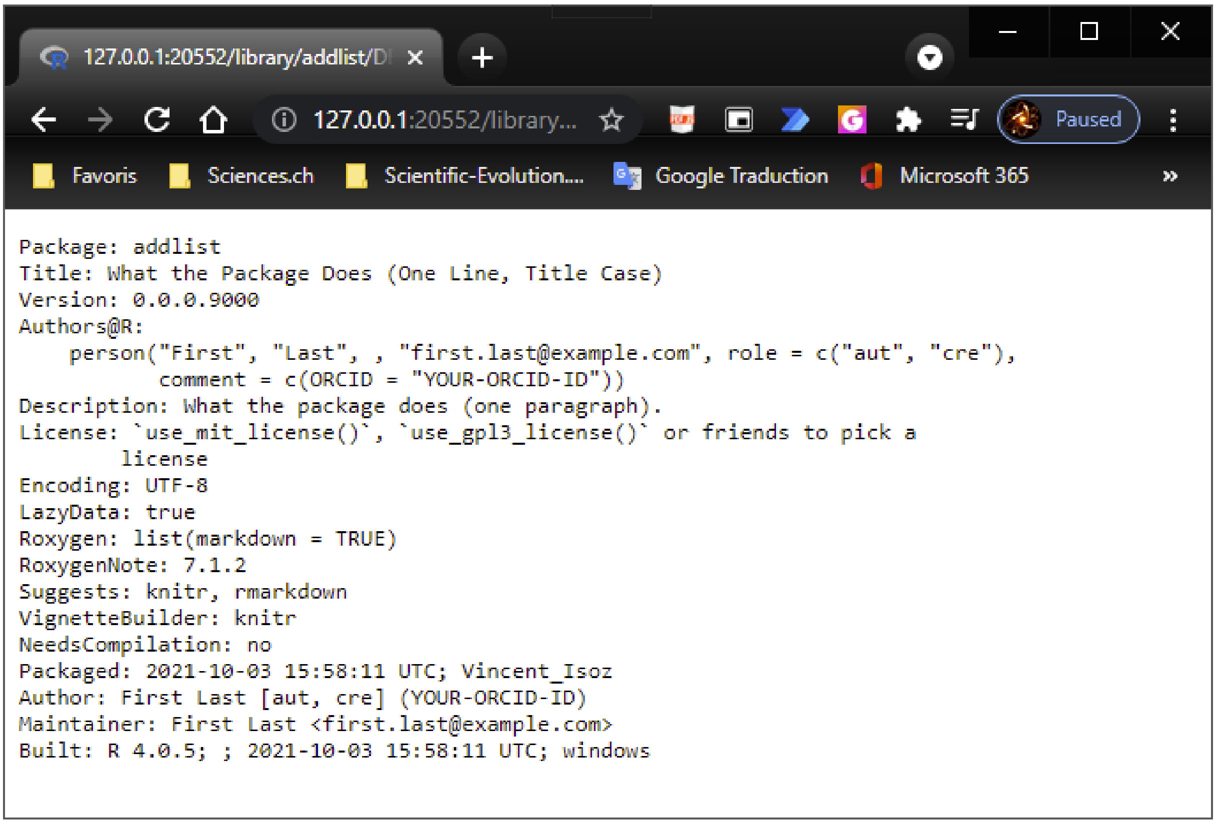
On a aussi:



qui donne:

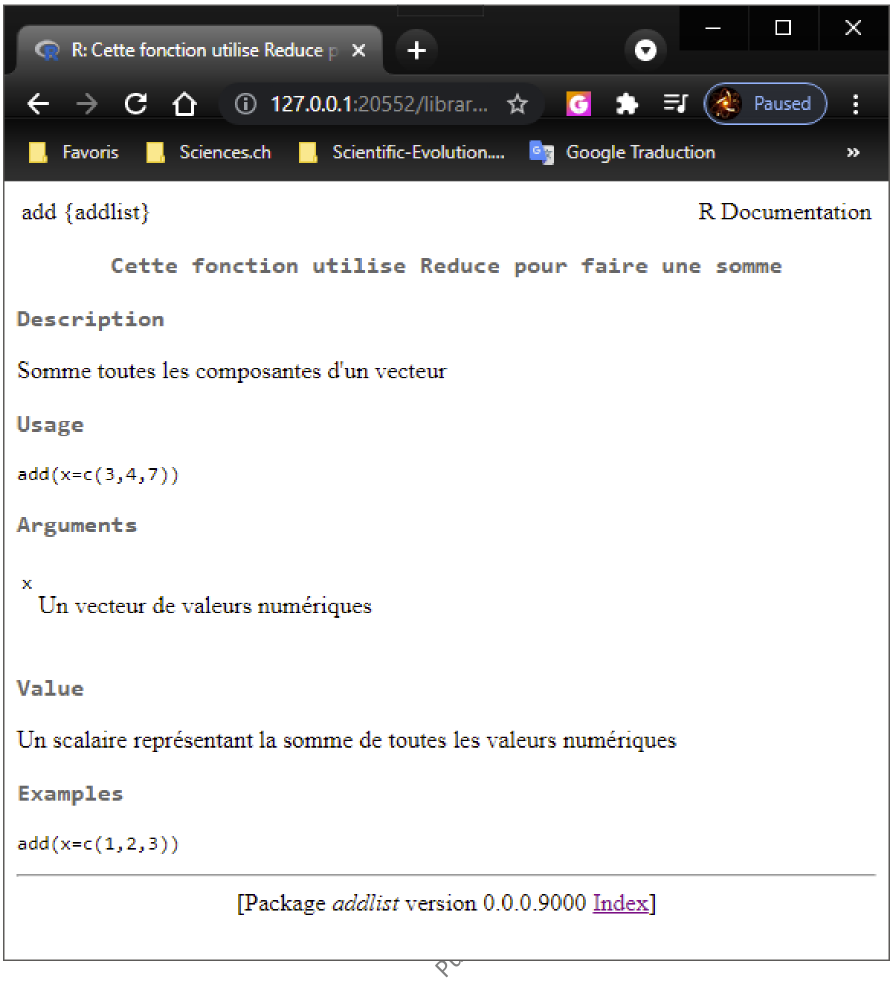


Si on clique sur *Description file* on obtient:



```
Package: addlist
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R:
    person("First", "Last", , "first.last@example.com", role = c("aut", "cre"),
           comment = c(ORCID = "YOUR-ORCID-ID"))
Description: What the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to pick a
        license
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.2
Suggests: knitr, rmarkdown
VignetteBuilder: knitr
NeedsCompilation: no
Packaged: 2021-10-03 15:58:11 UTC; Vincent_Isoz
Author: First Last [aut, cre] (YOUR-ORCID-ID)
Maintainer: First Last <first.last@example.com>
Built: R 4.0.5; ; 2021-10-03 15:58:11 UTC; windows
```

et si on clique sur *add* on retombe sur:

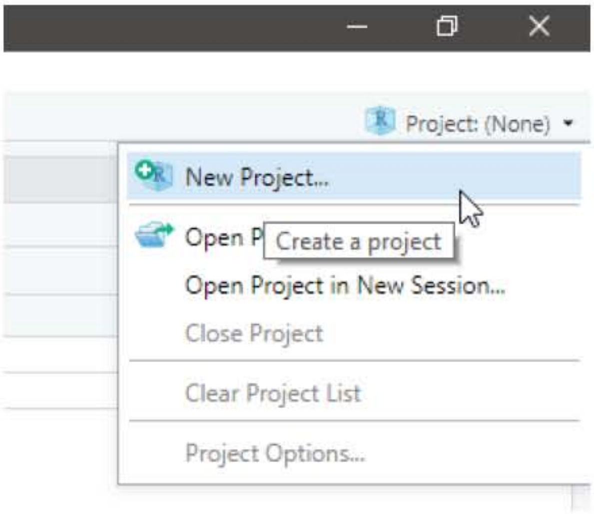


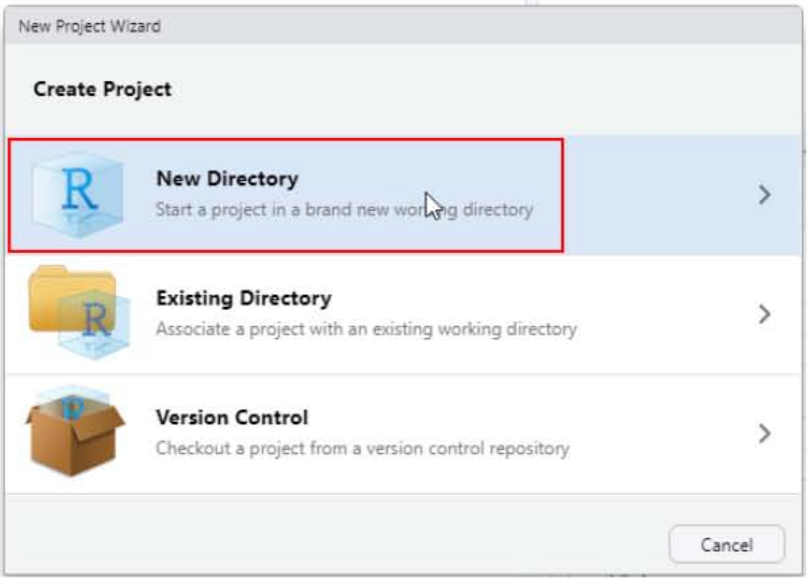
Si un jour nous avons le temps, nous montrerons comment rajouter un vignette à ce package.

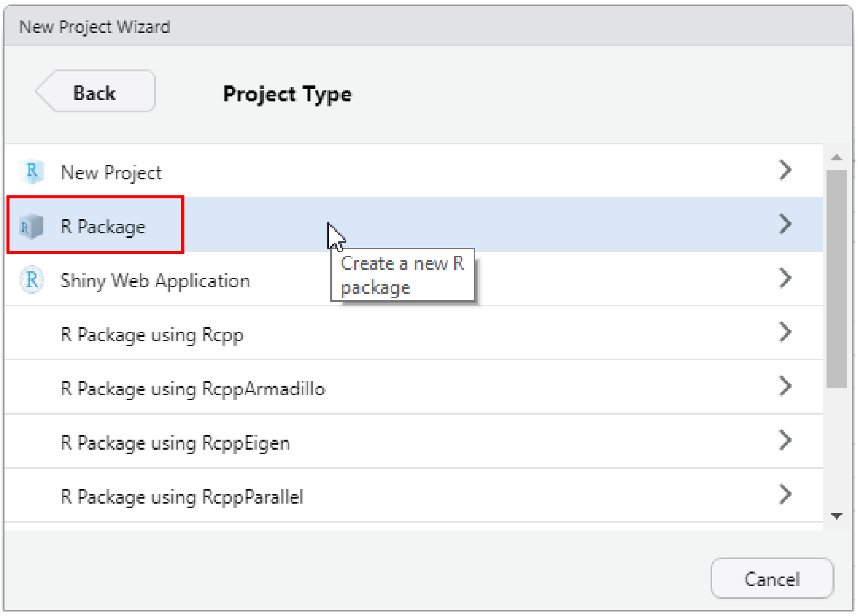
Une fois terminé, on peut soumettre le package à CRAN:

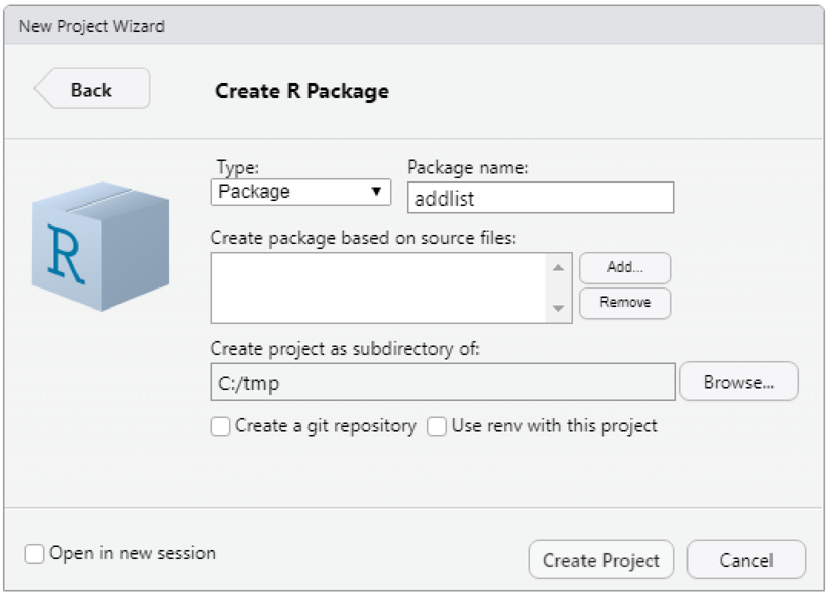
https://cran.r-project.org/submit.html

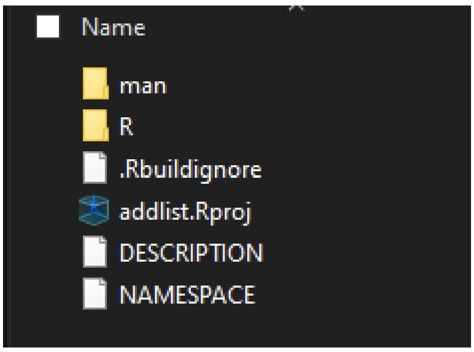
Avec R Studio

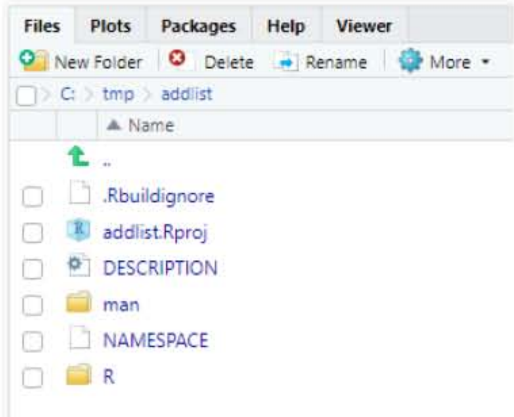


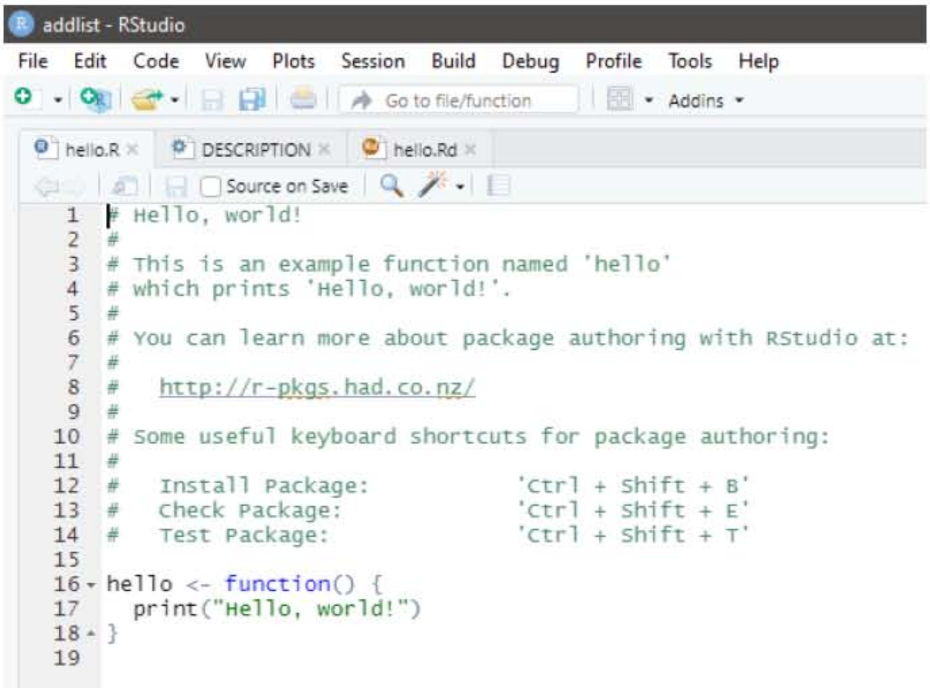


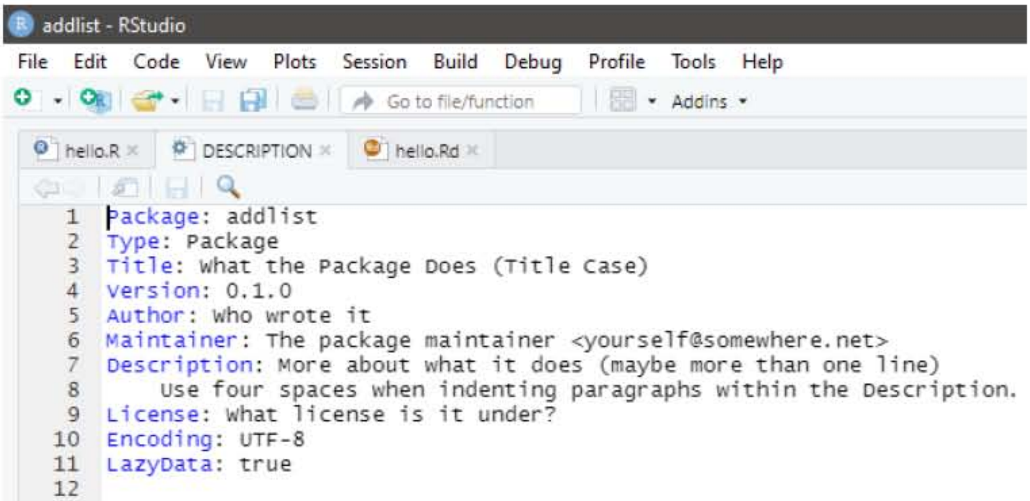

New Project Wizard
Back
Project Type
New Project
R Package
Create a new R package
Shiny Web Application
R Package using Rcpp
R Package using RcppArmadillo
R Package using RcppEigen
R Package using RcppParallel
Cancel

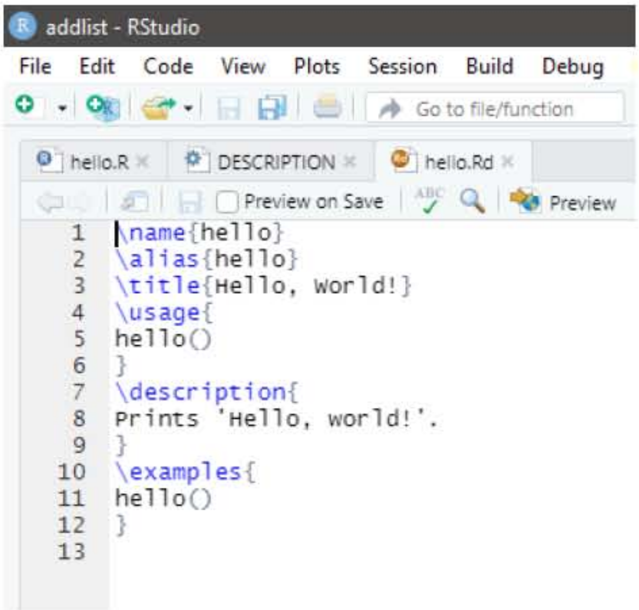

New Project Wizard
Back
Create R Package
Type:
Package
Package name:
addlist
Create package based on source files:
Add...
Remove
Create project as subdirectory of:
C:/tmp
Browse...
Create a git repository
Use renv with this project
Open in new session
Create Project
Cancel

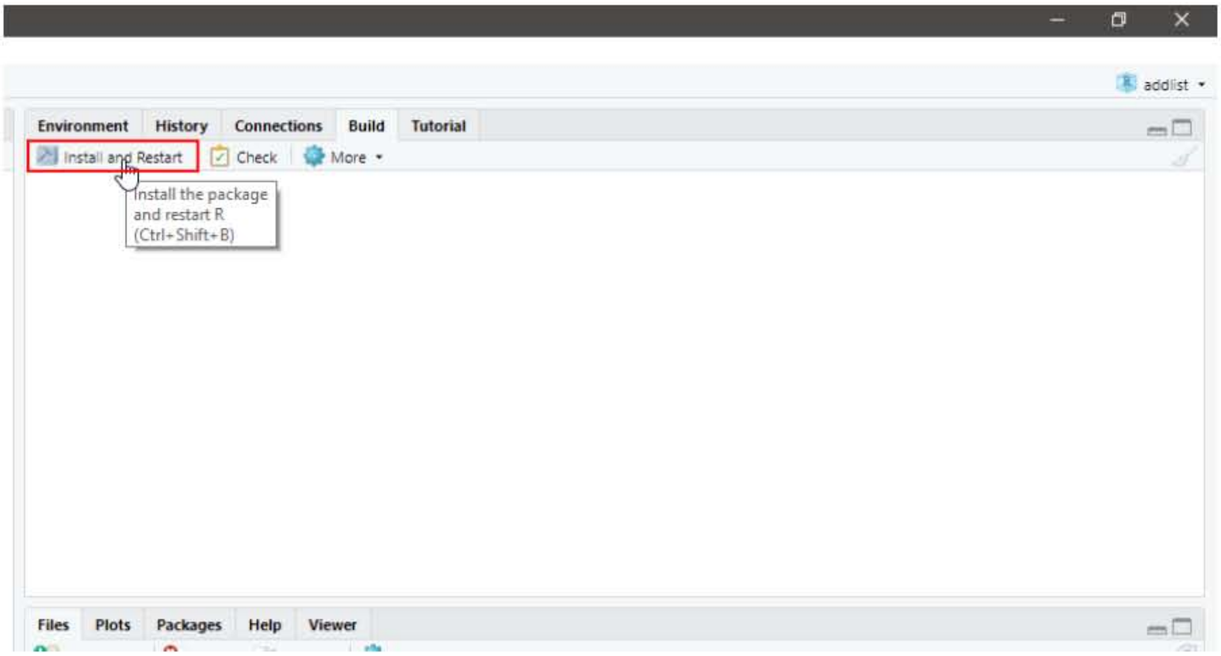

Name
man
R
.Rbuildignore
addlist.Rproj
DESCRIPTION
NAMESPACE

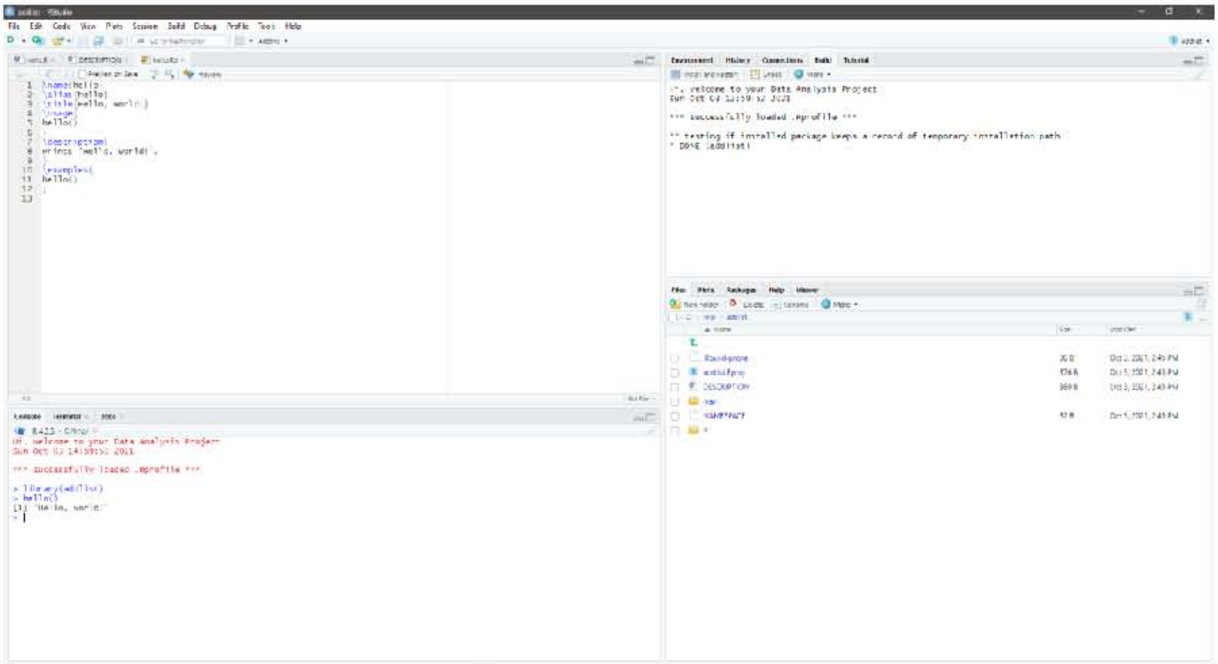

Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
C: > tmp > addlist
Name
..
.Rbuildignore
addlist.Rproj
DESCRIPTION
man
NAMESPACE
R

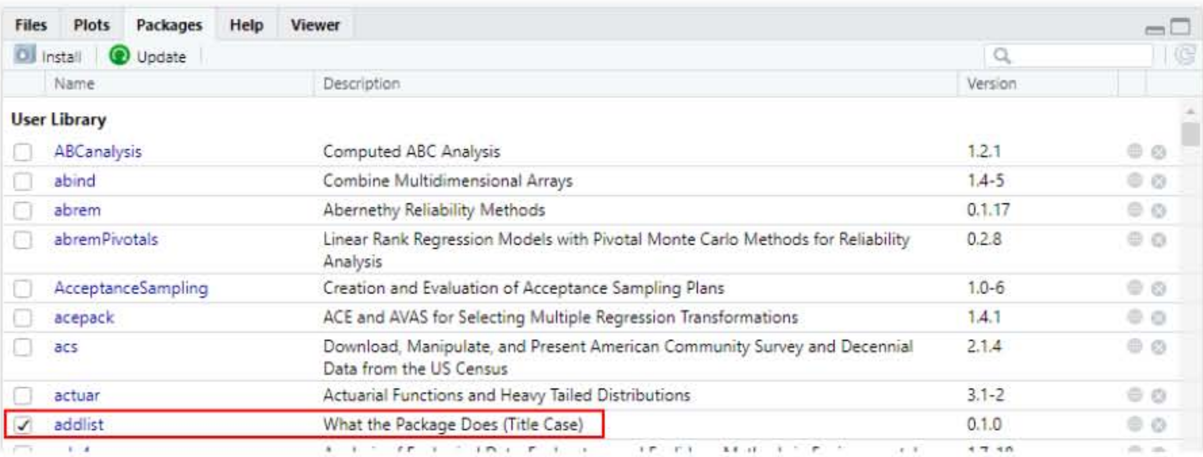

addlist - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
hello.R
DESCRIPTION
hello.Rd
Source on Save
Hello, world!
#
This is an example function named 'hello'
which prints 'Hello, world!'.
#
You can learn more about package authoring with RStudio at:
#
http://r-pkgs.had.co.nz/
#
Some useful keyboard shortcuts for package authoring:
#
Install Package: 'Ctrl + Shift + B'
Check Package: 'Ctrl + Shift + E'
Test Package: 'Ctrl + Shift + T'
hello <- function() {
  print("Hello, world!")
}

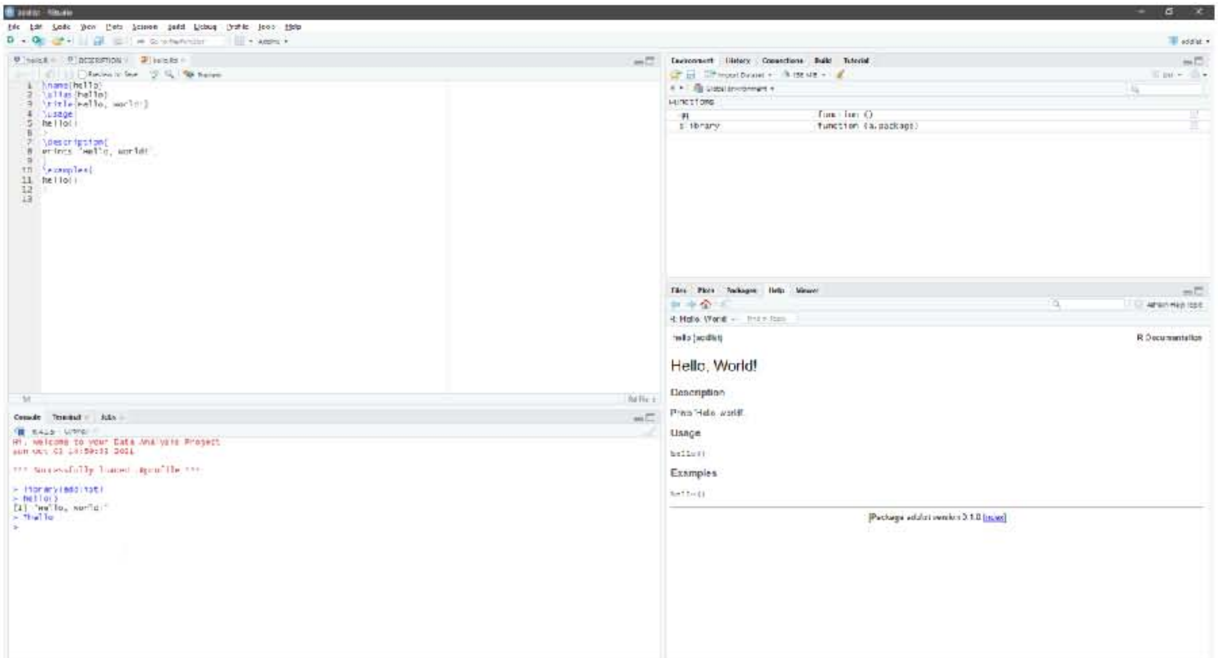

addlist - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
hello.R
DESCRIPTION
hello.Rd
Package: addlist
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true


addlist - RStudio
File Edit Code View Plots Session Build Debug
Go to file/function
hello.R
DESCRIPTION
hello.Rd
Preview on Save
Preview
1 \name{hello}
2 \alias{hello}
3 \title{Hello, World!}
4 \usage{
5 hello()
6 }
7 \description{
8 Prints 'Hello, world!'.
9 }
10 \examples{
11 hello()
12 }
13


addlist
Environment History Connections Build Tutorial
Install and Restart
Check
More
Install the package
and restart R
(Ctrl+Shift+B)
Files Plots Packages Help Viewer




Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
ABCanalysis
Computed ABC Analysis
1.2.1
abind
Combine Multidimensional Arrays
1.4-5
abrem
Abernethy Reliability Methods
0.1.17
abremPivotals
Linear Rank Regression Models with Pivotal Monte Carlo Methods for Reliability Analysis
0.2.8
AcceptanceSampling
Creation and Evaluation of Acceptance Sampling Plans
1.0-6
acepack
ACE and AVAS for Selecting Multiple Regression Transformations
1.4.1
acs
Download, Manipulate, and Present American Community Survey and Decennial Data from the US Census
2.1.4
actuar
Actuarial Functions and Heavy Tailed Distributions
3.1-2
addlist
What the Package Does (Title Case)
0.1.0

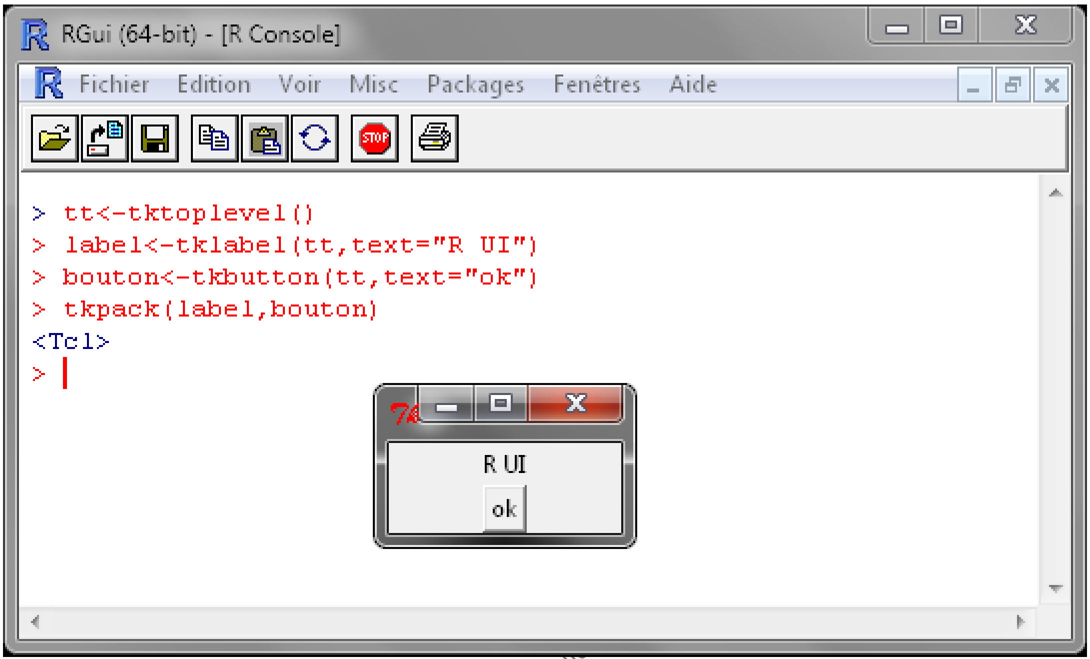

Hello, World!

Le reste est identique à la méthode sans R Studio.

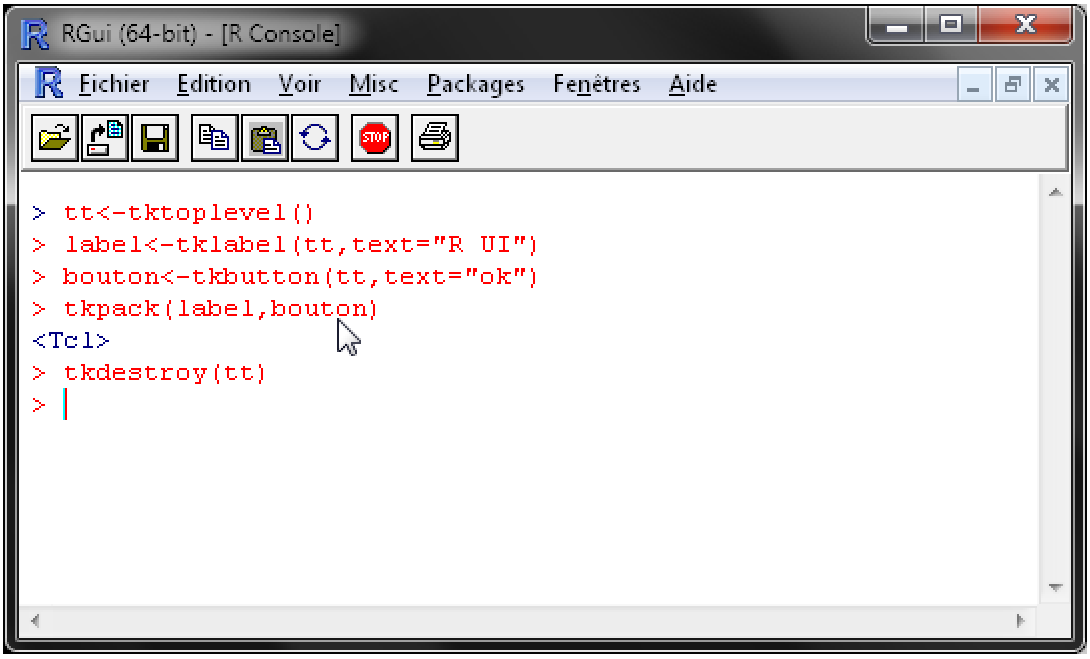
36. Interfaces pour les utilisateurs

Générées depuis R

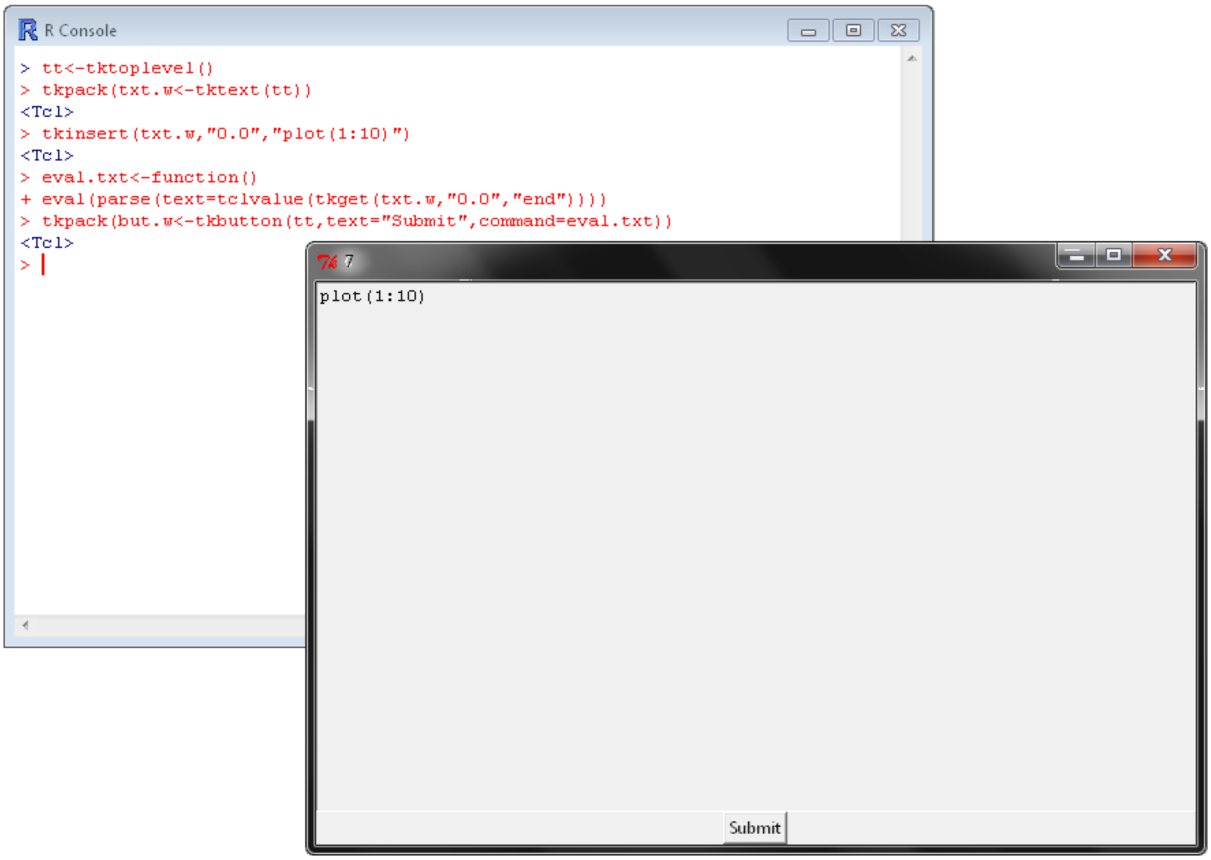
Générer des interfaces avec le package **tcltk2** (il s'agit en réalité simplement de **Tkinter** qui est aussi l'outil de base de création d'interface en Python et Linux en général!):



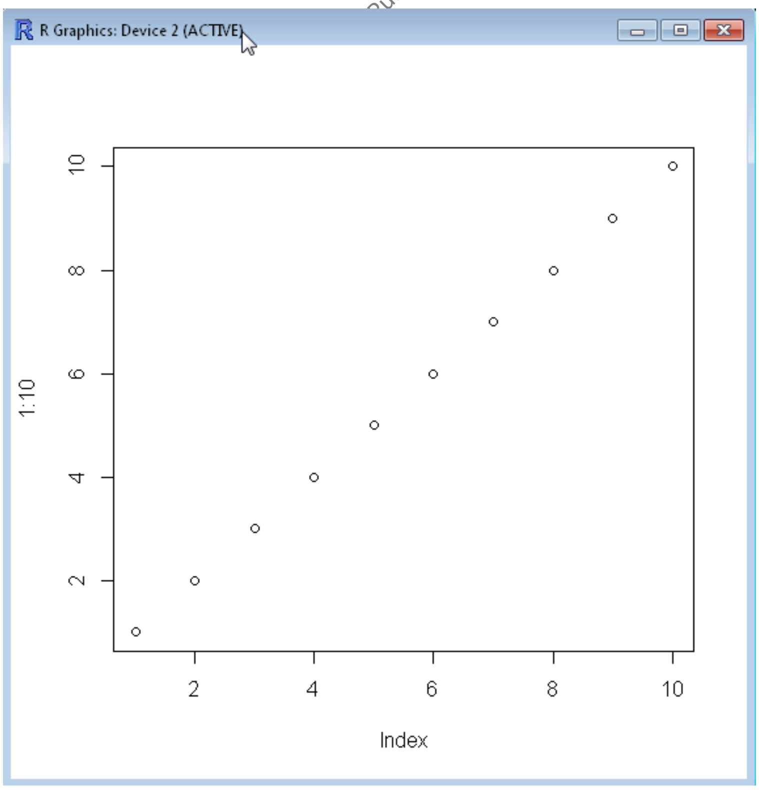
et pour ferme la fenêtre on utilisera la commande **tkdestroy( )** :



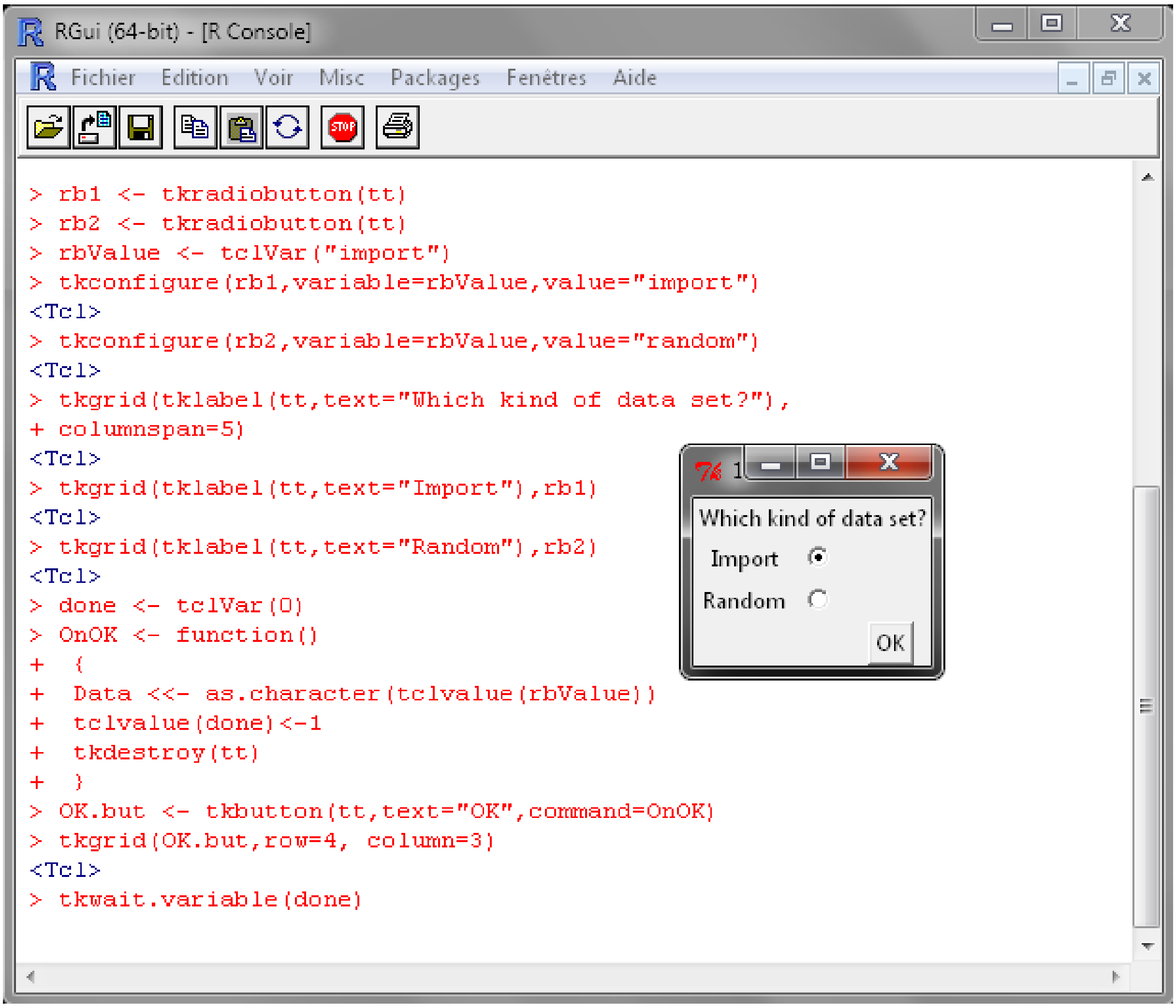
Autre exemple:



et si on clique sur **Submit** il vient:



Ou:



Un peu plus sympa:

```
require(tcltk)
require(tkrplot)
genplot = function(m,s,n){
     dat = c(rnorm(n,m,s))
          plot(density(dat),type='l',main='Simulation Loi Normale')
         }

doplot = function(...){
   m = as.numeric(tclvalue(mean))
   s = as.numeric(tclvalue(sd))
   n = as.numeric(tclvalue(n))
   genplot(m,s,n)
}

base = tktoplevel()
tkwm.title(base,'Simulation Loi Normale')
mainfrm = tkframe(base)

mean = tclVar(10)
sd   = tclVar(1)
n    = tclVar(1000)

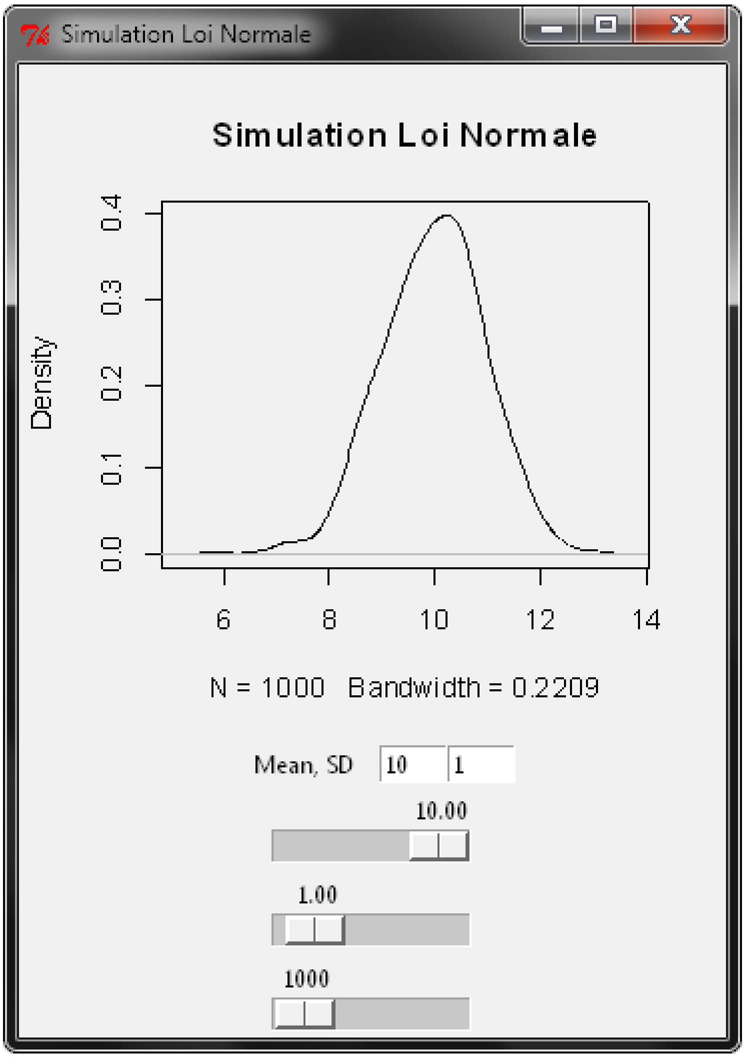
img = tkrplot(mainfrm,doplot)
scalefunc = function(...)tkrreplot(img)
sc_1 = tkscale(mainfrm,command=scalefunc,from=-10,to=10,showvalue=TRUE,
               variable=mean,resolution=.01,orient='horiz')
```

```
sc_2 = tkscale(mainfrm,command=scalefunc,from=0.1,to=10,showvalue=TRUE,
               variable=sd,resolution=.01,orient='horiz')
sc_3 = tkscale(mainfrm,command=scalefunc,from=1,to=100000,showvalue=TRUE,
               variable=n,resolution=10,orient='horiz')
tkpack(img)

m = tkframe(mainfrm)
tkpack(tklabel(m,text='Mean,  SD',width=10),side='left')
tkpack(tkentry(m,width=5,textvariable=mean),side='left')
tkpack(tkentry(m,width=5,textvariable=sd),side='left')
tkpack(m,side='top')
tkpack(sc_1)
tkpack(sc_2)
tkpack(sc_3)

tkpack(mainfrm)
```

Ce qui donne:



R via Microsoft Office Excel (RExcel)

Pour ceux qui ont absolument du mal à utiliser R en ligne de commande voici une solution possible pour utiliser R depuis Microsoft Excel:

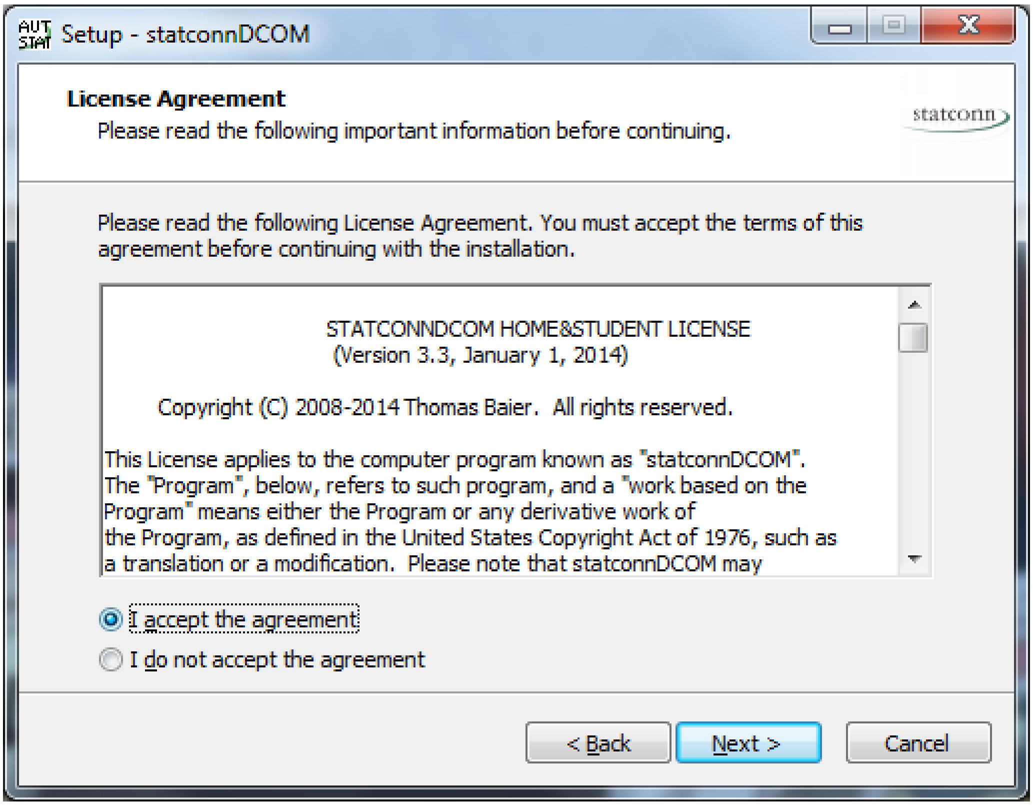
D'abord il vous faudra installer le logiciel statconnDCOM téléchargeable à l'adresse Internet suivante:

http://rcom.univie.ac.at/download/current/statconnDCOM.latest.exe

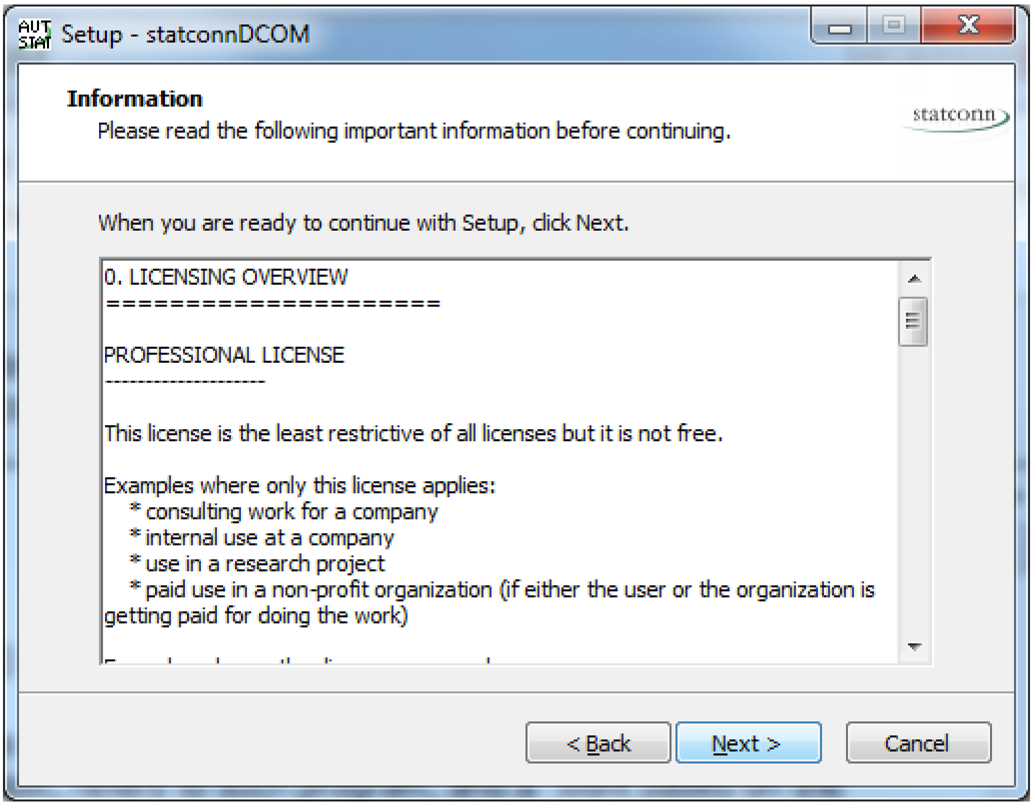
Une fois le fichier téléchargé vous lancez l'installation ce qui donnera sur un environnement Microsoft Windows:



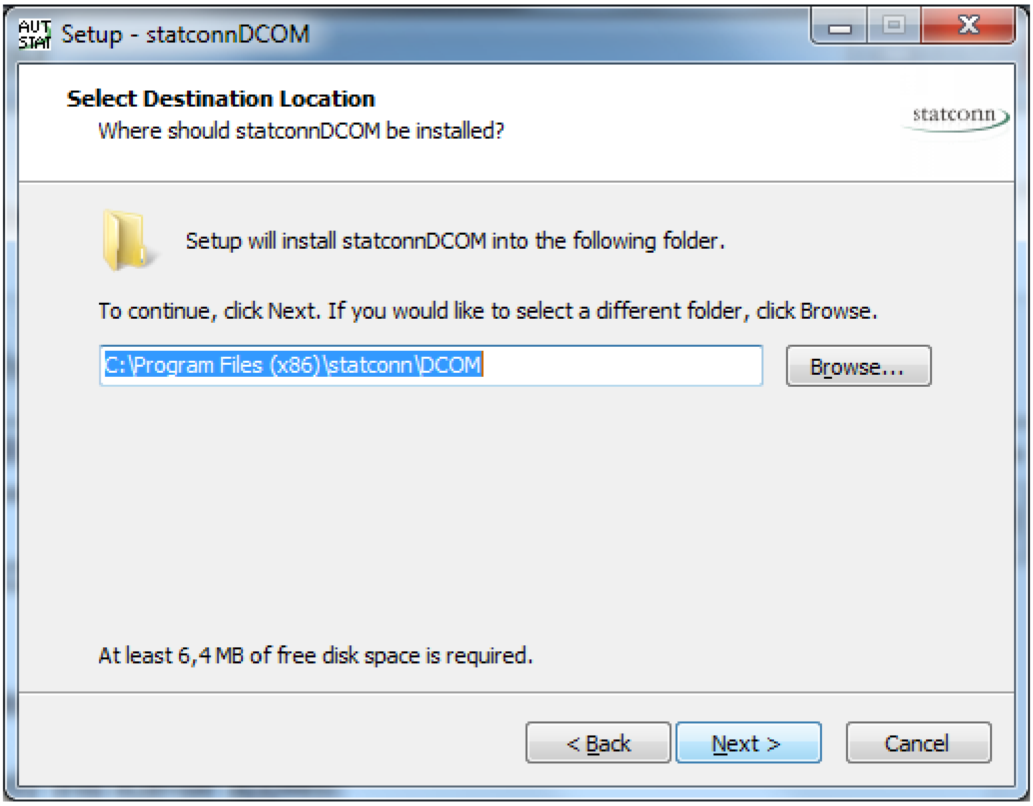
Vous cliquez sur **Next**:



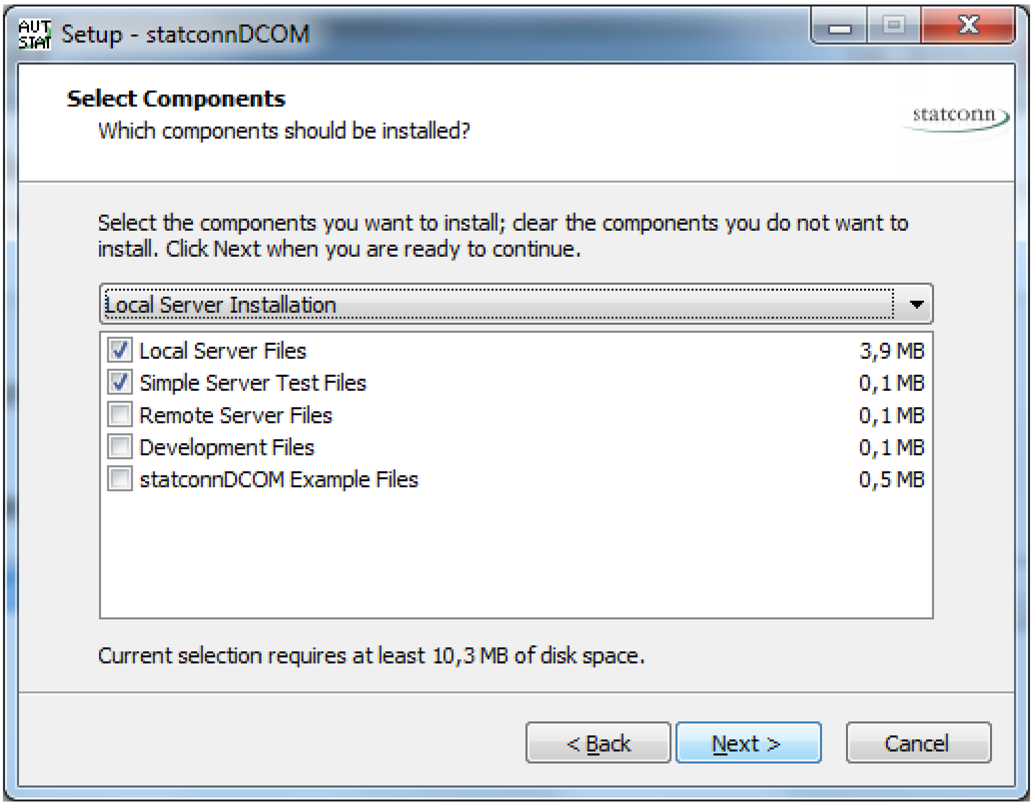
Vous cliquez sur **I accept the agreement** et sur **Next**:



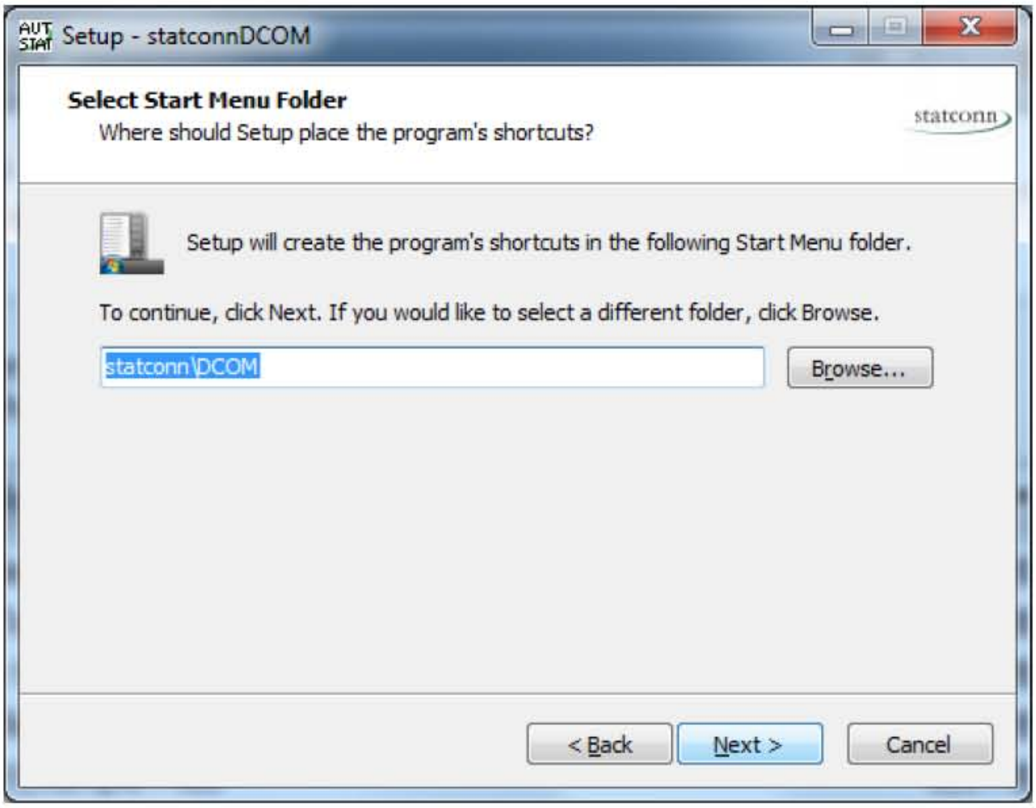
Vous cliquez à nouveau sur **Next**:



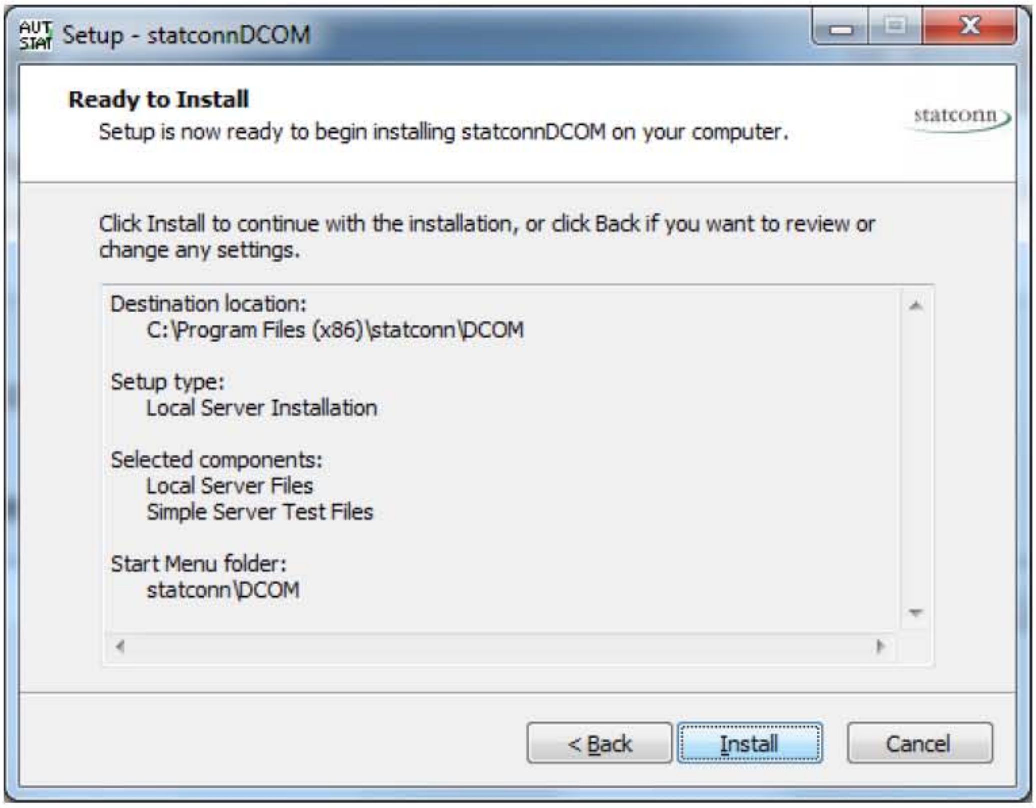
et **Next**...:



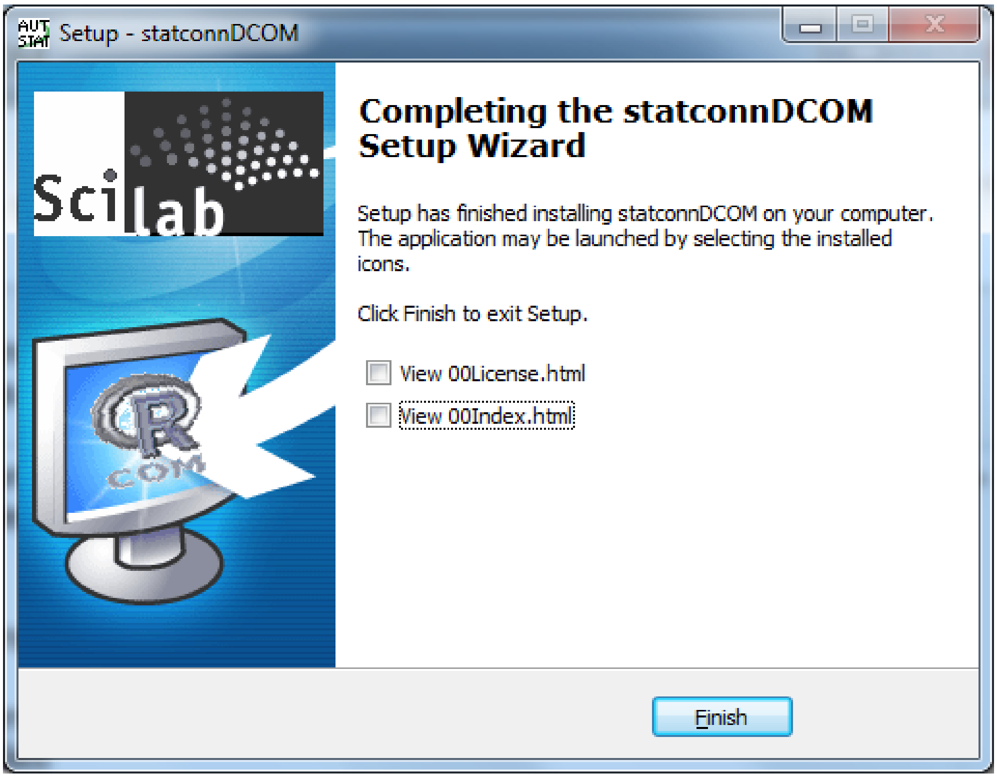
et **Next**:



et **Next**:



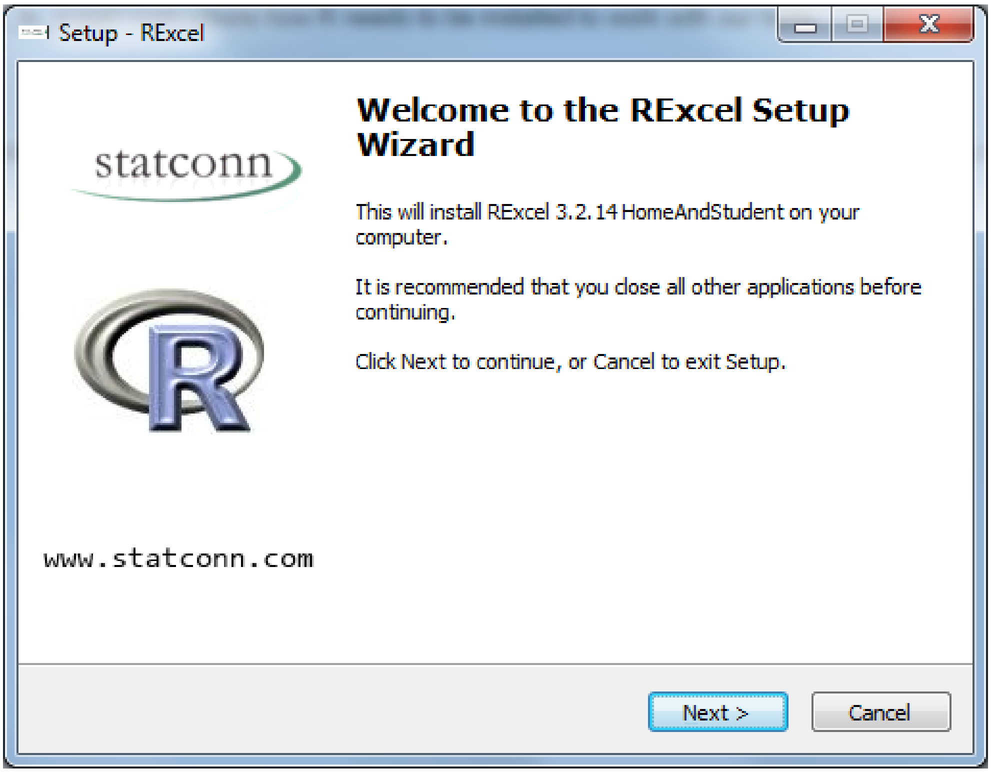
et enfin **Install** pour avoir après quelques secondes:



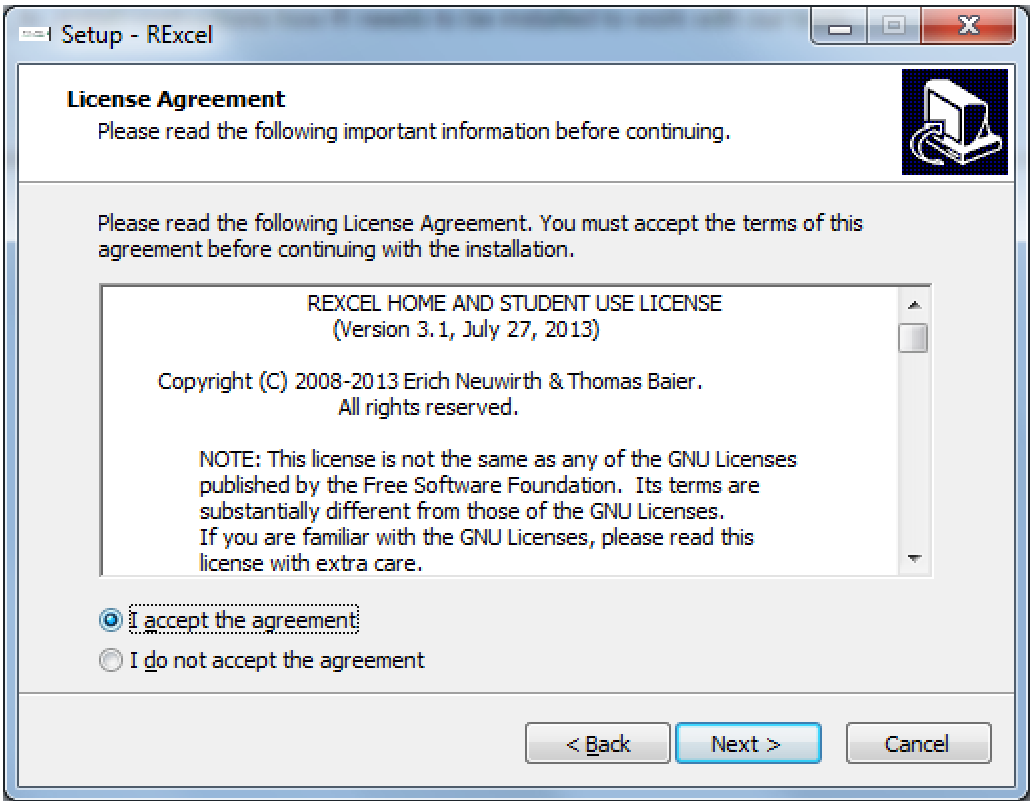
et on clique sur **Finish**. Une fois ceci fait, vous télécharger le logiciel RExcel sur le lien suivant:

http://rcom.univie.ac.at/download.html

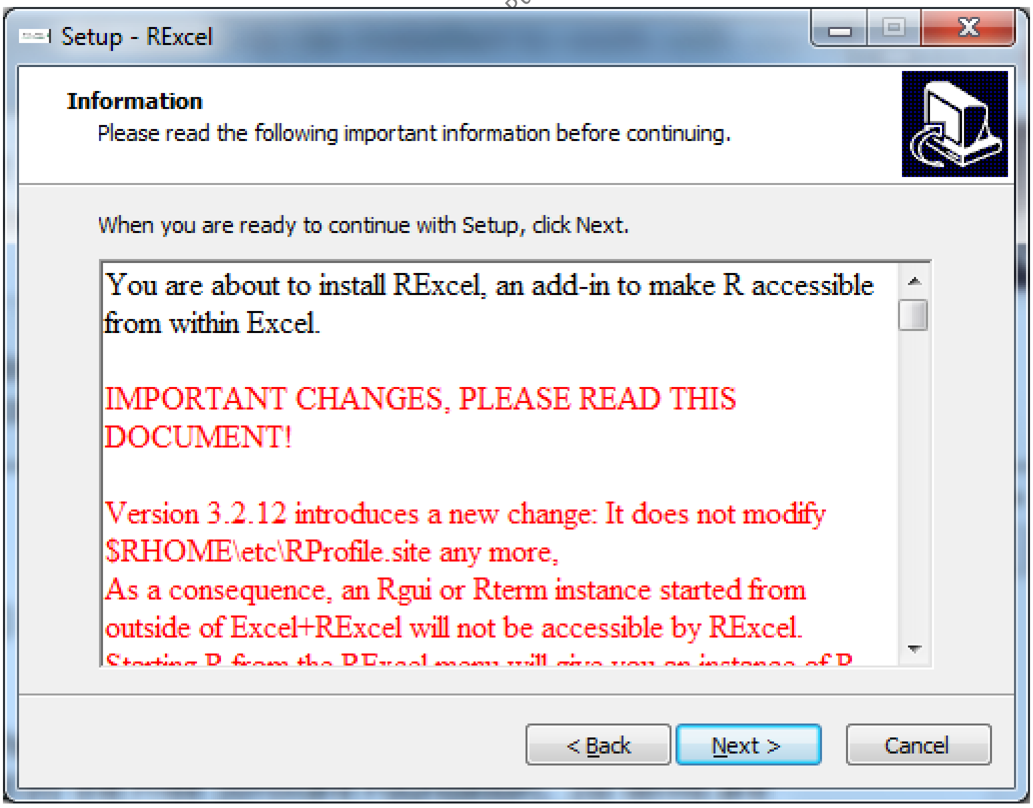
Vous aurez alors après le téléchargement et à l'exécution de l'installation, la boîte de dialogue suivante:



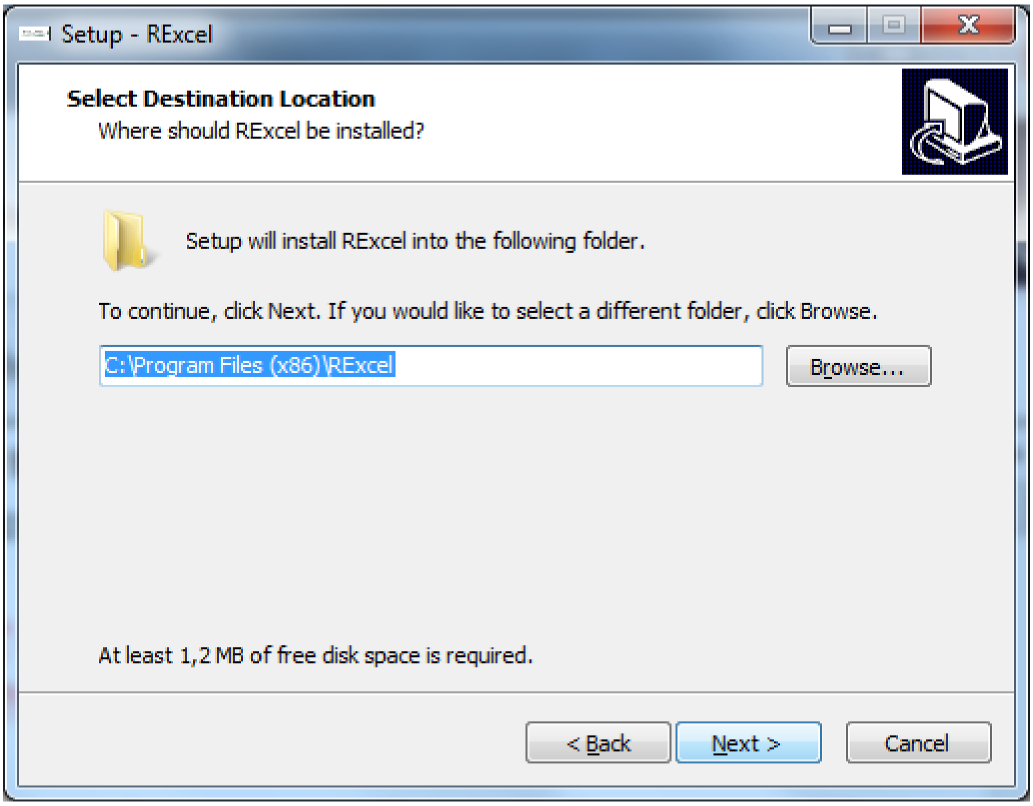
Vous cliquez sur **Next**:



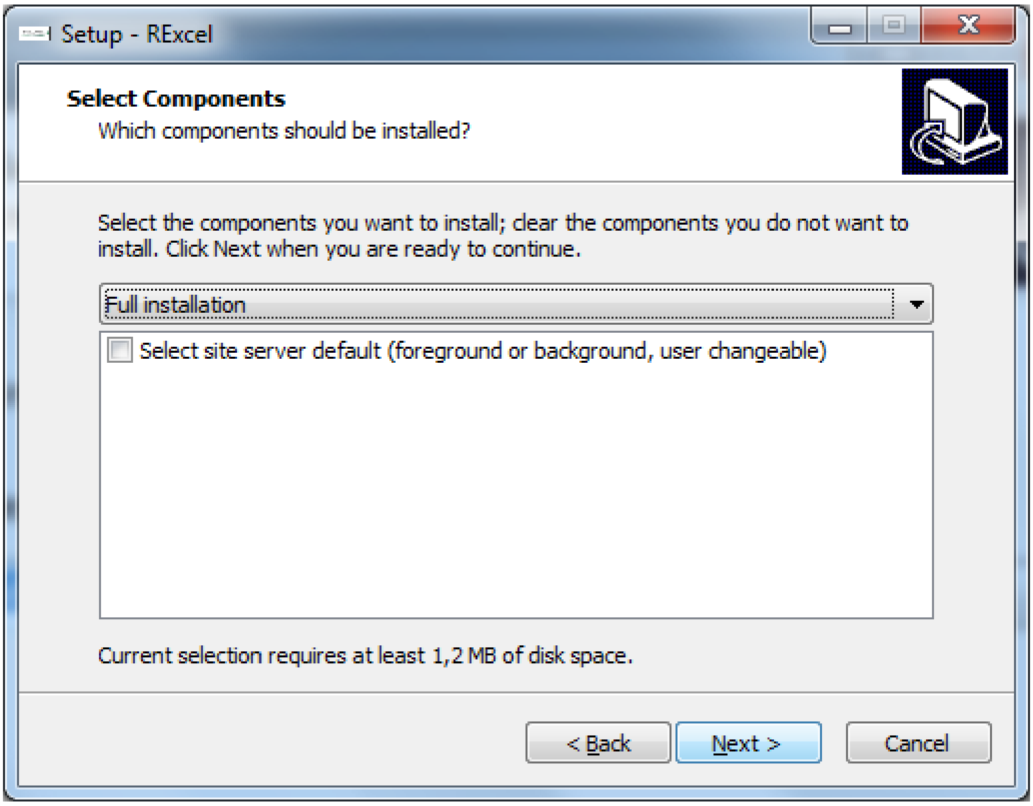
Ci-dessus vous cliquez sur **I accept the agreement** et ensuite sur **Next**:



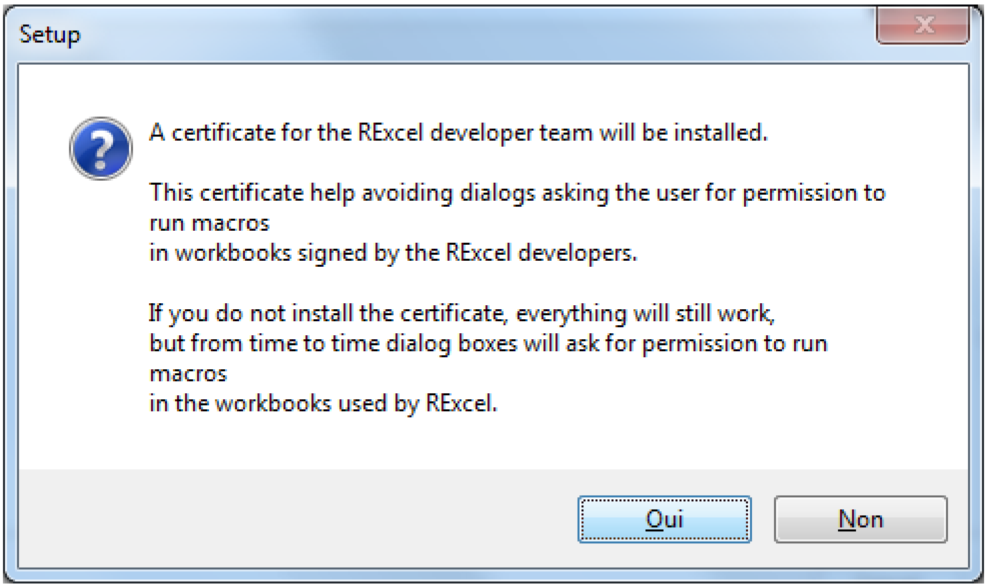
Vous lisez très attentivement ce qui est écrit en rouge et cliquez sur **Next**:



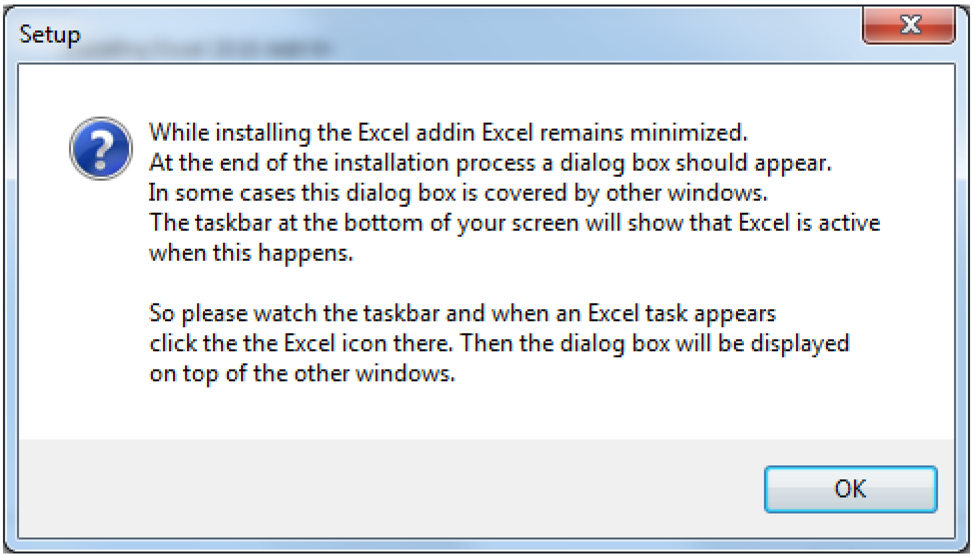
Vous cliquez sur **Next**:



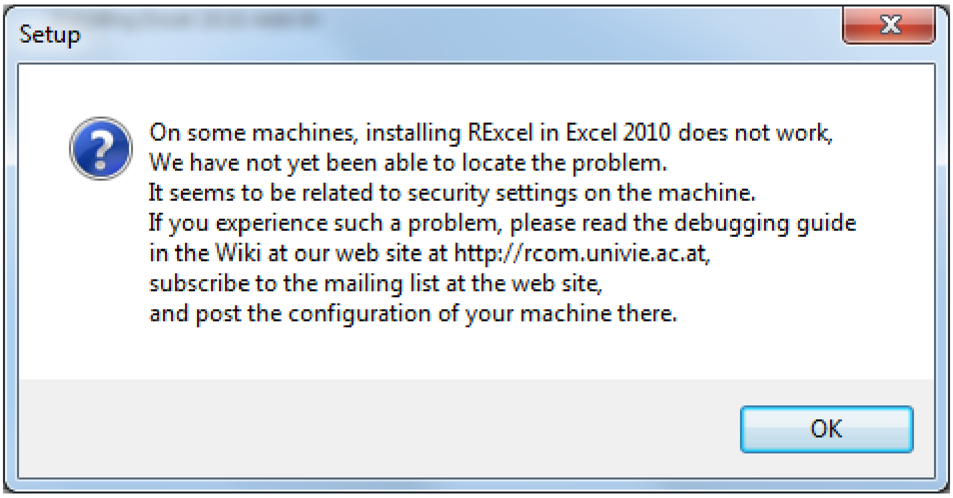
Encore sur **Next** et apparaît le message suivant qu'il faut lire attentivement!:



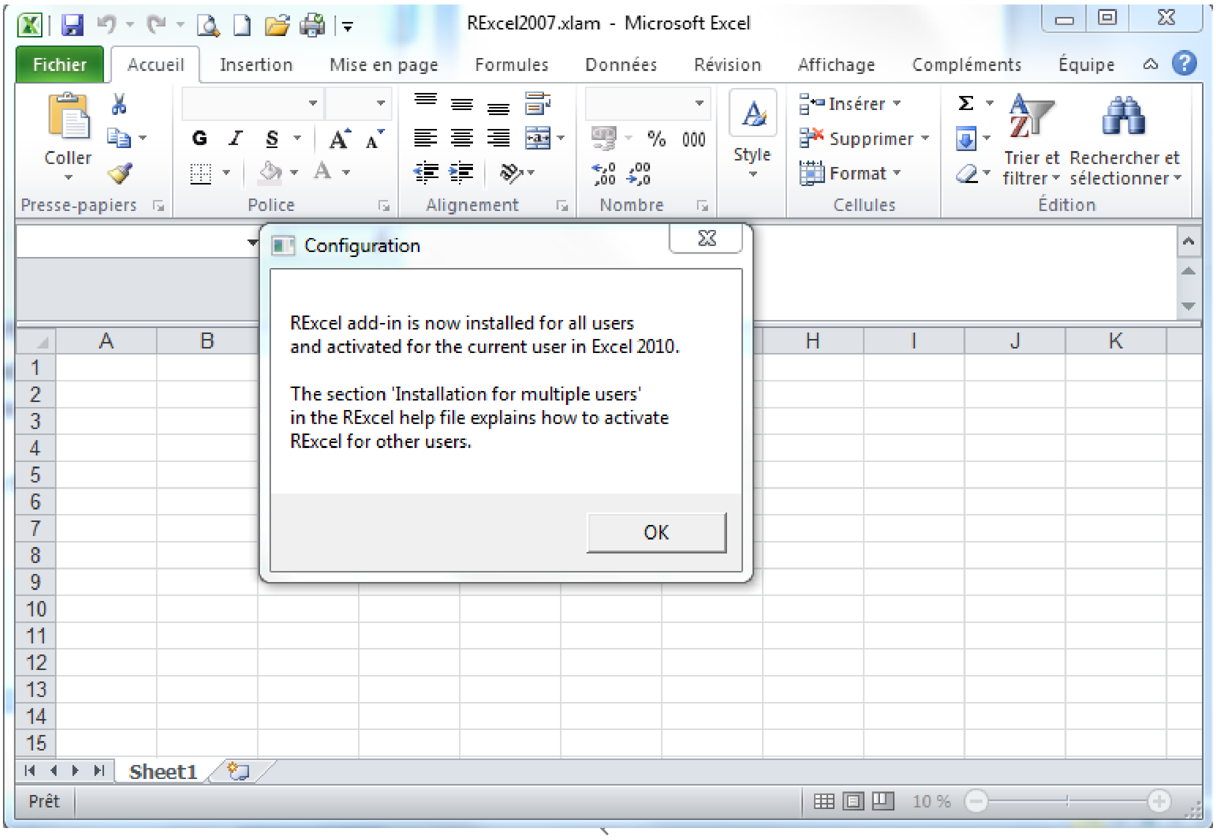
Vous cliquez alors sur **Oui** et apparaît le second message suivant qu'il faut lire attentivement!:



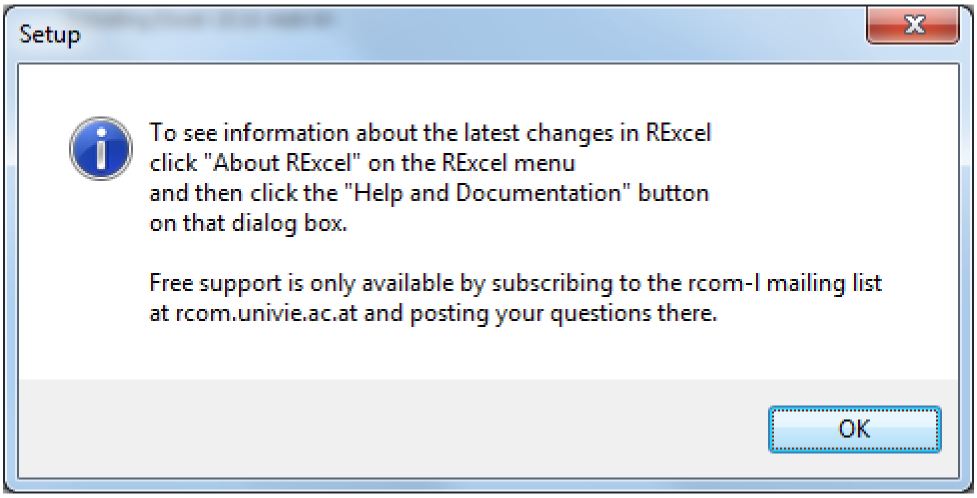
On clique sur OK (pas le choix...) et on lit à nouveau attentivement le message suivant:



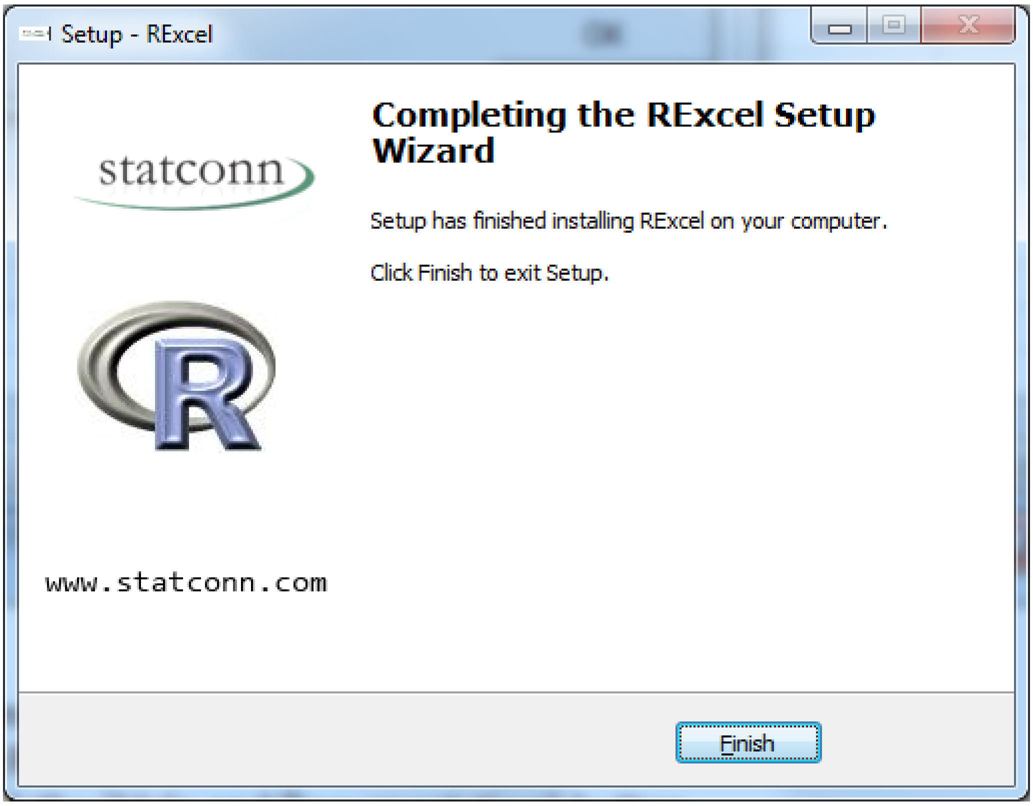
Quand vous cliquerez sur **OK**, après quelques secondes l'icône de Microsoft Excel se mettra à clignoter dans la barre des tâches et il faudra cliquer sur l'icône de Microsoft Excel pour cliquer sur **OK** dans la boîte de dialogue qui apparaît:



Il vient ensuite le message suivant:



Nous cliquons sur **OK**:

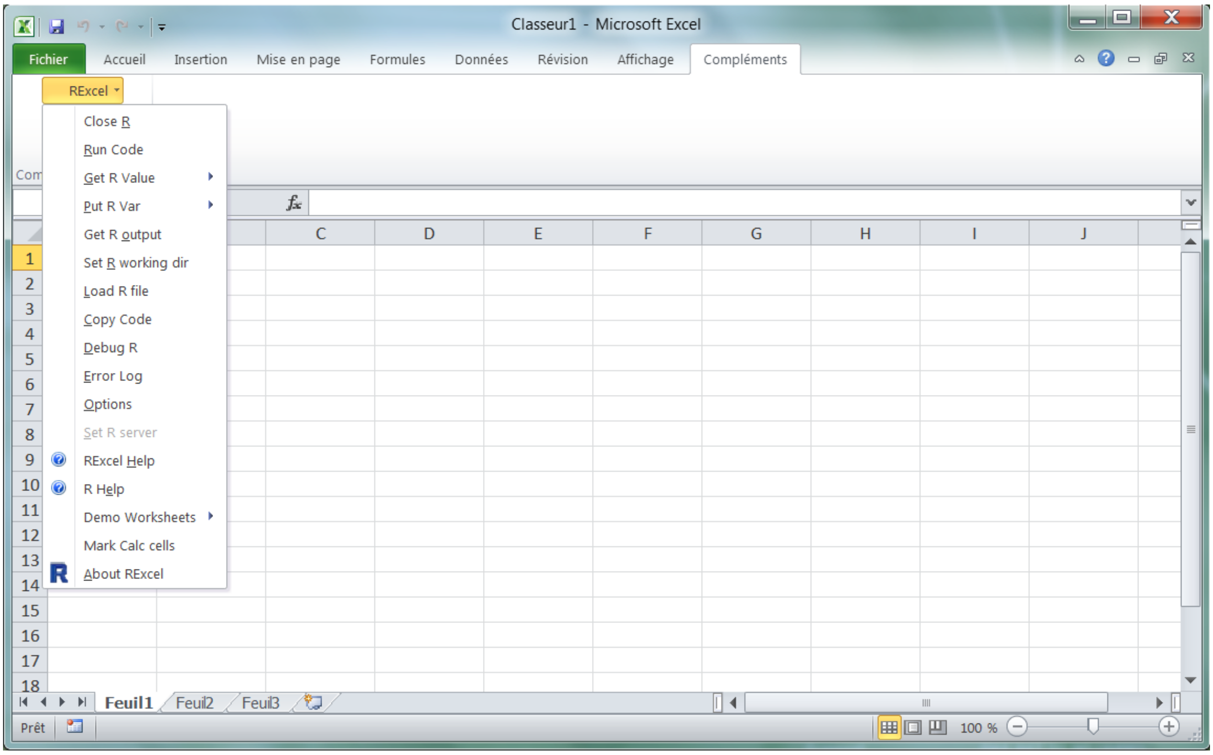


et enfin sur **Finish**.

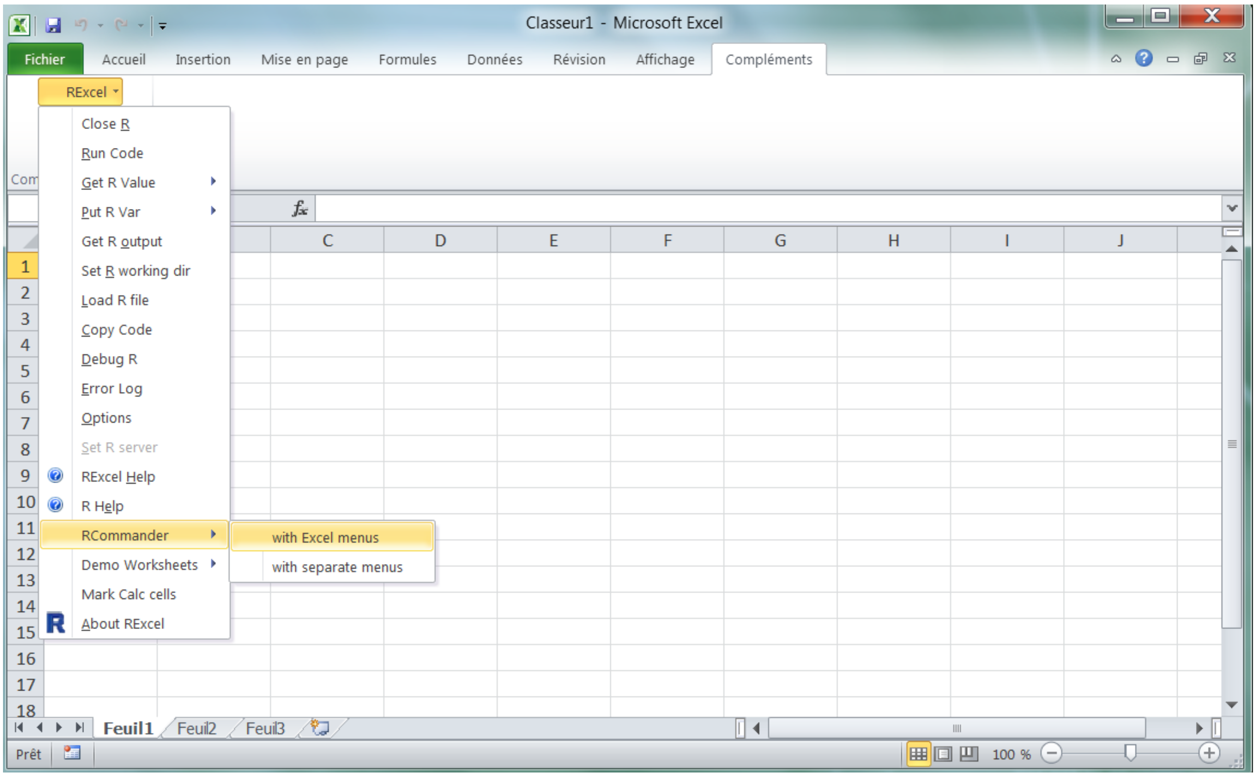
Normalement vous aurez alors sur le bureau l'icône spéciale RExcel... qui aura été rajoutée automatiquement:



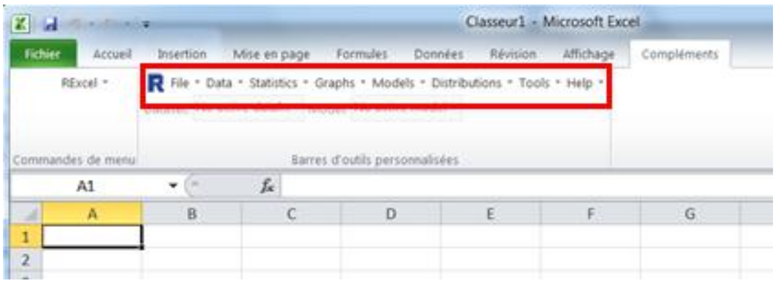
Quand Microsoft Excel s'ouvre, nous avons alors dans le ruban **Complément** le menu **RExcel** qui est visible:



Cependant, une fois le package **Rcmdr** installé dans R (**install.packages("Rcmdr")**), nous avons une option en plus nommée **RCommander/with Excel menus**:

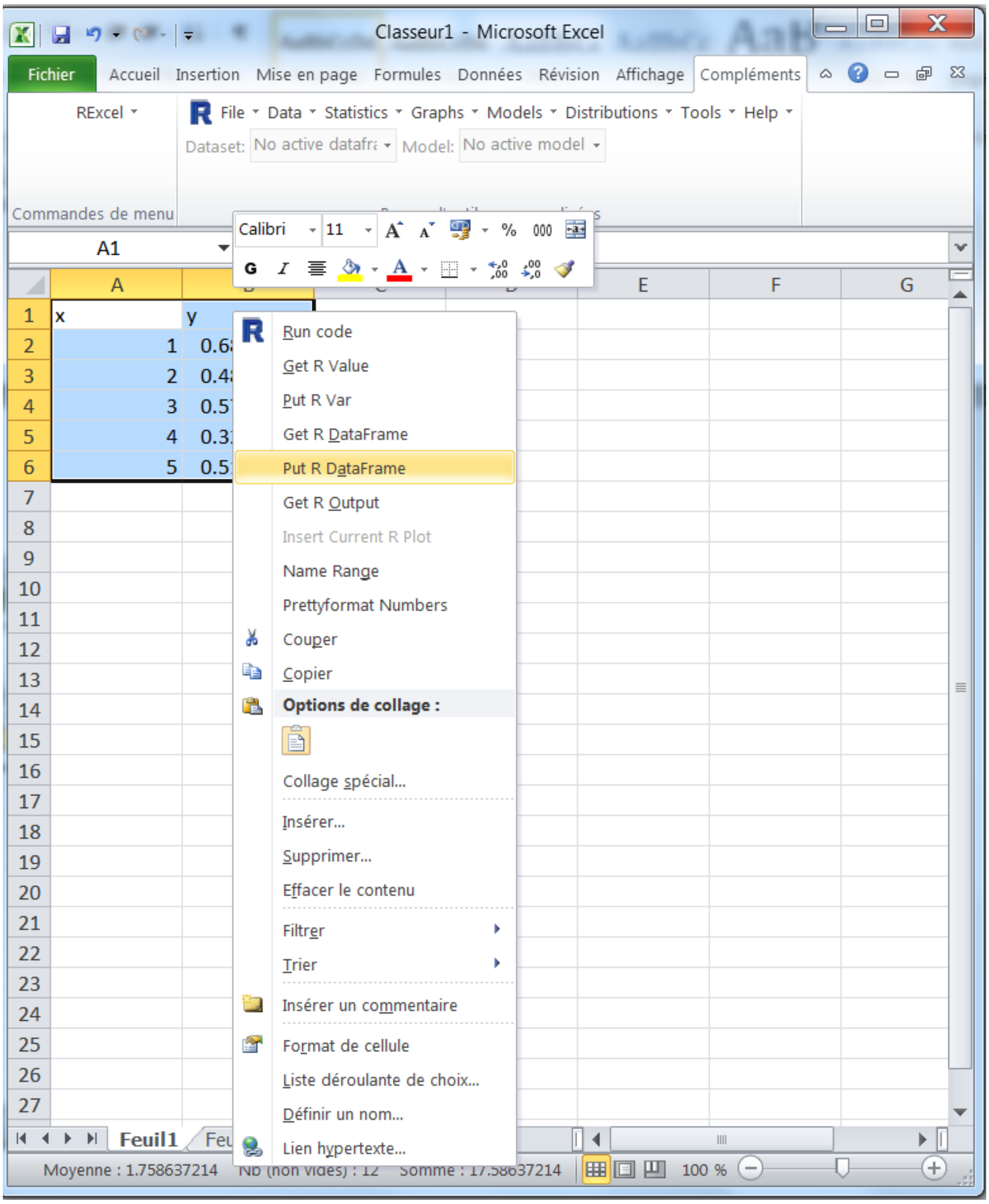


Si nous cliquons dessus apparaît alors une barre d'outils de menus:

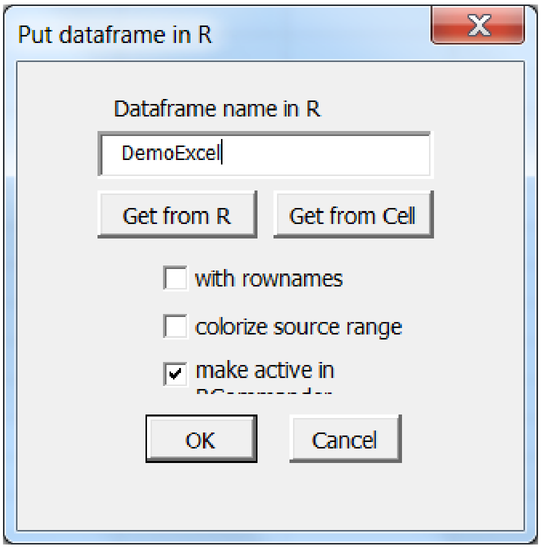


Maintenant voyons les manips les plus importantes à savoir faire.

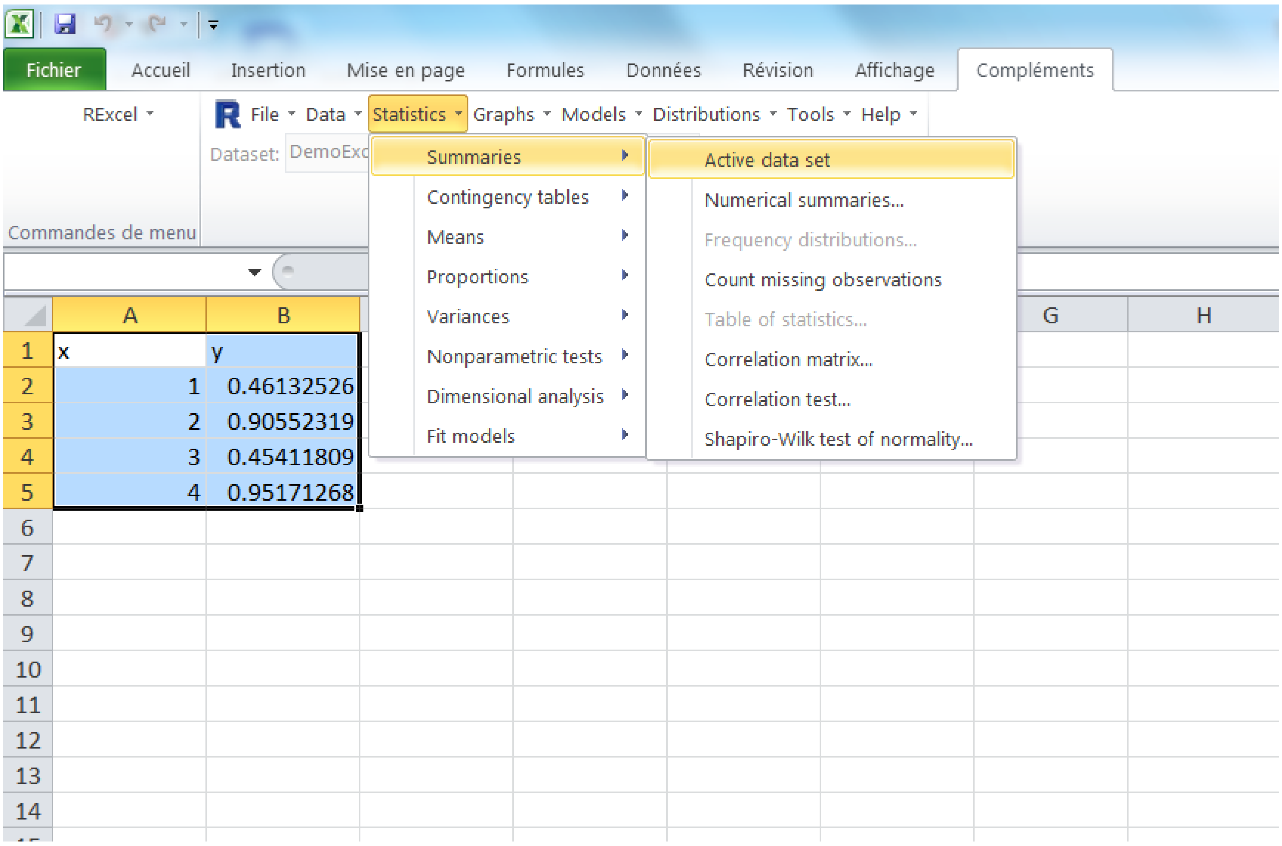
Le minimum minimorum est de savoir convertir une plage de données Microsoft Excel en un data frame R. Pour cela, après avoir saisi des données, on fait un clic droit et on choisit **Put R DataFrame**:



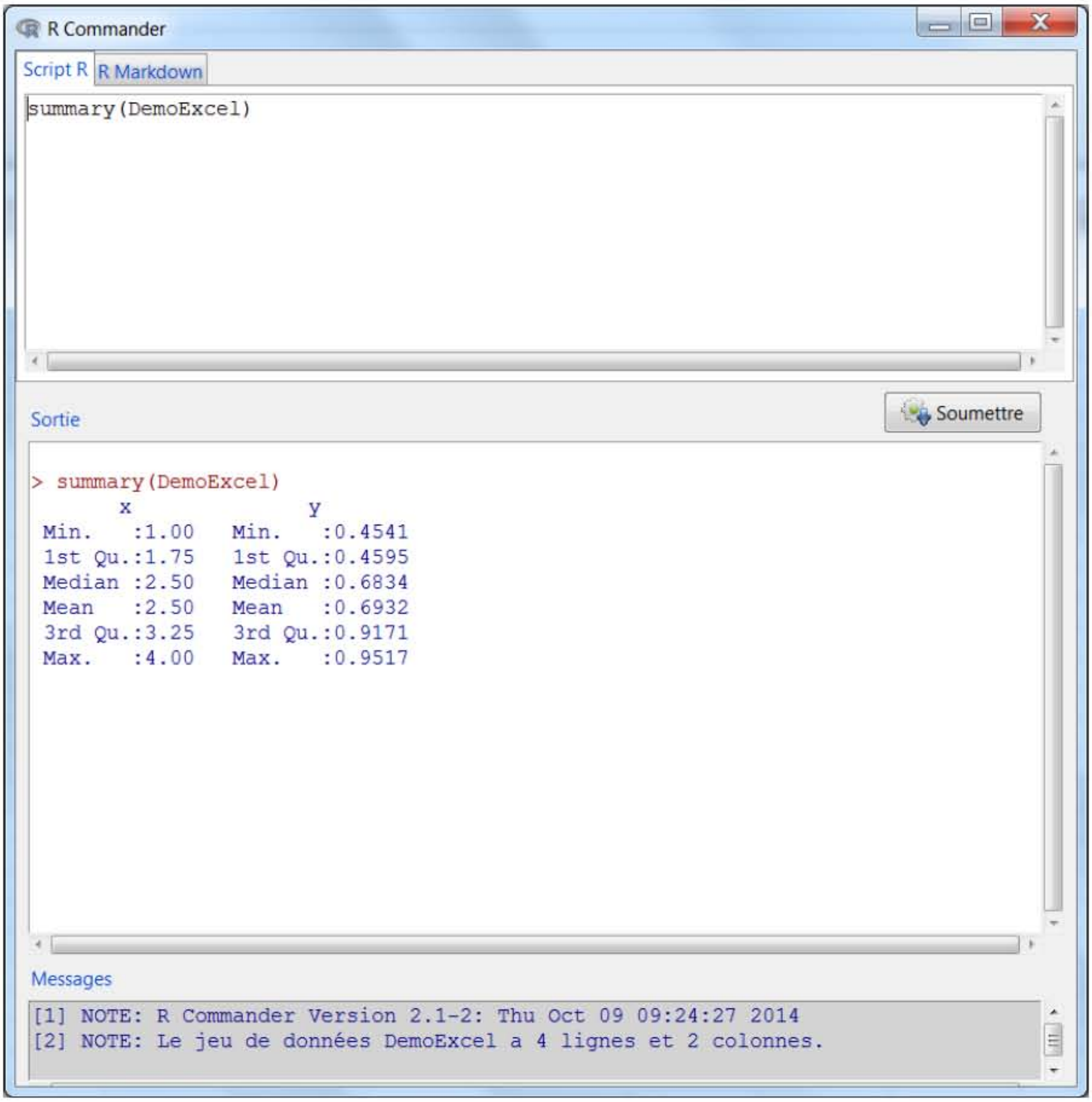
Il vient alors la boîte de dialogue suivante où on va saisir *DemoExcel* comme nom du data frame:



Ensuite, nous pouvons vérifier que ce data frame a bien été créé en faisant une petite analyse élémentaire:

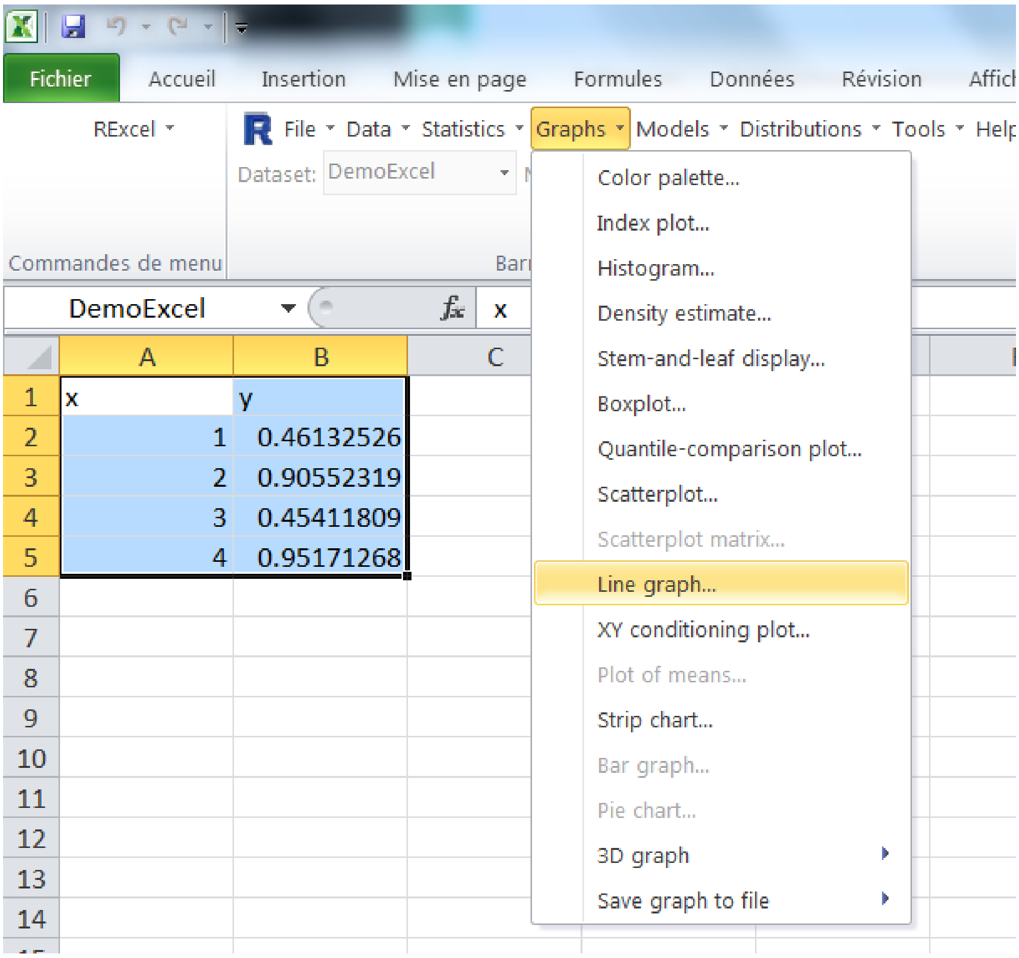


Nous avons alors bien dans la fenêtre R Commander:

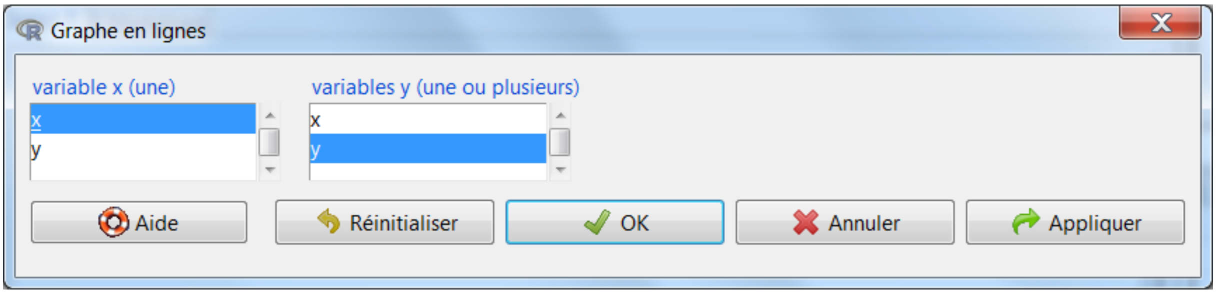


Sinon autre manipulation trivial: Faire un graph.

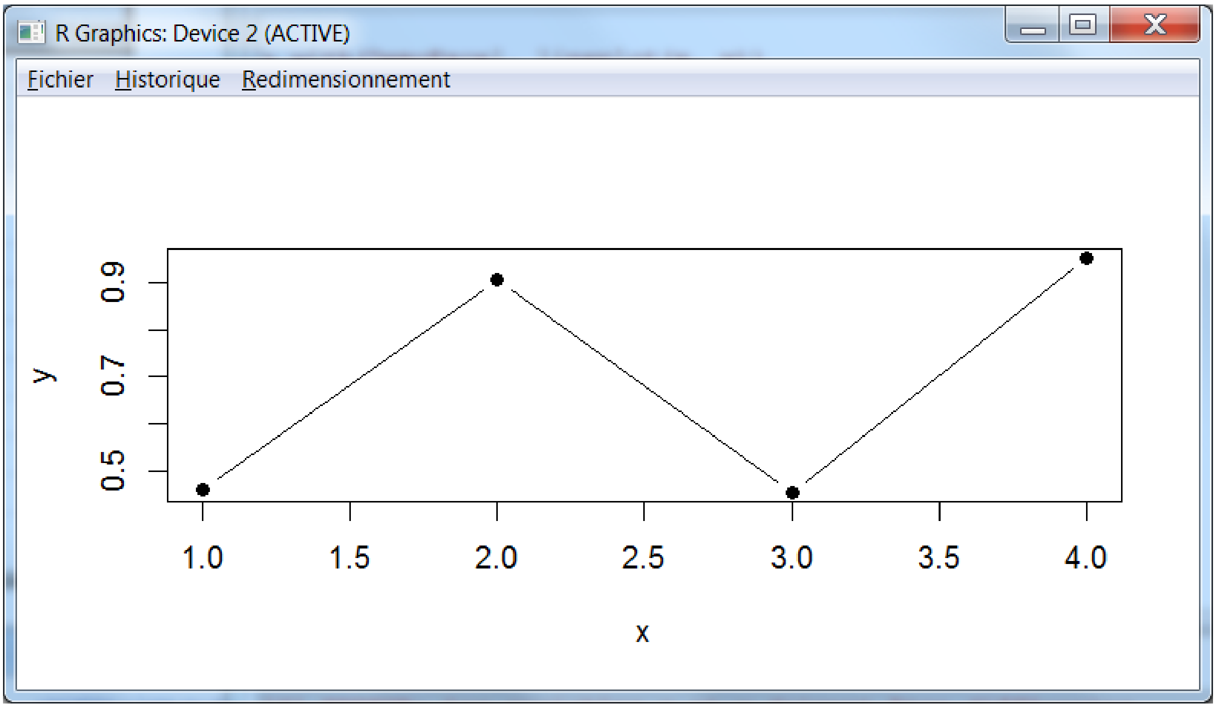
Voyons comment procéder à partir d'un exemple générique:



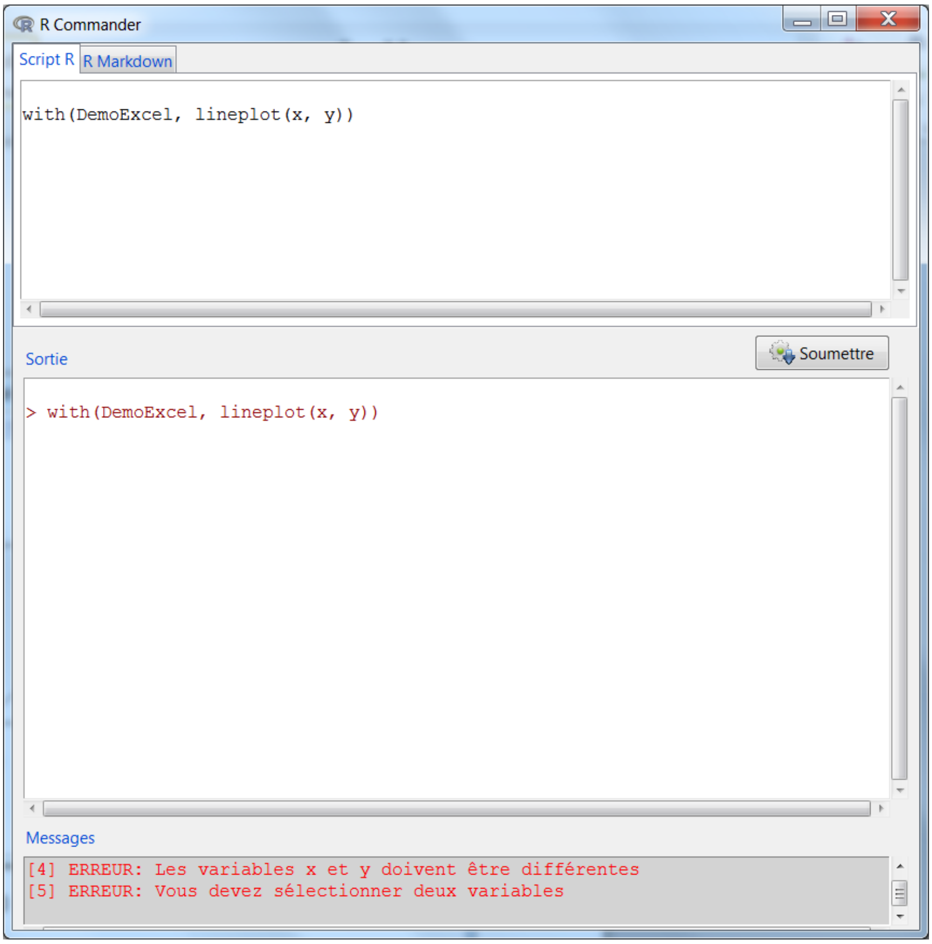
Il vient alors la boîte de dialogue ci-dessous dans laquelle il faut sélectionner en bleu les variables voulues pour que RCmdr sache ce qu'il doit prendre comme variables en entrée et en sortie:



Une fois que nous validons sur **OK**, nous obtenons:

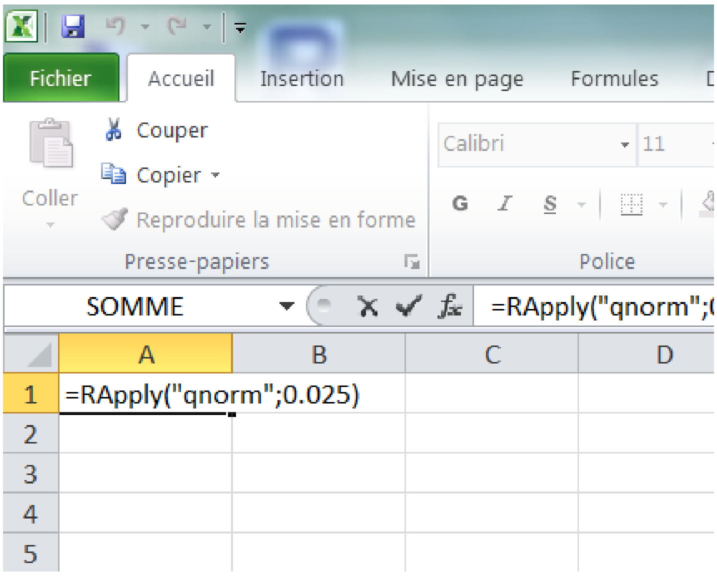


Et dans la fenêtre R Commander il vient respectivement:

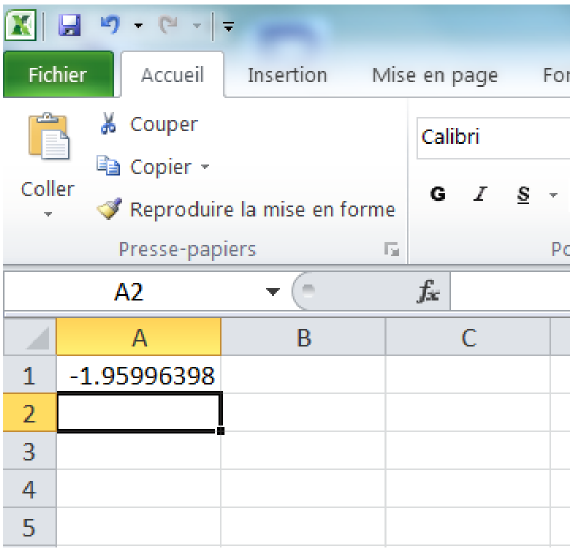


Maintenant, voyons une possibilité beaucoup plus importante! La possibilité d'utiliser les fonctions R directement dans les cellules Microsoft Excel.

Pour cela, il suffit d'utiliser d'appeler la fonction **RApply( )** en passant en argument le nom de la fonction R et ensuite les paramètres correspondants:



et en validant, nous avons alors:



Cela marche donc bien!

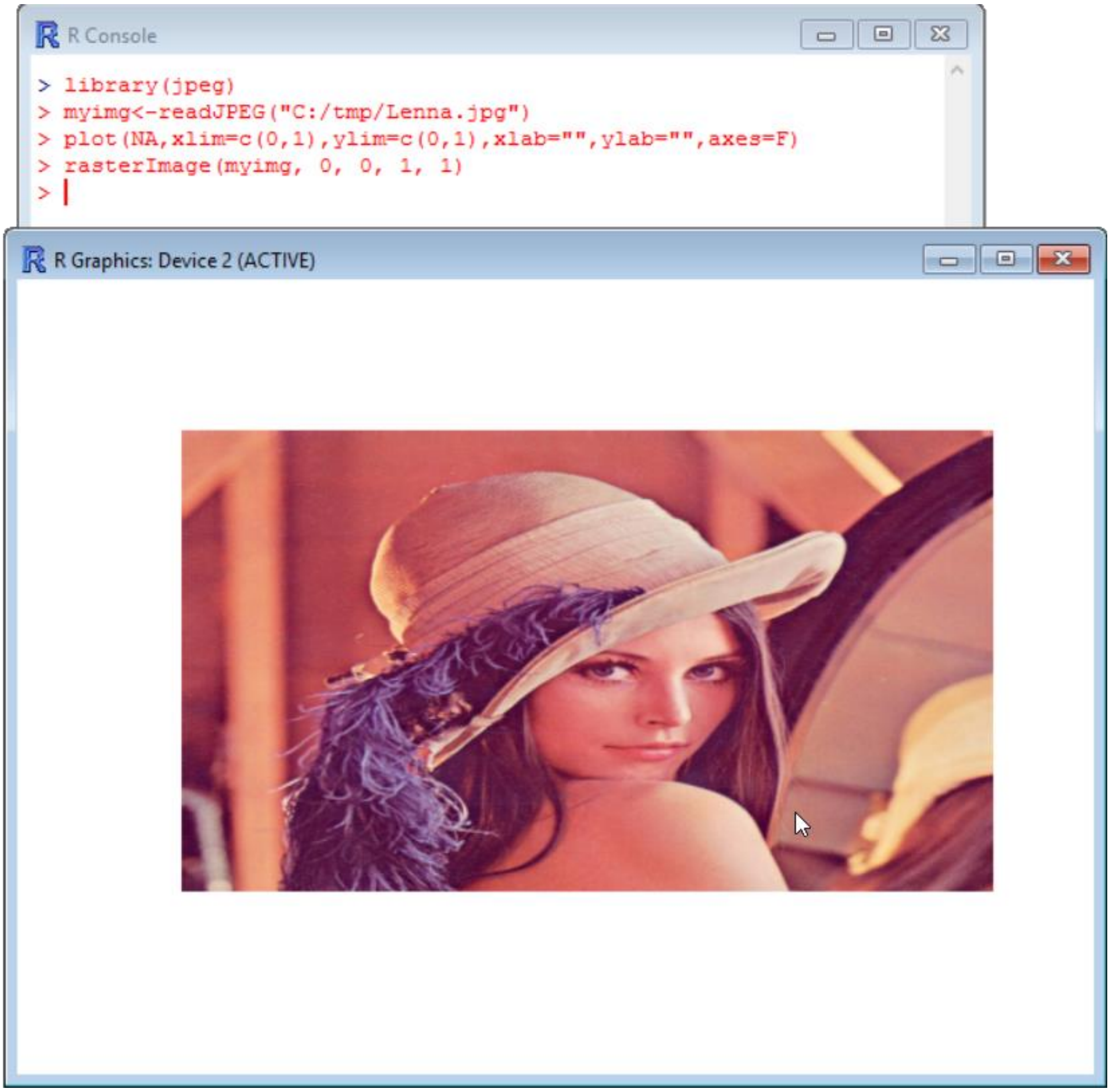
37. Divers

Dans ce chapitre nous avons souhaité mettre des scripts si possibles petits dont l'importance nous semble majeure pour certains besoins métiers et dont il n'existe pas à notre connaissance de packages.

Traitement d'images jpeg

Bon évidemment R ce n'est pas MATLAB au niveau des images mais évidemment il y a un certain potentiel… donc voyons ce que nous pouvons en tirer à ce jour.

D'abord lire et afficher une image à l'aide du package **jpeg** et sa fonction **readJPEG( )**:



Nous pouvons bien évidemment extraire une des trois couches RGB. Par exemple ici le vert (2ème couche):



Après on peut attaquer donc les contenus de chacun des matrices de chacune des couches de couleur à loisir…

Voyons maintenant ce que met à disposition le package **EBImage** de **Bioclite**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

> library("BiocManager")
> BiocManager::install("EBImage")
Bioconductor version 3.9 (BiocManager 1.30.10), R 3.6.3 (2020-02-29)
Installing package(s) 'EBImage'
Warning: package 'EBImage' is in use and will not be installed
Installation path not writeable, unable to update packages: boot, class,
  KernSmooth, lattice, MASS, nlme, nnet, spatial, survival
Old packages: 'arm', 'gridSVG', 'labelled', 'RcppArmadillo', 'RCurl',
  'rootSolve', 'VGAM', 'xml2'
Update all/some/none? [a/s/n]: n
> |
```

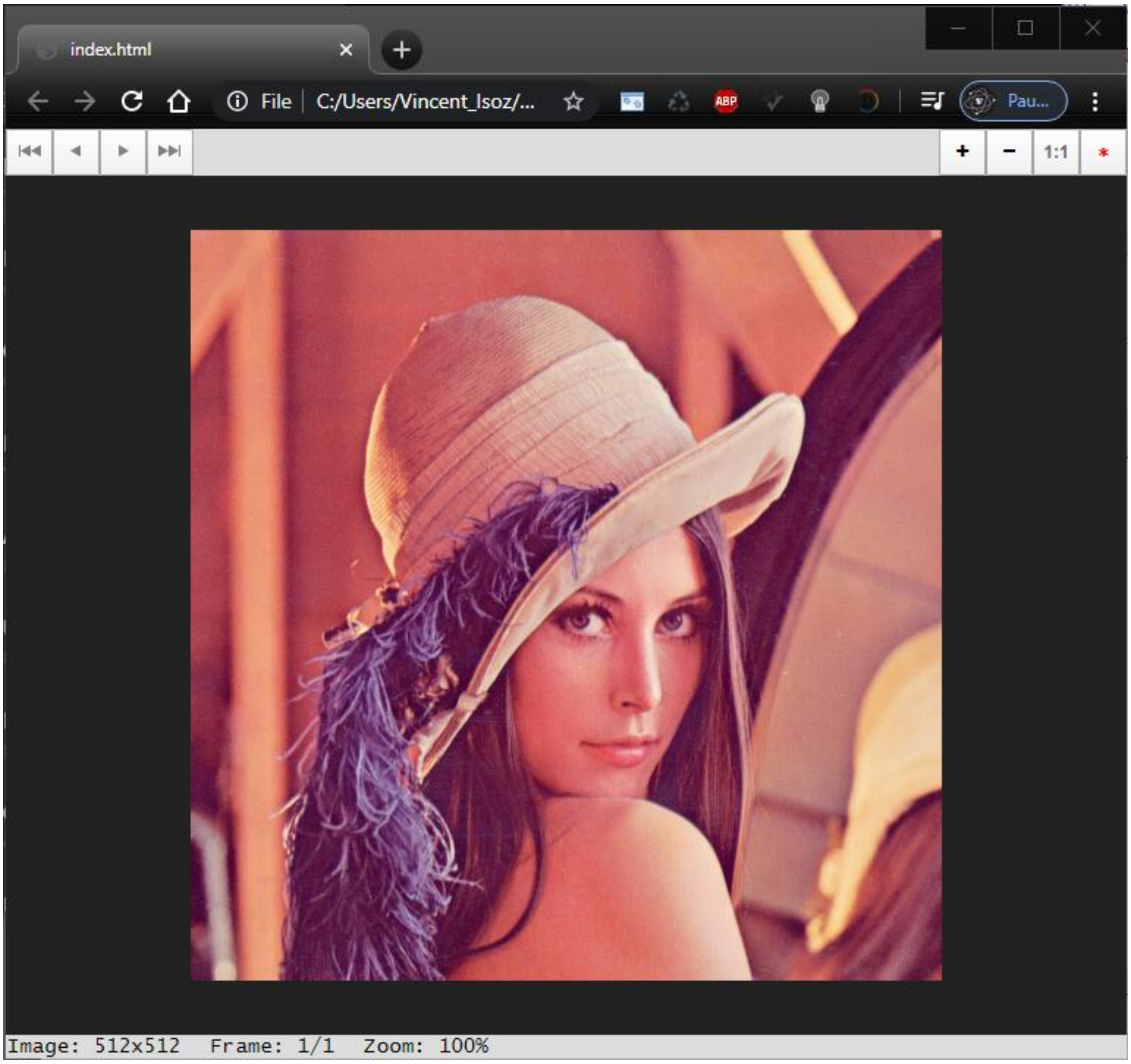
Une fois installé, chargeons un image avec la fonction **readImage( )** et affichons là à l'écran avec la fonction **display( )**:

```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

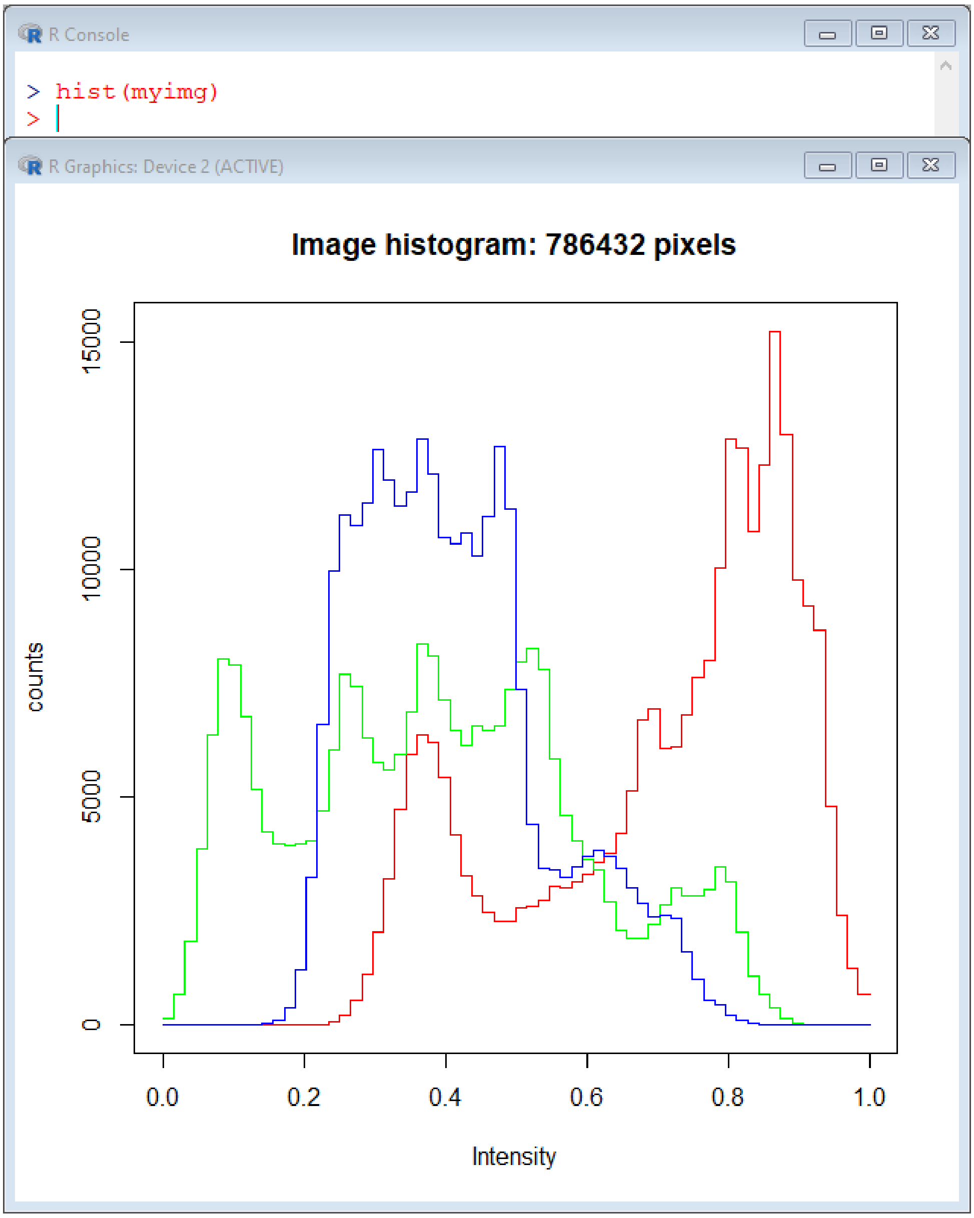
> library("EBImage")
> myimg<-readImage("c:/tmp/Lenna.jpg")
> print(myimg)
Image
  colorMode     : Color
  storage.mode : double
  dim          : 512 512 3
  frames.total : 3
  frames.render: 1

imageData(object)[1:5,1:6,1]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.8823529 0.8705882 0.8705882 0.8784314 0.8862745 0.8901961
[2,] 0.8784314 0.8784314 0.8784314 0.8823529 0.8901961 0.8901961
[3,] 0.8745098 0.8784314 0.8823529 0.8823529 0.8823529 0.8823529
[4,] 0.8745098 0.8784314 0.8784314 0.8784314 0.8705882 0.8666667
[5,] 0.8901961 0.8862745 0.8862745 0.8823529 0.8823529 0.8823529
> display(myimg)
> |
```

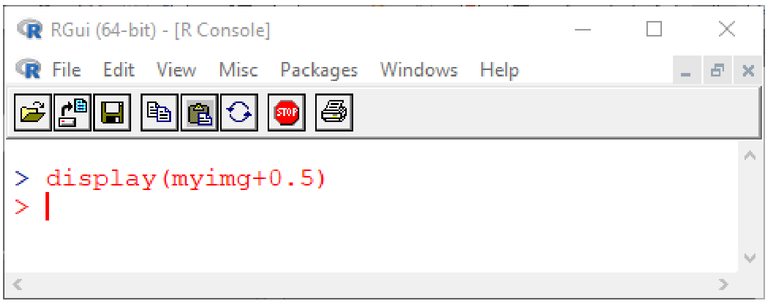
Ce qui ouvre le navigateur:



On peut afficher l'histogramme du spectre RVB:



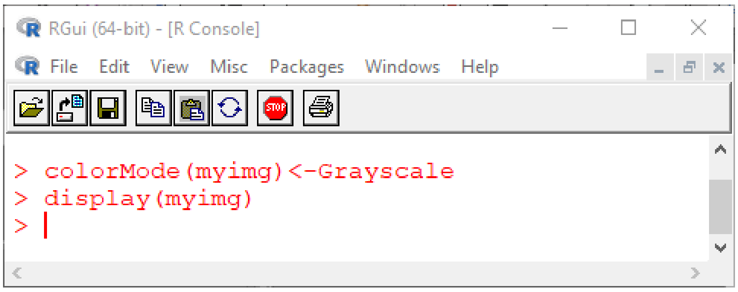
et attaquer (bien évidemment!) les composantes des trois matrices RVB soit globalement soit individuellement comme l'illustre l'exemple simpliste suivant:



Ce qui donne à nouveau dans le navigateur:



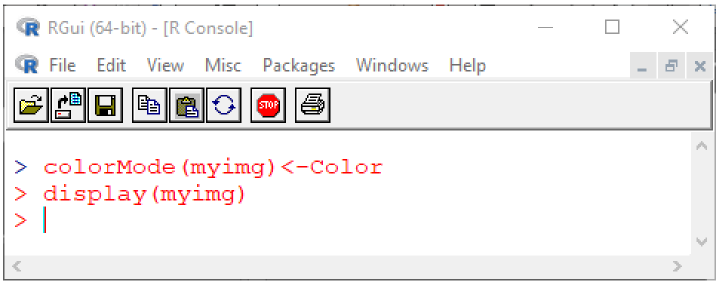
On peut additionner des images avec ce package (sous la condition qu'elles ont les mêmes dimensions bien évidemment!).



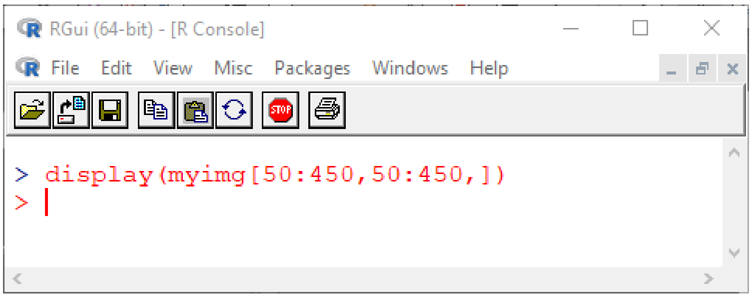
Ce qui donne dans le navigateur:



Pour revenir à la couleur, nous écrirons:



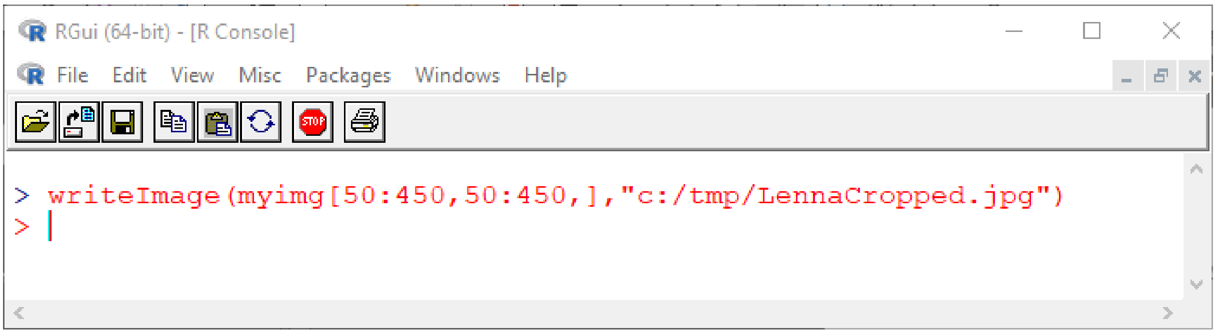
On peut aussi bien évidemment rogner l'image:



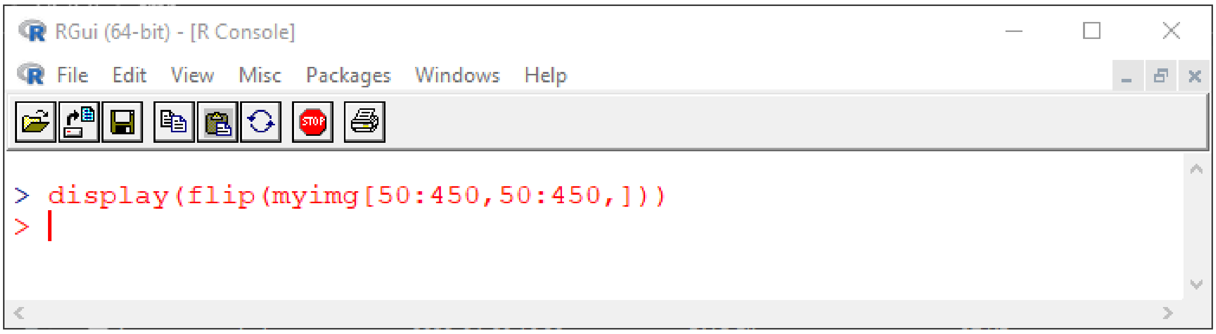
Ce qui donne dans le navigateur:



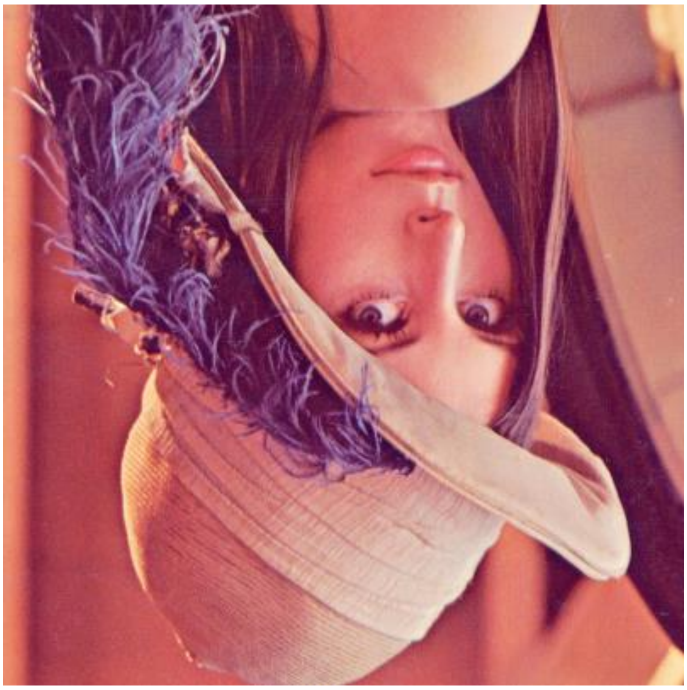
On peut sauvegarder le résultat avec la fonction **writeImage( )**:



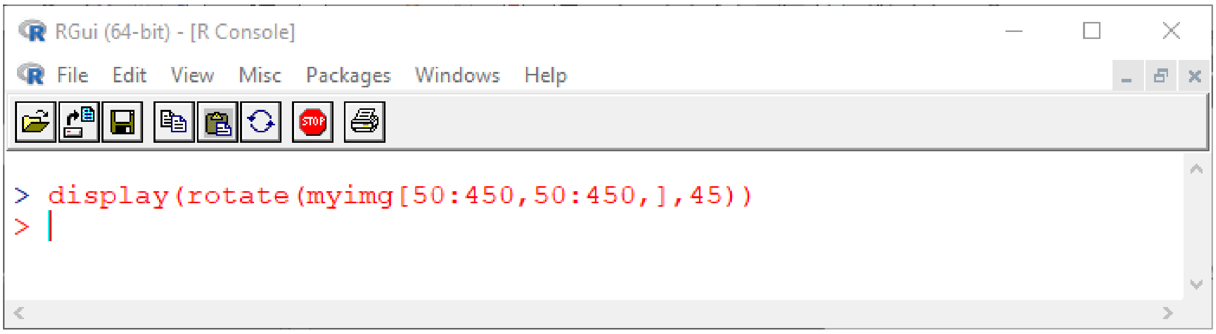
On peut aussi tourner une image ce qui donnera:



Ce qui donnera dans le navigateur:



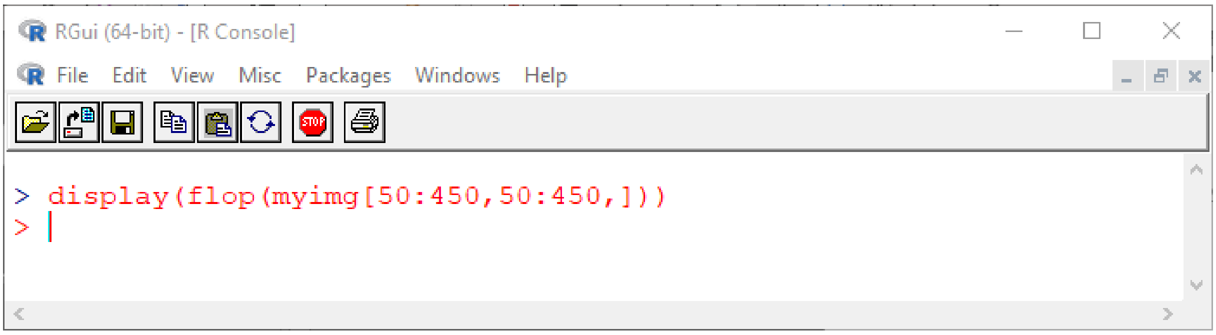
ou faire une rotation:



Ce qui donnera dans le navigateur:



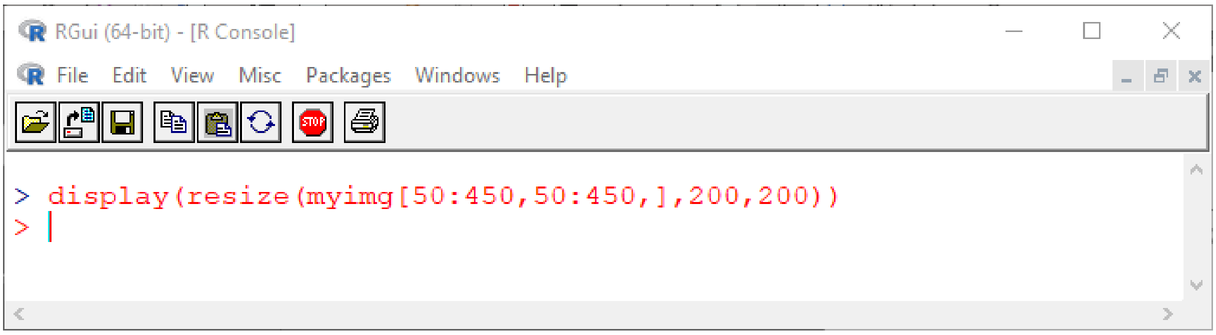
Ou faire un mirroir (symétrie verticale):



Ce qui donne dans le navigateur:



On peut aussi redimensionner:



Ce qui donnera dans le navigateur:



Il y aussi des filtres et bien d'autres fonctionnalités que le lecteur pourra trouver dans la documentation du package correspondant!

Reconnaissance de texte sur image (OCR)

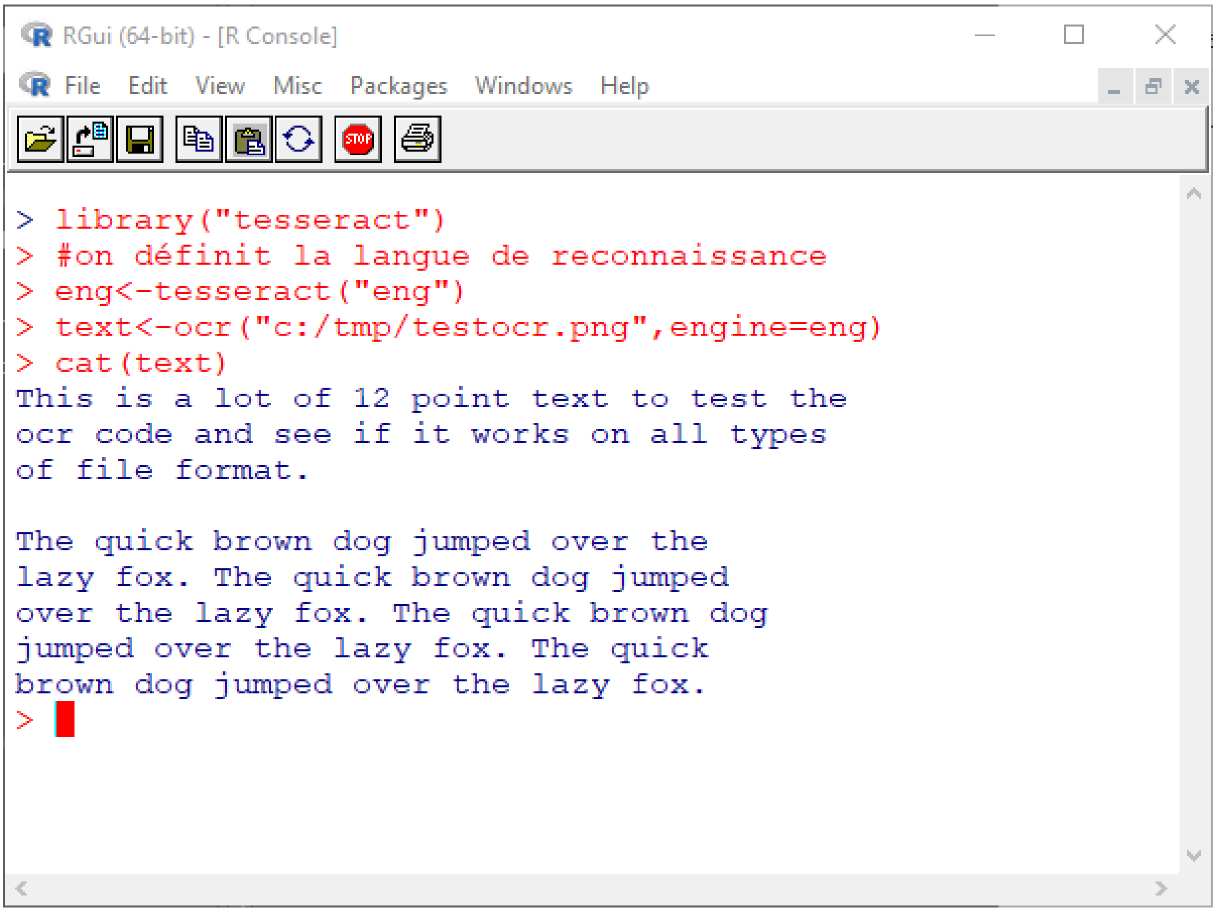
Ce qui suit n'est qu'une petite version adaptée de la vignette disponible ici:

https://cran.r-project.org/web/packages/tesseract/vignettes/intro.html

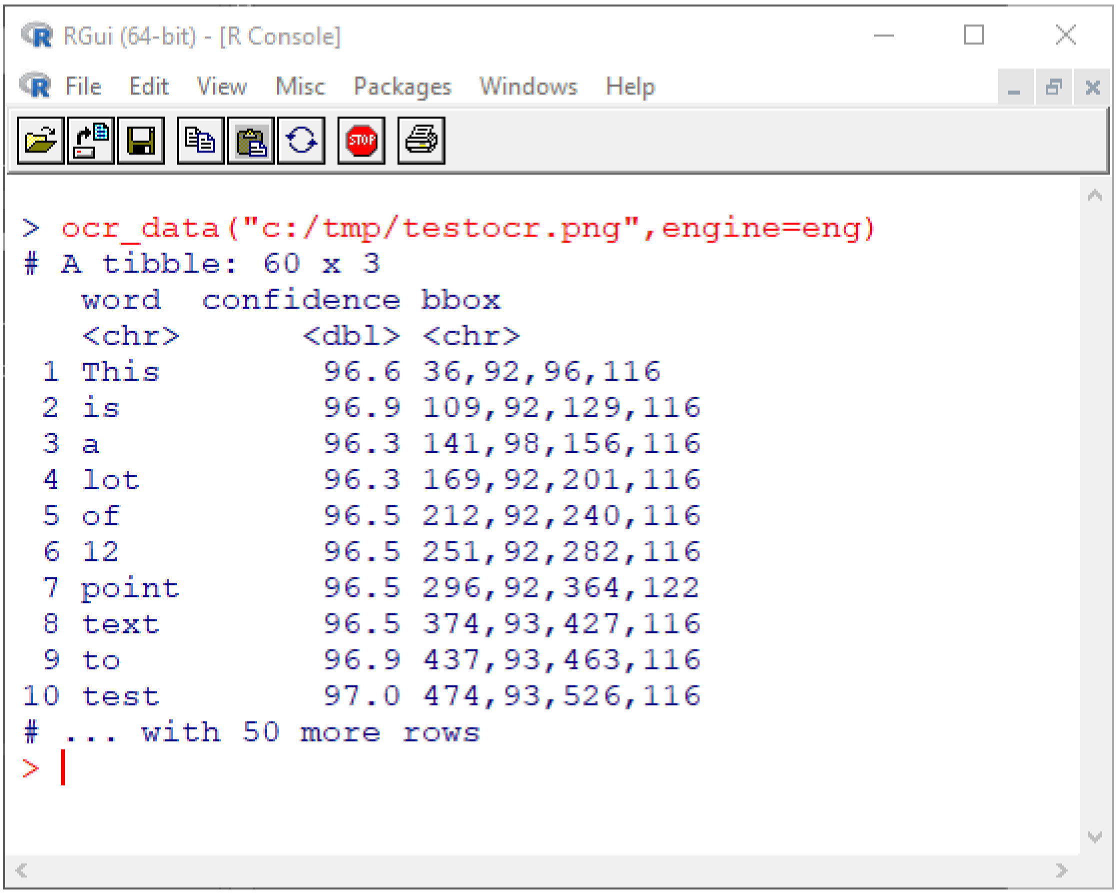
Considérons l'image suivante:

This is a lot of 12 point text to test the
ocr code and see if it works on all types
of file format.
The quick brown dog jumped over the
lazy fox. The quick brown dog jumped
over the lazy fox. The quick brown dog
jumped over the lazy fox. The quick
brown dog jumped over the lazy fox.

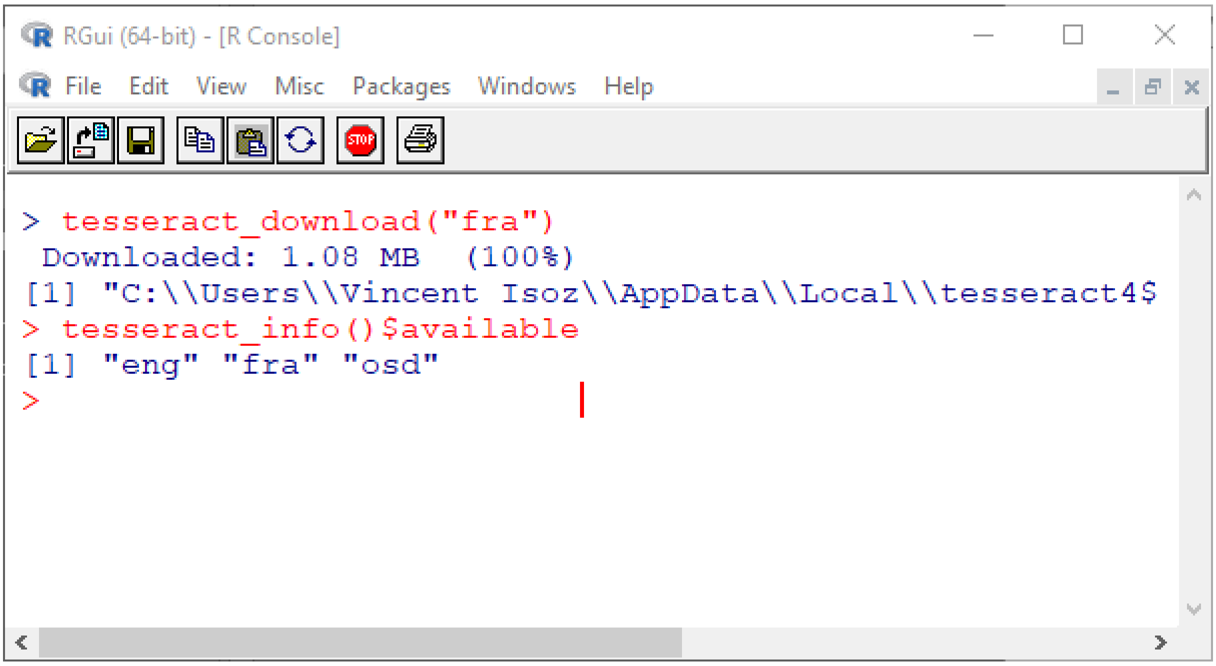
Nous souhaitons en extraire le texte. Pour cela nous utilisons la fonction **ocr( )** du package **tesseract**:



La fonction **ocr_data( )** renvoie tous les mots de l'image avec un cadre de sélection et un taux de confiance:



On peut télécharger d'autres langues avec la commande **tesseract_download( )** et vérifier qu'elle soit bien installée à l'aide de la fonction **tesseract_info( )**:



Géolocaliser une adresse IP

Nous avons déjà vu comment utiliser des coordonnées de latitude et de longitude pour les représenter sur une carte géographique. Mais… dans le cadre du marketing de plus en plus de community manager doivent analyser des adresses IP et les obtenir leurs coordonnées.

Voici un script pour cela:

```
freegeoip <- function(ip, format =
ifelse(length(ip)==1,'list','dataframe'))
{
    if (1 == length(ip))
    {
        # a single IP address
        require(rjson)
        url <- paste(c("http://freegeoip.net/json/", ip), collapse='')
        ret <- fromJSON(readLines(url, warn=FALSE))
        if (format == 'dataframe')
            ret <- data.frame(t(unlist(ret)))
        return(ret)
    } else {
        ret <- data.frame()
        for (i in 1:length(ip))
        {
            r <- freegeoip(ip[i], format="dataframe")
            ret <- rbind(ret, r)
        }
        return(ret)
    }
}
```


Il suffit alors d'exécuter la commande suivante dans R:

>freegeoip('169.254.38.80')

Ce qui permet d'obtenir plusieurs informations comme la latitude, longitude, le pays et autres…

Logique Floue

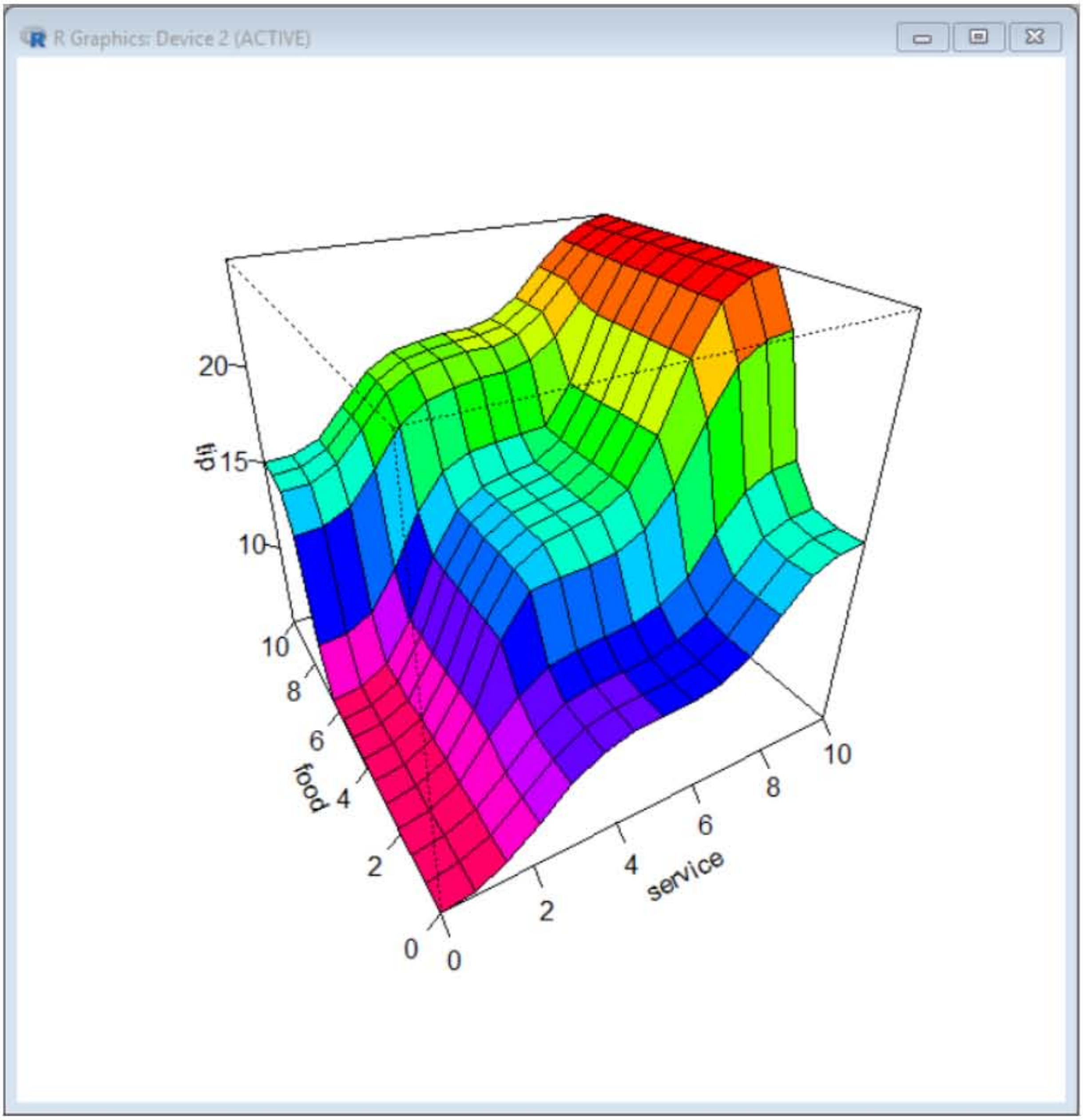
Comme nous l'avons mentionné dans le cours théorique, la logique floue est significativement utilisée en ingénierie (robotique, mécanique) mais aussi en Data Mining/Machine Learning et psychologie et autres.

Nous avons aussi mentionné que l'exemple classique est celui du "problème du pourboire" que nous avons détaillé dans le cours théorique. Le but ici est de voir comment le reproduire dans R!

Nous allons donc utiliser le package **FuzzyR** avec les fonctions correspondantes **addvar( )** , **addmf( )** etc.:

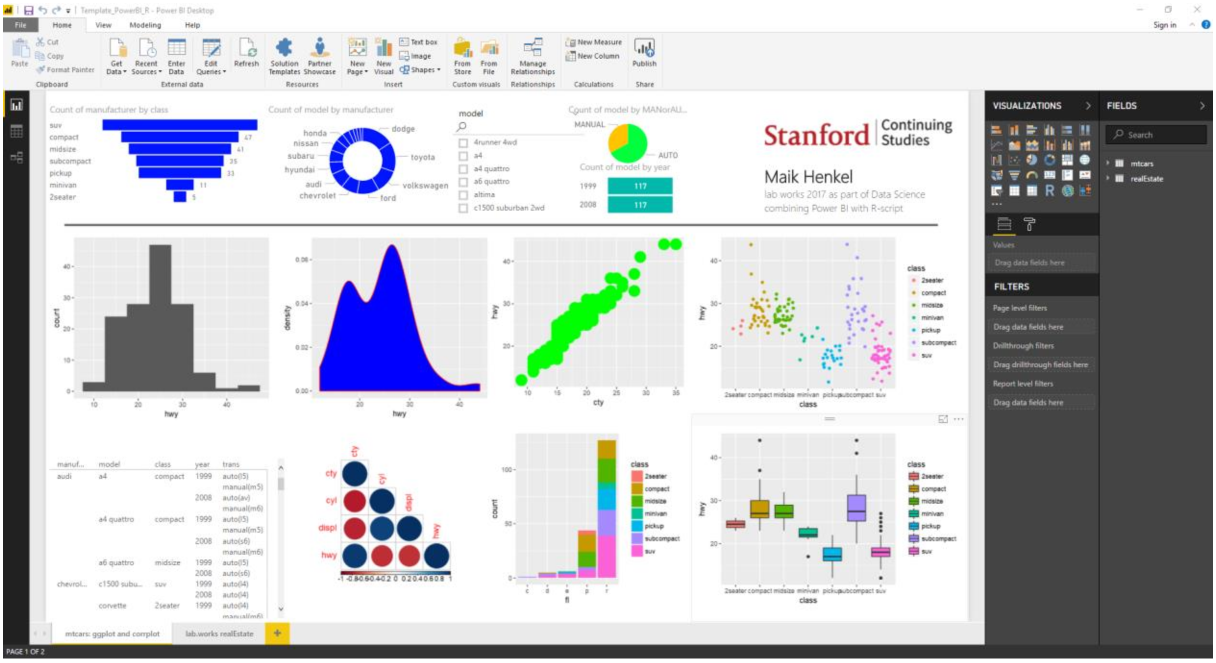
```
> library(FuzzyR)
> fis= newfis('tipper')
>
> fis= addvar(fis, 'input', 'service', c(0,10))
> fis= addvar(fis, 'input', 'food', c(0,10))
> fis= addvar(fis, 'output', 'tip', c(0,30))
> fis= addmf(fis, 'input', 1, 'poor', 'gaussmf', c(1.5,0))
> fis= addmf(fis, 'input', 1, 'good', 'gaussmf', c(1.5,5))
> fis= addmf(fis, 'input', 1, 'excellent', 'gaussmf', c(1.5,10))
> fis= addmf(fis, 'input', 2, 'rancid', 'trapmf', c(0,0,1,3))
> fis= addmf(fis, 'input', 2, 'delicious', 'trapmf', c(7,9,10,10))
> fis= addmf(fis, 'output', 1, 'cheap', 'trimf', c(0,5,10))
> fis= addmf(fis, 'output', 1, 'average', 'trimf', c(10,15,20))
> fis= addmf(fis, 'output', 1, 'generous', 'trimf', c(20,25,30))
>
> rules= rbind(c(1,1,1,1,2), c(2,0,2,1,1), c(3,2,3,1,2))
>
> fis= addrule(fis, rules)
>
> showrule(fis)
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)
> evalfis(c(1,2), fis)
         [,1]
[1,] 5.558586
> evalfis(rbind(c(3,5), c(2,7)), fis)
          [,1]
[1,] 12.218379
[2,]  7.788532
> gensurf(fis)
>
```

et la non moins fameux graphique correspondant:


R Graphics: Device 2 (ACTIVE)
tip
20
15
10
food
0
2
4
6
8
10
service
0
2
4
6
8
10

Microsoft Power BI Desktop

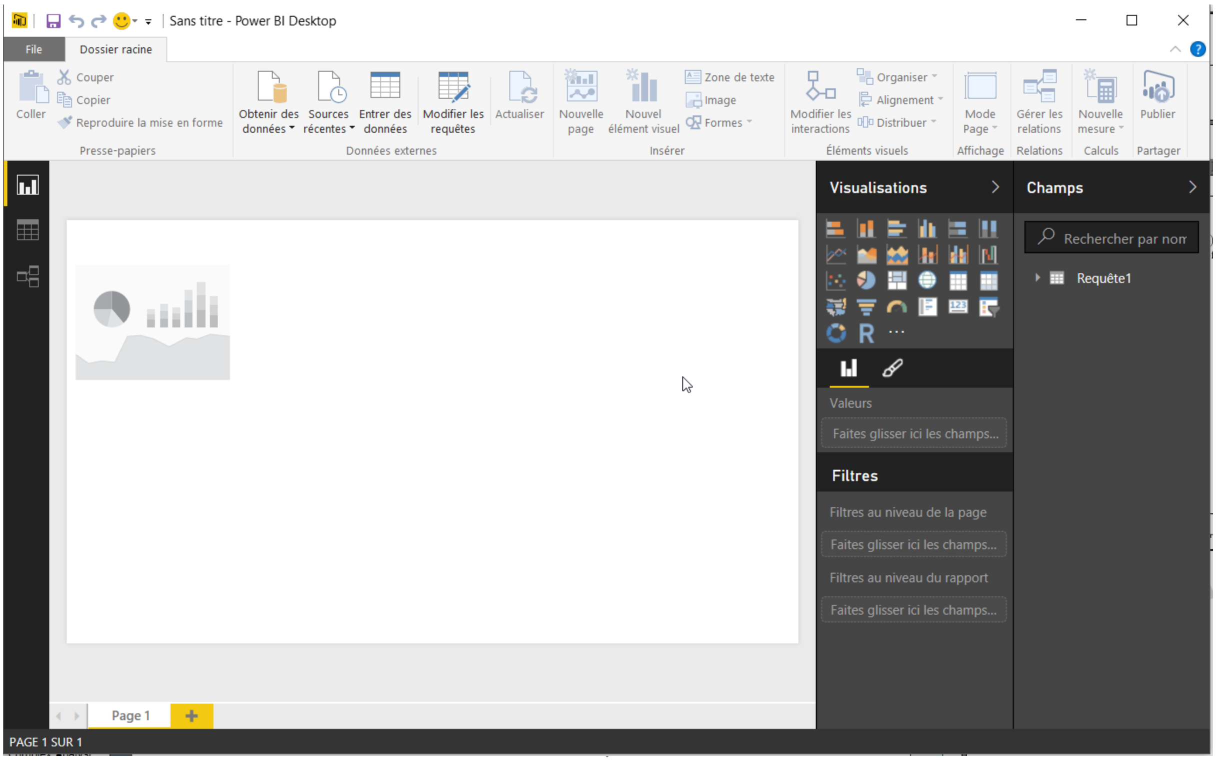
Les fans de PowerPivot, Power View, Power Map et Power Query dans Microsoft Excel connaissent bien Power BI. Il s'agit de l'outil bureautique client de Microsoft pour le développement de tableaux de bords publiables ensuite si besoin sur une plateforme Internet Power BI (cloud) ou sur un serveur On-Premise hebergé dans les locaux de votre entreprise. Depuis la version de Décembre 2015 il est possible de connecter Power BI à des scripts R comme nous allons le voir (au jour où nous écrivons ces lignes cette fonctionnalité est en beta).



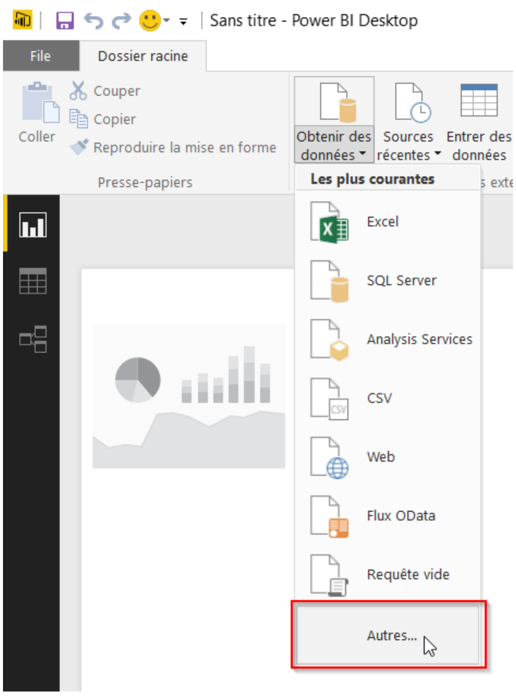
Attention!!! Suivant la situation il vaut mieux inclure les différencts scripts R de Power BI dans des fichiers externes *.R et charger ces derniers avec la fonction **source( )** . Cette pratique est cependant à éviter si vous utilisez Power BI Service.

Exemple 1.: Importer (et traiter) des données avec R
MRO 3.4.2

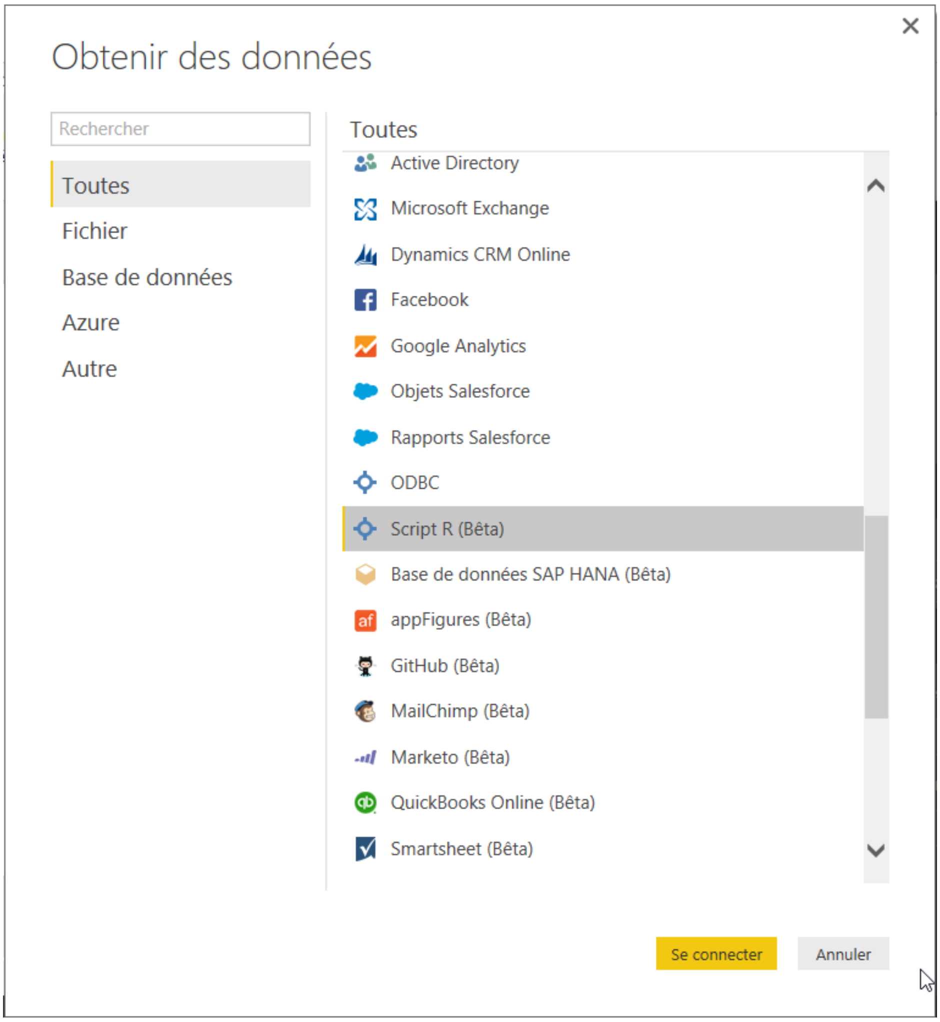
Maintenant, nous ouvrons Power BI:



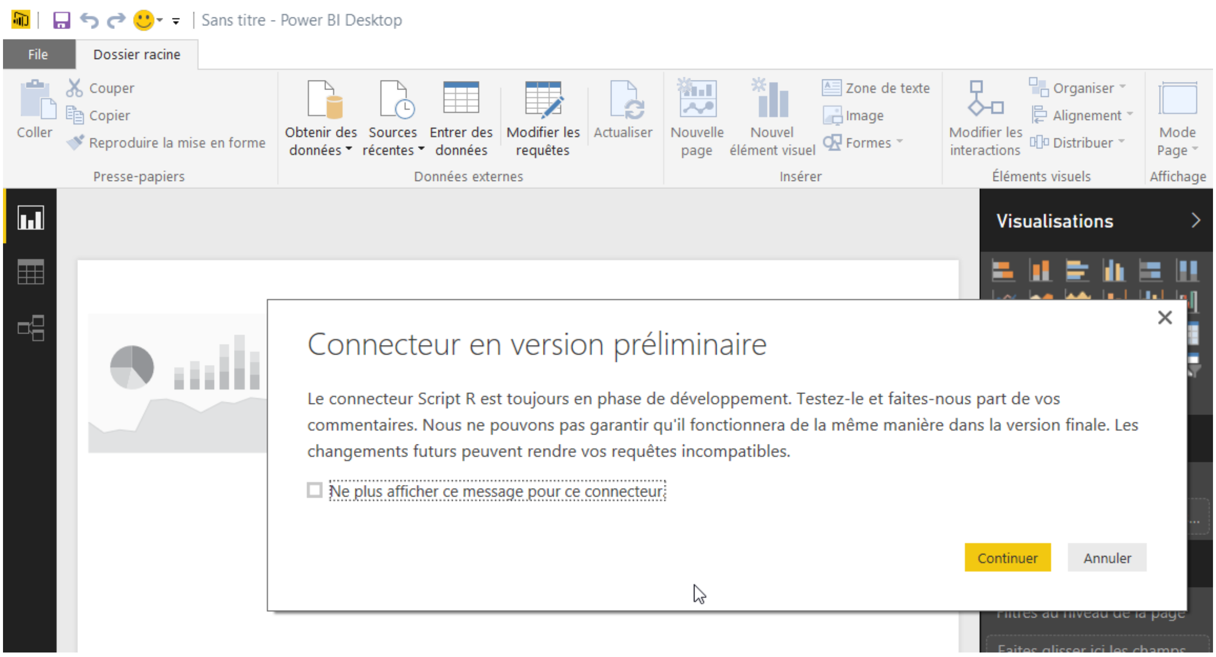
Nous cliquons sur le bouton **Obtenir des données** et **Autres…**:



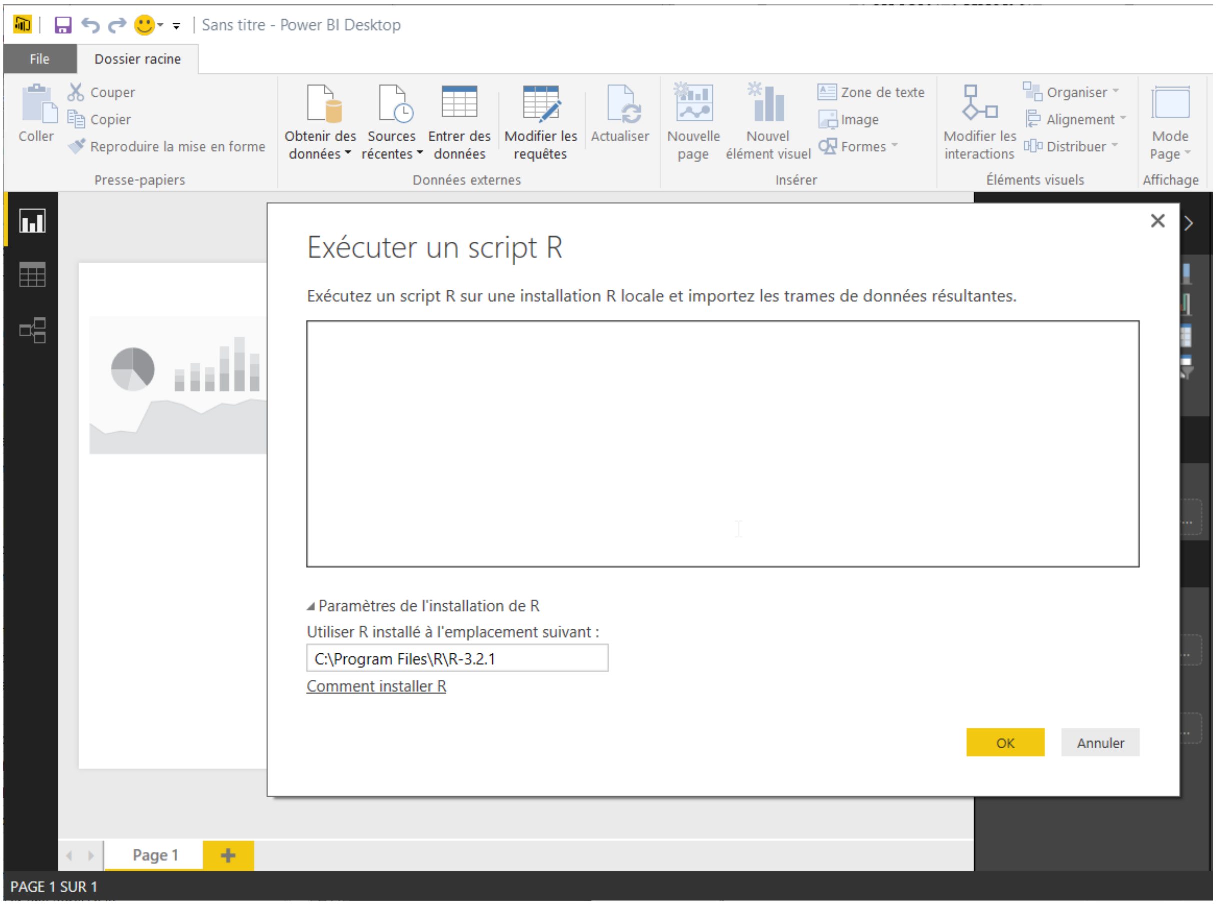
Ensuite:



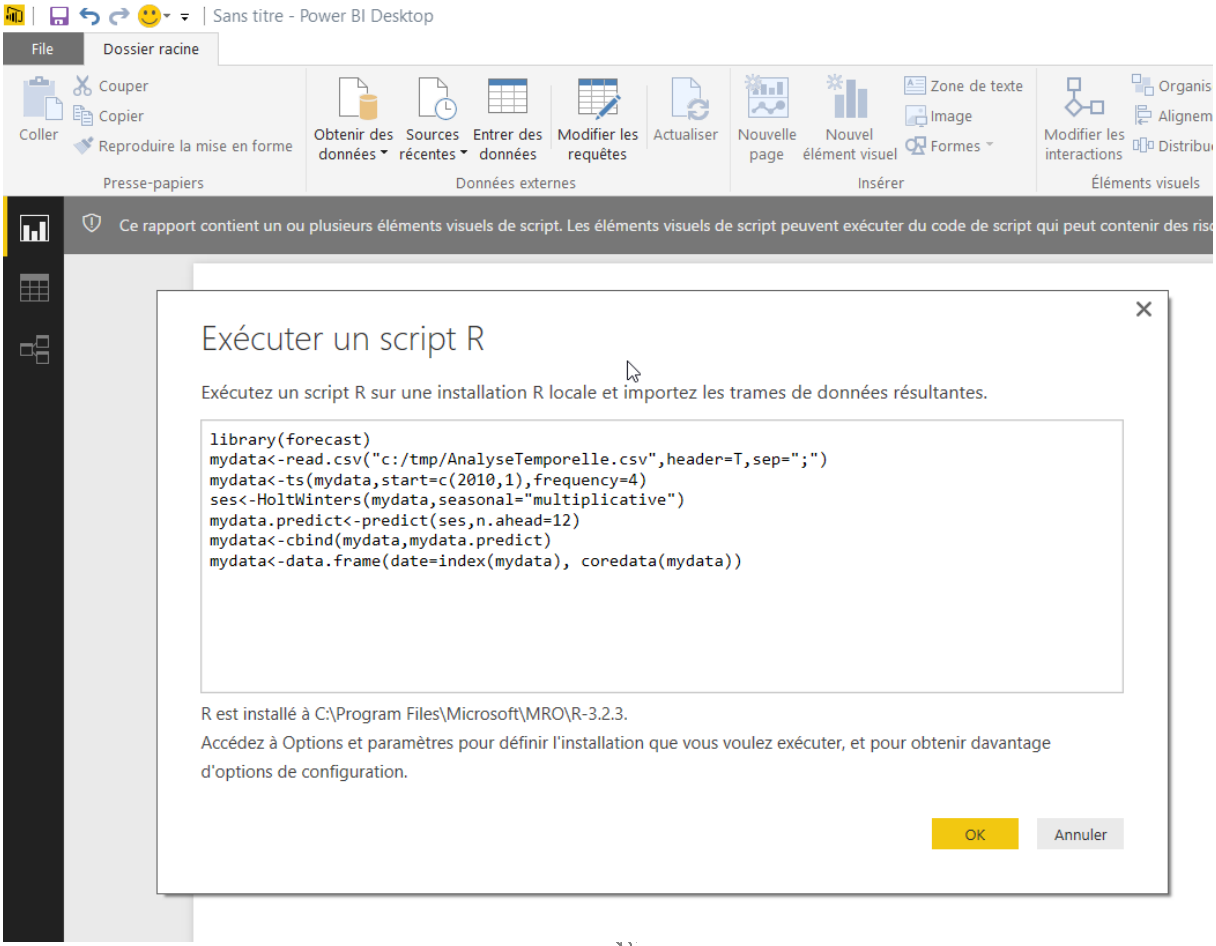
Nous cliquons sur **Se connecter**:



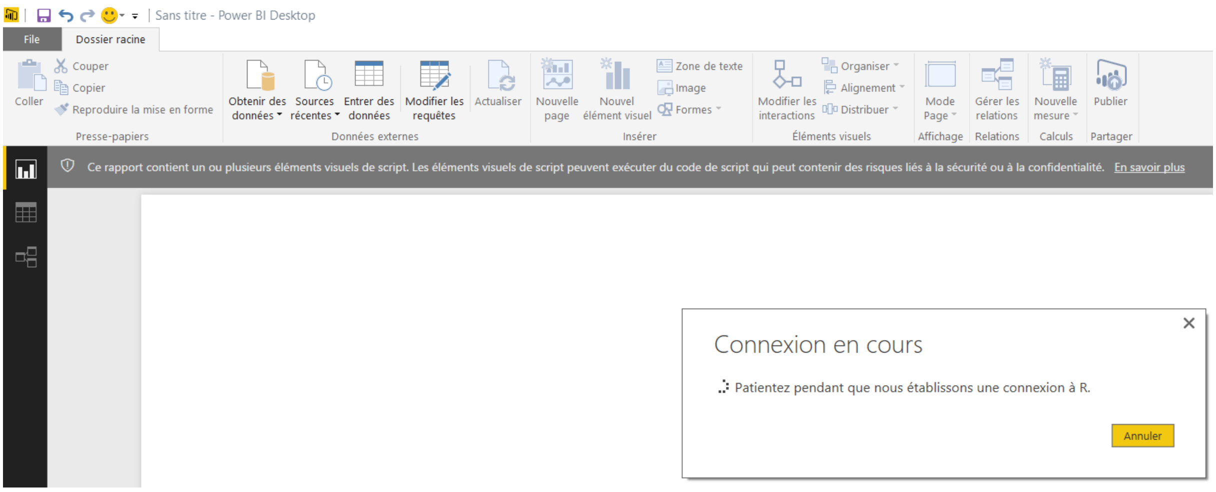
et nous cliquons sur **Continuer**:



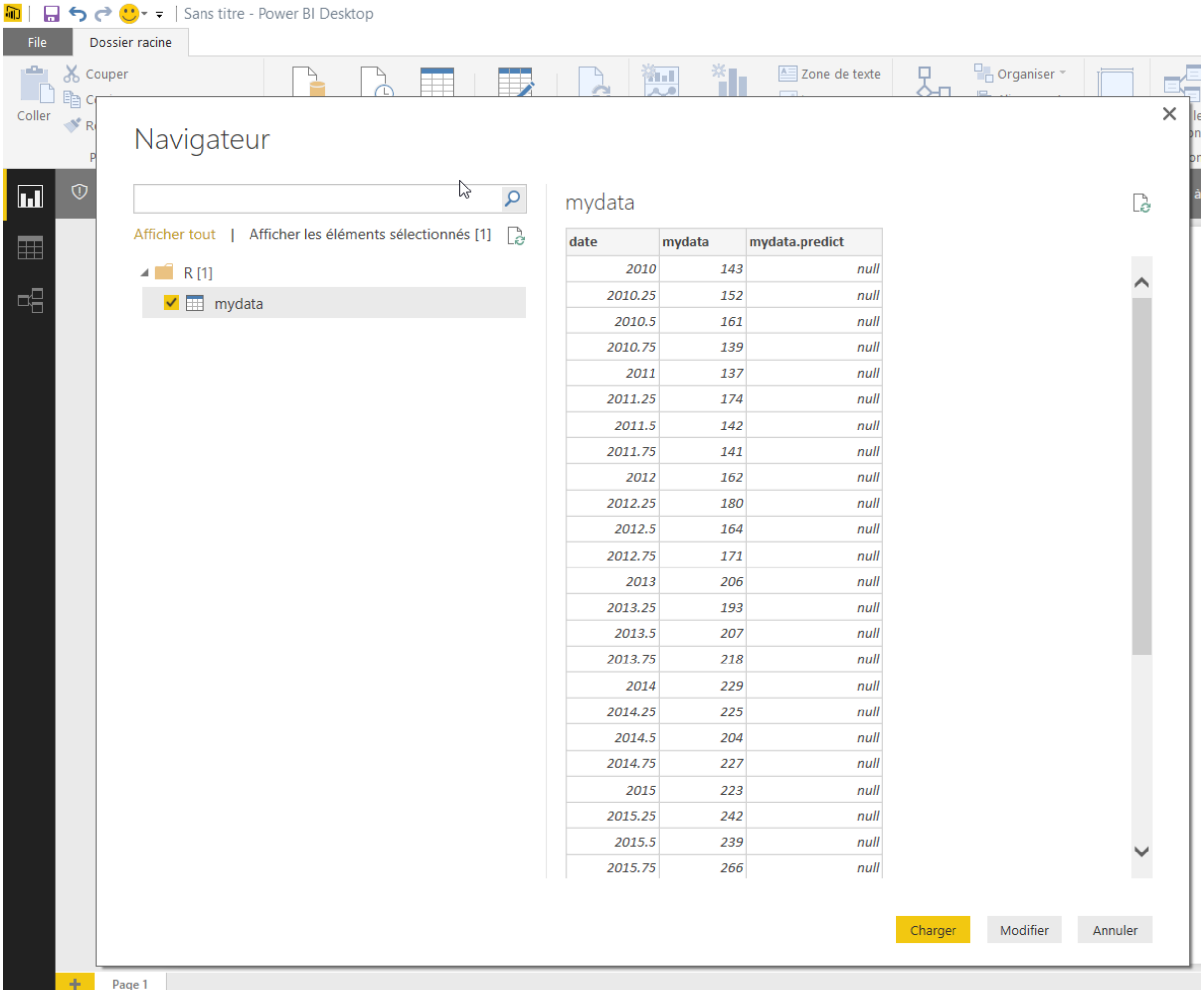
On met le script par exemple suivant (simple projection financière temporelle utilisant la méthode de Holts et Winters):



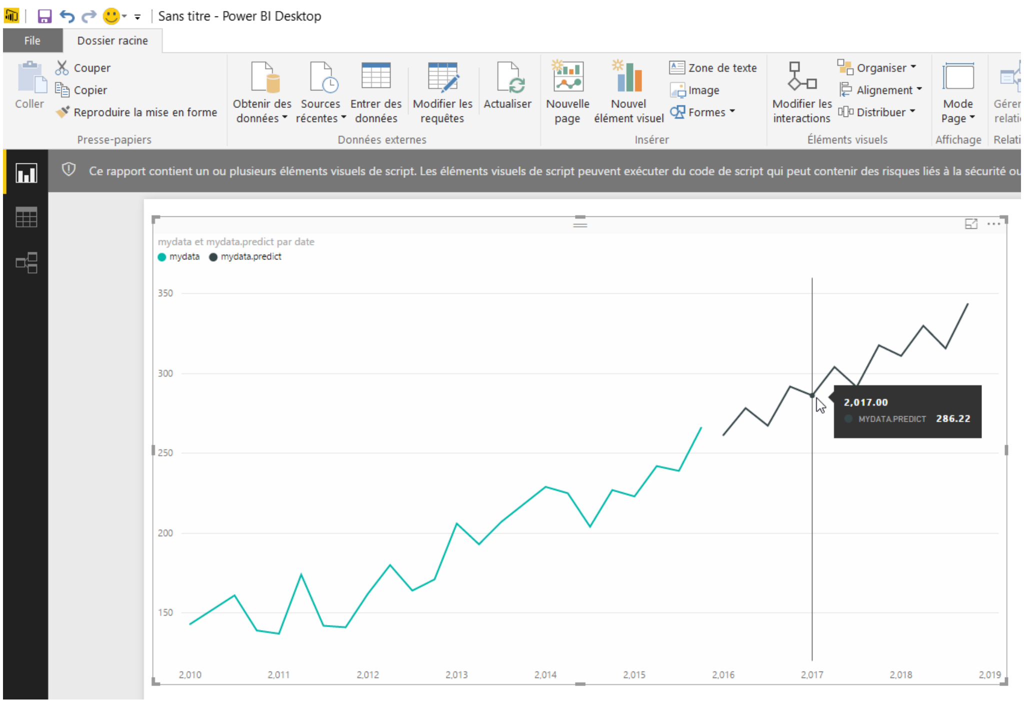
Après quelques instants:



Nous avons alors:



Et y'a plus qu'à faire des graphiques de type standards Power BI (voir exercice suivant au besoin):



Performances (benchmarking)

Nous allons voir maintenant comment Power BI performe par rapport à R. Pour cela nous allons travailler avec un fichier nommé *Ventes.csv* d'environ 1.1 GB et contenant environ 12 millions de lignes:

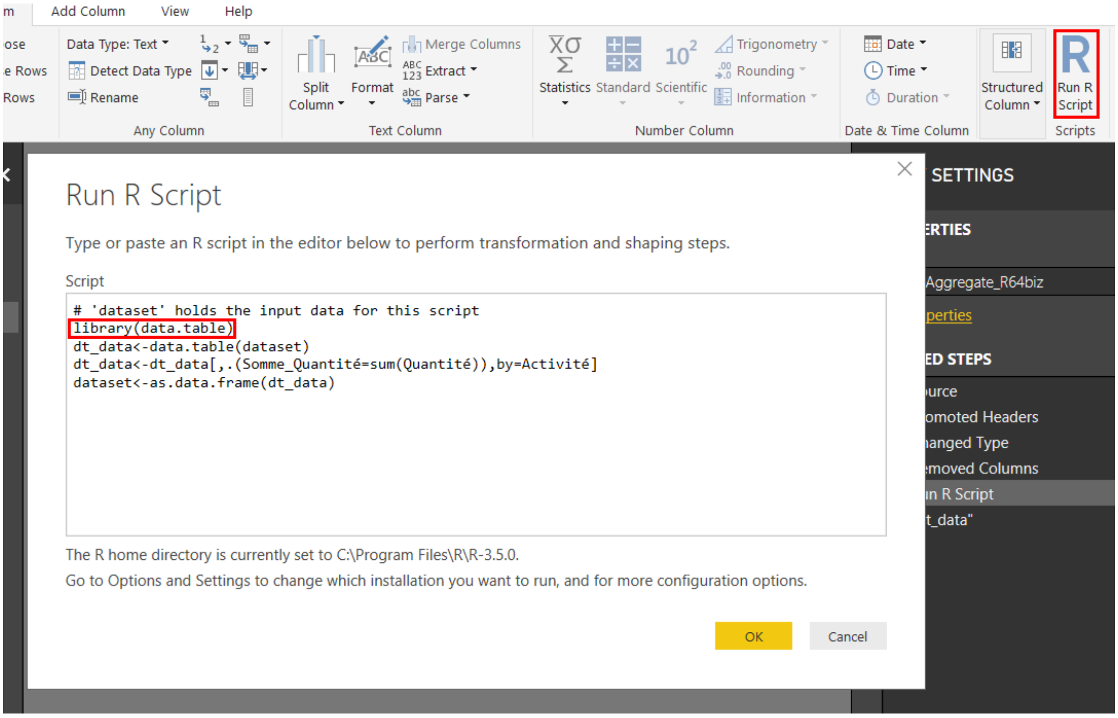
Ventes.csv 1,162,848 KB Fichier CSV Microsoft Excel

Et importer ce fichier et grouper les données comme ci-dessous:

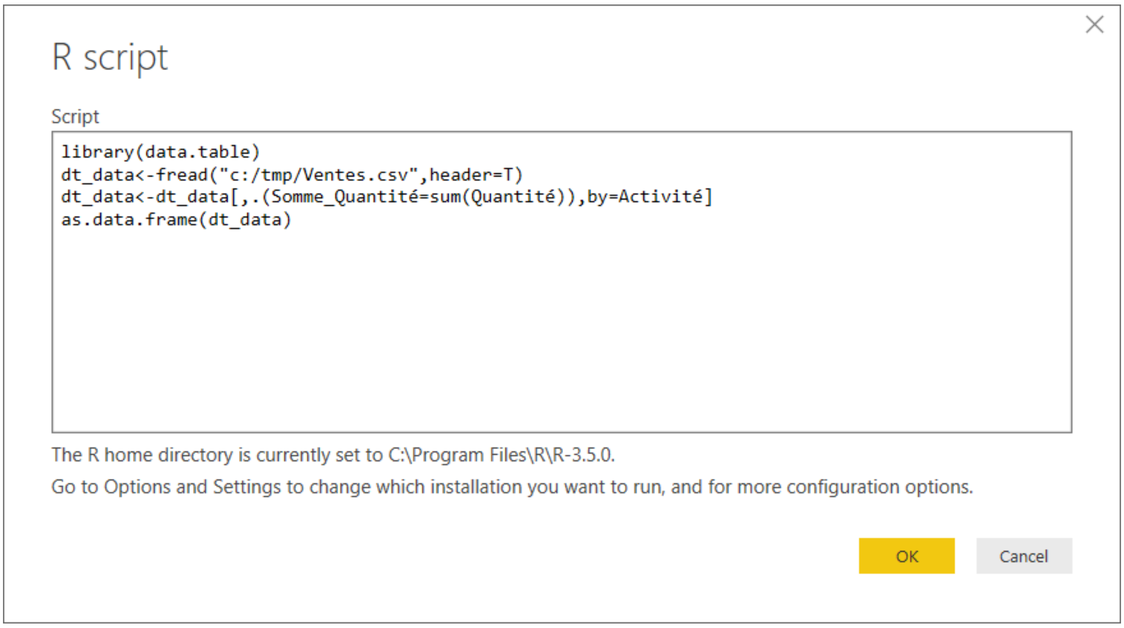
	Activité	Somme_Quantité
1	Assurances	9610517
2	Machines/Outils	19409657
3	Alimentaire	6165916
4	Construction	21399771
5	Pharmaceutique	15241384
6	Distribution	14144231
7	Éducation	20779379
8	Banques	7437726

Voici ce qu'il faut savoir concernant ce fichier (sur une machine de test unique servant d'étalon à l'exemple!):

1. L'import classique avec le groupement dans **Power BI 32** bits après 6 minutes mène à un échec du logiciel (message d'erreur).
2. L'import classique avec le groupement dans **Power BI 64** bits après 5 minutes 29 secondes mène à une réussite du logiciel (voir capture d'écran ci-dessus!)
3. L'import classique dans **Power BI 64** bits avec le groupement fait via un script **R 64 bit** (voir figure ci-dessous) après 3 minutes et 45 secondes mène à une réussite:



4. L'import et le traitement direct via R selon la méthode vue dans l'exercice précédent:



après 1 minute 2 secondes amène à une réussite…

On ne peut que conclure que le traitement de données directement via R est beaucoup… beaucoup… beaucoup plus performant et ceci sans vraiment d'étonnement…

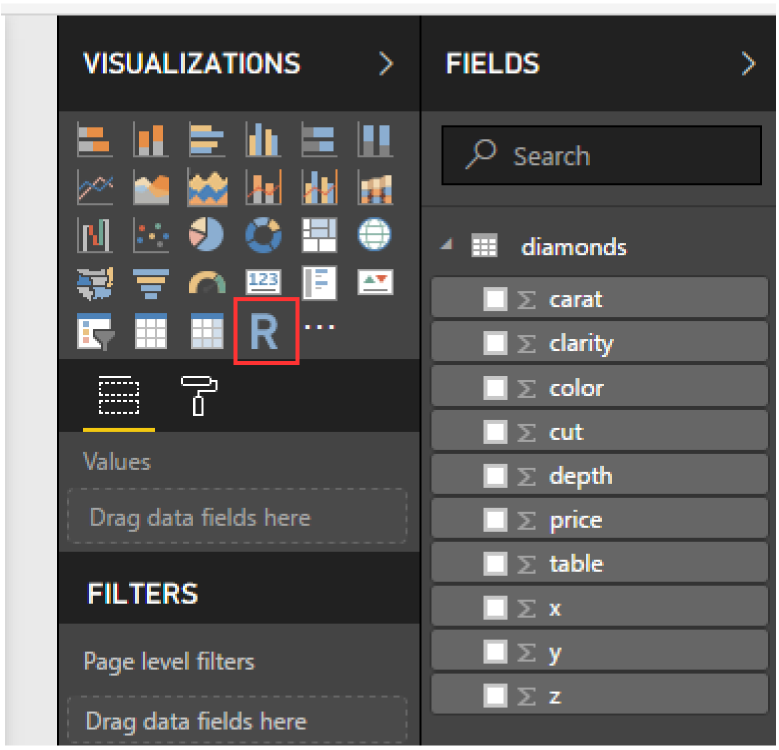
Exemple 2.: Générer des graphiques avec R

MRO 3.4.2

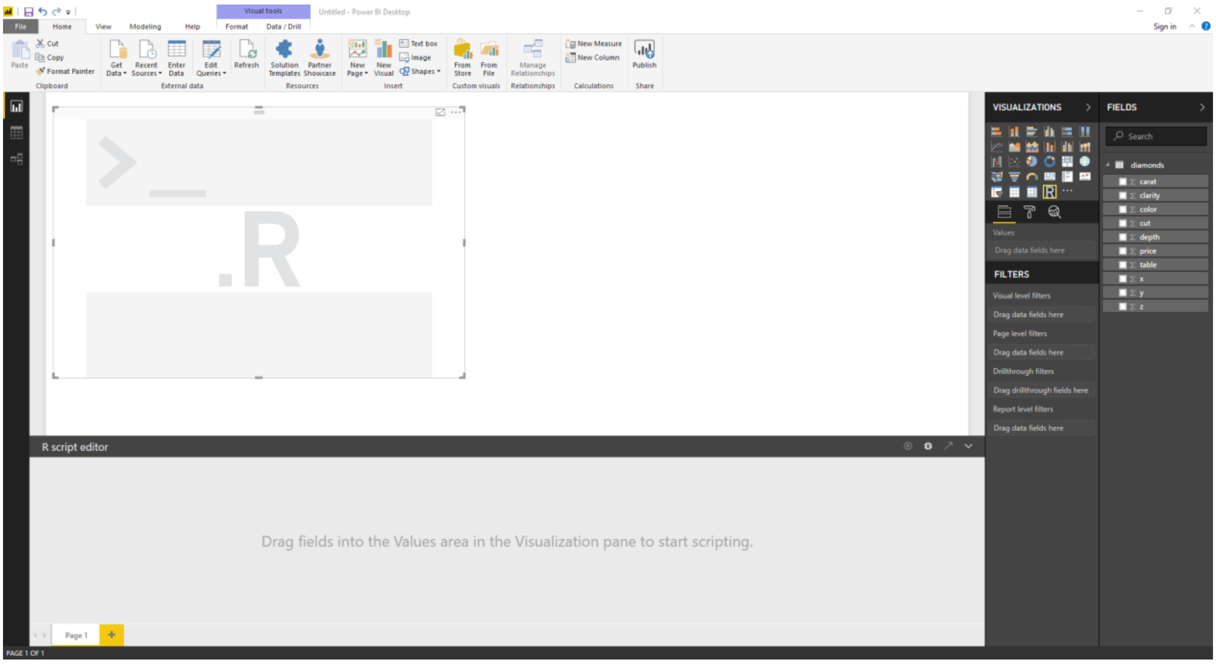
Oui vous ne rêvez pas! On peut afficher dans Microsoft Power BI Desktop, des graphiques natifs R!!!!

Voyons cela!

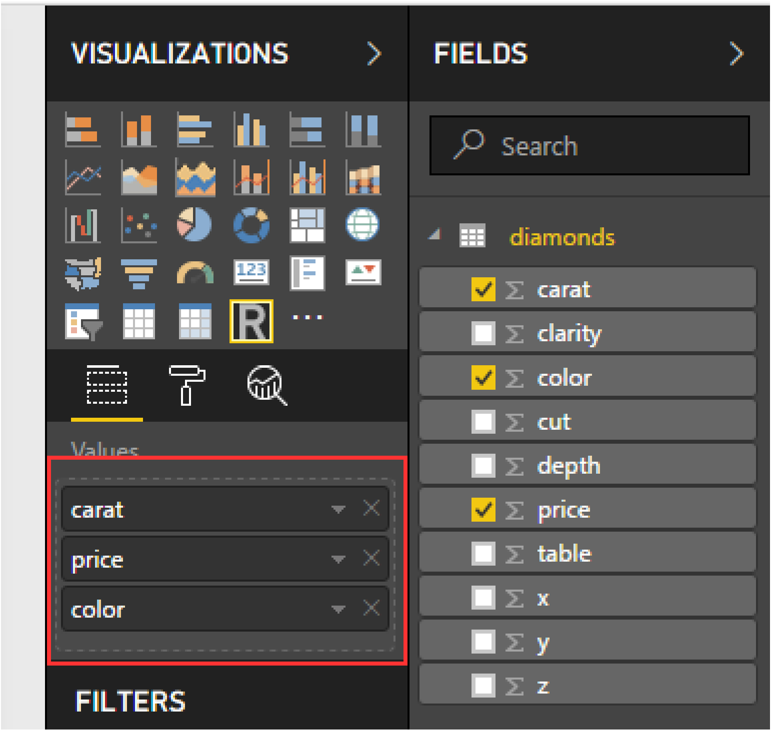
Vous cliquez sur la petite icône R visible dans le panneau de droite (après avoir importé des données):



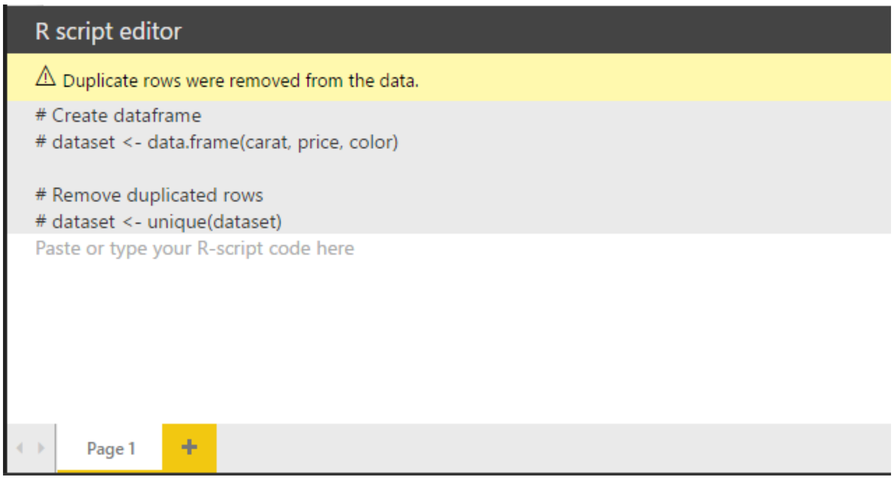
Vous aurez alors:



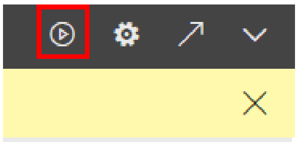
Et comme indiqué on doit glisser les champs qui nous intéressent pour commencer le scripting:



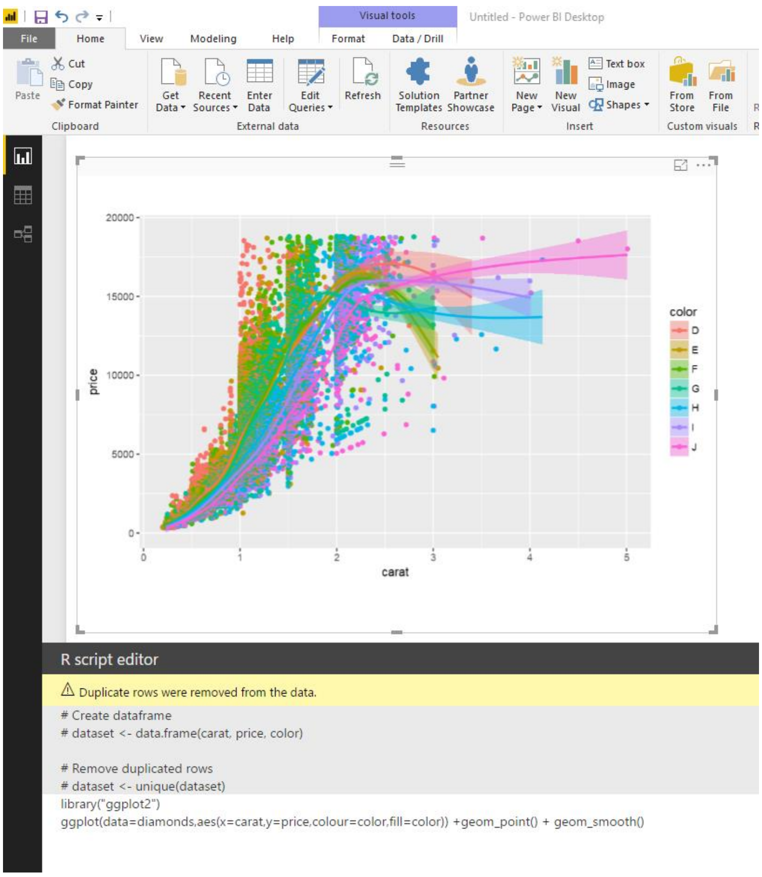
Le volet inférieur s'affichera alors comme ci-dessous:



Et ensuite y'a plus qu'à écrire le script comme à l'habitude sans oublier d'exécuter le script en cliquant sur le bouton correspondant:



Ce qui donne:



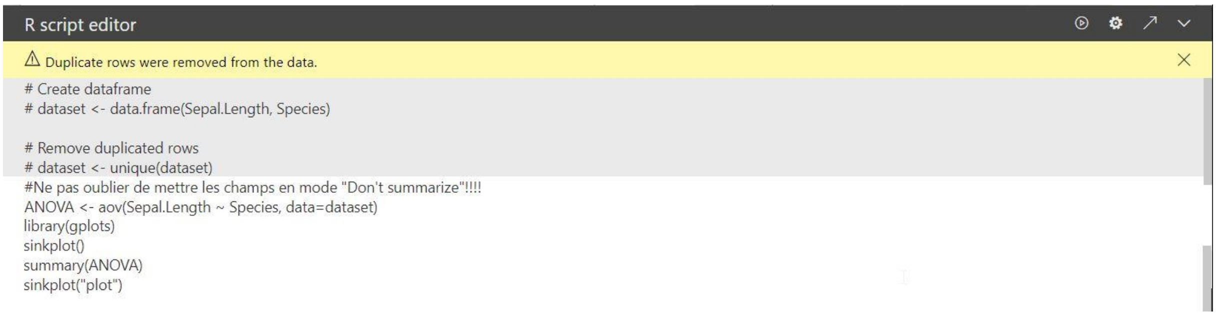
Attention!!!! Si vous souhaitez que les filtres Power BI fonctionnent avec vos graphiques, il faut alors utiliser le data frame nommé ***dataset*** (créé par Power BI) dans le script R et non pas votre non de data frame initial!

Exemple 3.: Afficher des analyses statistiques "avancées" avec R (avec gplots)

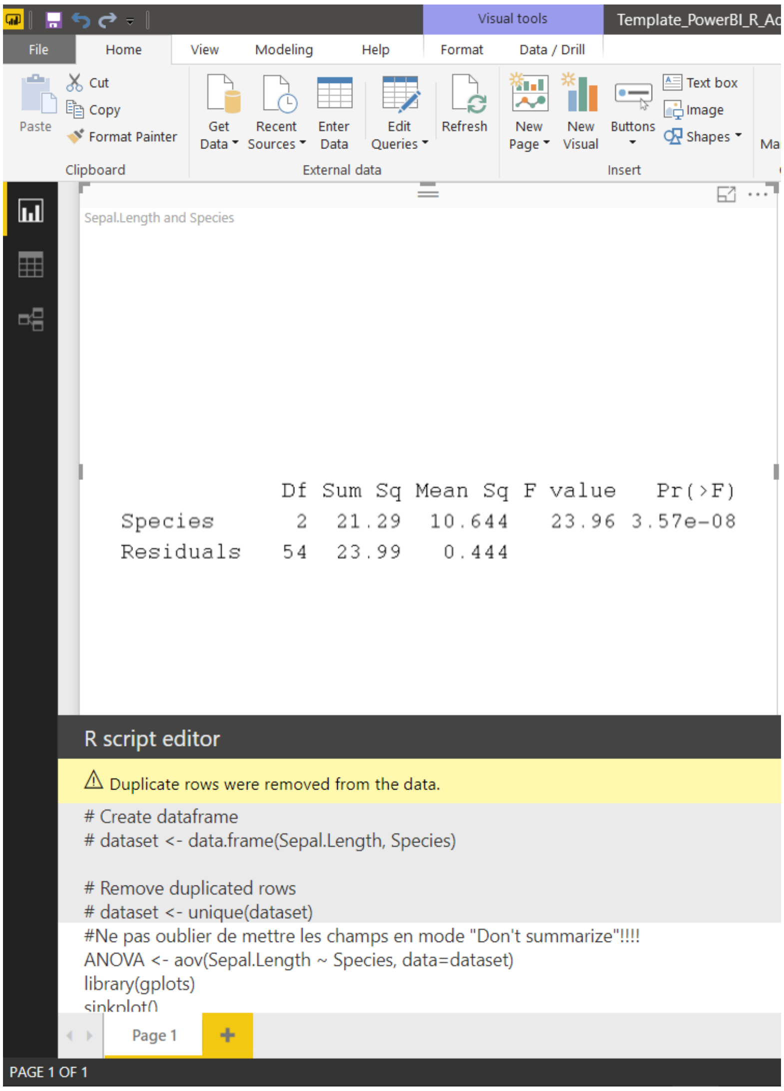
MRO 3.4.2

En procédant de même qu'avant pour importer des données et pour exécuter un script R et en utilisant le package **gplots** et sa fonction **sinkplot( )** , nous arrivons parfaitement à afficher une ANOVA dans Power BI.

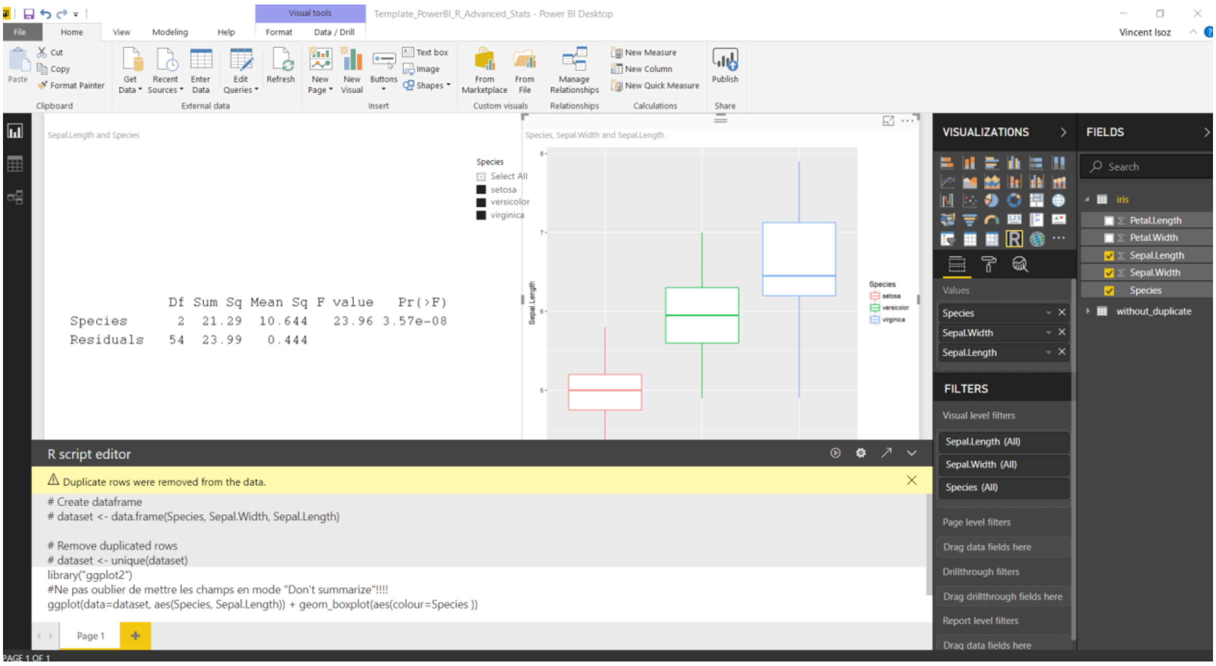
Le script à mettre dans Power BI sera:



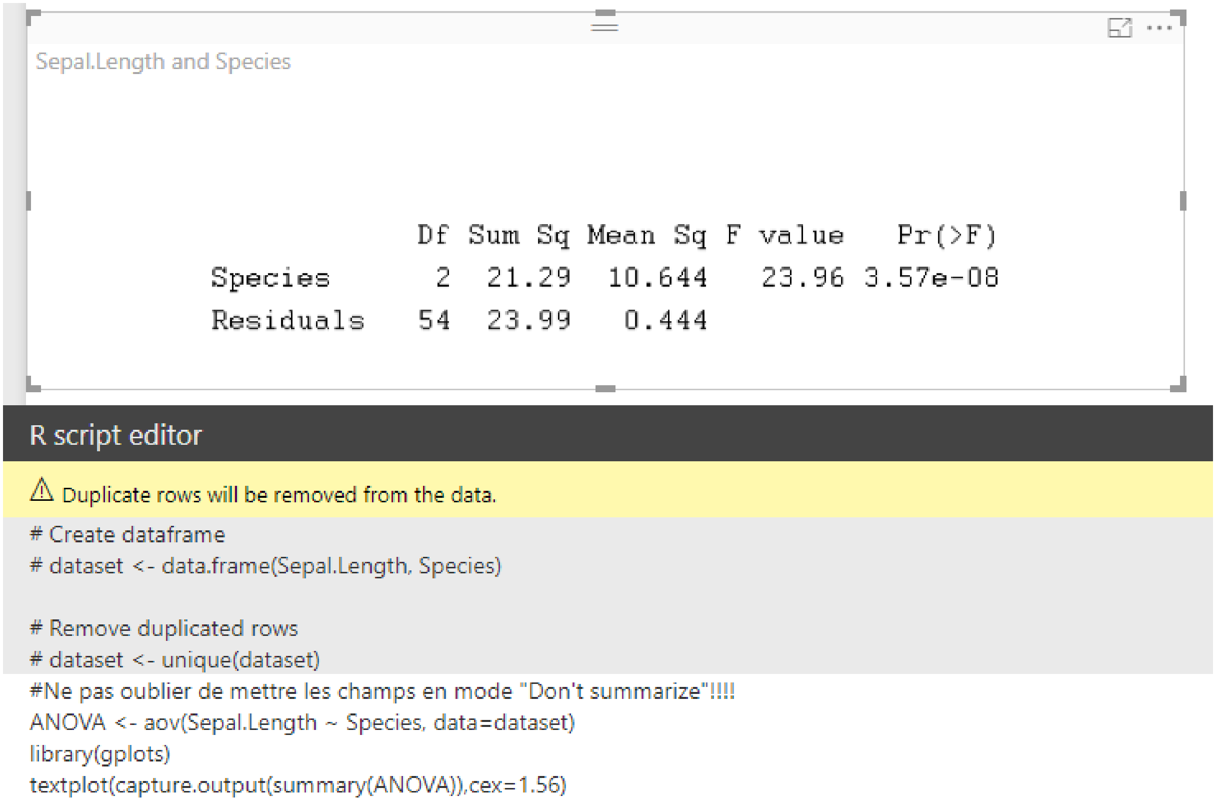
Ce qui donnera:



Et bien évidemment on peut ajouter un graphique Box-Whisker et ajouter au tout un slicer (filtre général) afin d'obtenir un résultat du genre suivant:



Sinon une très bonne alternative est d'utiliser la fonction **textplot( )** avec **capture.output( )** qui a un meilleur rendu à l'écran et permet aussi de contrôler la taille de la typo:



Bref il est possible de faire de la "vraie" statistique inférentielle (bayésienne, non-bayésienne, paramétrique, non-paramétrique, etc.) que de la statistique descriptive scolaire niveau collège…!

Pour afficher des structures tabulaires nous pouvons sinon utiliser le package **gridExtra** et sa fonction grid.table( ):

Custom

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa :50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	NA
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	NA
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	NA

R script editor

⚠ Duplicate rows will be removed from the data.

```
# Remove duplicated rows
# dataset <- unique(dataset)
library(gridExtra)
data(iris)
grid.table(summary(iris))
```

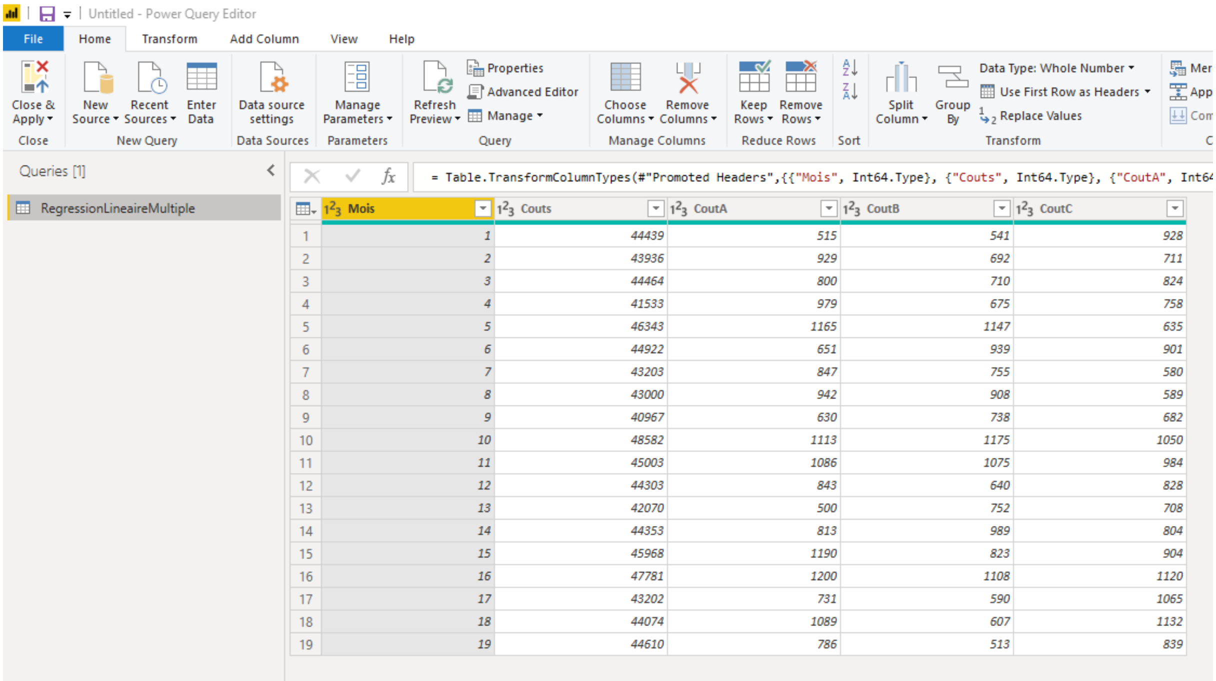

Benoit Fedit a résumé différentes approches sur son blog disponible ici:

https://datakuity.com/2022/02/15/show-text-results-from-an-r-visual-in-power-bi/

Exemple 4.: Manipulation de tables avec R (régression linéaire simple)

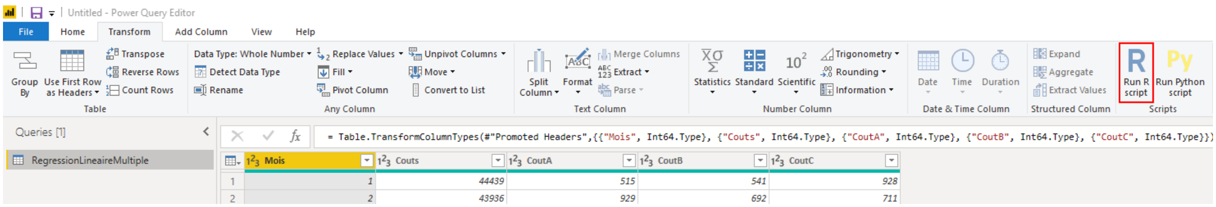
MRO 3.5.3

Considérons la table suivant dans Power Query:

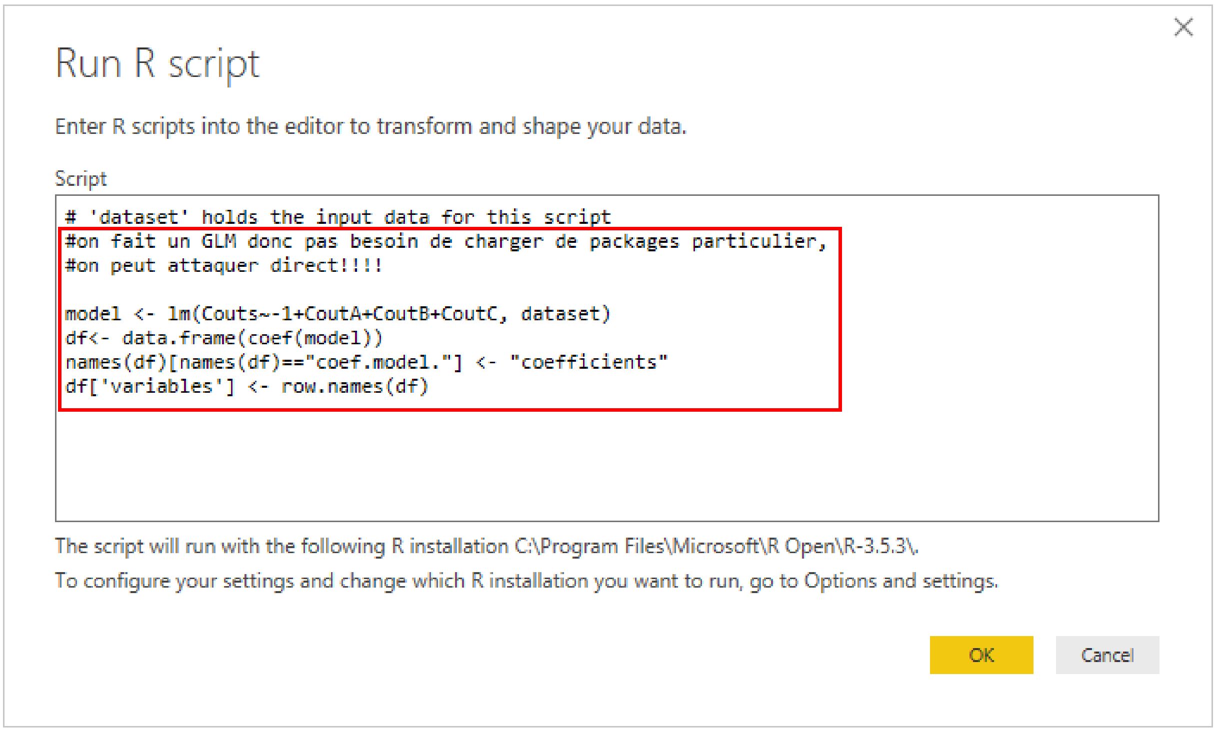


	Mois	Couts	CoutA	CoutB	CoutC
1	1	44439	515	541	928
2	2	43936	929	692	711
3	3	44464	800	710	824
4	4	41533	979	675	758
5	5	46343	1165	1147	635
6	6	44922	651	939	901
7	7	43203	847	755	580
8	8	43000	942	908	589
9	9	40967	630	738	682
10	10	48582	1113	1175	1050
11	11	45003	1086	1075	984
12	12	44303	843	640	828
13	13	42070	500	752	708
14	14	44353	813	989	804
15	15	45968	1190	823	904
16	16	47781	1200	1108	1120
17	17	43202	731	590	1065
18	18	44074	1089	607	1132
19	19	44610	786	513	839

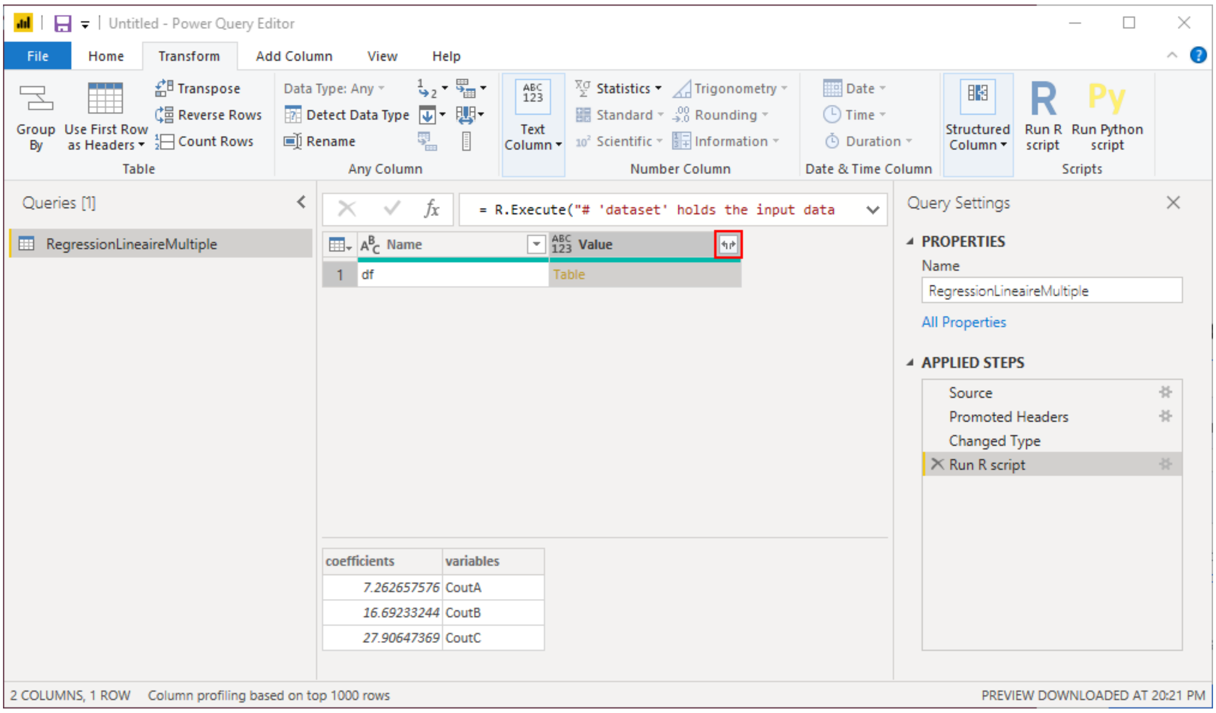
Nous souhaiterions récupérer les coefficients de régression linéaire simple en utilisant R. Pour ce faire nous cliquons d'abord sur le bouton:



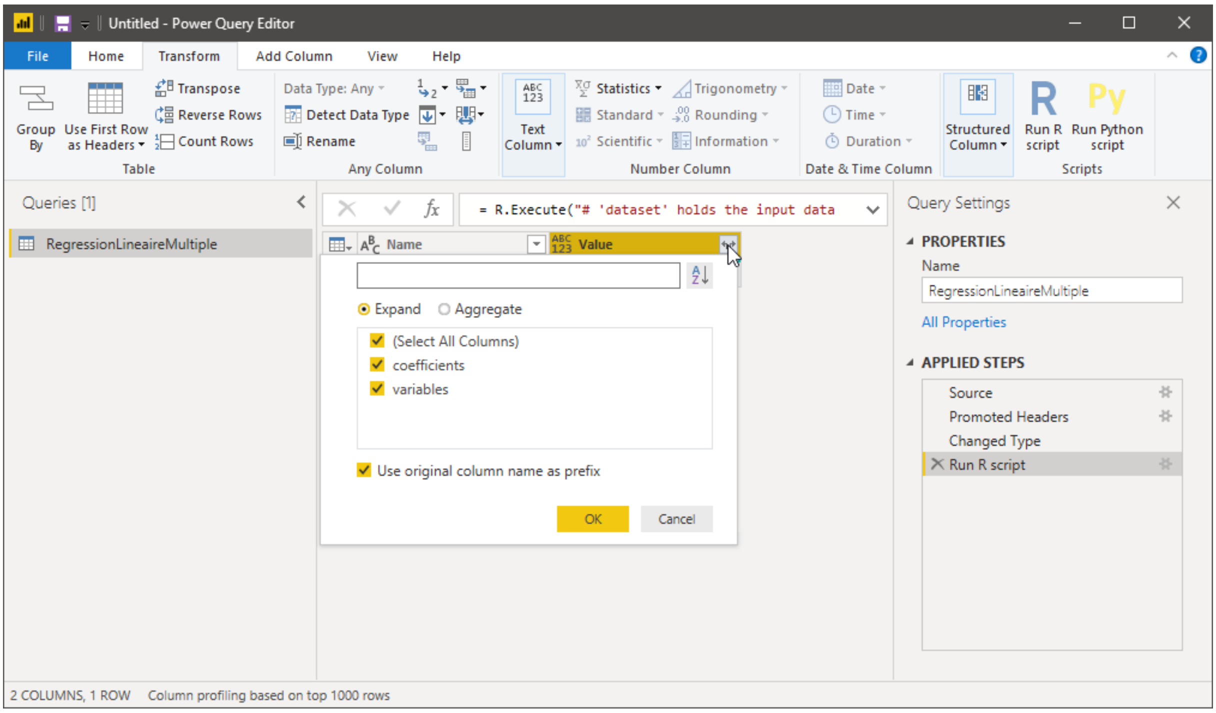
Dans la boîte de dialogue qui apparaît, nous mettons alors:



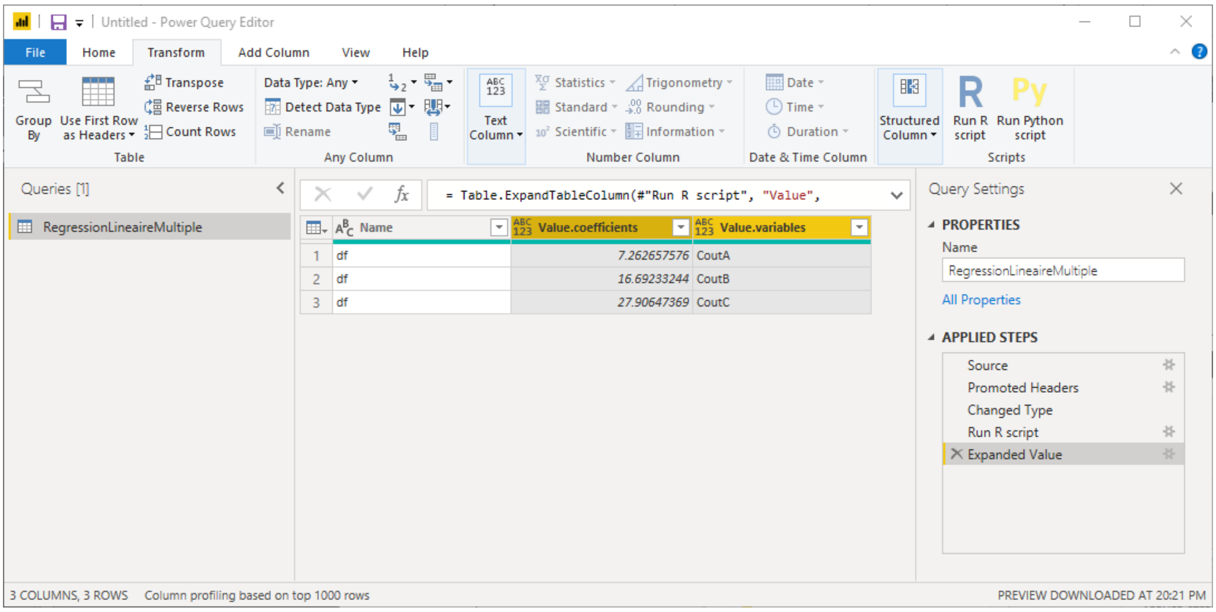
Ce qui donne après validation:



On clique alors sur l'expandeur:



Et on valide par OK:



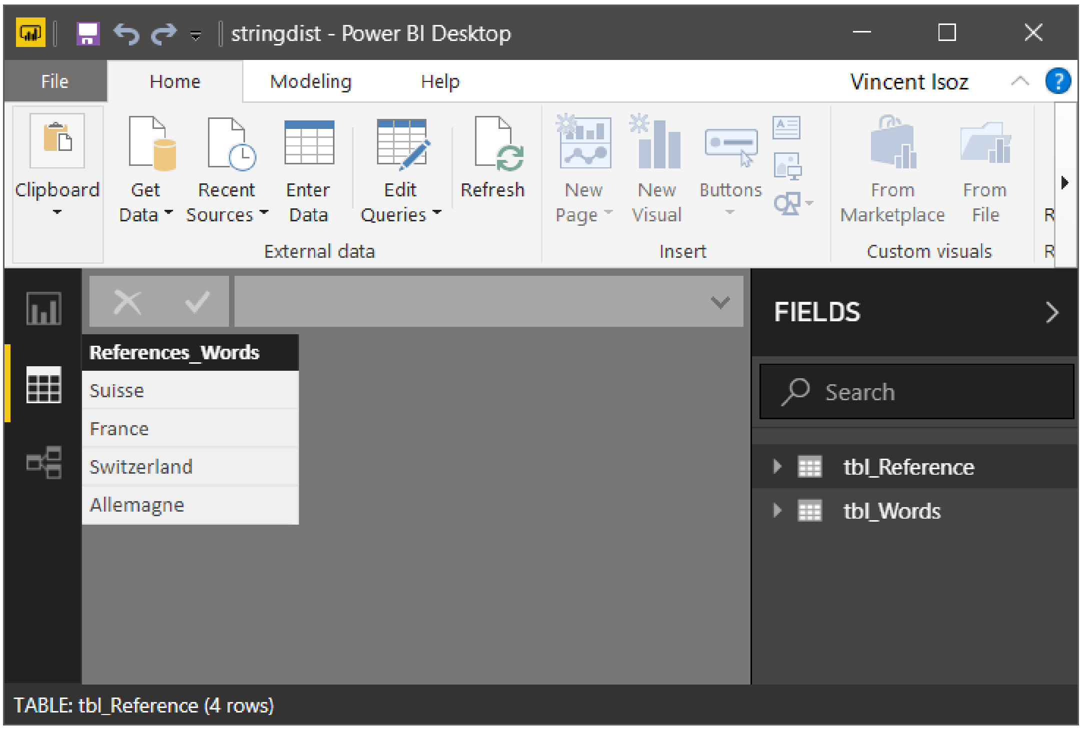
Ce qui donne bien le résultat attendu! On peut bien évidemment supprimer la première colonne!

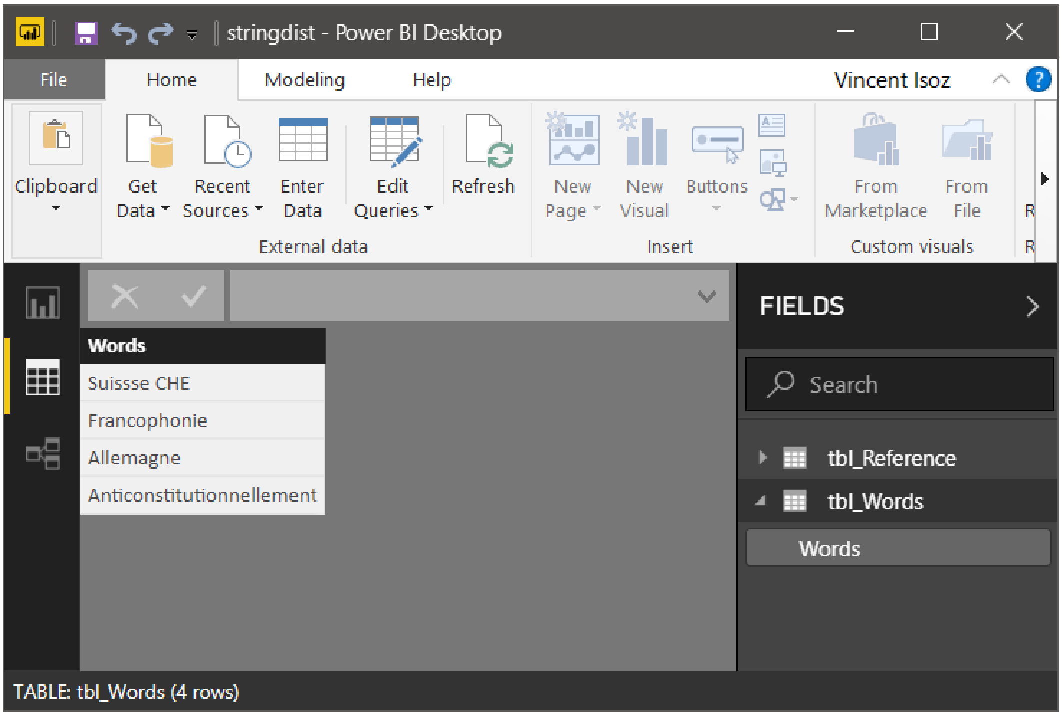
Exemple 5.: Manipulation de deux tables avec R

MRO 3.4.2

Nous allons voir ici comment on peut manipuler avec R dans Power BI plusieurs tables différentes (car ce n'est pas trivial) avec un exemple particulier de "similarity matching" de chaînes de caractères.

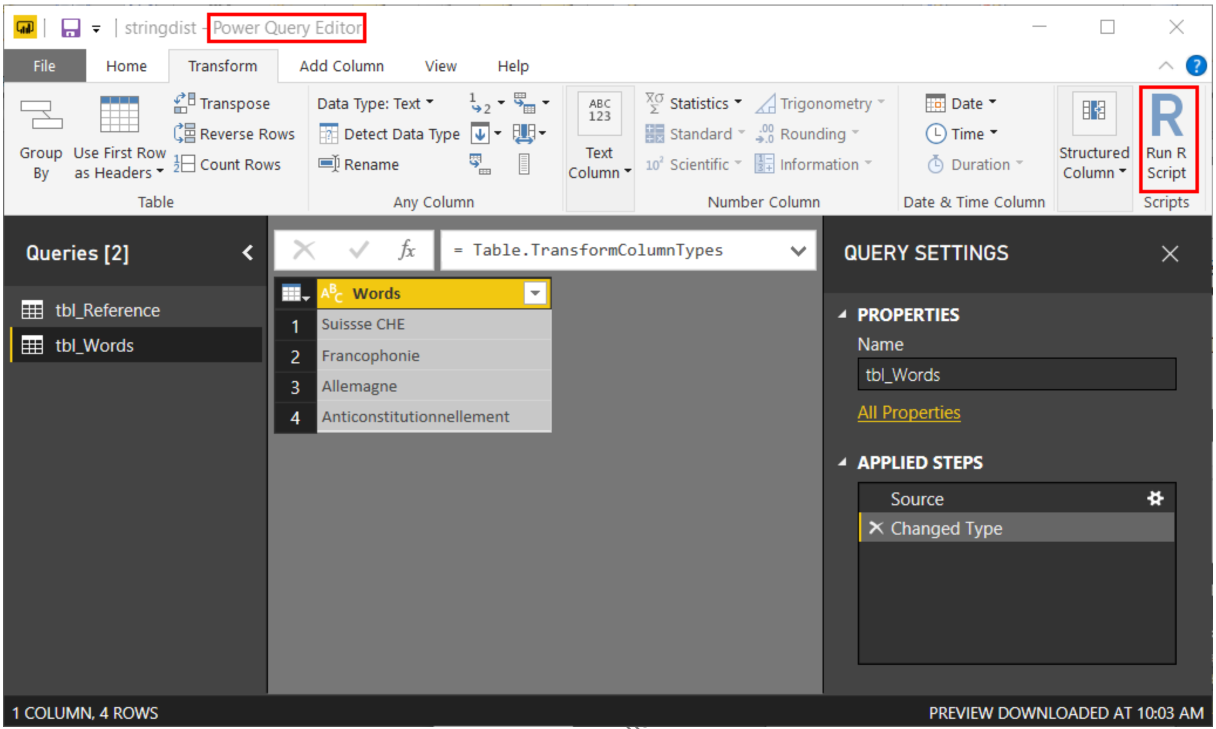
Considérons les deux petites tables suivantes saisies manuellement (via le bouton *Enter Data* de Power BI):



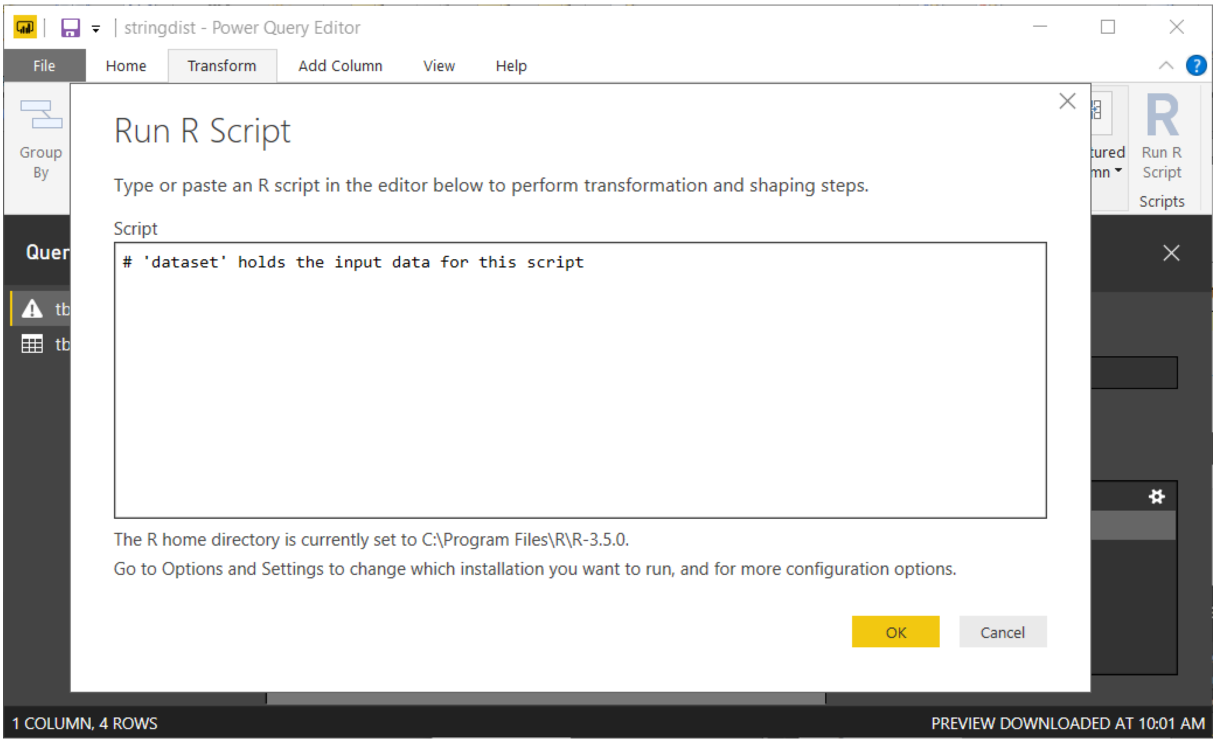


Avec R, nous voulons dans la seconde table *tbl_Words* utiliser la méthode Jaro-Winkler pour faire de la similarité de termes et faire les remplacements au mieux!

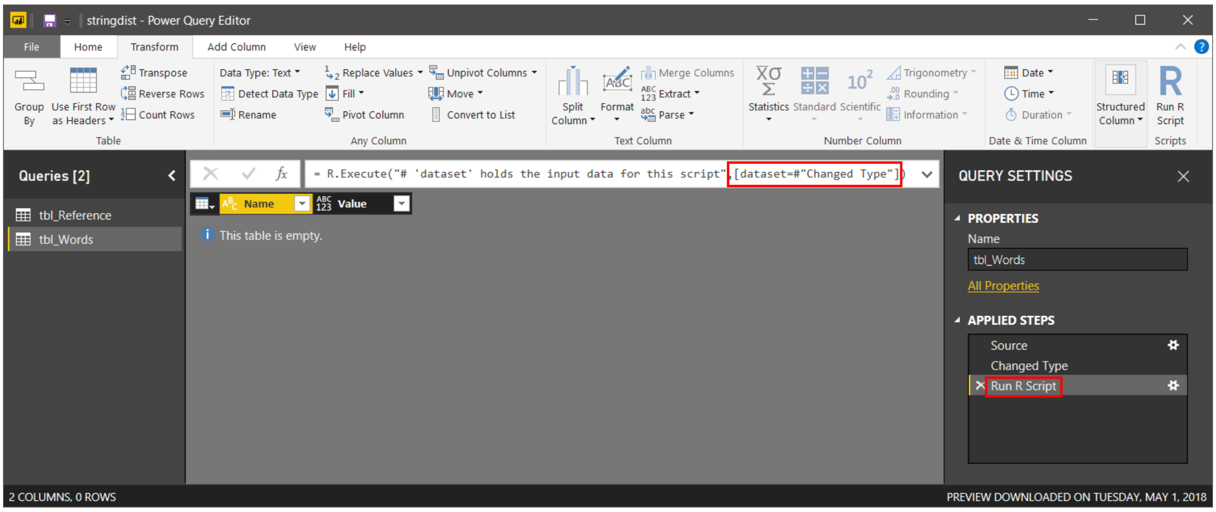
Pour ce faire, nous ouvrons la dernière table dans Power Query en cliquant sur le bouton *Edit Queries*:



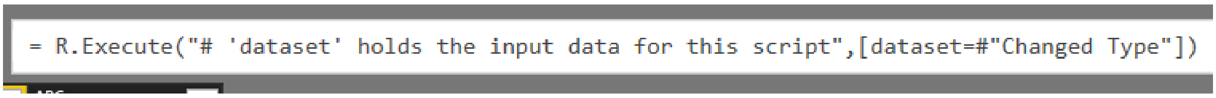
Cliquez sur le bouton *Run R script*:



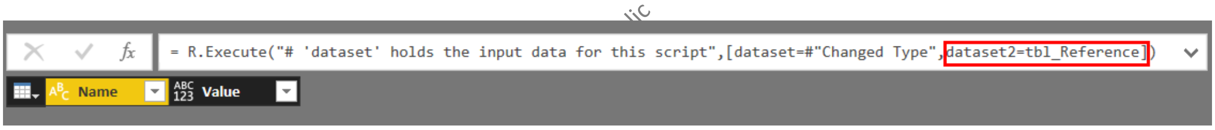
et fermez de suite après en cliquant sur *OK*. Observez ce que nous avons (une nouvelle étape et quelque chose d'intéressant dans la barre de formules):



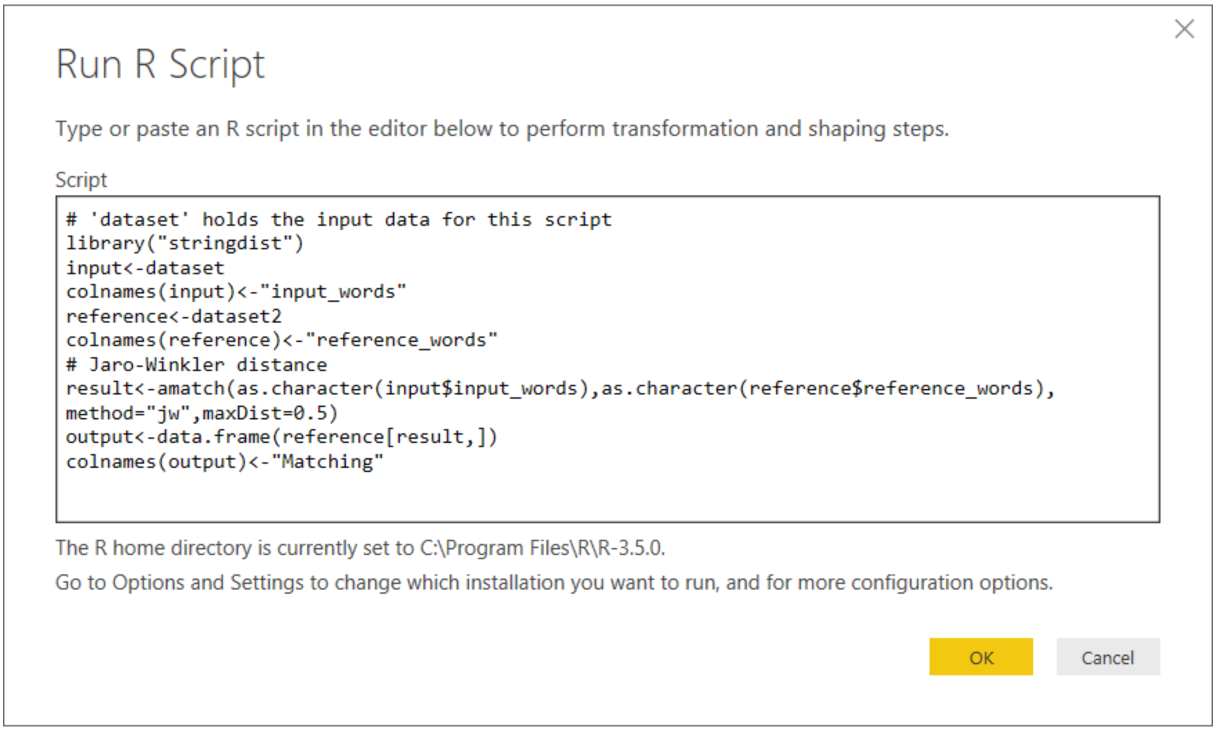
Faisons un zoom sur la barre de formules:



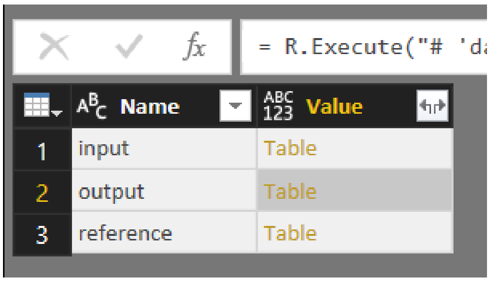
On voit que c'est via la formule que Power BI charge le dataset en cours dans le script R. Pour aller y prendre une autre table, nous allons modifier cette dernière comme suit:



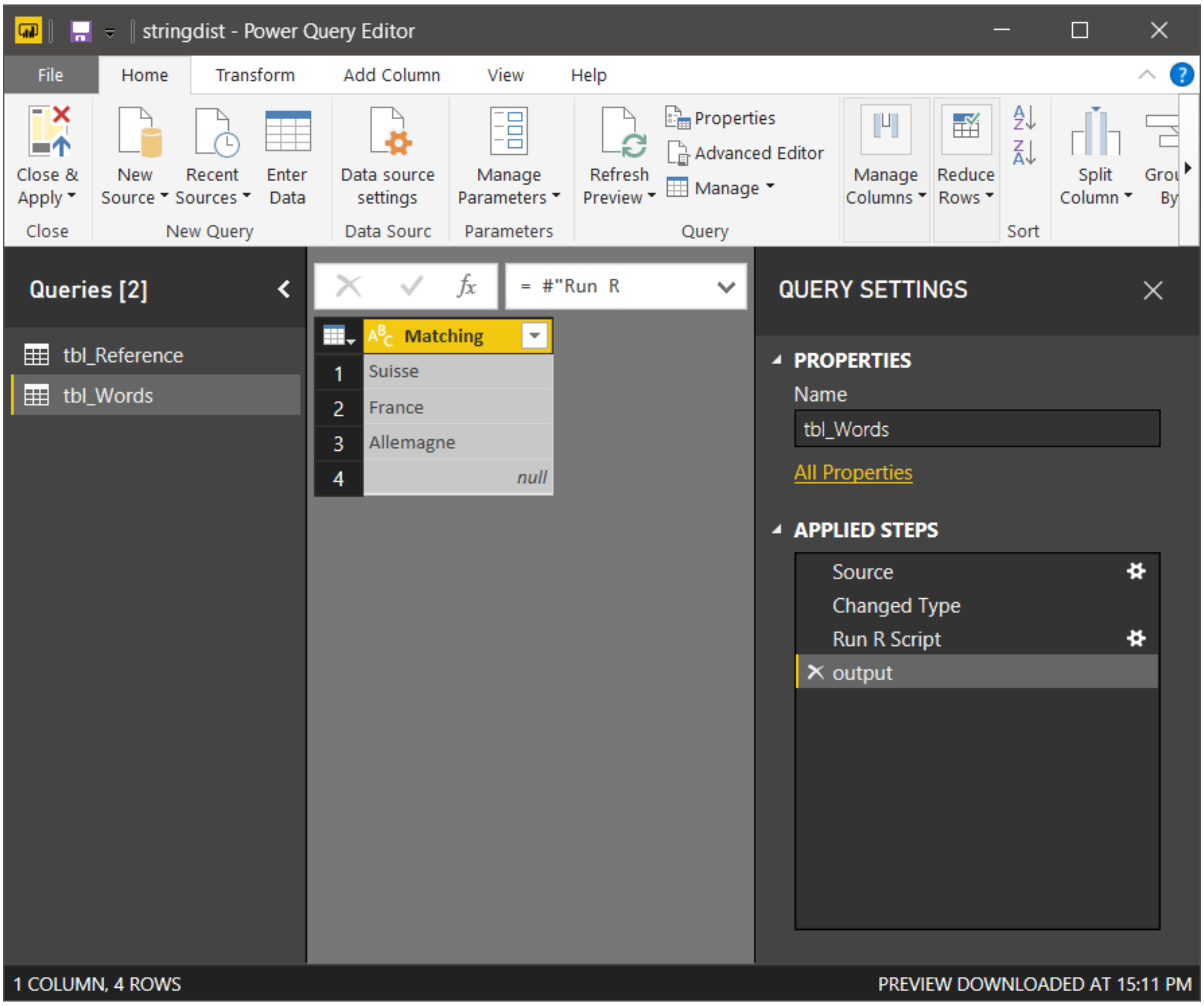
et on retourne dans la fenêtre de script pour écrire:



Ce qui donnera:



En cliquant sur *Table* de la ligne *ouptut* on obtiendra:

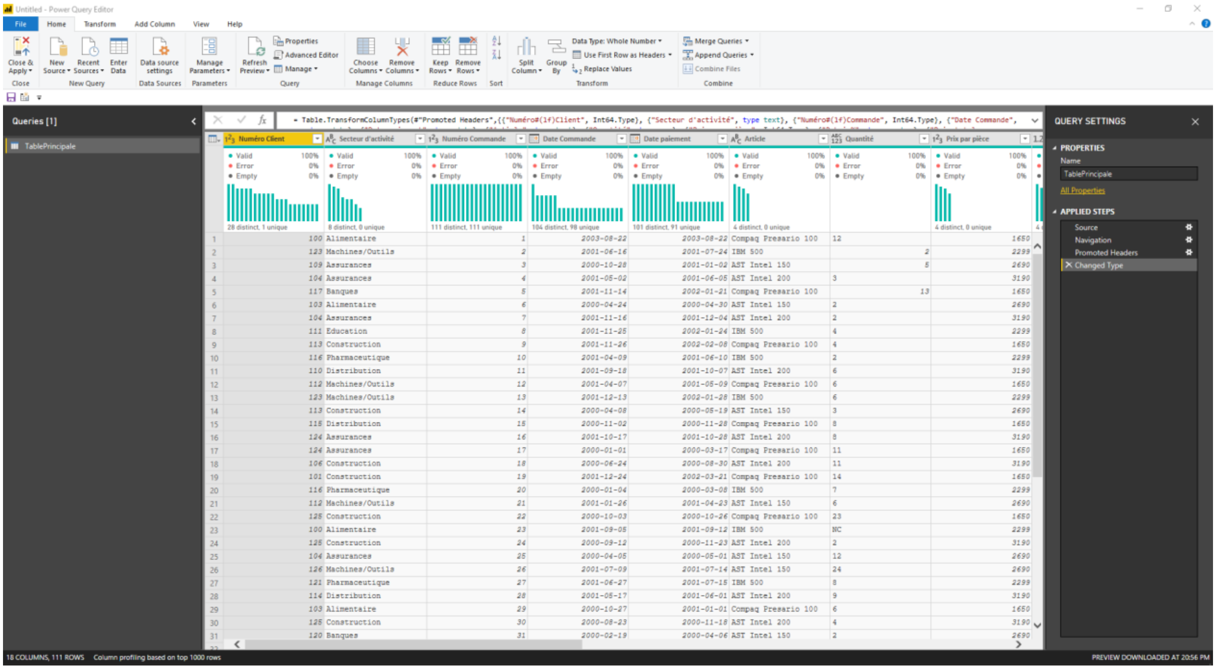


Exemple 6.: Export de données avec R

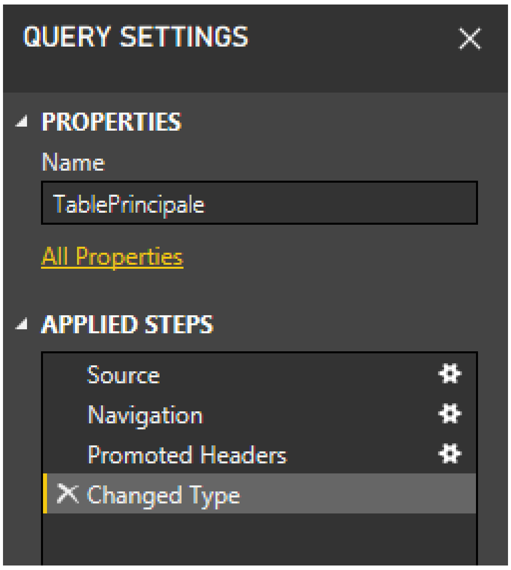
MRO 3.6.0

Considérons le simple cas suivant d'un import de fichier CSV d'environ 11 millions de lignes (**sans erreurs**!!!) que nous souhaitons immédiatement exporter.

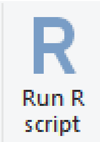
Nous partons donc de Power Query:



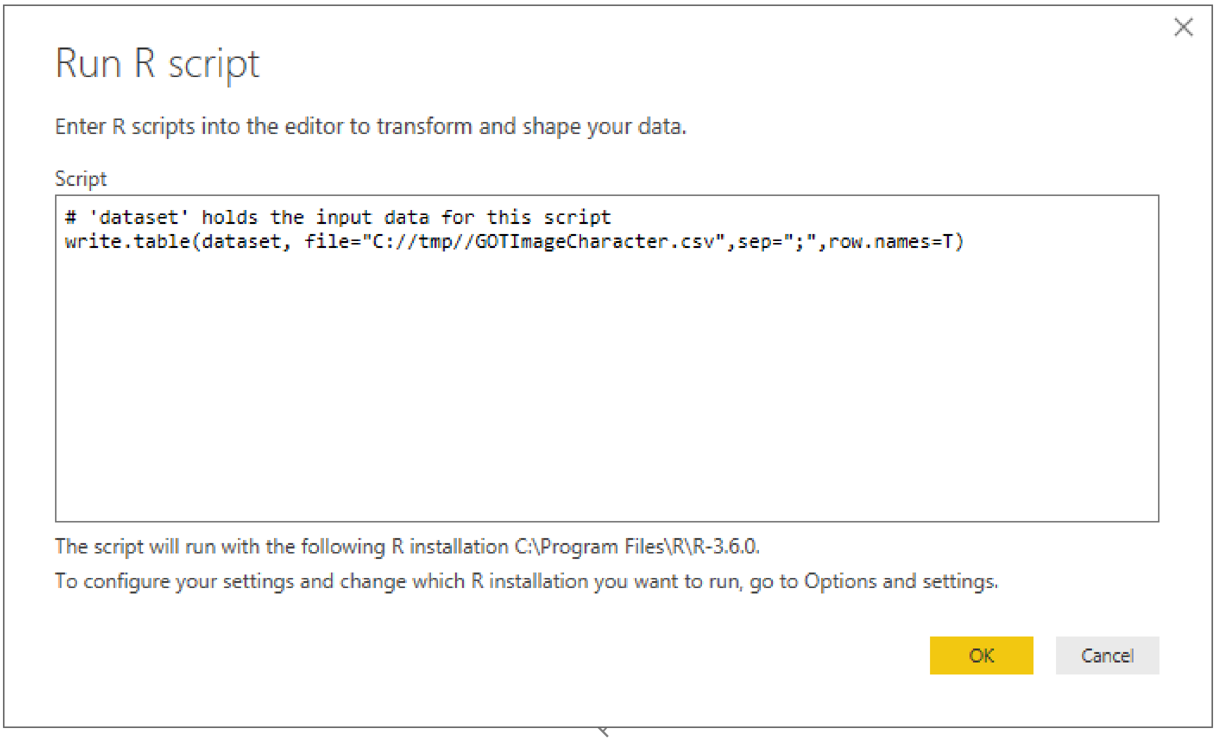
Observez bien les étapes actuelles:



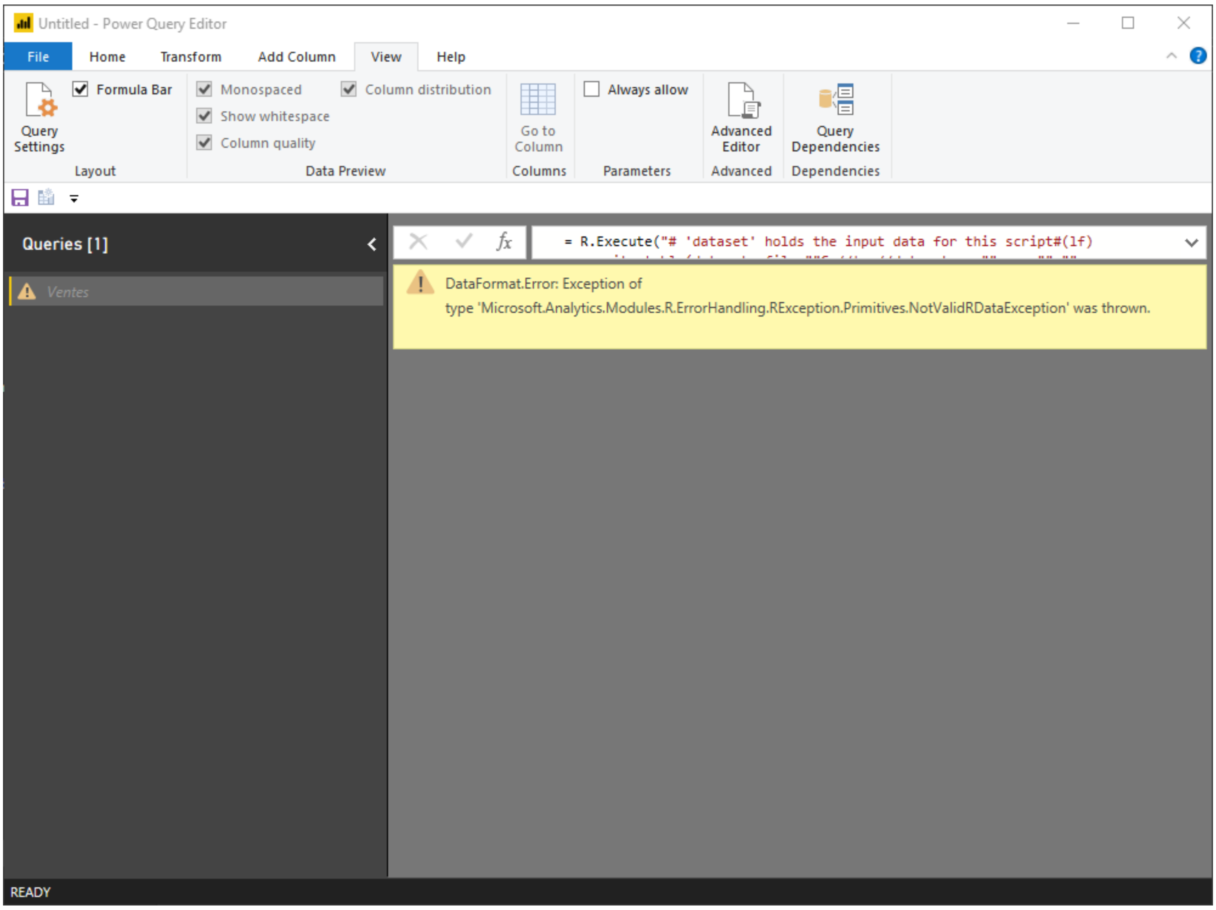
Nous voulons rajouter une étape de script R qui exporte les données! Pour ce faire nous cliquons dans l'onglet **Transform** sur:



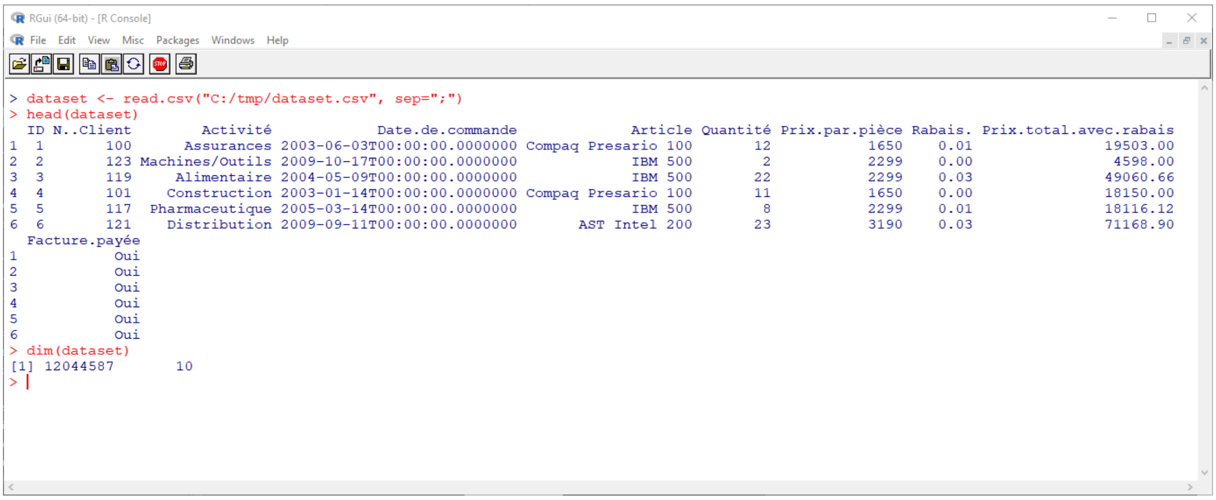
Et on y met:



L'erreur suivante apparaitra:



Mais l'export aura fonctionné! Comme le prouve la manipulation suivante dans R:



Cependant il faudra poursuivre en langage M pour la suite concernant Power Query (ou l'utilisateur de base pourra supprimer la ligne R incriminée!)…

Il faut savoir que cette méthode n'est pas exempte de problèmes et de défauts à ce jour malheureusement…

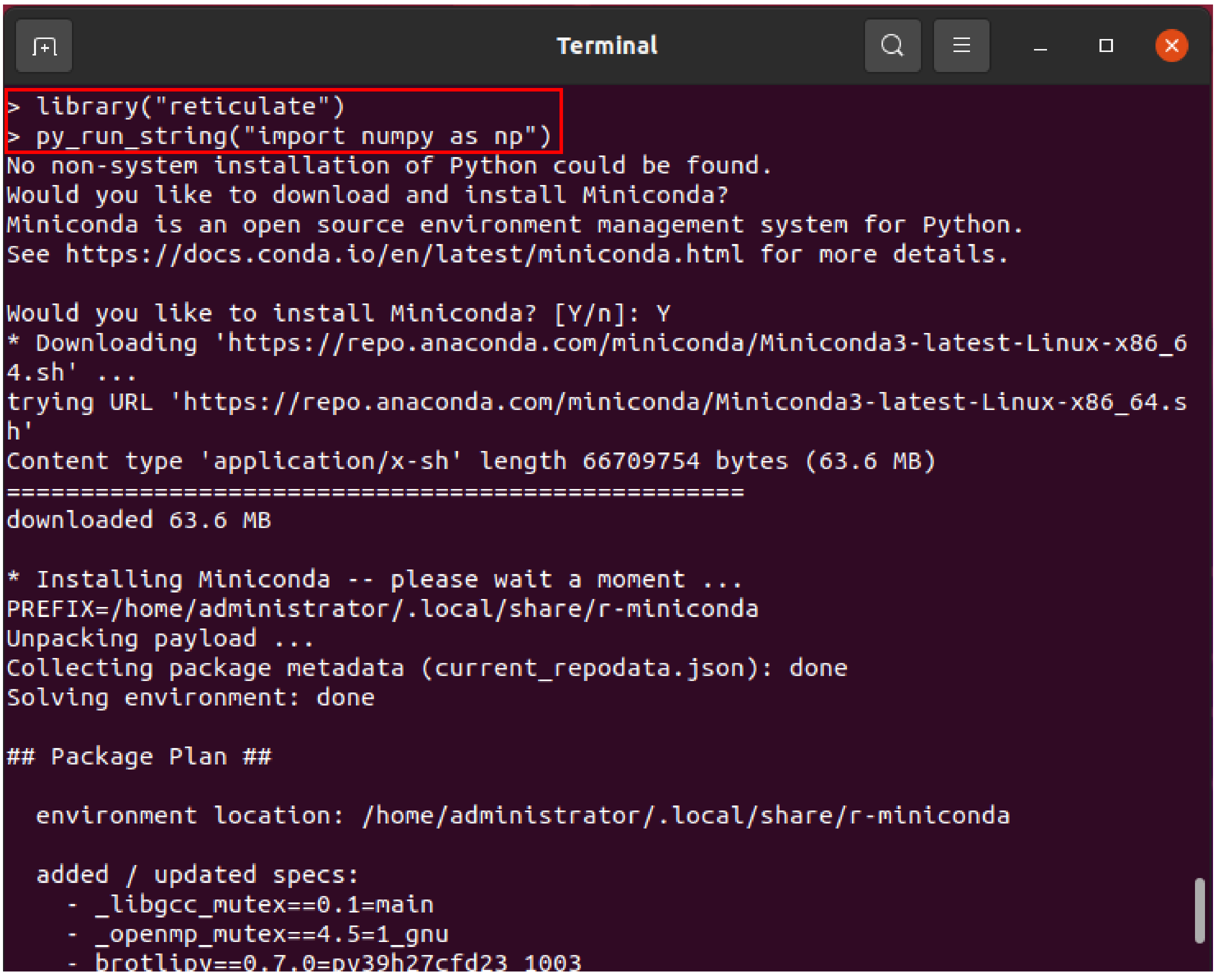
Reticulate

Si vous voulez travailler sur un IDE unique (ici RStudio) et utiliser à la fois R & Python, la solution s'appelle reticulate (https://rstudio.github.io/reticulate/index.html).

C'est un package R qui permet de gérer le langage Python de plusieurs manières :

- Code Python dans un fichier Rmd (R Markdown), les chunks peuvent être en R ou en Python et les objets peuvent être appelés dans les deux sens (R <=> Python)
- Pour récupérer un objet Python en R : *py$MonObjetPy*
- Pour récupérer un objet R en Python : *r.MonObjetR*
- Importer des modules Python et les utiliser en R (utiliser le $ pour accéder aux fonctions d'un module) à l'aide de **import( )**.
- Sourcer des scripts Python à l'aide de **source_python( )**
- Remplacer la console R en console Python à l'aide de la fonction **repl_python( )**

Voyons un petit exemple (ensuite il n'y a plus qu'à généraliser) sur Linux car dans un environnement Microsoft Windows il y a souvent que des problèmes (pour changer…):



```
libgomp-11.2.0          | 427 KB    | ##################################### | 100%
ld_impl_linux-64-2.3    | 667 KB    | ##################################### | 100%
python_abi-3.6          | 4 KB      | ##################################### | 100%
pip-21.3.1              | 1.2 MB    | ##################################### | 100%
setuptools-49.6.0       | 936 KB    | ##################################### | 100%
libzlib-1.2.11          | 59 KB     | ##################################### | 100%
ca-certificates-2021    | 139 KB    | ##################################### | 100%
python-3.6.13           | 38.3 MB   | ##################################### | 100%
wheel-0.37.0            | 31 KB     | ##################################### | 100%
ncurses-6.2             | 985 KB    | ##################################### | 100%
libstdcxx-ng-11.2.0     | 4.2 MB    | ##################################### | 100%
zlib-1.2.11             | 86 KB     | ##################################### | 100%
readline-8.1            | 295 KB    | ##################################### | 100%
libffi-3.4.2            | 57 KB     | ##################################### | 100%
libgcc-ng-11.2.0        | 887 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate r-reticulate
#
# To deactivate an active environment, use
#
#     $ conda deactivate

* Miniconda has been successfully installed at '/home/administrator/.local/share
/r-miniconda'.
>
```

et ensuite on peut commencer à faire le job:

```
> py_run_string("my_python_array=np.array([2,4,6,8])")
> my_python_array
Error: object 'my_python_array' not found
> print(my_python_array)
Error in print(my_python_array) : object 'my_python_array' not found
> py_run_string("print(my_python_array)")
[2 4 6 8]
>
```

R Markdown

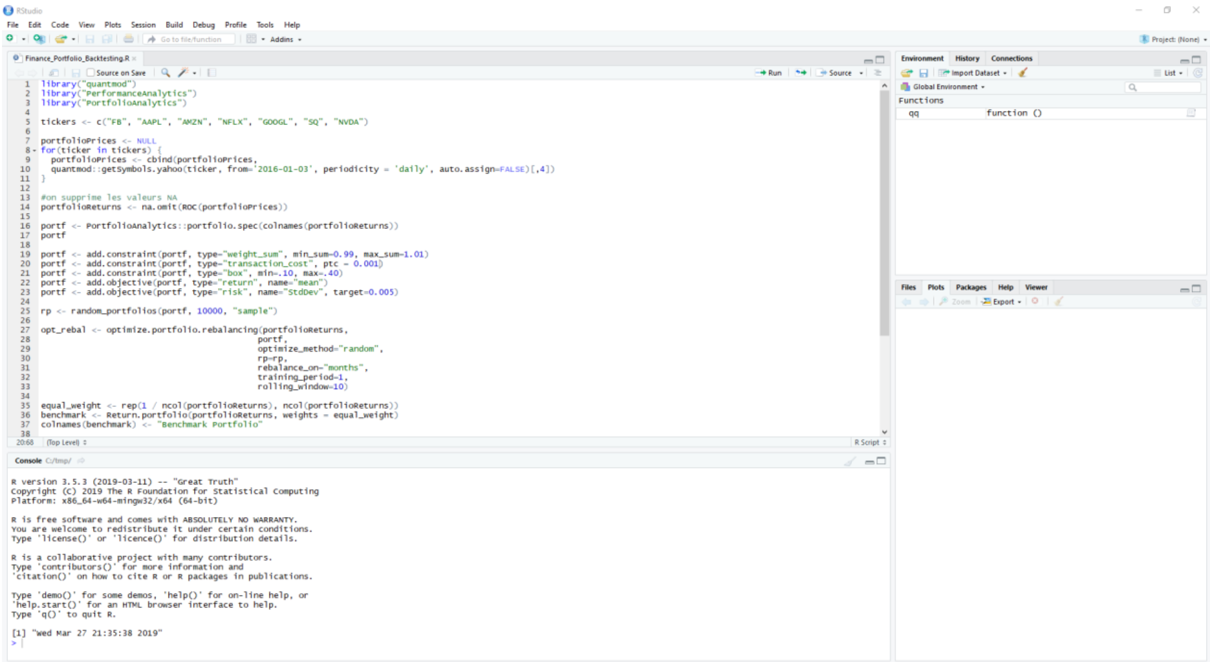
Ceux qui codent en Python et qui connaissent JupyterLab savent combien il est agréable d'écrire une ligne de code et avoir son résultat (nombre, vecteur, matrice, data frame ou graphique) juste en-dessous, et pouvoir poursuivre l'écriture du code en-dessous de l'objet précédemment obtenu et toujours avoir ainsi une vue d'ensemble sur:

1. Une ligne de code (input)

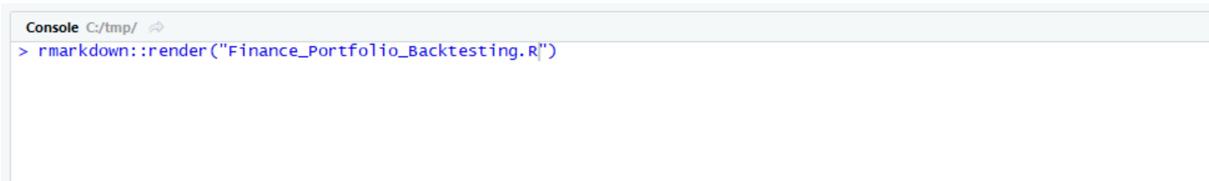
2. Une sortie (output) correspondante

Et finalement pouvoir exporter le résultat en HTML ou PDF de qualité tout à fait acceptable en termes de résolution et d'esthétique.

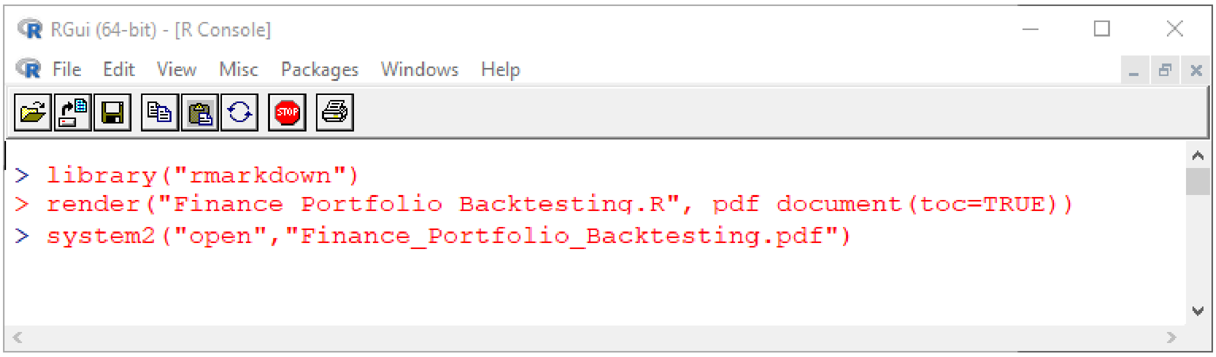
Comme l'utilisation de R Markdown est beaucoup plus simple dans R Studio, considérons donc le cas suivant:



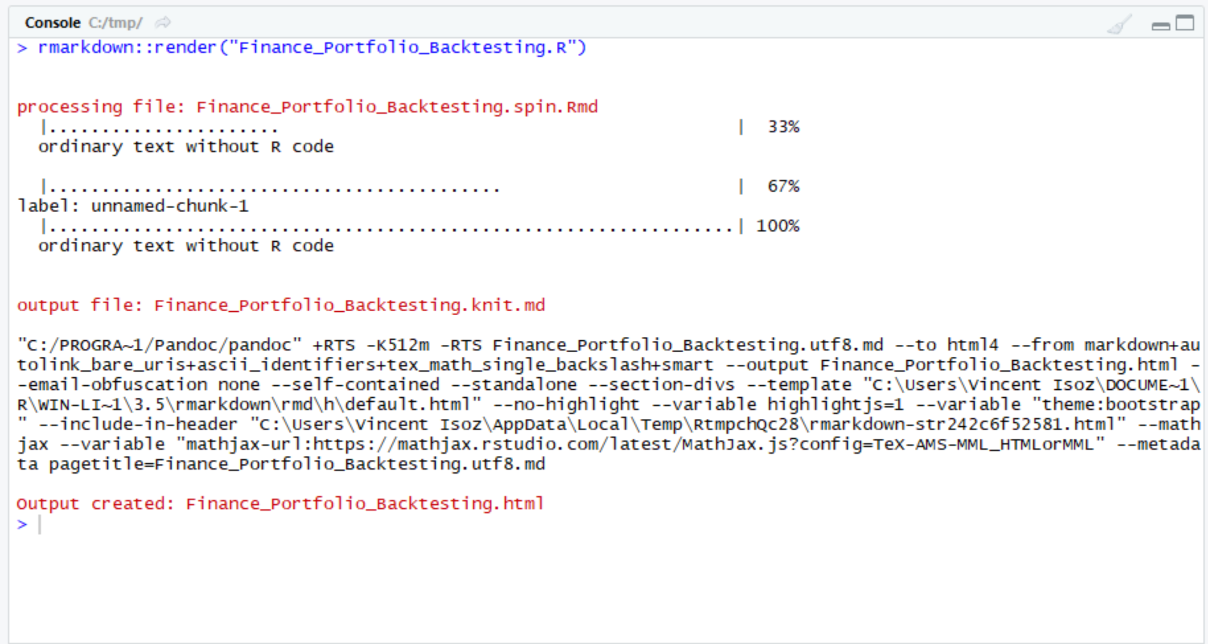
Nous souhaiterions que le script visible soit affiché en tant qu'output R Markdown, pour cela nous écrivons soit dans la console, soit à la fin de notre script:



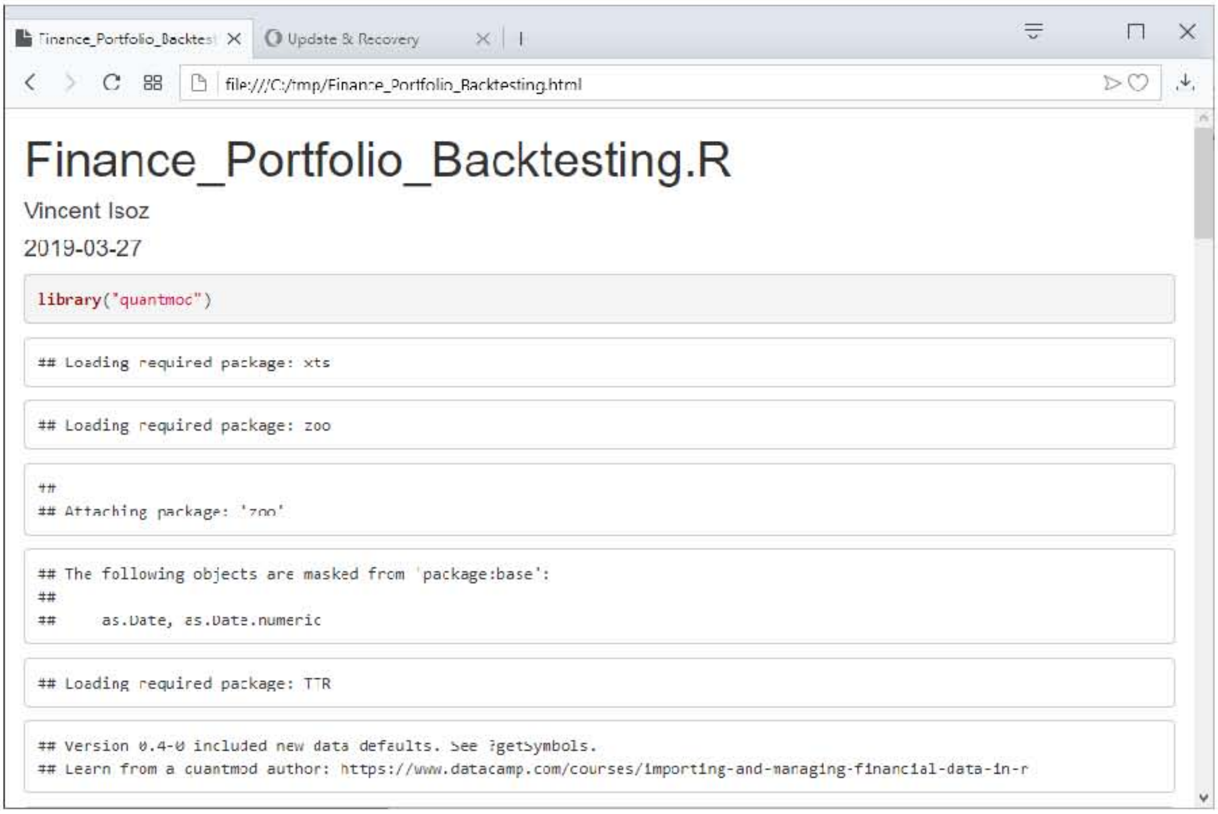
Sinon, dans la console R, nous pouvons mettre aussi:



Et une fois exécuté, nous obtenons:



Ce qui donnera en ouvrant le fichier HTML:



Finance_Portfolio_Backtesting.R

Vincent Isoz

2019-03-27

```
library("quantmod")
```

```
## Loading required package: xts
```

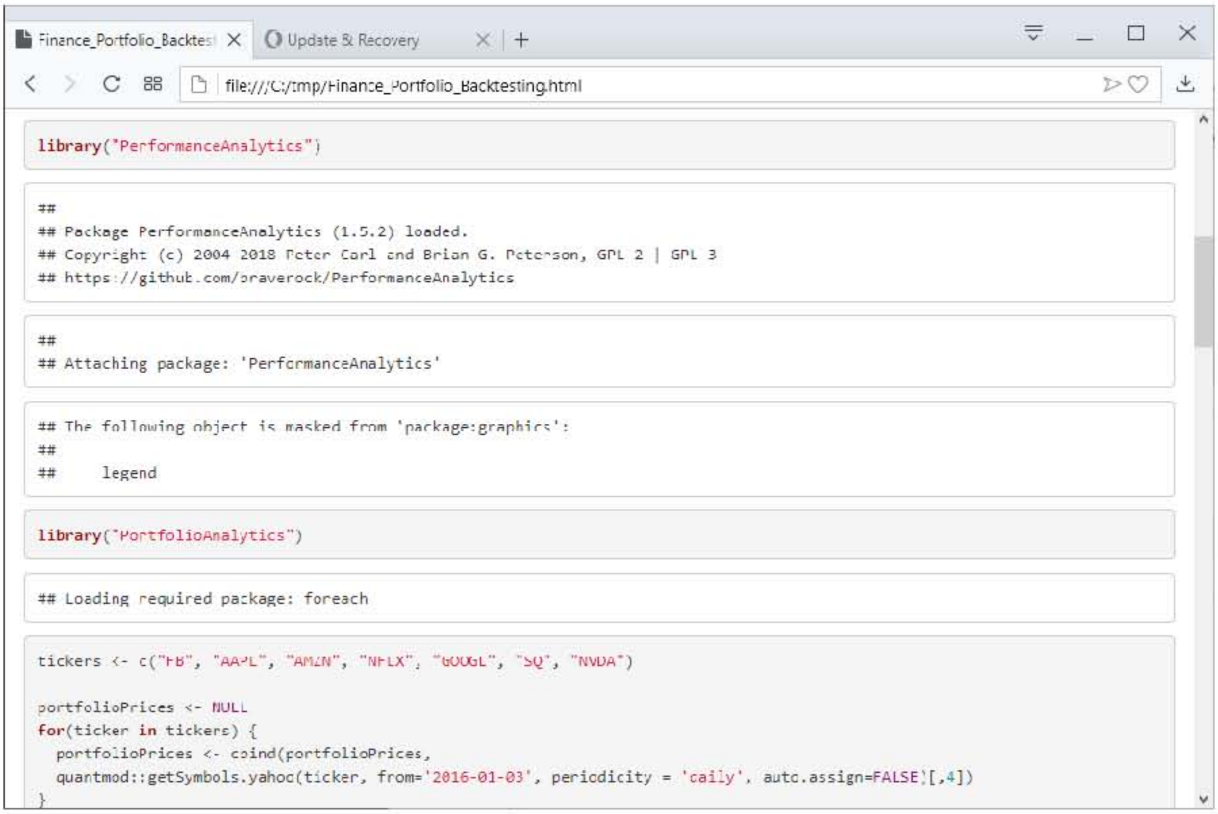
```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
## Learn from a quantmod author: https://www.datacamp.com/courses/importing-and-managing-financial-data-in-r
```



```
library("PerformanceAnalytics")
```

```
##
## Package PerformanceAnalytics (1.5.2) loaded.
## Copyright (c) 2004 2018 Peter Carl and Brian G. Peterson, GPL 2 | GPL 3
## https://github.com/braverock/PerformanceAnalytics
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
library("PortfolioAnalytics")
```

```
## Loading required package: foreach
```

```
tickers <- c("FB", "AAPL", "AMZN", "NFLX", "GOOGL", "SQ", "NVDA")

portfolioPrices <- NULL
for(ticker in tickers) {
  portfolioPrices <- cbind(portfolioPrices,
  quantmod::getSymbols.yahoo(ticker, from='2016-01-03', periodicity = 'daily', auto.assign=FALSE)[,4])
}
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
#on supprime les valeurs NA
portfolioReturns <- na.omit(ROC(portfolioPrices))

portf <- PortfolioAnalytics::portfolio.spec(colnames(portfolioReturns))
portf
```

```
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## PortfolioAnalytics::portfolio.spec(assets = colnames(portfolioReturns))
##
## Number of assets: 7
## Asset Names
## [1] "FB.Close"    "AAPL.Close"  "AMZN.Close"  "NFLX.Close"  "GOOGL.Close"
## [6] "SQ.Close"    "NVDA.Close"
```

```
portf <- add.constraint(portf, type="weight_sum", min_sum=0.99, max_sum=1.01)
portf <- add.constraint(portf, type="transaction_cost", ptc = 0.001)
```

```
portf <- add.constraint(portf, type="box", min=.10, max=.40)
portf <- add.objective(portf, type="return", name="mean")
portf <- add.objective(portf, type="risk", name="StdDev", target=0.005)

rp <- random_portfolios(portf, 10000, "sample")

opt_rebal <- optimize.portfolio.rebalancing(portfolioReturns,
                                            portf,
                                            optimize_method="random",
                                            rp=rp,
                                            rebalance_on="months",
                                            training_period=1,
                                            rolling_window=10)
```

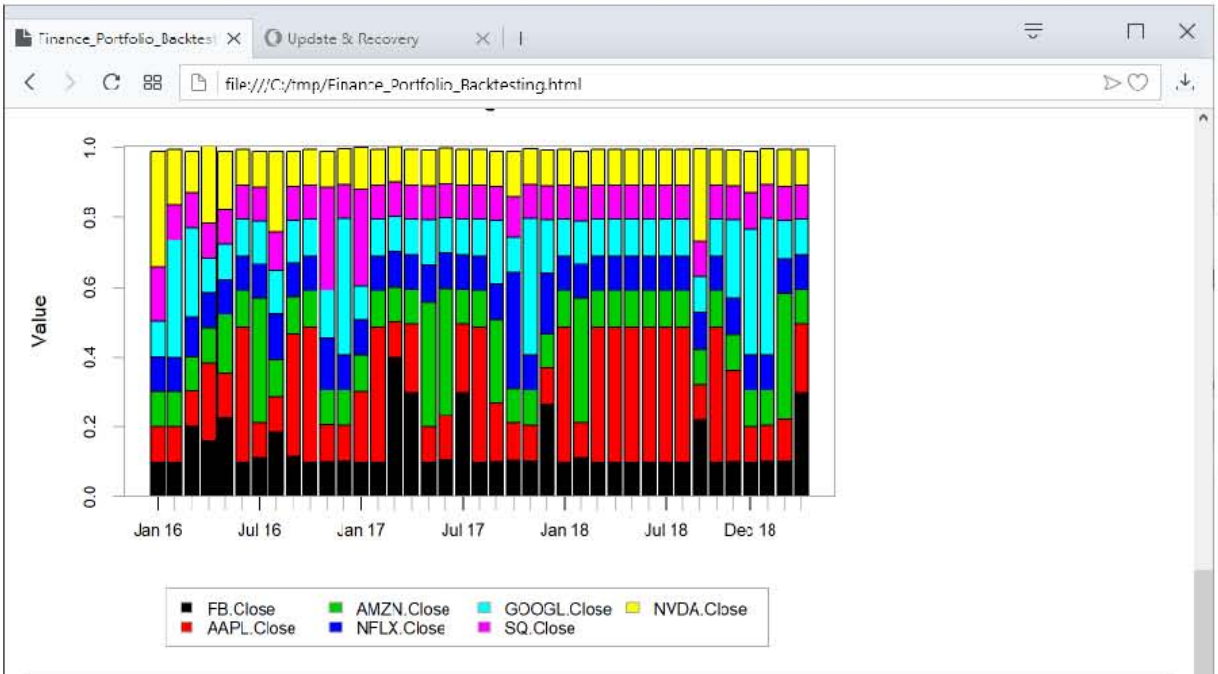
```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
equal_weight <- rep(1 / ncol(portfolioReturns), ncol(portfolioReturns))
benchmark <- Return.portfolio(portfolioReturns, weights = equal_weight)
colnames(benchmark) <- "Benchmark Portfolio"

sp500prices <- getSymbols.yahoo("SPY", from='2016-01-03', periodicity = 'daily', auto.assign=FALSE)[,4]
sp500Rets <- na.omit(ROC(sp500prices))
sp500Rets <- as.xts(sp500Rets)

chart.Weights(opt_rebal, main="Rebalanced Weights Over Time")
```

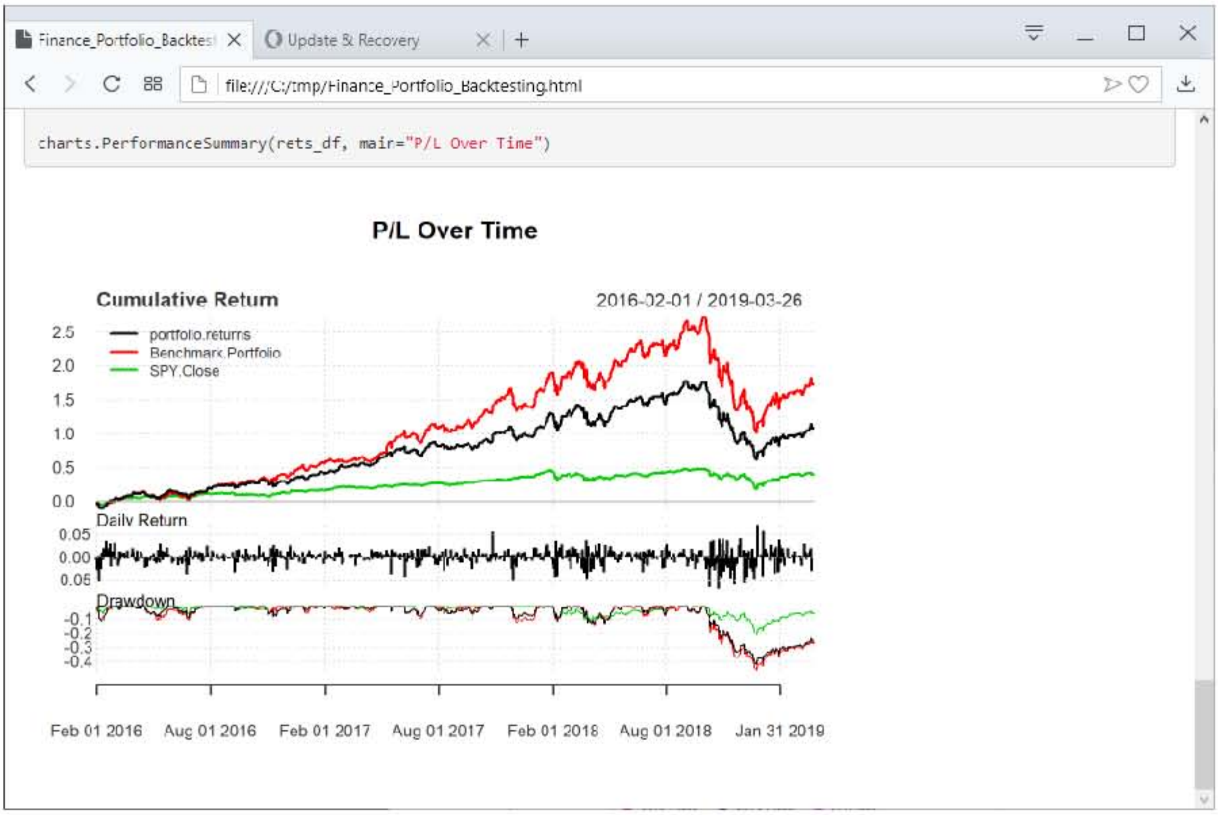
Rebalanced Weights Over Time




```
rebal_weights <-extractWeights(opt_rebal)
rebal_returns <- Return.portfolio(portfolioReturns, weights=rebal_weights)

rets_df <- cbind(rebal_returns, benchmark, sp500Rets)
```

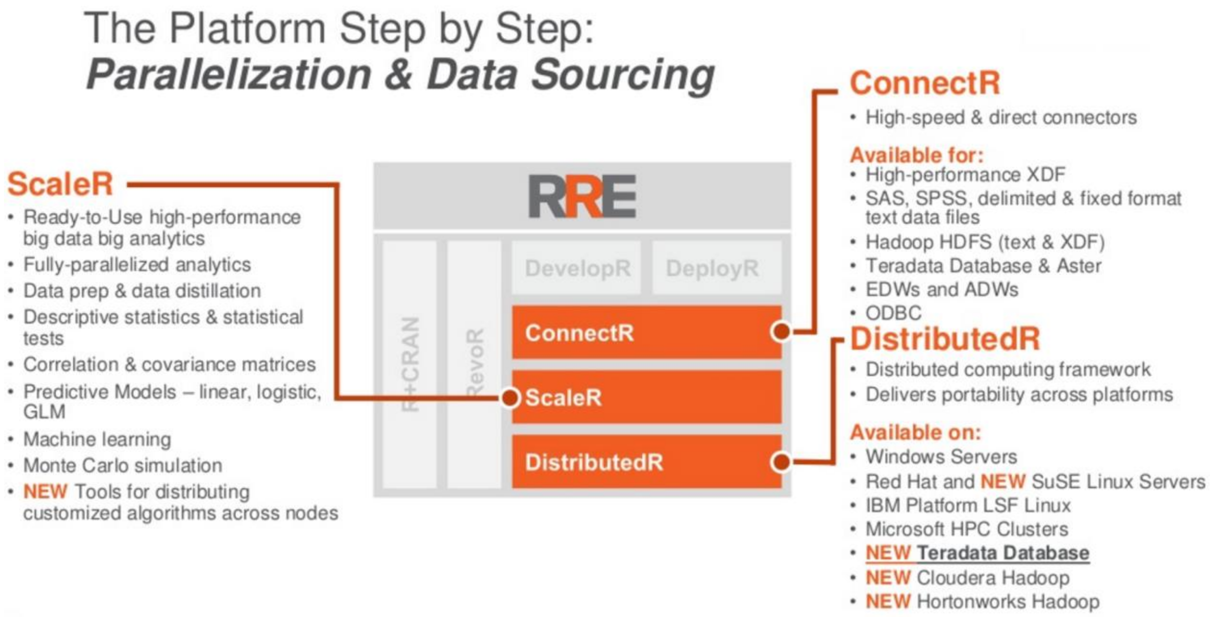
```
charts.PerformanceSummary(rets_df, main="P/L Over Time")
```



38. Revolution R (Microsoft R Open)

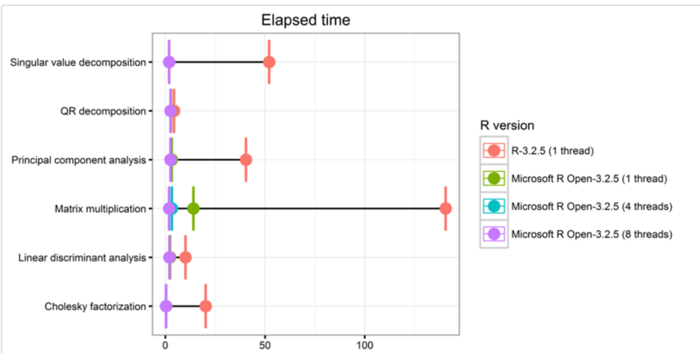
Revolution R Open qui est sous licence GPLv2 est la version "Entreprise" et "Scientifique" de R pour Mac, Windows et Linux dans le sens que les commandes sont reproductibles à l'aide d'un jeu de commande particulier (tout en restant cependant compatible avec TOUS les packages du CRAN et RStudio), que certaines fonctions ont été reprogrammées et que le logiciel gère automatiquement de gros jeux de données en utilisant le disque dur plutôt que la mémoire vive (ce qui est largement suffisant lorsqu'on ne cherche pas à avoir des résultats en temps réel).

Pour reprendre une vieille diapo de chez Revolution R:



et un comparatif:

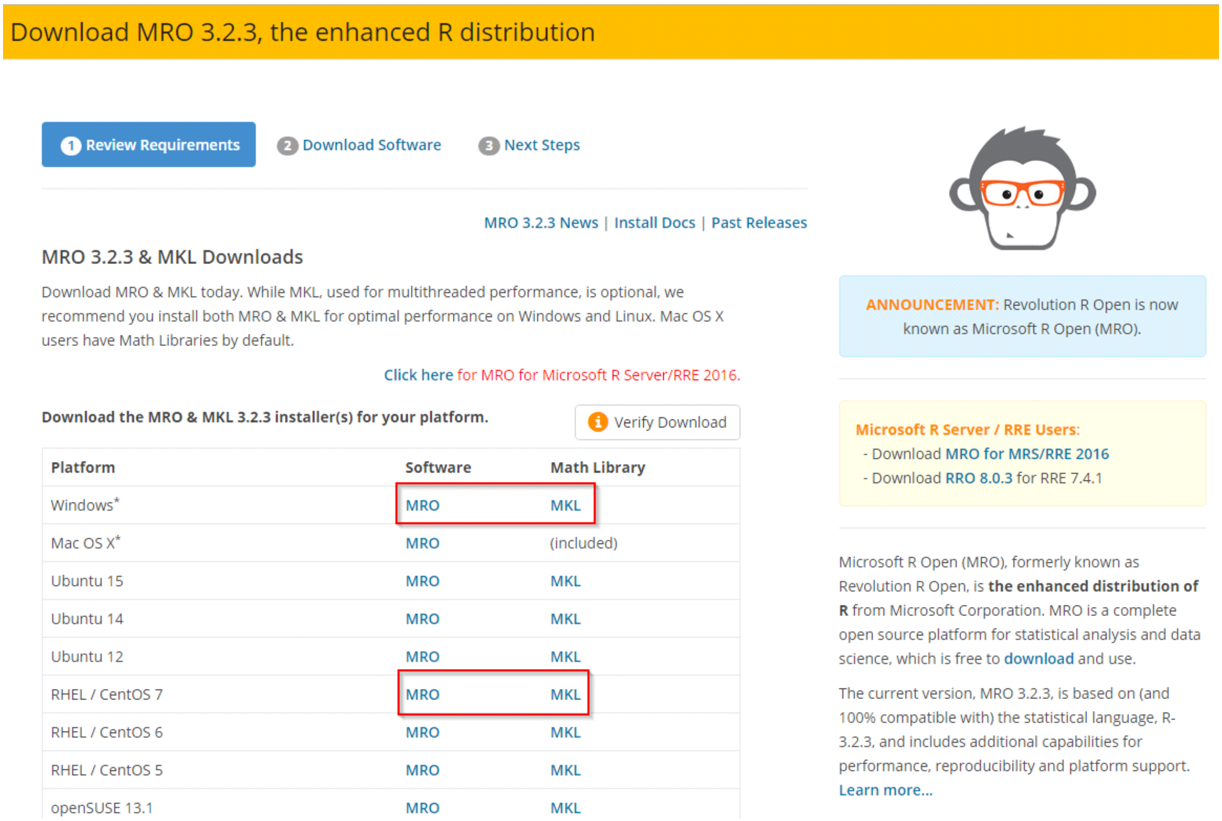
	R-3.2.5 (1 thread)	Microsoft R Open-3.2.5 (1 thread)	Microsoft R Open-3.2.5 (4 threads)	Microsoft R Open-3.2.5 (8 threads)
Matrix multiplication	140.63	14.11	3.27	1.89
Cholesky factorization	20.27	0.42	0.35	0.36
QR decomposition	4.36	2.78	2.59	2.63
Singular value decomposition	52.07	1.91	1.89	1.90
Principal component analysis	40.46	3.25	2.67	2.55
Linear discriminant analysis	10.14	2.36	2.04	2.03



Installation

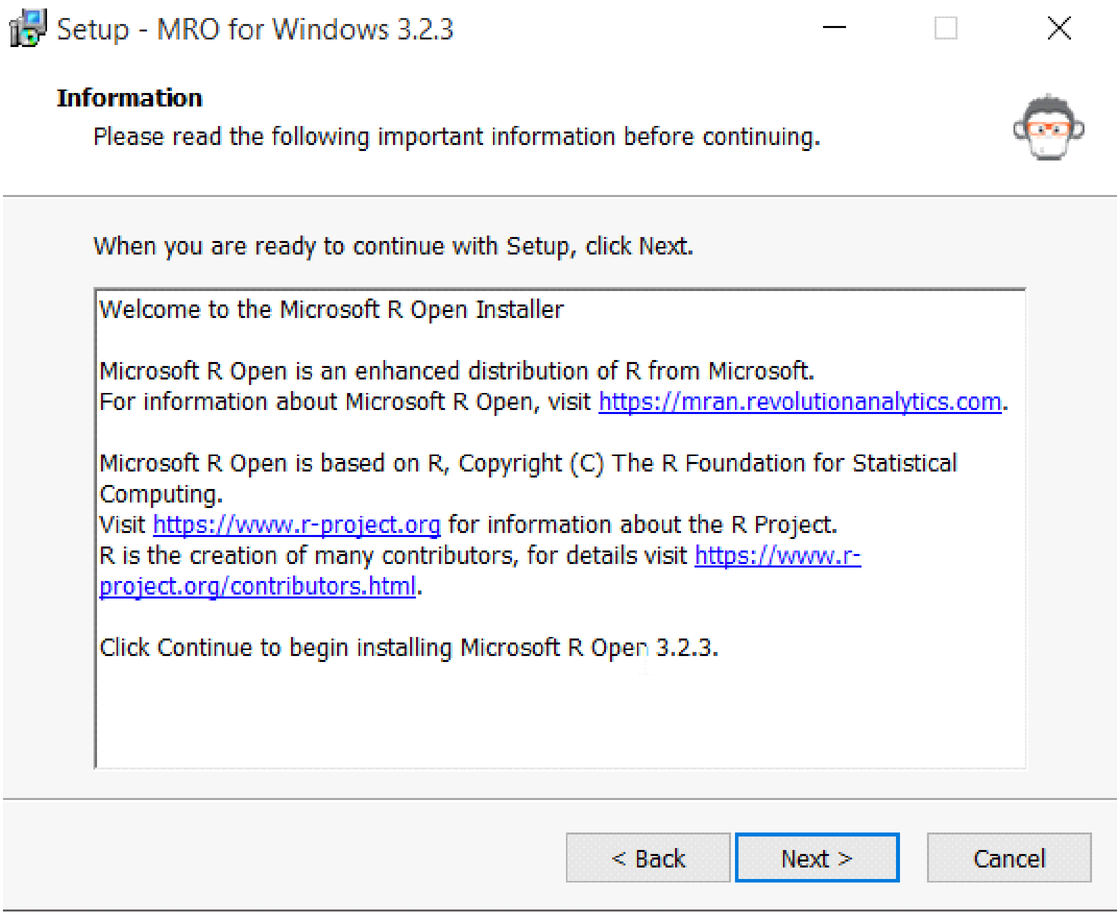
Avant tout il faut faire bien attention à installer les deux exécutables **MRO et MKL**:

https://mran.revolutionanalytics.com/download/

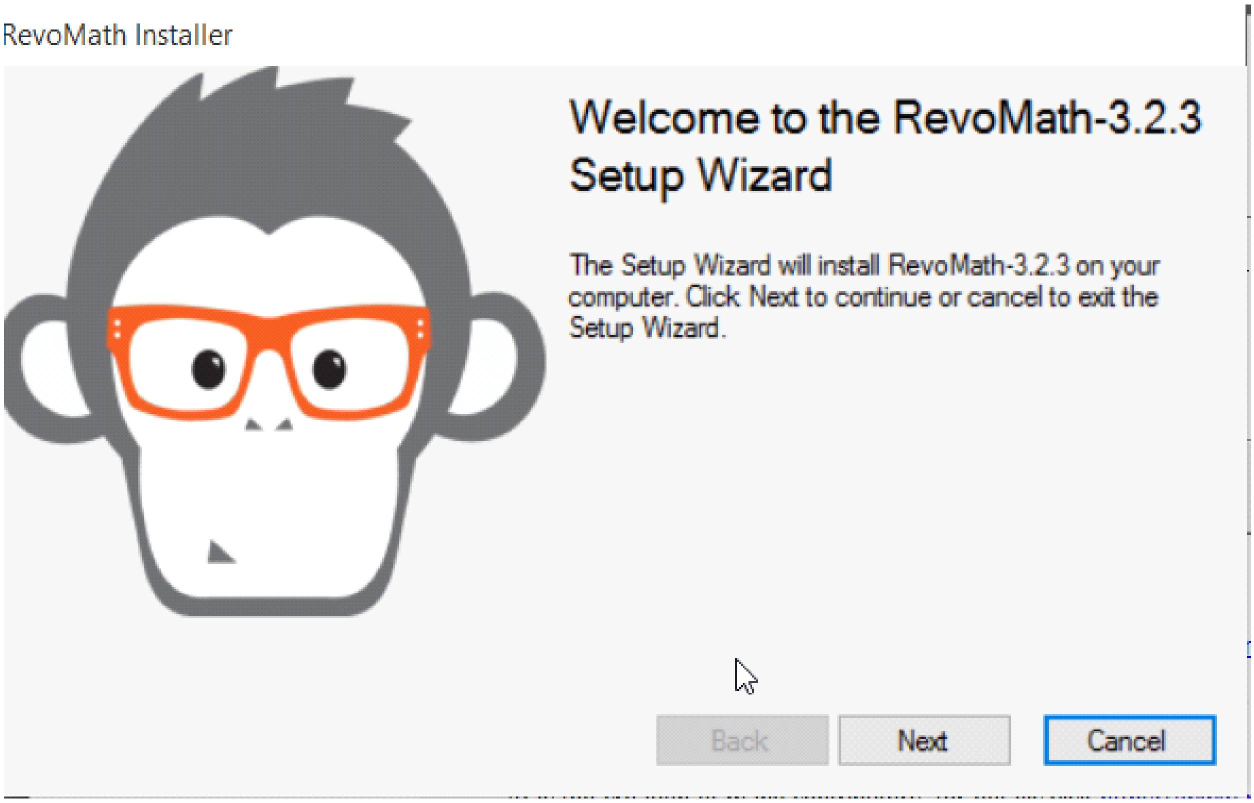


ATTENTION LE PROJET A ÉTÉ RETIRÉ EN JUIN 2023!!!!

L'installation de Microsoft R Open (MRO) est standard, nous l'avons déjà traitée au début de cet ouvrage:



Ensuite l'installation de RevoMath est toute aussi simple:

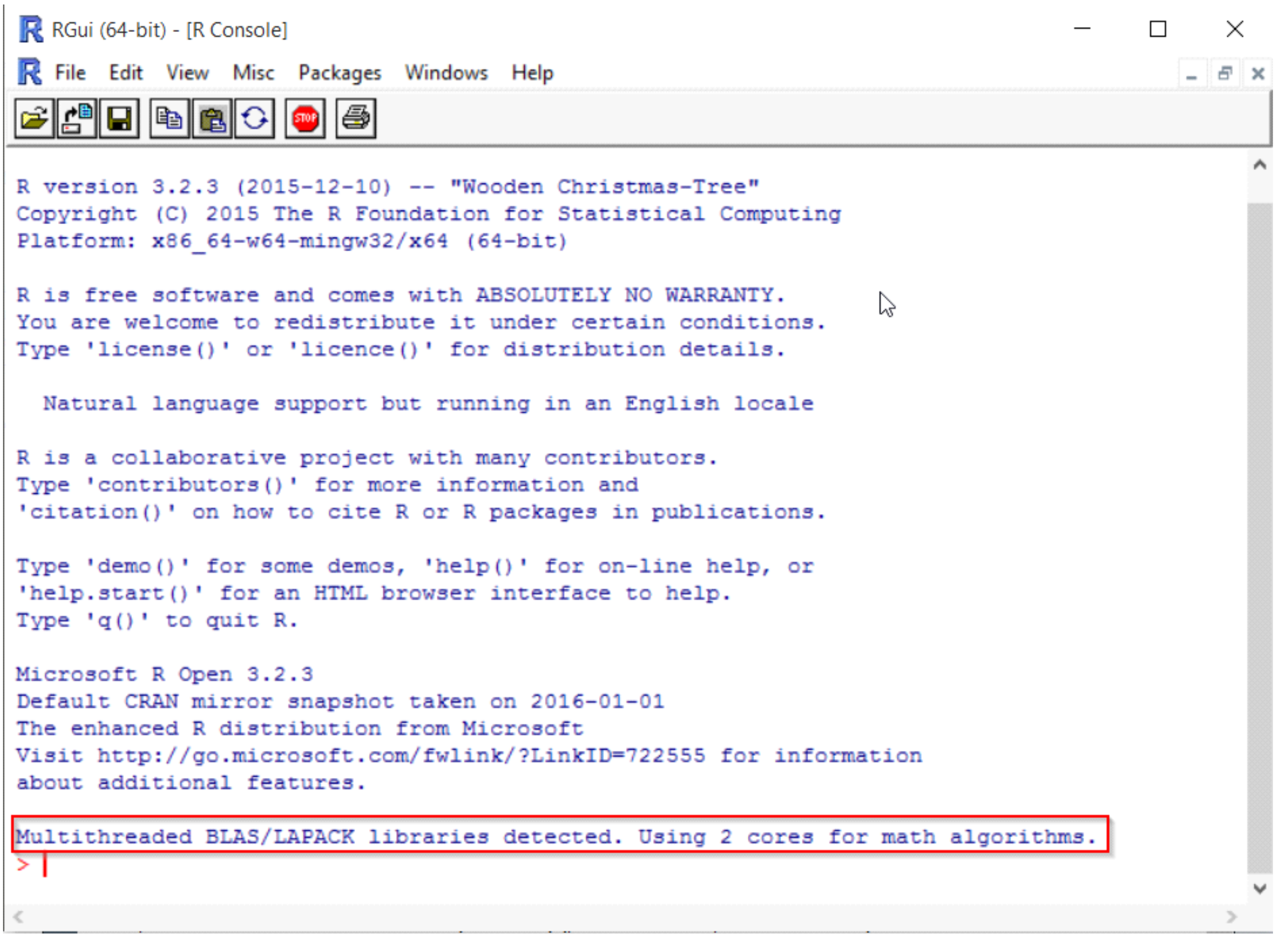


Remarque!!!: Sur certains environnements Windows 10 cette installation ne marche pas!!! Il faut alors aller chercher le fichier *.msi correspondant dans :

C:\Users\XXXXX\AppData\Local\Temp

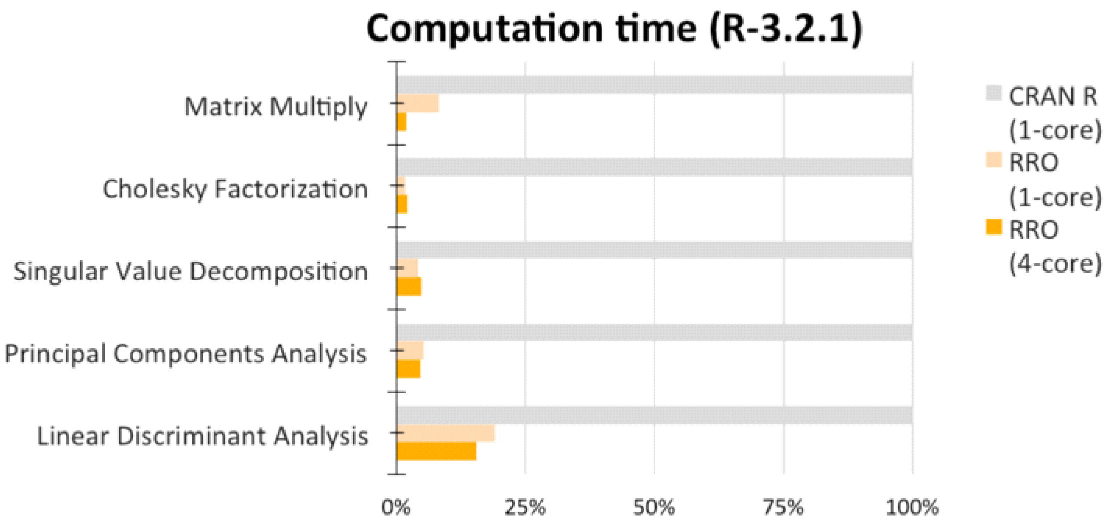
et directement installer depuis là…

Si l'installation se passe correctement, au lancement de MRO vous aurez:



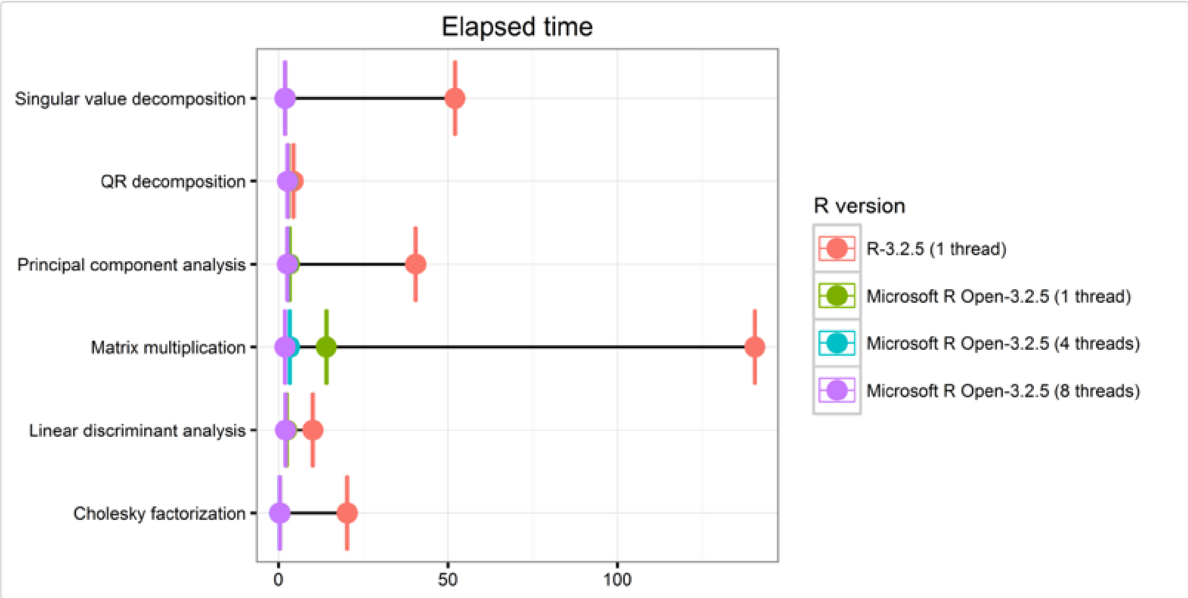
Différenciation en performance calculatoire

Selon Microsoft (Revolution R) nous avons au niveau des performances les comparaisons suivantes:

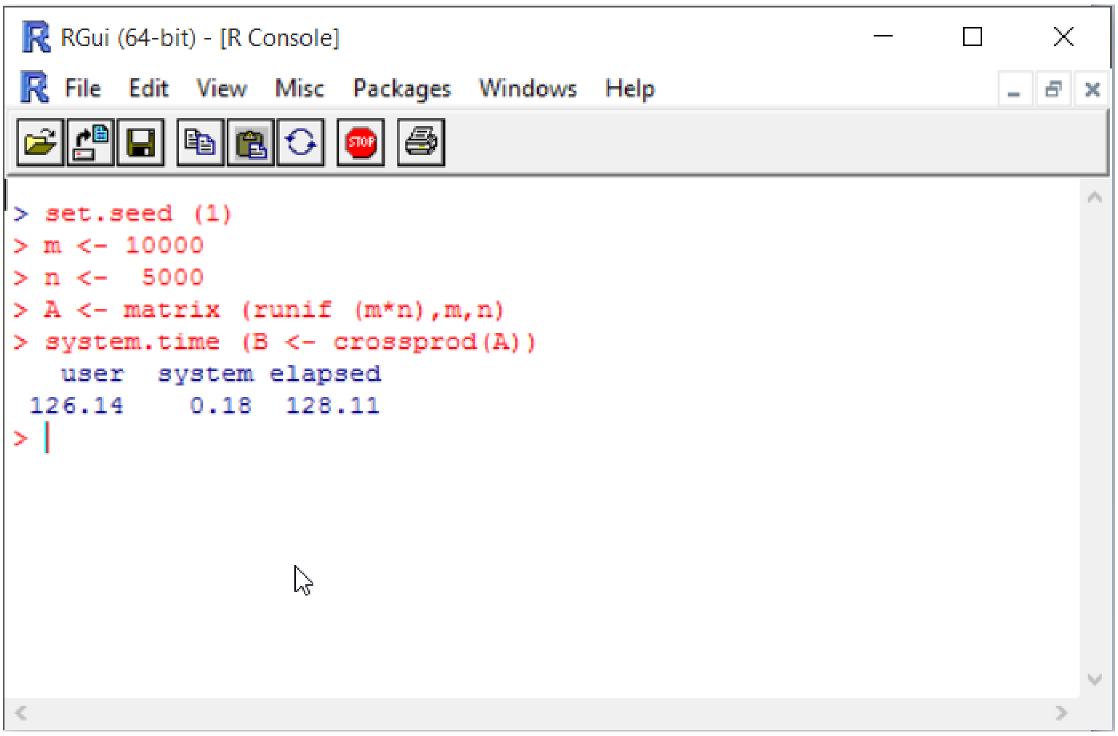


Ou encore:

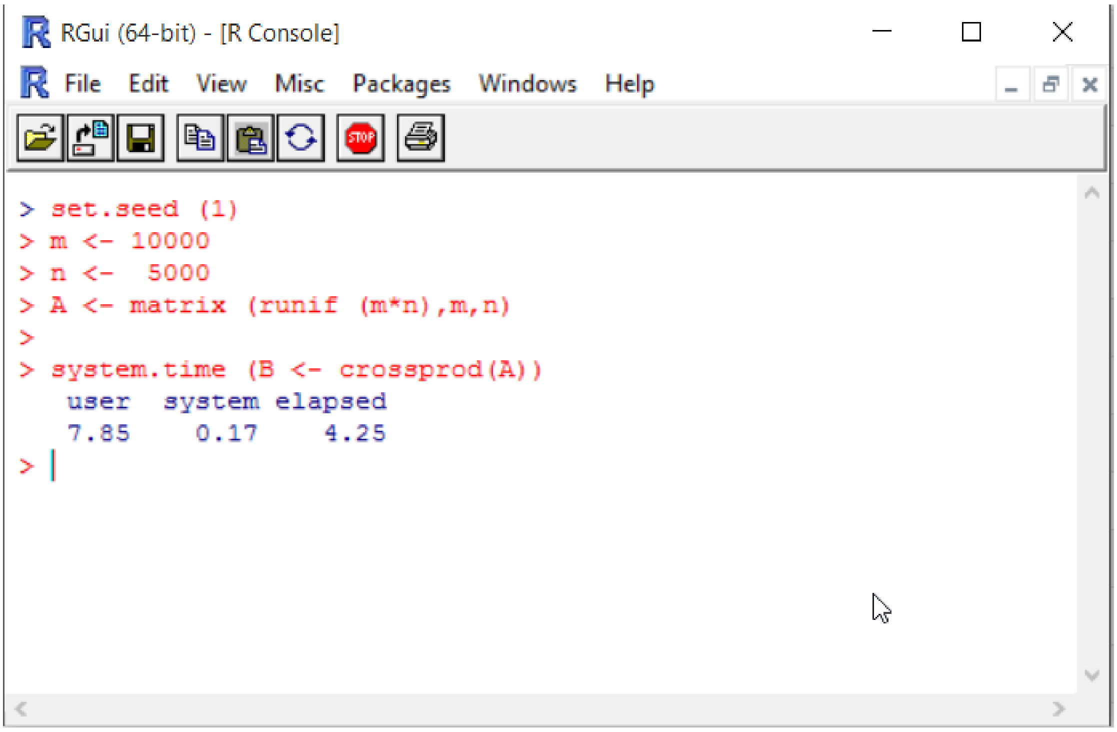
	R-3.2.5 (1 thread)	Microsoft R Open-3.2.5 (1 thread)	Microsoft R Open-3.2.5 (4 threads)	Microsoft R Open-3.2.5 (8 threads)
Matrix multiplication	140.63	14.11	3.27	1.89
Cholesky factorization	20.27	0.42	0.35	0.36
QR decomposition	4.36	2.78	2.59	2.63
Singular value decomposition	52.07	1.91	1.89	1.90
Principal component analysis	40.46	3.25	2.67	2.55
Linear discriminant analysis	10.14	2.36	2.04	2.03



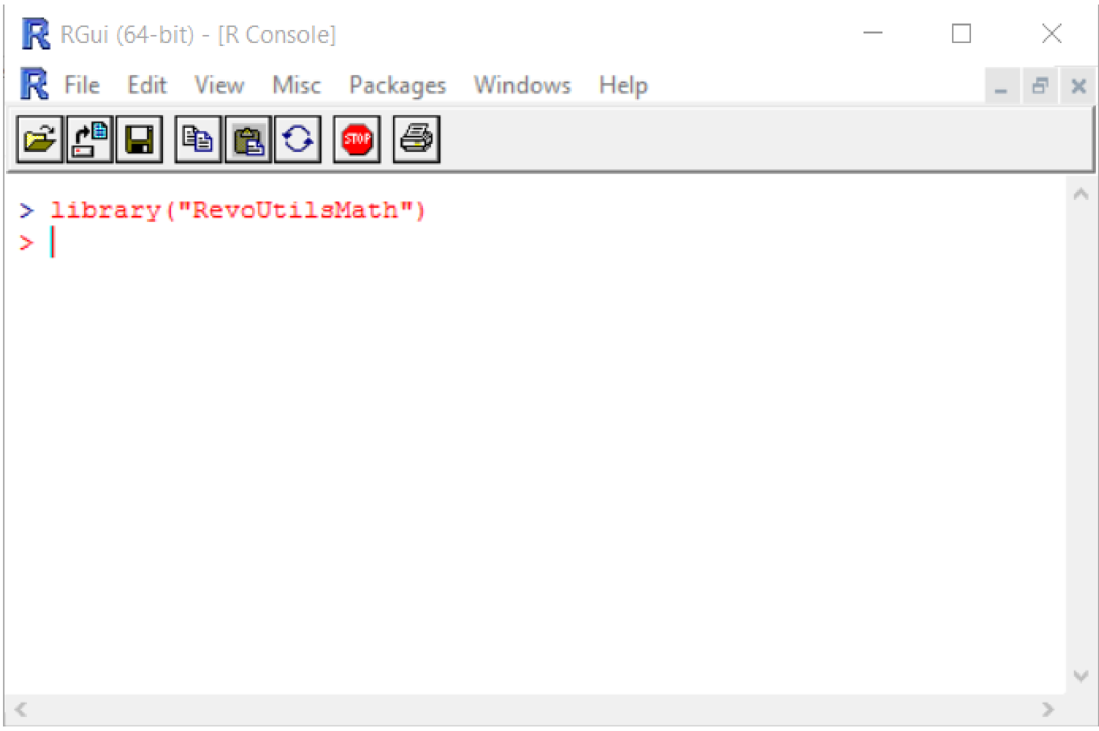
Voyons un cas pratique:



et le même calcul avec MRO:



Par sécurité et par rigueur scientifique je recommande toutefois de charger le package **RevoUtilsMaths** quand même avant:



Simon Urbanek, contributeur au noyau de R, a publié un code fameux qui permet de comparer la performance (code disponible ici: http://r.research.att.com/benchmarks/). Les résultats ont donné:

	R-3.2.1	RRO-3.2.1 (4 cores)	Performance gain
I. Matrix calculation			
Create, transpose and deform matrix	1.19	1.19	0.0
Matrix computation	0.49	0.50	0.0
Sort random values	0.89	0.89	0.0
Cross product	14.92	0.37	39.7
Linear regression	6.97	0.15	46.5
II. Matrix functions			
Fast Fourier Transform	0.42	0.35	0.2
Compute eigenvalues	0.94	0.37	1.5
Calculate determinant	3.70	0.15	24.2
Cholesky decomposition	5.63	0.14	39.2
Matrix inverse	2.48	0.13	17.6
III. Programmation			
Vector calculation	0.69	0.65	0.1
Matrix calculation	0.25	0.26	0.0
Recursion	0.46	0.43	0.1
Loops	0.39	0.40	0.0
Mixed control flow	0.41	0.36	0.1
Total test time	201.16	44.22	3.5

Selon Revolution R, leurs clients constatent un accroissement de performance de 3 à 50 entre la version CRAN de R et la version de MRAN de R et ce sans changer la moindre ligne de code.

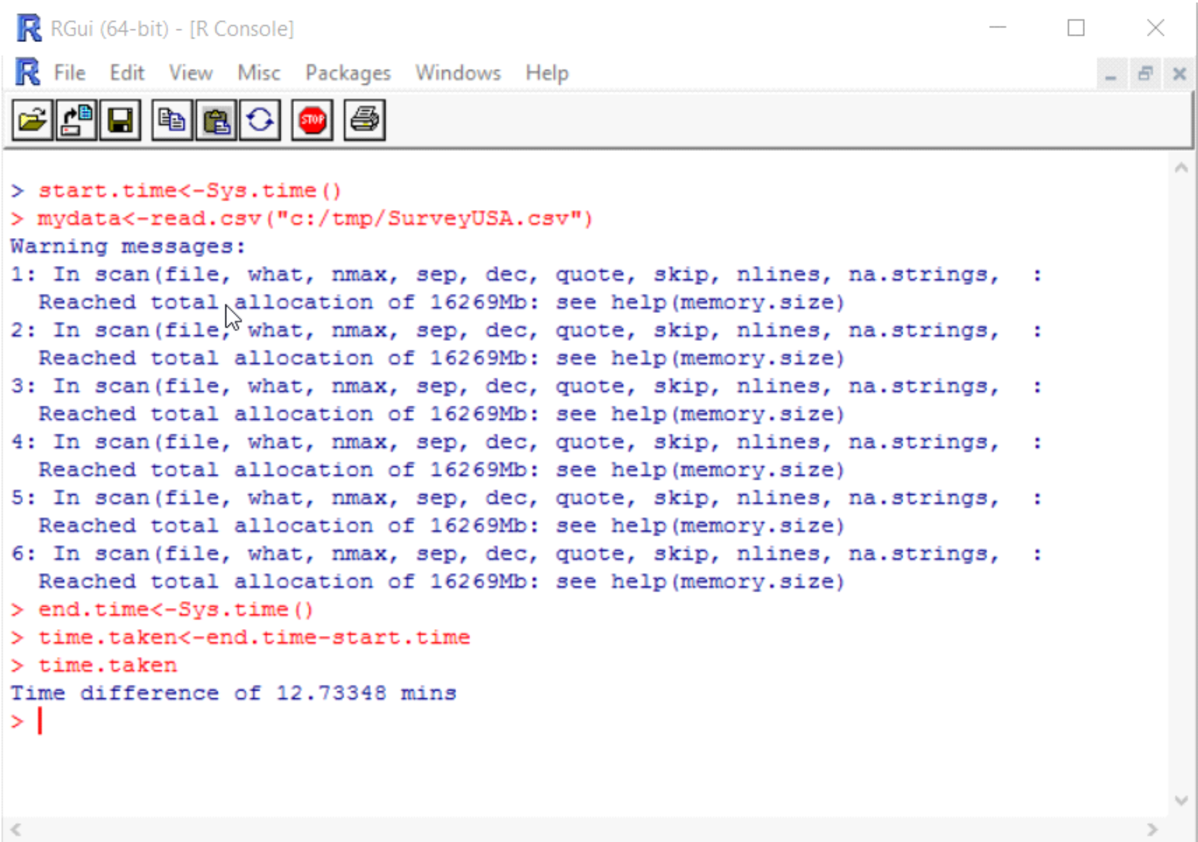
Attention! Pas de conclusions hâtives. Actuellement cette conclusion ne s'applique pas au data mash up et data munging puisque cela n'implique pas d'algèbre linéaire!

Différenciation chargement de gros jeux de données

Commençons par les gros jeux de données:



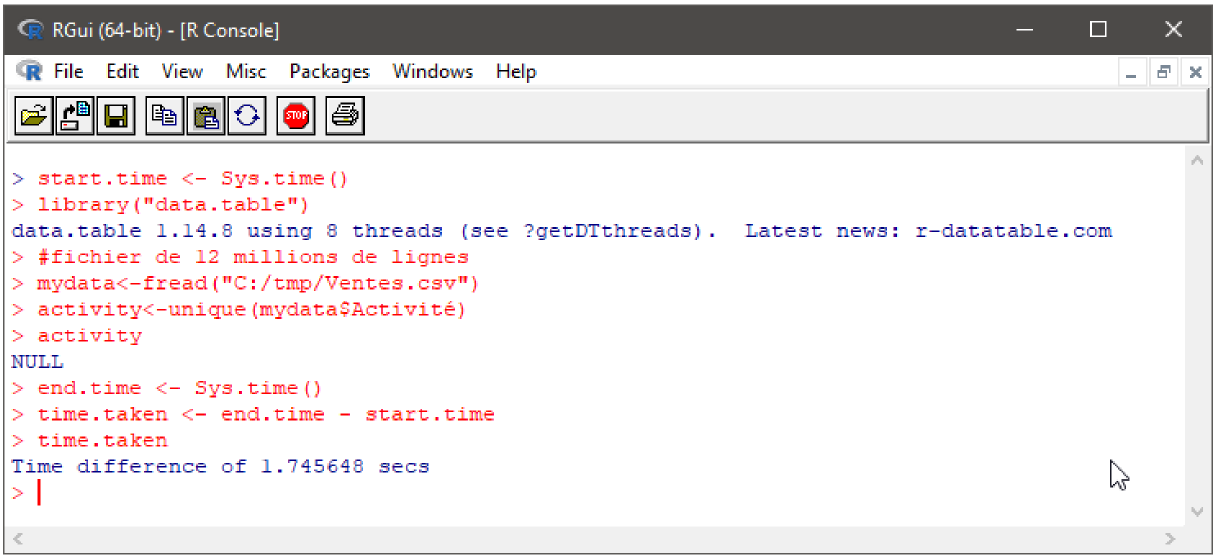
Ce fichier nous le savons à une peine folle à être chargée dans une machine R3.2.1 x64 équipé de 16GB de RAM (Alienware 15):



Alors qu'avec Revolution R Open:

ToDo

Mais avec Data.Table sur un Alienware Area 51M boosté:



Alors qu'avec Microsoft R Open:



Comme quoi….

39. R Shiny/Flexdashboard

R Shiny est un outil permettant de construire des pages web interactives avec des statistiques produites par R et ce avec des options beaucoup plus puissantes et avancées que Tableau ou Microsoft Power BI sans avoir à connaître l'HTML, le JavaScript et le CSS pour les fonctionnalités de base.

Remarque: Notez que l'auteur principal de ce PDF préfère largement Microsoft Power BI à R Shiny ou Flexdashobard pour plusieurs milliers de raisons!

L'idée est que certains end-users n'aient pas à exécuter de script pour visualiser ou jouer avec différents paramètres statistiques et ce de façon interactive, dynamique et si possible ludique!

Déployer des apps Shiny peut se faire de trois manières:

- Installer le serveur Shiny (AGPLv3):

  https://github.com/rstudio/shiny-server

- Acheter un serveur Shiny Server "Pro" avec l'avantage d'avoir nativement des fonctions d'authentification sécurisées (avec ou sans LDAP), support SSL, de gestions et monitoring de la performance et bien évidemment d'avoir le support direct de R Studio, Inc.:

  https://www.rstudio.com/pricing

- Héberger ces applications dans le cloud Shiny chez rstudio.com

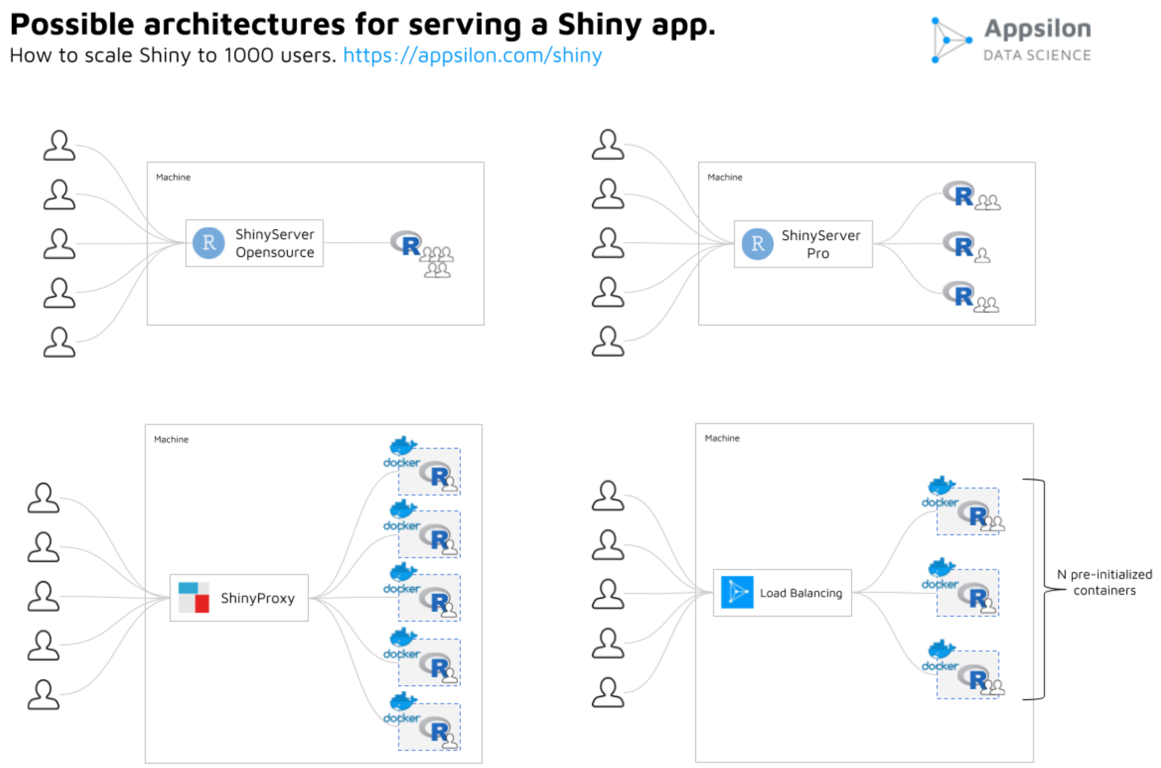
  http://www.shinyapps.io

- Installer et déployer Shiny Proxy (necessite Docker cependant)

  https://www.shinyproxy.io/

Pour les différences entre les solutions, n'hésitez pas à consulter:

https://appsilon.com/alternatives-to-scaling-shiny/



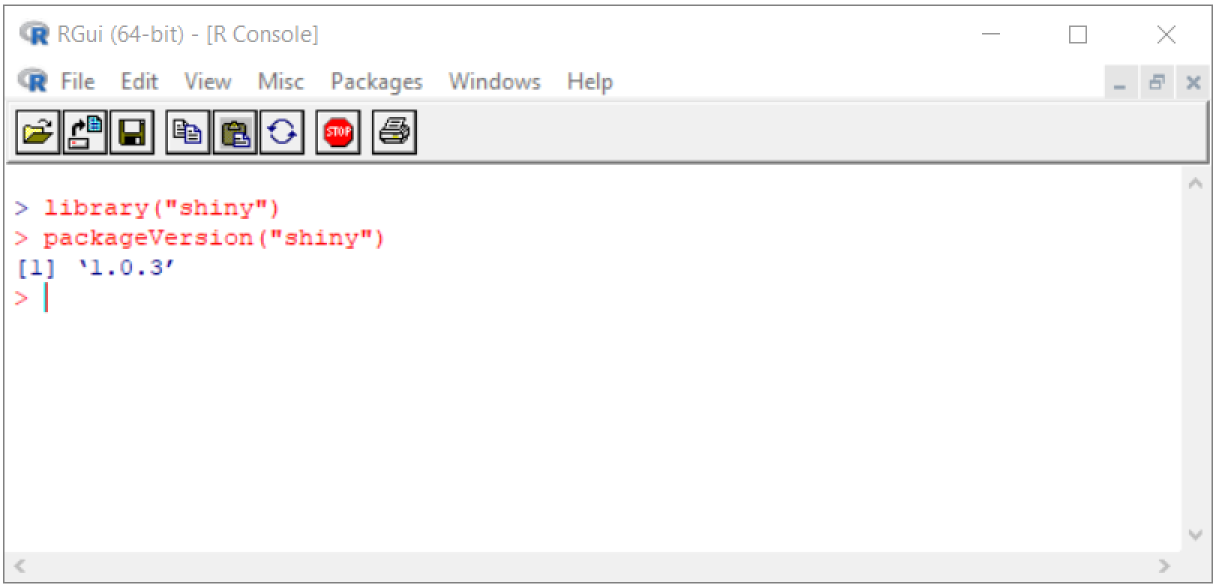
Par définition, une App Shiny est une page web (IU) connectée à un ordinateur/serveur exécutant une session R en live (serveur):



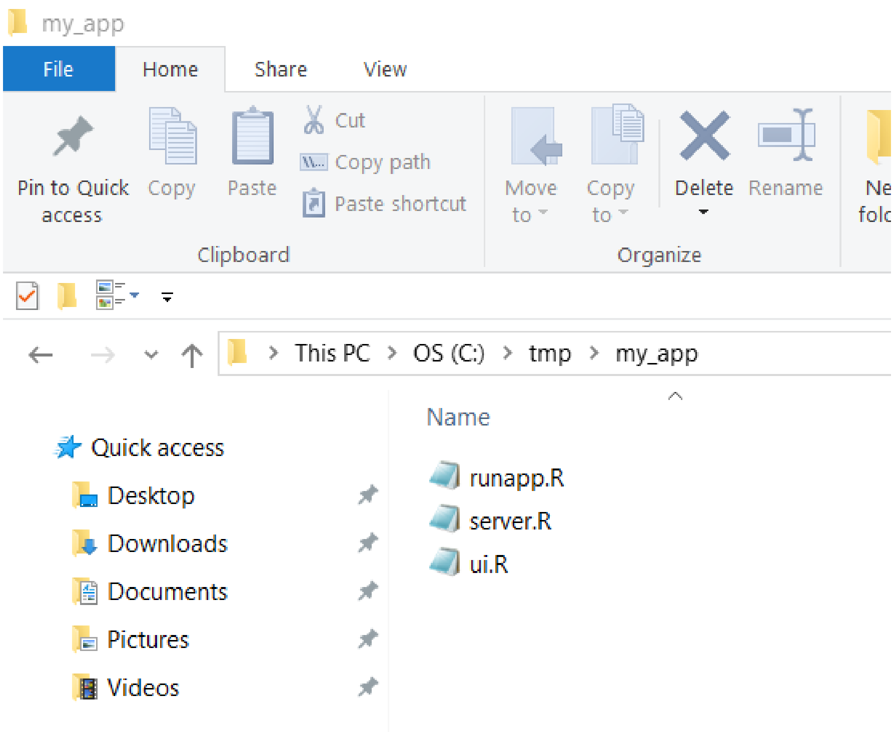
L'utilisateur peut manipuler l'IU ce qui va causer le serveur à mettre à jour l'UI en exécutant le code R correspondant.

Installer R Shiny

Pour installer le package c'est très simple comme à l'habitude (R 3.4.1) ensuite il suffit de charger le package:



Ensuite, considérons les trois fichiers suivants R (c'est une bonne pratique de séparer tout dans trois fichiers différents mais ce n'est pas obligatoire comme nous le verrons plus loin!):



Il faut savoir qu'il est à ce jour obligatoire pour Shiny d'avoir le fichier *ui.R* et *server.R* le (le troisième étant utilisé pour une astuce dont nous allons parler plus loin) qui ont respectivement le contenu suivant:

ui.R - Notepad

File Edit Format View Help

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

server.R - Notepad

File Edit Format View Help

```
library(shiny)

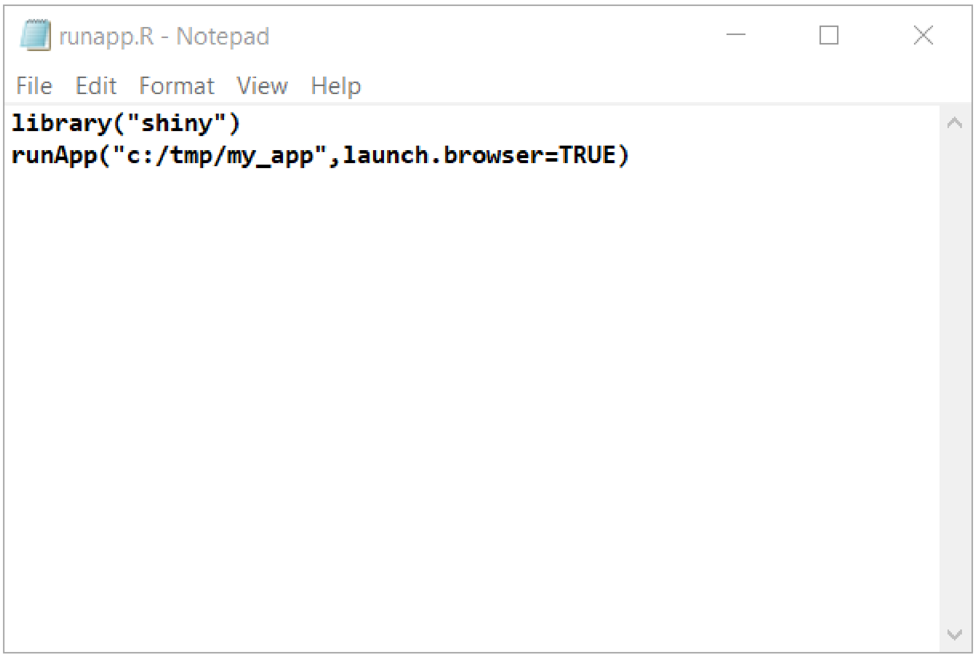
# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

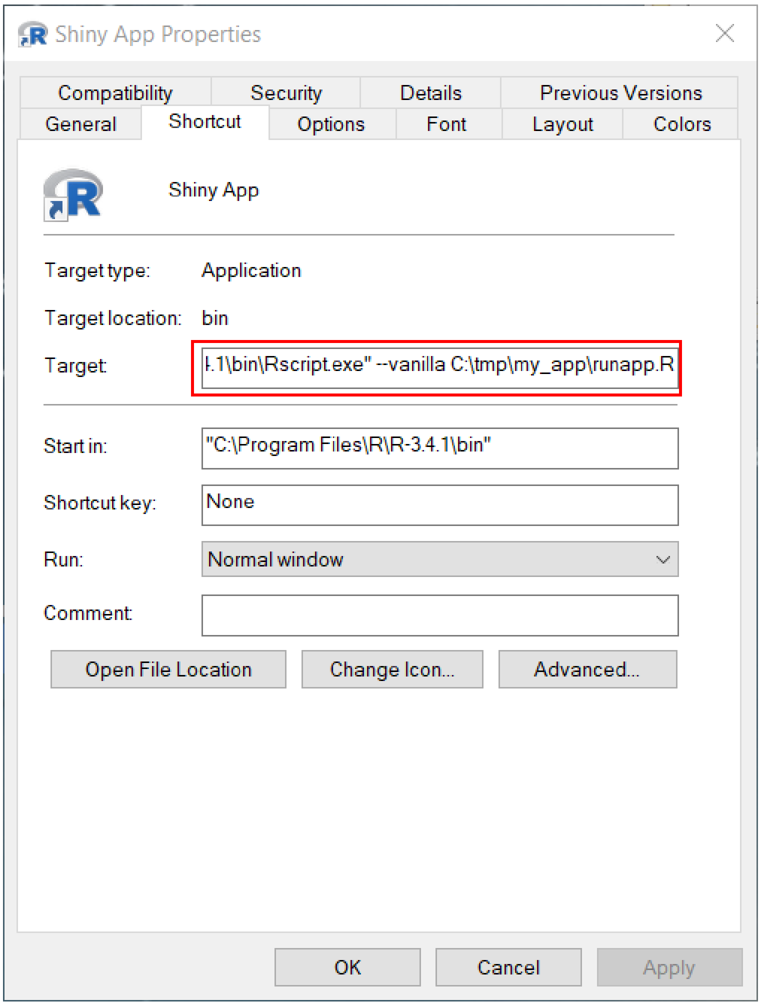
Et le dernier fichier n'est pas spécifique à Shiny, c'est juste une manière élégante pour un utilisateur de lancer une application shine à partir d'un raccourci Windows (et peu importe le nom de fichier contrairement aux deux précédents!):



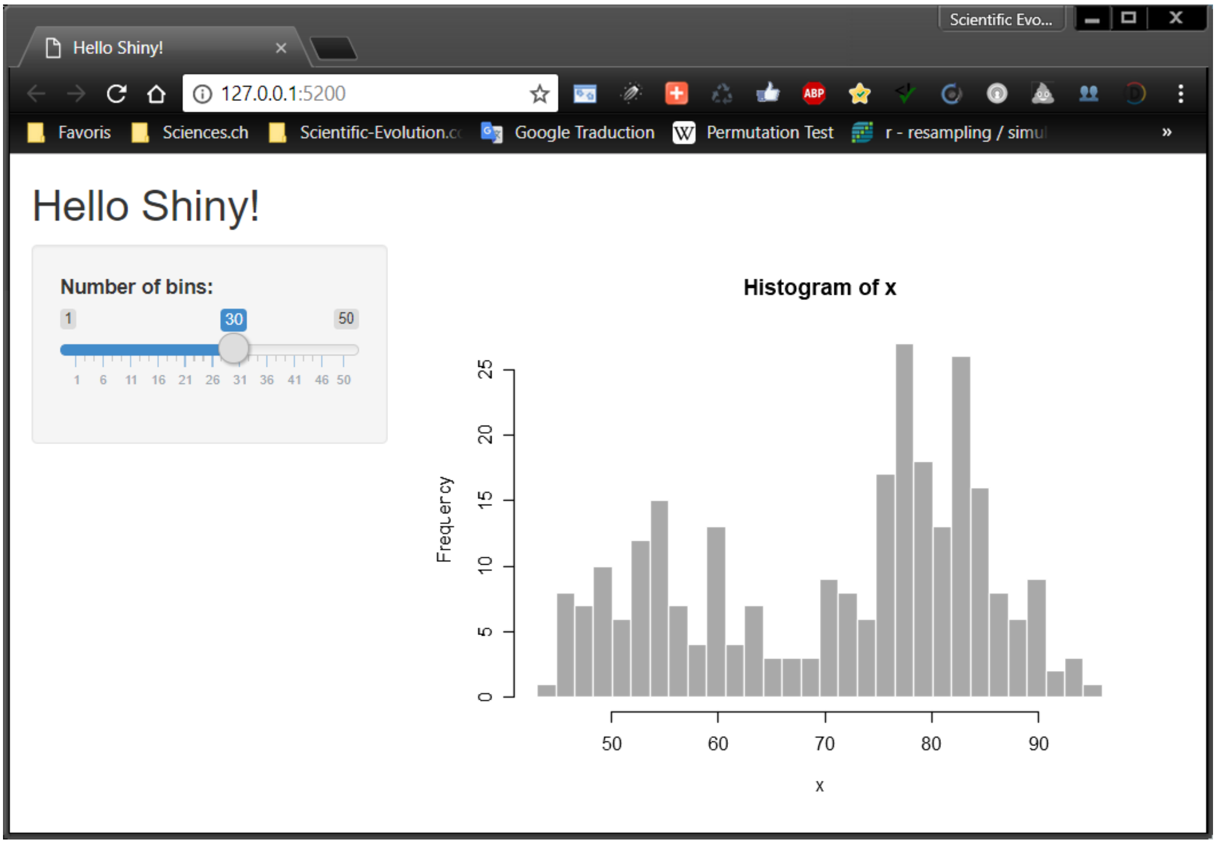
Et enfin on construit sur le bureau de Windows le raccourci suivant:



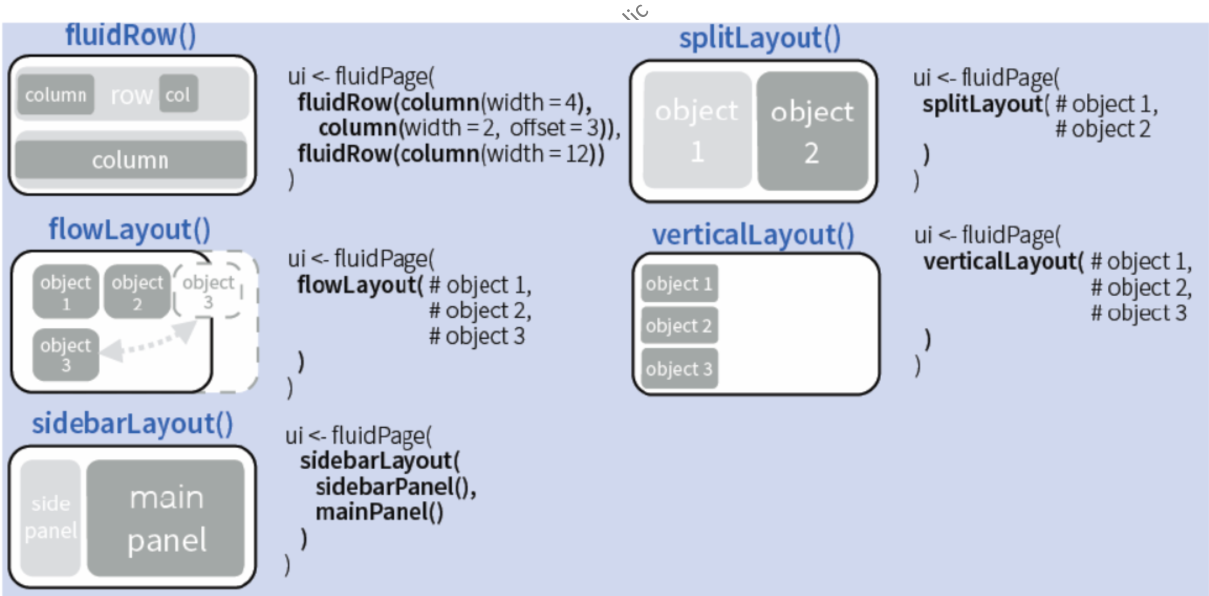
Avec les propriétés:



Après un double clic sur le raccourci, nous obtenons le résultat tant attendu (!):



Signalons que différentes structures de panneaux sont disponibles:



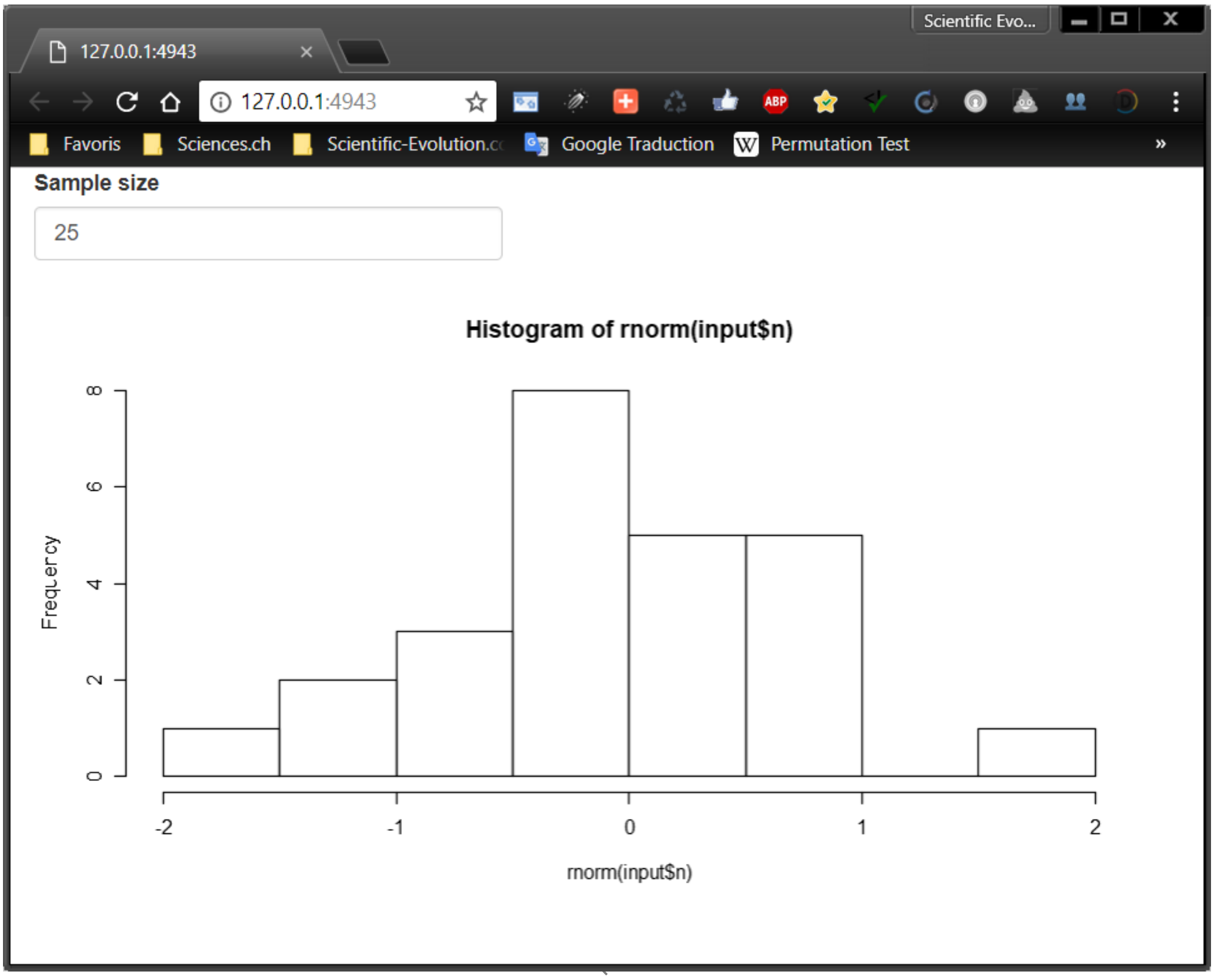
On a aussi les structures tabulaires suivantes sinon:



Sinon voici un petit exemple où tout est écrit et exécuté d'une seule traite à l'aide de la commande **shinyApp( )** :



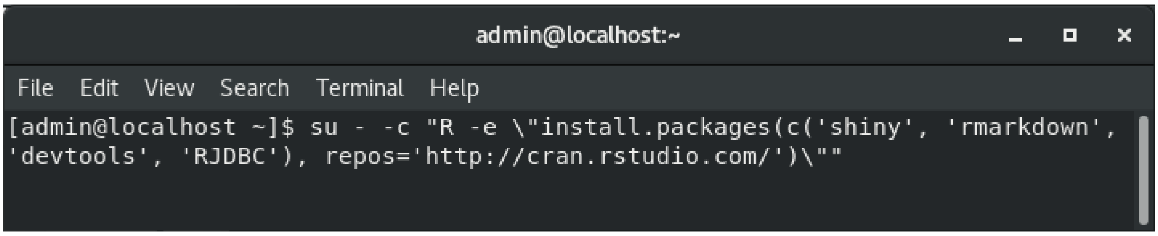
Ce qui donne:



Installer R Shiny Server

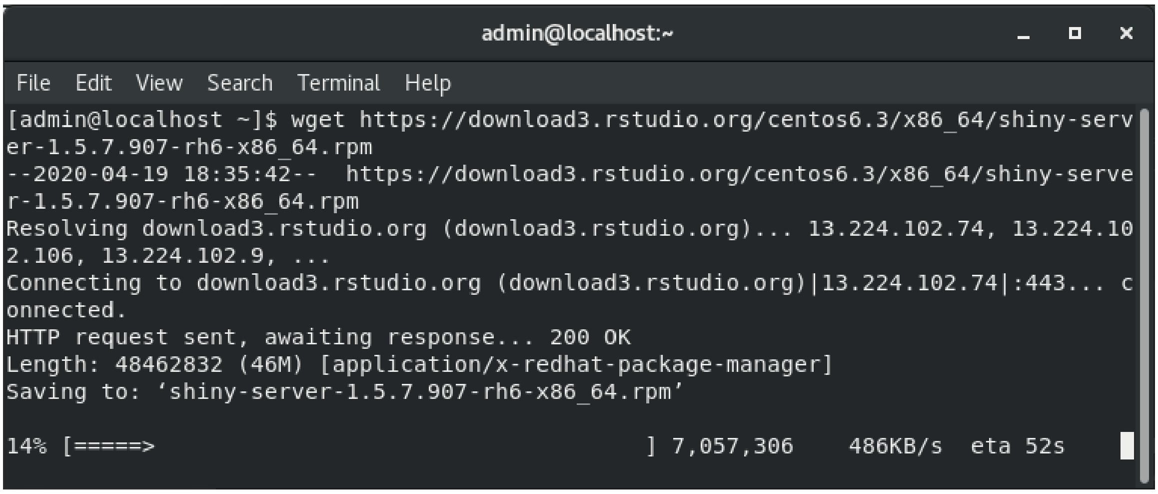
Voyons comment installer (mais pas configurer car cela est un métier à part entière surtout à cause de la partie "sécurité" des serveurs) R Shiny Server sur Cent OS, Scientific Linux et Redhat.

Pour commençons par installer quelques package R utiles pour la suite:

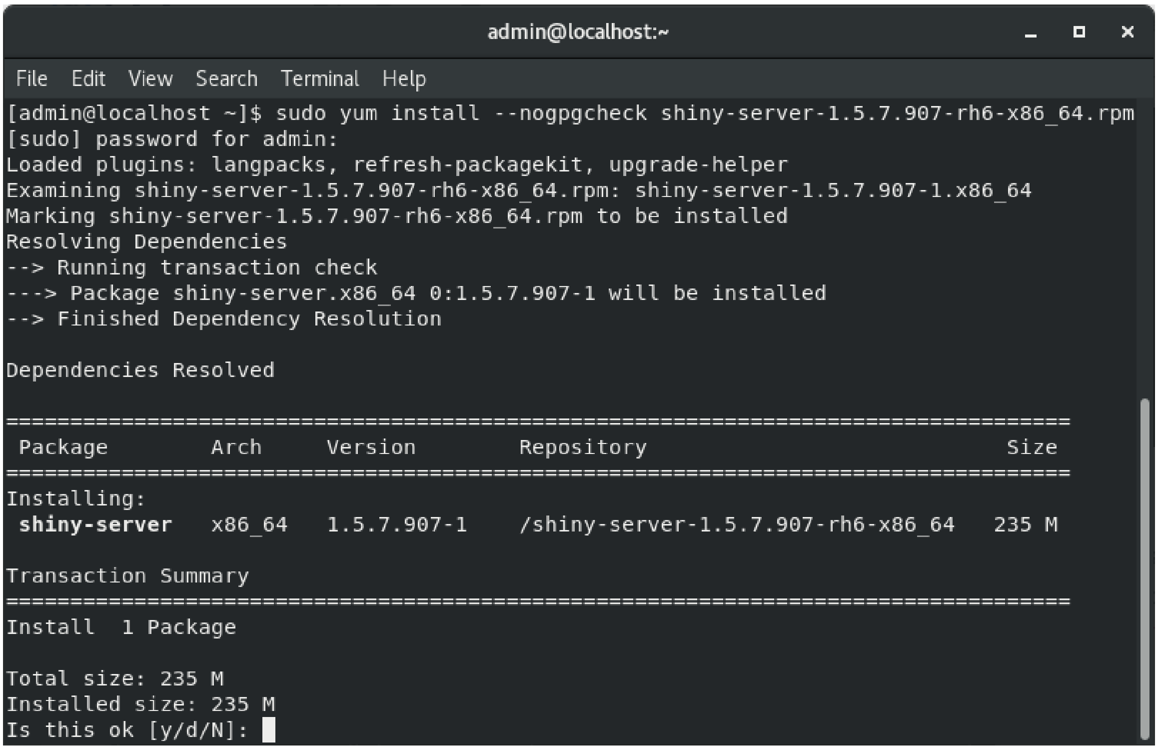


Cela peut prendre entre 5 et 35 minutes suivant la machine.

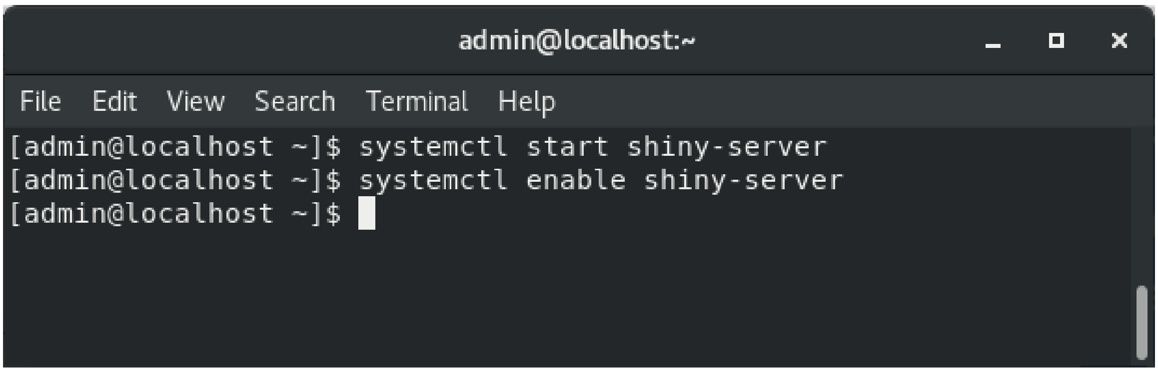
Ensuite, nous téléchargeons le fichier dínstallation rpm de Shiny Server:



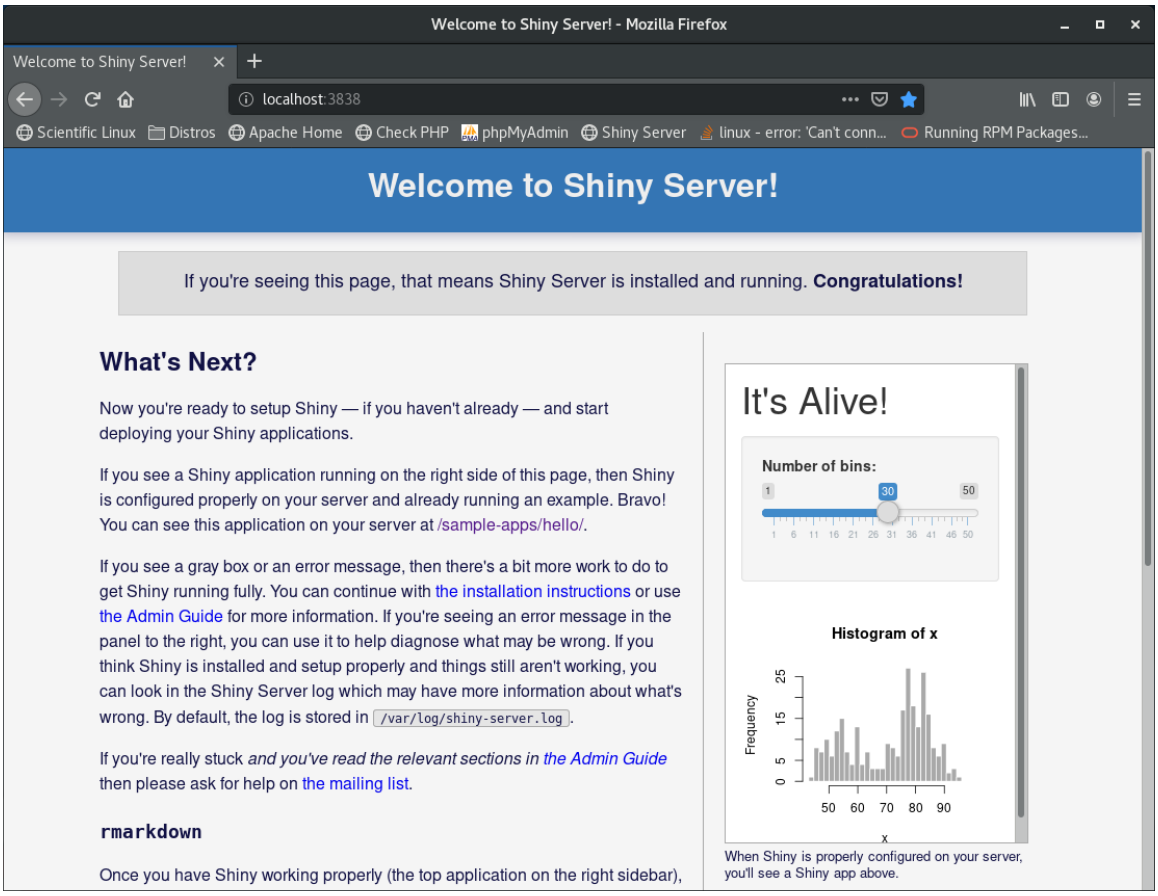
Une fois téléchargé, nous lançons l'installation:



Ensuite, nous devons lancer les services R Shiny:



Ensuite, il suffit d'ouvrir un navigateur et de mettre l'adresse par défaut **localhost:3838** pour lancer R Shiny Server:



Pour vérifier la version installée, on peut écrire à tout moment:

```
admin@localhost:~
File Edit View Search Terminal Help
[admin@localhost ~]$ R

R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

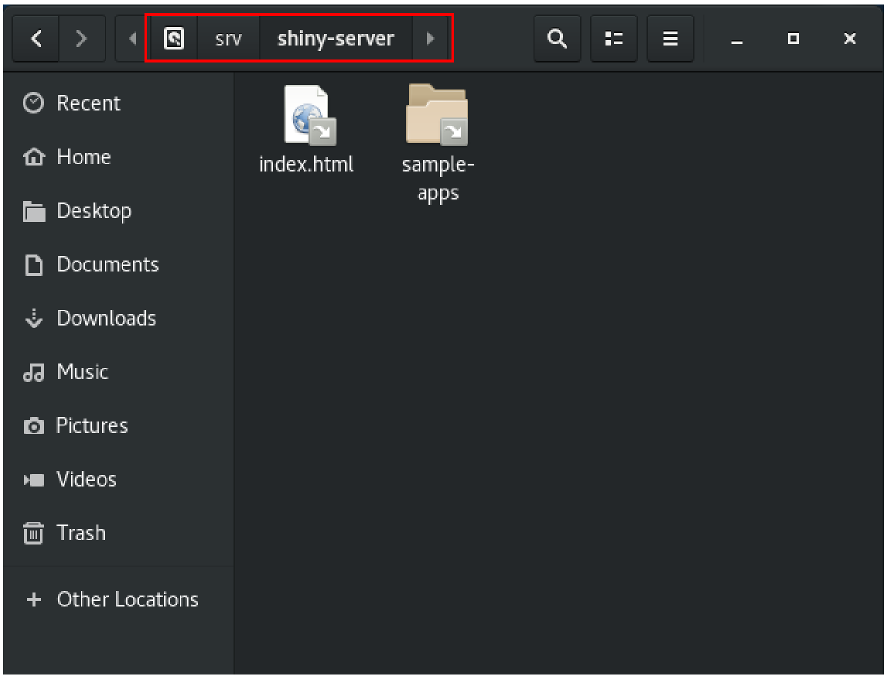
  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

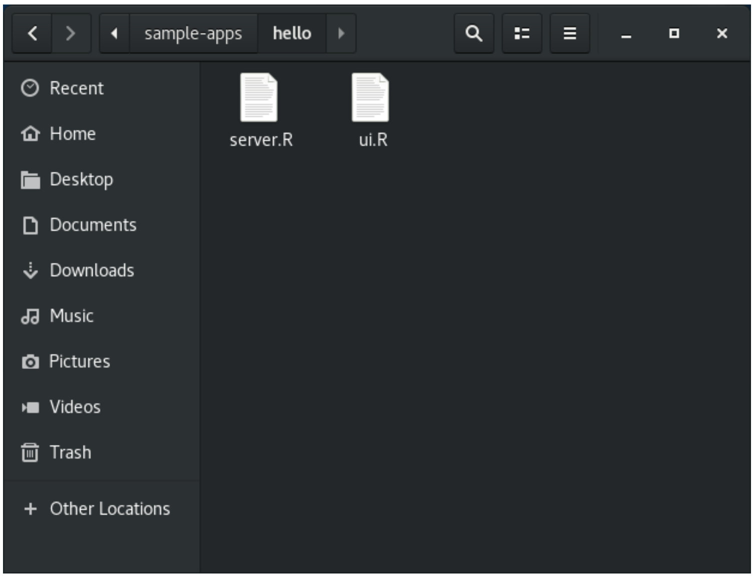
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> system('shiny-server --version', intern = TRUE)
[1] "Shiny Server v1.5.7.907" "Node.js v8.10.0"
> 
```

Le dossier par défaut où mettre les applications est dans **/srv/shiny-server/**:



Tout dossier qui sera mis en-dessous de **/srv/shiny-server/** sera gérée comme un app:



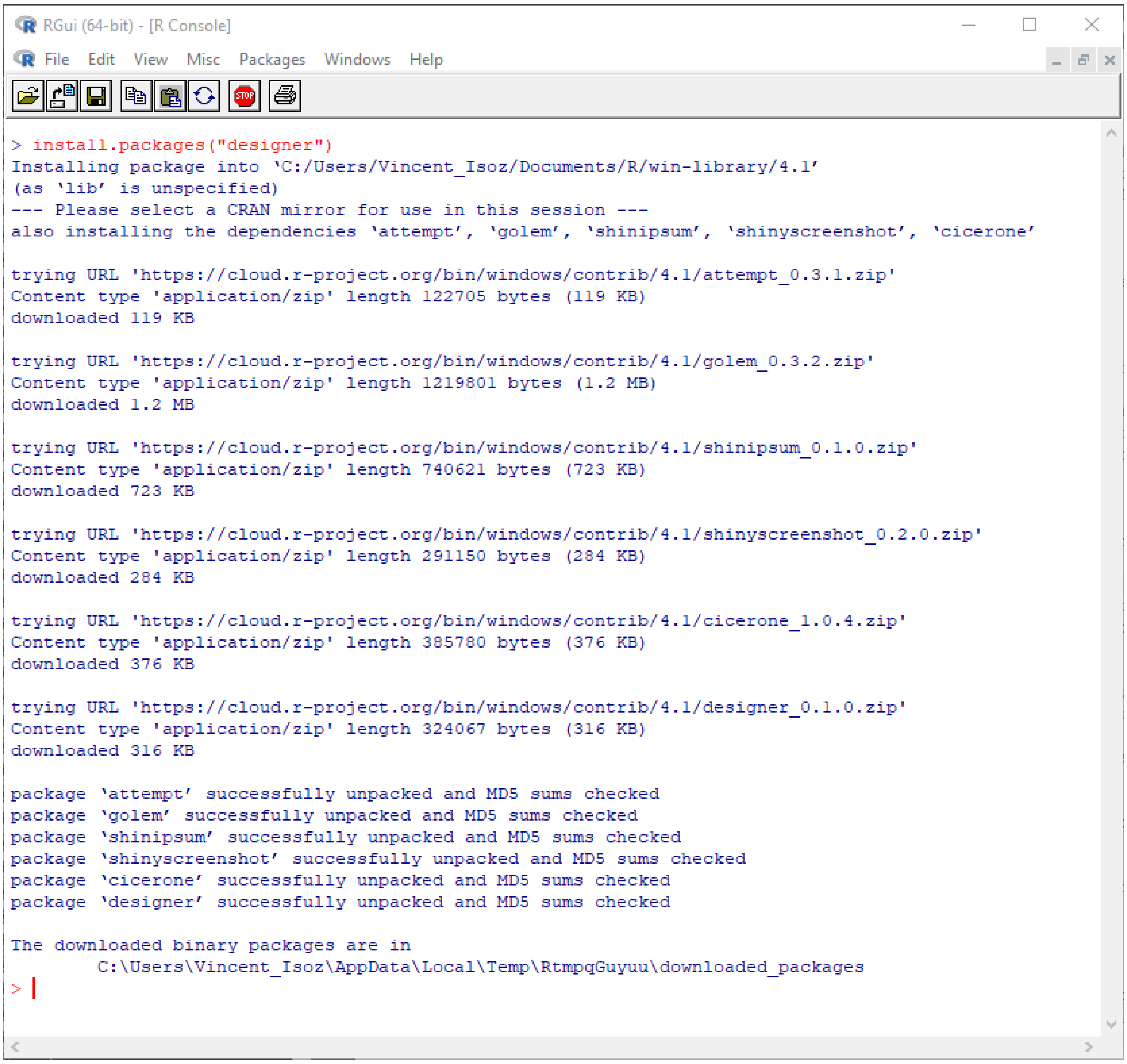
Exemples

Voyons maintenant quelques exemples d'applications de R Shiny.

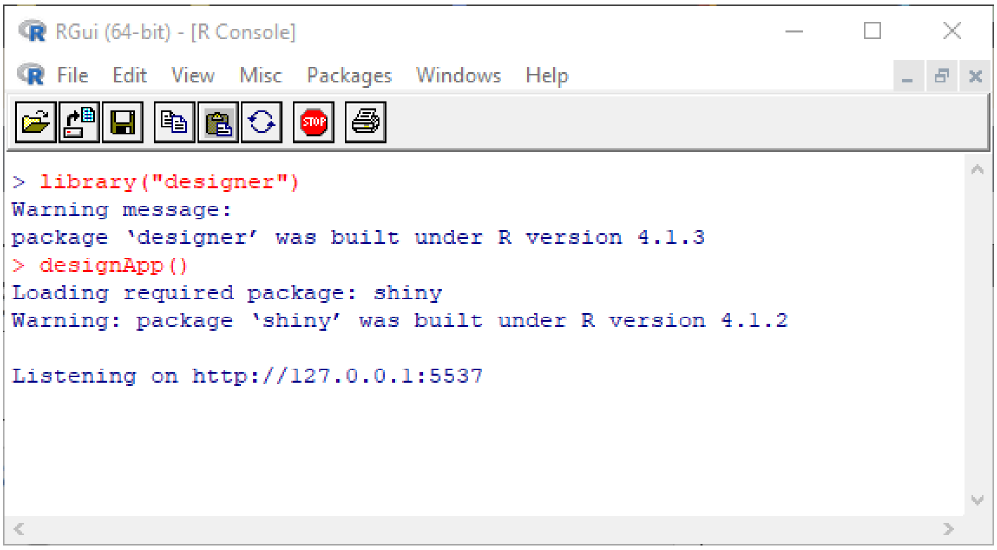
Exemple 7.: Designer un site R Shiny avec un assistant

R 4.1.1

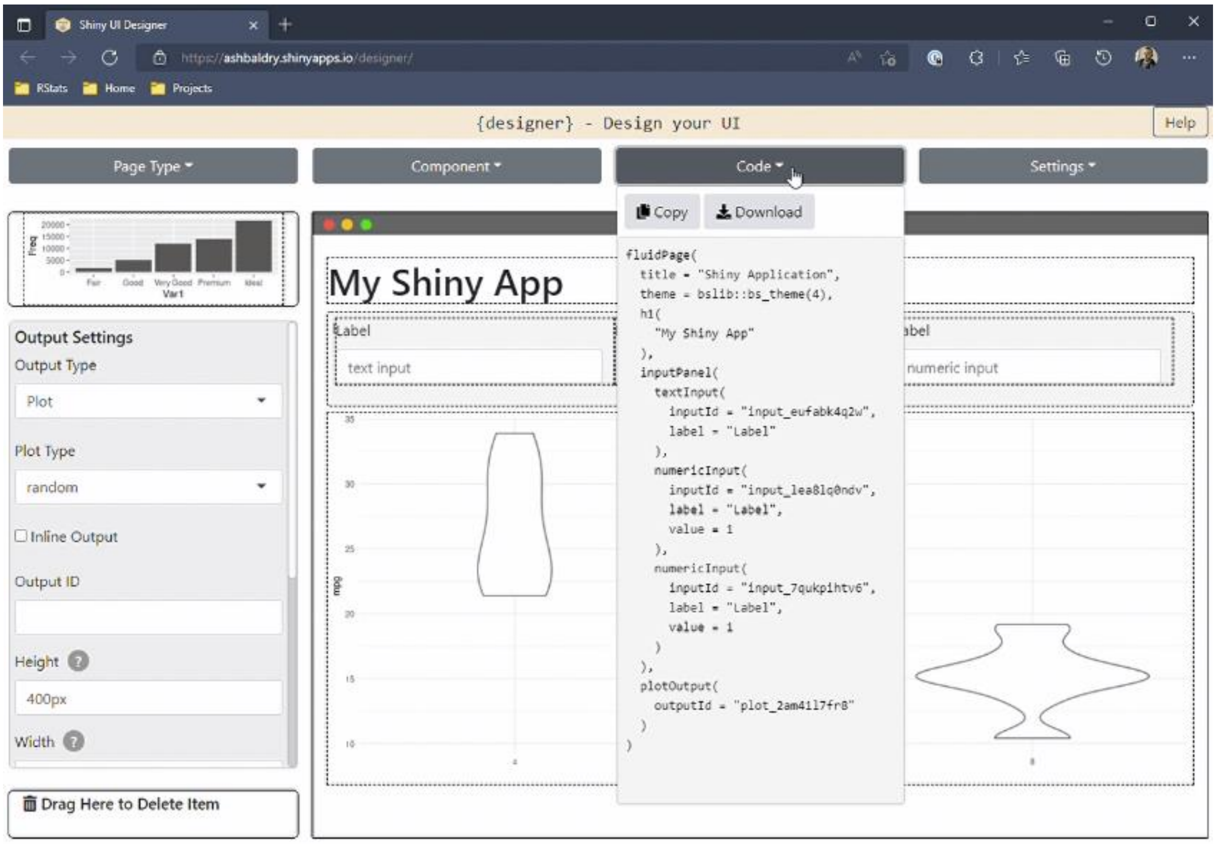
Avant de coder complément de A à Z un site R Shiny, il est utilse de savoir qu'il existe un assistant qui permet de répondre aux besoins les plus courants sans écrire la moindre ligne de script R Shiny. Pour cela, nous allons utiliser et installer le package **designer**:



Nous chargeons le package designer et utiliser la fonction designApp( ) pour lancer l'éditeur:



Ce qui ouvre:



Une fois ouvert, créez l'application selon vos besoins jusqu'à ce que vous soyez satisfait de la mise en page de l'application, puis copiez le code dans le fichier R approprié

Pour plus d'informations, voir:

https://ashbaldry.github.io/designer/

Exemple 8.: Requêter de façon interactive SQL Server Express
R 4.0.5

Pour cet exemple nous recommandons très fortement d'abord au lecteur de relire comment se connecter à SQL Server Express avec le package RODBC à la page 453.

```
library("RODBC") # Also loads RODBC
library("shiny")

ui <- shinyUI(

  pageWithSidebar(

    headerPanel("Hide Side Bar example"),
    sidebarPanel(
      textInput("Id", "Enter Employee ID below"),
      submitButton(text="Submit")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Data", tableOutput("tbTable"))
      )
    )
  )
)

server <- function(input, output, session) {
  myData <- reactive({
    req(input$Id)

    #se connecter à la base de données comme on l'a vue dans la section
    #traitant de la connexion à SQL Server Express avec RODBC
    con  = odbcConnect("SQL Server Express", rows_at_time = 1)

    #contruction de la requête dynamique
    query = paste0("SELECT EmployeeID, LastName, FirstName FROM Employees
WHERE EmployeeID= ", input$Id)

    #stockage du résultat
    res<-sqlQuery(con,query )

    #fermeture de la connexion
    odbcClose(con)
    #retour des résultats
    res
  })

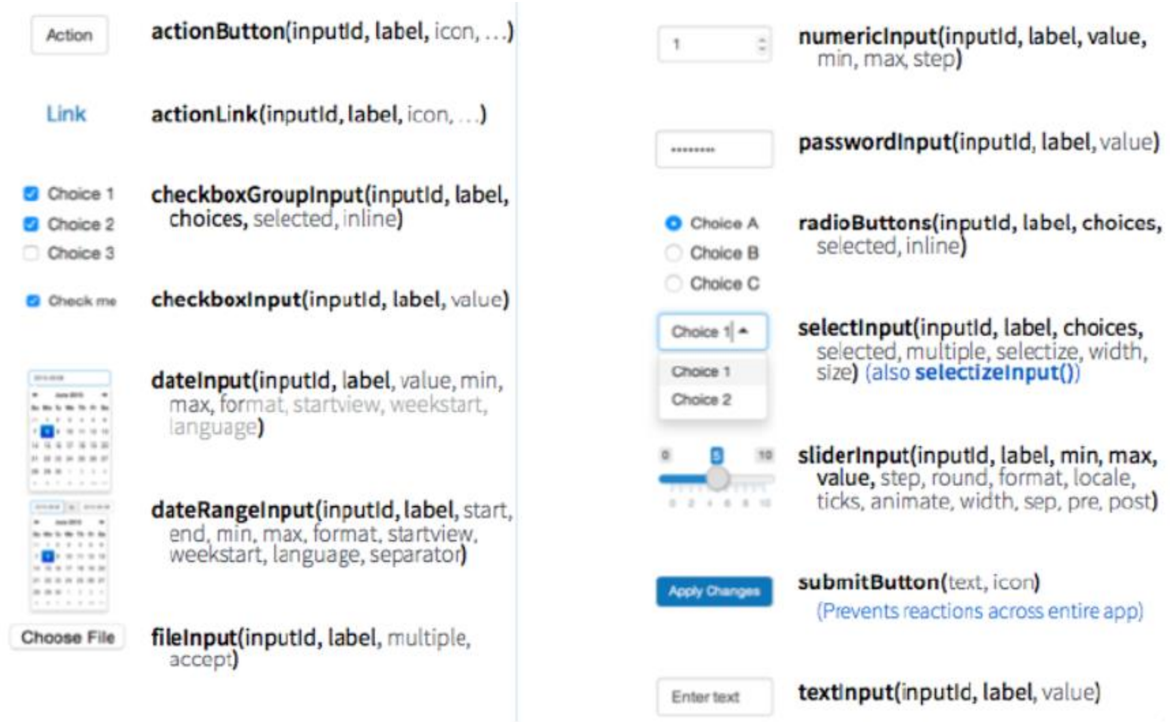
  output$tbTable <-
    renderTable(
      myData()
    )
}

shinyApp(ui = ui, server = server)
```


Ce qui donne visuellement:


Hide Side Bar example
Enter Employee ID below
3
Submit
Data
EmployeeID
LastName
FirstName
3
Leverling
Janet

Types de composants



Types de balises

On peut utiliser les balises suivantes pour compléter notre tableau de bord:

tags$a	tags$data	tags$h6	tags$nav	tags$span
tags$abbr	tags$datalist	tags$head	tags$noscript	tags$strong
tags$address	tags$dd	tags$header	tags$object	tags$style
tags$area	tags$del	tags$hgroup	tags$ol	tags$sub
tags$article	tags$details	tags$hr	tags$optgroup	tags$summary
tags$aside	tags$dfn	tags$HTML	tags$option	tags$sup
tags$audio	tags$div	tags$i	tags$output	tags$table
tags$b	tags$dl	tags$iframe	tags$p	tags$tbody
tags$base	tags$dt	tags$img	tags$param	tags$td
tags$bdi	tags$em	tags$input	tags$pre	tags$textarea
tags$bdo	tags$embed	tags$ins	tags$progress	tags$tfoot
tags$blockquote	tags$eventsource	tags$kbd	tags$q	tags$th
tags$body	tags$fieldset	tags$keygen	tags$ruby	tags$thead
tags$br	tags$figcaption	tags$label	tags$rp	tags$time
tags$button	tags$figure	tags$legend	tags$rt	tags$title
tags$canvas	tags$footer	tags$li	tags$s	tags$tr
tags$caption	tags$form	tags$link	tags$samp	tags$track
tags$cite	tags$h1	tags$mark	tags$script	tags$u
tags$code	tags$h2	tags$map	tags$section	tags$ul
tags$col	tags$h3	tags$menu	tags$select	tags$var
tags$colgroup	tags$h4	tags$meta	tags$small	tags$video
tags$command	tags$h5	tags$meter	tags$source	tags$wbr

Leur usage est simple comme en atteste le petit exemple ci-dessous:

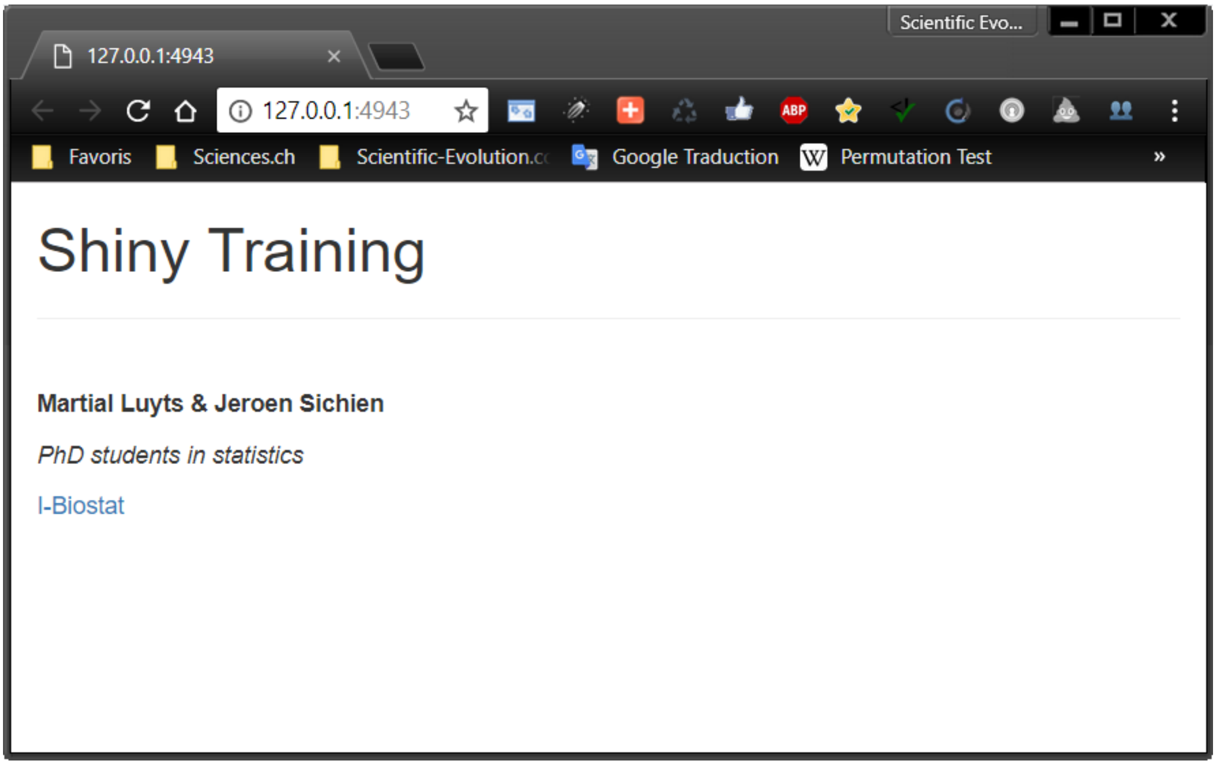
RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
> library(shiny)
> ui<-fluidPage(
+ tags$h1("Shiny Training"),
+ tags$hr(),
+ tags$br(),
+ tags$p(strong("Martial Luyts & Jeroen Sichien")),
+ tags$p(em("PhD students in statistics")),
+ tags$a(href="http://ibiostat.be","I-Biostat"))
> server<-function(input,output){}
> shinyApp(ui=ui,server=server)

Listening on http://127.0.0.1:4943
```

qui donne:

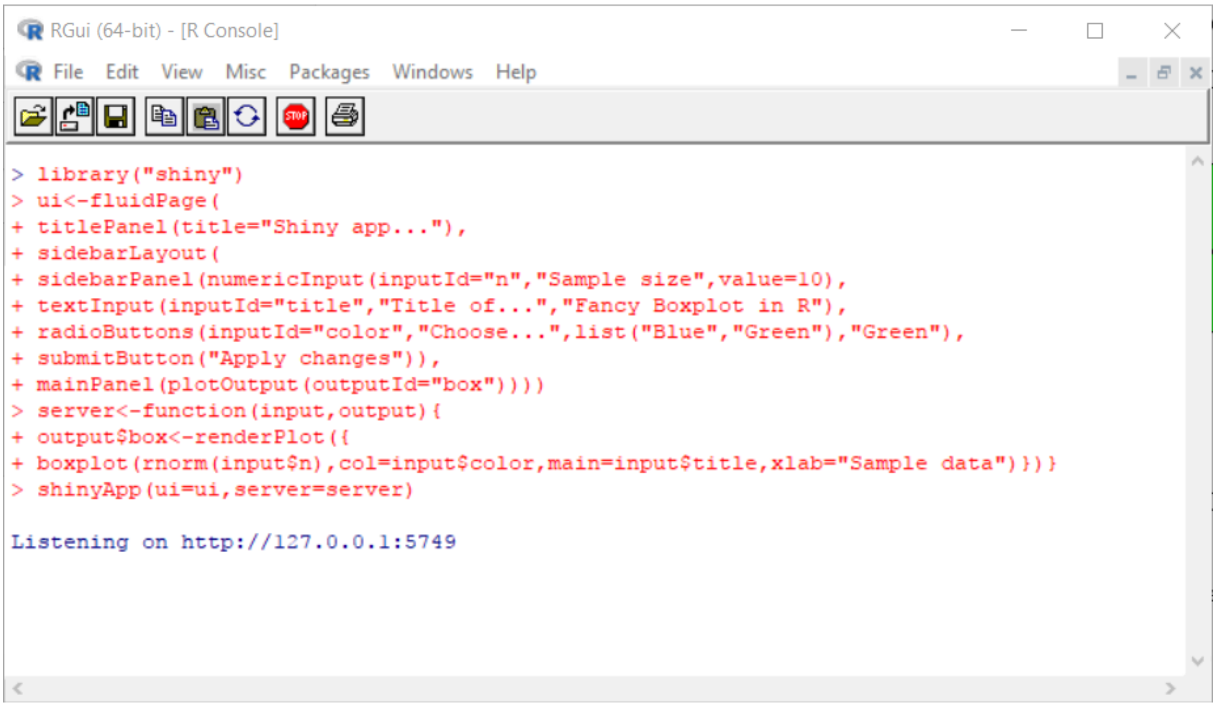


Exemples

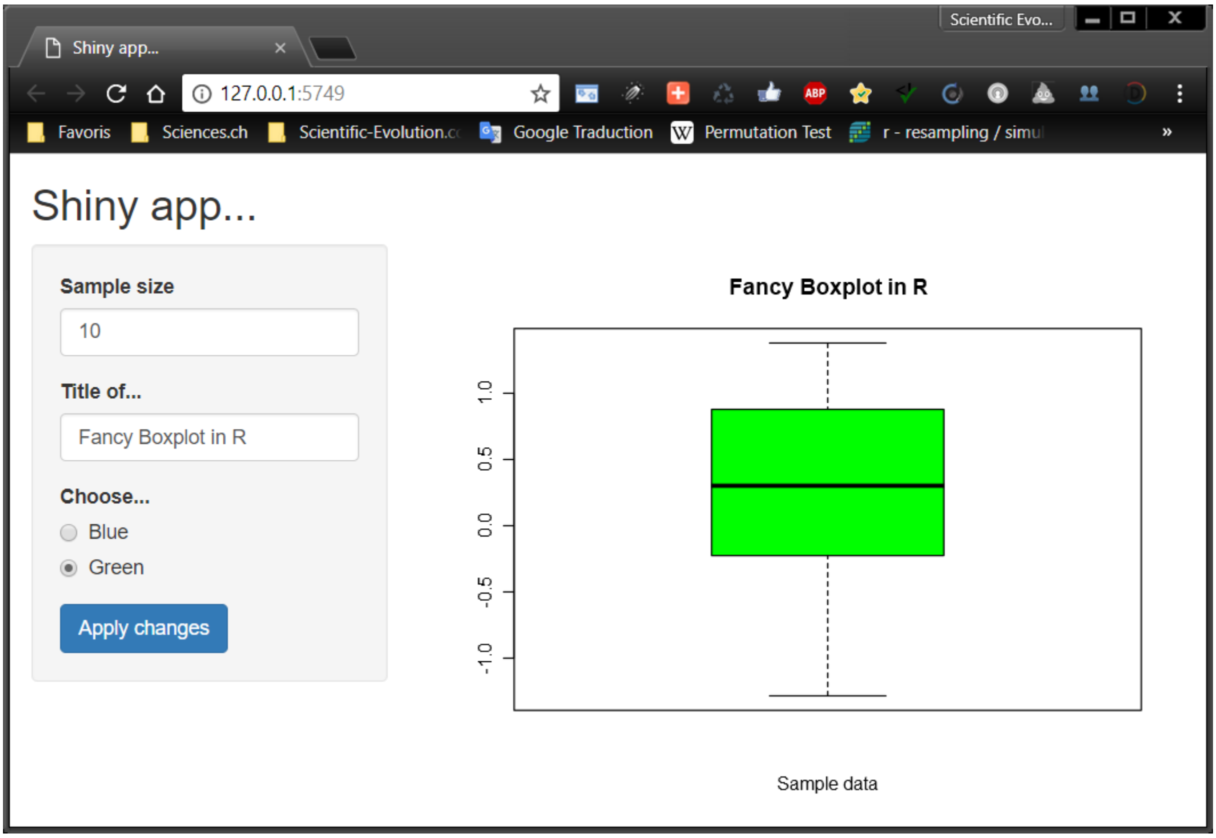
Voici une galerie d'exemples glanés de l'Internet:

Exemple R Shiny 1 (graphiques et champs)

Avec R Shiny 1.0.5 et R 3.4.1:



Ce qui donne:


Shiny app...
127.0.0.1:5749
Favoris
Sciences.ch
Google Traduction
Permutation Test
Shiny app...
Sample size
10
Title of...
Fancy Boxplot in R
Choose...
Blue
Green
Apply changes
Fancy Boxplot in R
1.0
0.5
0.0
-0.5
-1.0
Sample data

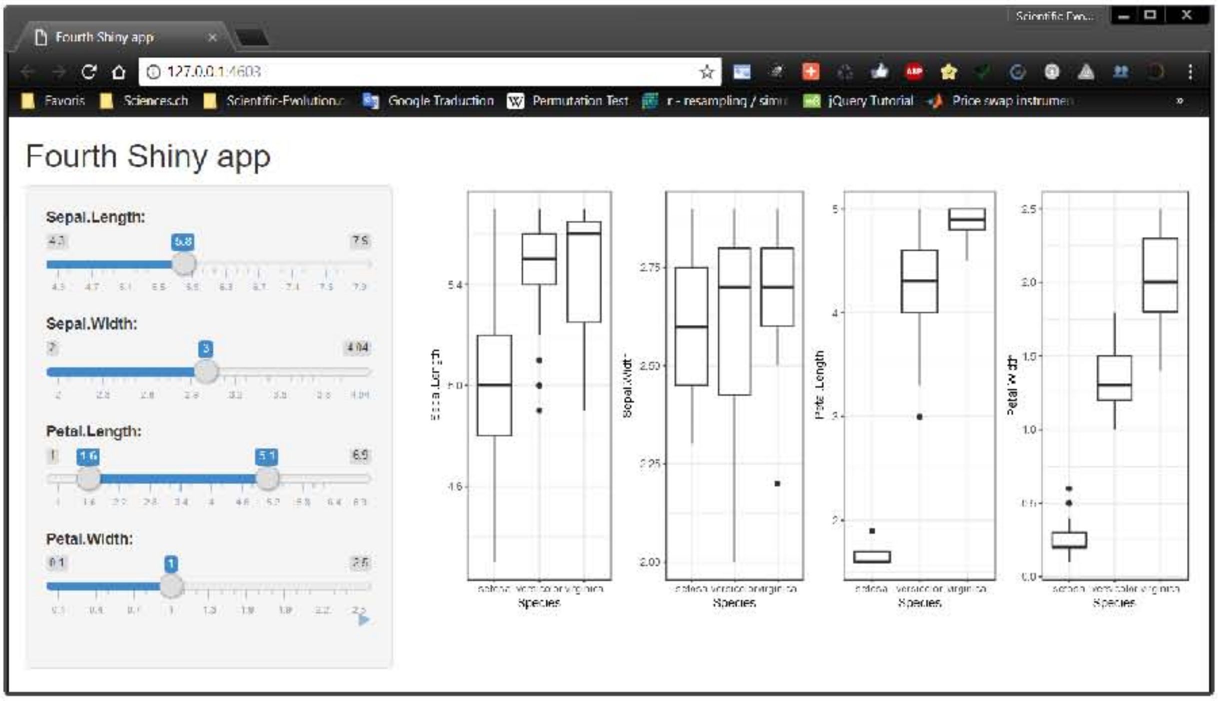
Exemple R Shiny 2 (ggplot2)

Utilisation de GGPlot2 dans R Shiny 1.0.5 et R 3.4.1:

```
> library(shiny)
> library(ggplot2)
> library(gridExtra)
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
>
> ui<-fluidPage(
+ titlePanel(title="Fourth Shiny app"),
+ sidebarLayout(
+ sidebarPanel(
+ sliderInput(inputId="sepallength","Sepal.Length:",min=4.3,max=7.9,value=5.8,step=0.1),
+ sliderInput(inputId="sepalwidth","Sepal.Width:",min=2,max=4.04,value=3,step=0.1),
+ sliderInput(inputId="petallength","Petal.Length:",min=1,max=6.9,value=c(1.6,5.1)),
+ sliderInput(inputId="petalwidth","Petal.Width:",min=0.1,max=2.5,value=1.3,step=0.3,
+ animate=animationOptions(interval=2600,loop=TRUE))),
+ mainPanel(plotOutput(outputId="box"))))
> server<-function(input,output){
+ output$box<-renderPlot({
+ sepall<-subset(iris,Sepal.Length>=4.3 & Sepal.Length<input$sepallength,
+ select=c(Sepal.Length,Species))
+ sepalw<-subset(iris,Sepal.Width>=2 & Sepal.Width<input$sepalwidth,
+ select=c(Sepal.Width,Species))
+ petall<-subset(iris,Petal.Length>=input$petallength[1] & Petal.Length<input$petallength[2],
+ select=c(Petal.Length,Species))
+ petalw<-subset(iris,Petal.Width>=0.1 && Petal.Width<input$petalwidth,select=c(Petal.Width,Species))
+ test<-ggplot(sepall,aes(x=Species,y=Sepal.Length))+geom_boxplot()+theme_bw()
+ testb<-ggplot(sepalw,aes(x=Species,y=Sepal.Width))+geom_boxplot()+theme_bw()
+ testc<-ggplot(petall,aes(x=Species,y=Petal.Length))+geom_boxplot()+theme_bw()
+ testd<-ggplot(petalw,aes(x=Species,y=Petal.Width))+geom_boxplot()+theme_bw()
+ grid.arrange(test,testb,testc,testd,nrow=1)
+ })}
> shinyApp(ui=ui,server=server)

Listening on http://127.0.0.1:4603
```

Ce qui va donner:


Fourth Shiny app
Sepal.Length:
Sepal.Width:
Petal.Length:
Petal.Width:
Species

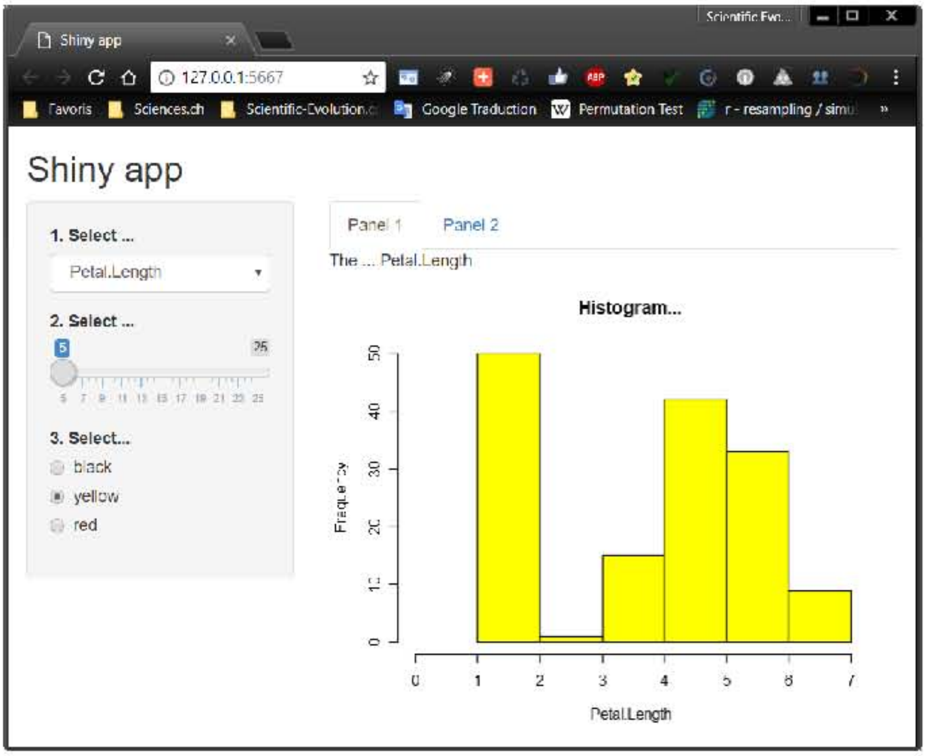
Exemple R Shiny 3 (onglets)

Il s'agit ici de voir comment créer des onglets avec des contenus différents avec Shiny 1.0.5 et R 3.4.1:

```
> library("shiny")
> ui<-fluidPage(
+ titlePanel(title="Shiny app"),
+ sidebarLayout(
+ sidebarPanel(
+ selectInput(inputId="var","1. Select ...", choices=c(
+ "Sepal.Length"=1,"Sepal.Width"=2,"Petal.Length"=3,"Petal.Width"
+ =4),selected=3,selectize=FALSE),
+ sliderInput(inputId="bin","2. Select ...",
+ min=5,max=25,value=5),
+ radioButtons(inputId="colour",label="3. Select...",
+ choices=c("black","yellow","red"),selected="yellow")
+ ),
+ mainPanel(
+ tabsetPanel(type="tab",
+ tabPanel("Panel 1",textOutput(outputId="text1"),
+ plotOutput(outputId="myhist")),
+ tabPanel("Panel 2",tableOutput(outputId="summary"))
+ ))))
> server<-function(input,output){
+ output$text1<-renderText({
+ colm=as.numeric(input$var)
+ paste("The ...",names(iris[colm]))})
+ output$myhist<-renderPlot({
+ colm=as.numeric(input$var)
+ hist(iris[,colm],col=input$colour,
+ xlim=c(0,max(iris[,colm])),
+ main="Histogram...",breaks=input$bin,
+ xlab=names(iris[colm]))})
+ output$summary<-renderTable({summary(iris)})
+ }
> shinyApp(ui=ui,server=server)

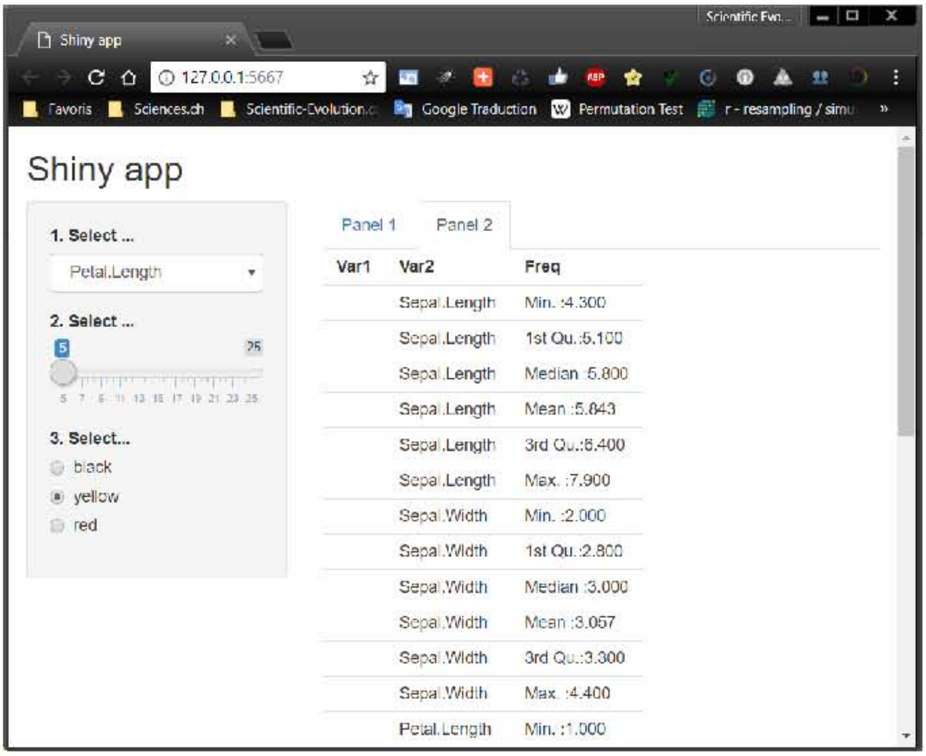
Listening on http://127.0.0.1:5667
```

Ce qui donne:



Shiny app
127.0.0.1:5667
Shiny app
1. Select ...
Petal.Length
2. Select ...
3. Select...
black
yellow
red
Panel 1
Panel 2
The ... Petal.Length
Histogram...
Frequency
Petal.Length

et:

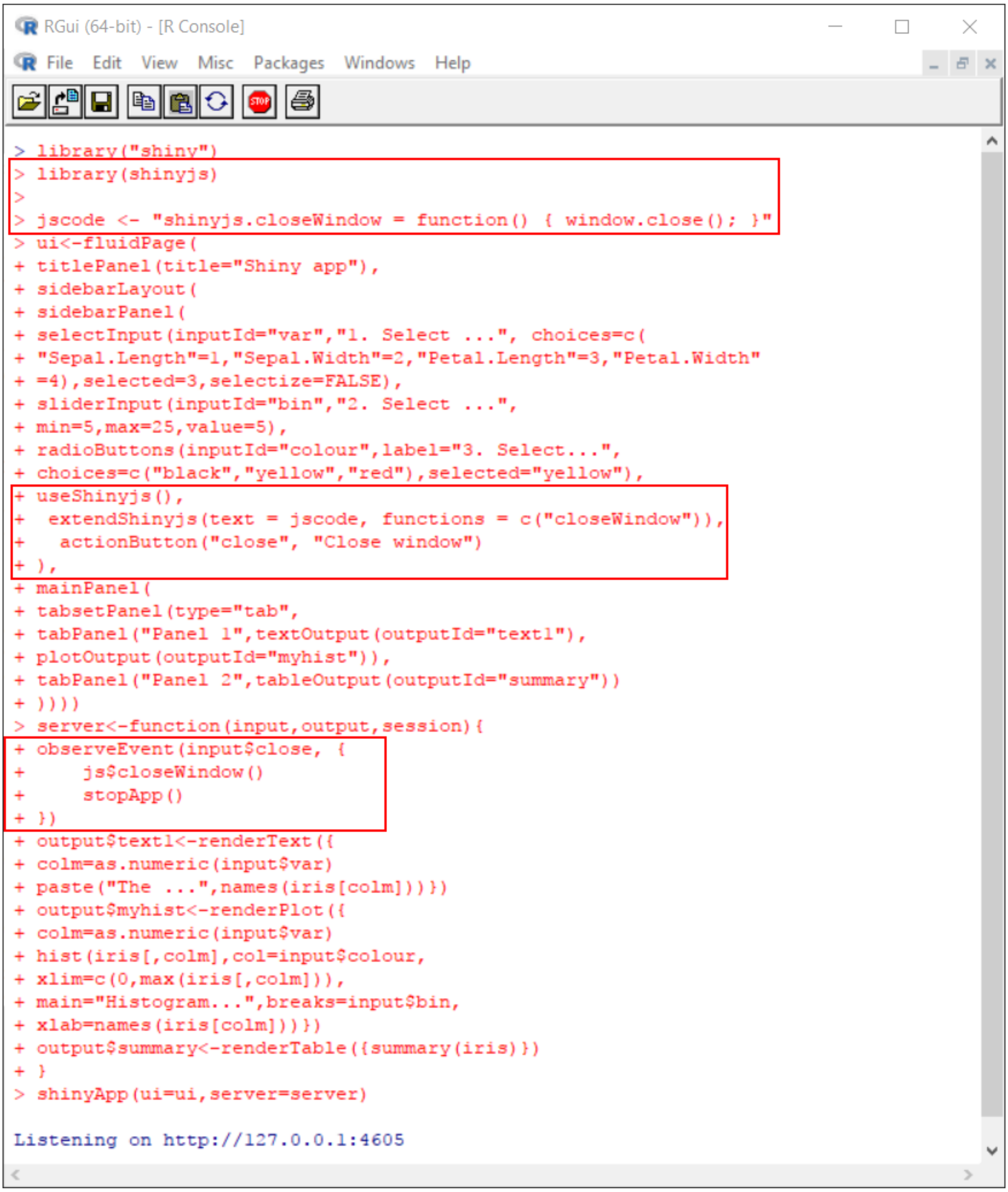


Shiny app
127.0.0.1:5667
Shiny app
1. Select ...
Petal.Length
2. Select ...
3. Select...
black
yellow
red
Panel 1
Panel 2
Var1 Var2 Freq
Sepal.Length Min. :4.300
Sepal.Length 1st Qu.:5.100
Sepal.Length Median :5.800
Sepal.Length Mean :5.843
Sepal.Length 3rd Qu.:6.400
Sepal.Length Max. :7.900
Sepal.Width Min. :2.000
Sepal.Width 1st Qu.:2.800
Sepal.Width Median :3.000
Sepal.Width Mean :3.057
Sepal.Width 3rd Qu.:3.300
Sepal.Width Max. :4.400
Petal.Length Min. :1.000

Exemple R Shiny 3 (bouton de fermeture)

Nous allons voici ici comment traiter une question très fréquente avec Shiny 1.0.5 et R 3.4.1: *comment ajouter un bouton de fermeture de la fenêtre du navigateur et également qui arrête l'application Shiny*?

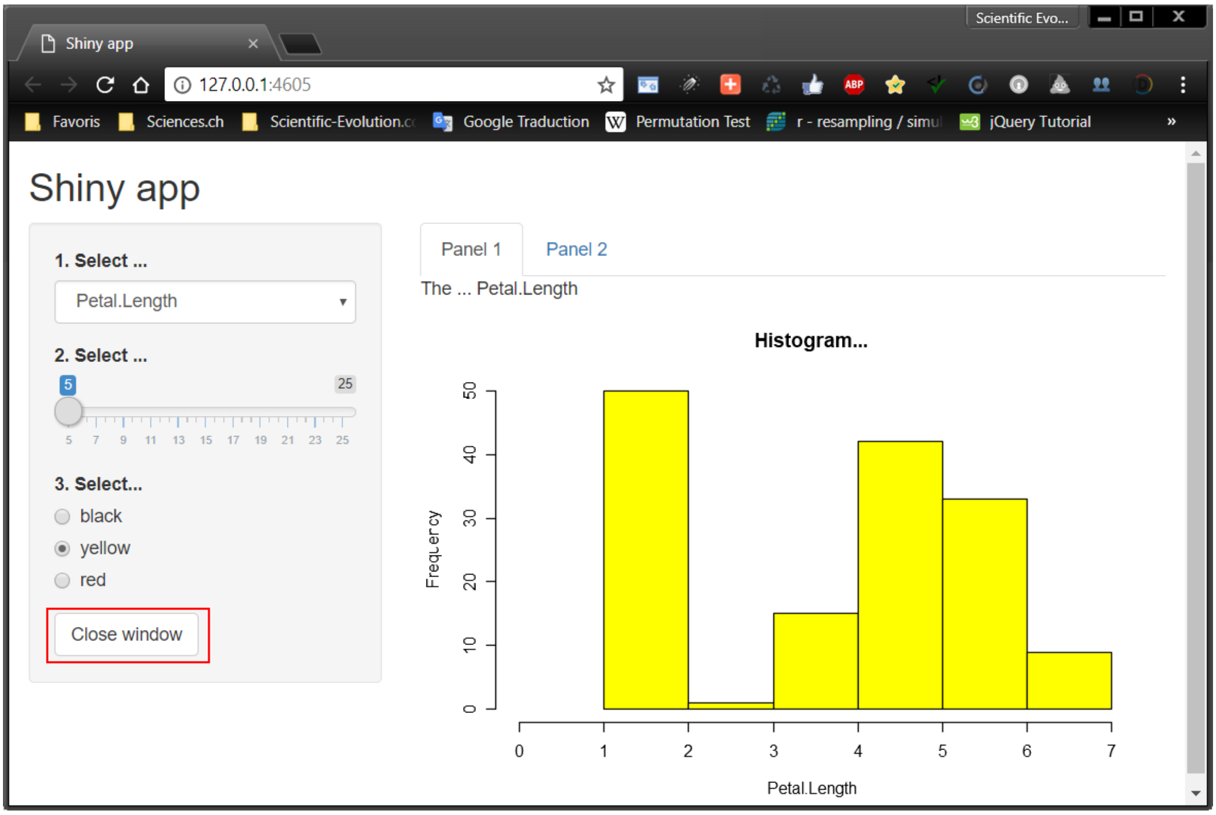
Pour ceci nous allons reprendre l'exemple précédent et mettre en évidence les parties du code qui sont nouvelles:



```
> library("shiny")
> library(shinyjs)
>
> jscode <- "shinyjs.closeWindow = function() { window.close(); }"
> ui<-fluidPage(
+ titlePanel(title="Shiny app"),
+ sidebarLayout(
+ sidebarPanel(
+ selectInput(inputId="var","1. Select ...", choices=c(
+ "Sepal.Length"=1,"Sepal.Width"=2,"Petal.Length"=3,"Petal.Width"
+ =4),selected=3,selectize=FALSE),
+ sliderInput(inputId="bin","2. Select ...",
+ min=5,max=25,value=5),
+ radioButtons(inputId="colour",label="3. Select...",
+ choices=c("black","yellow","red"),selected="yellow"),
+ useShinyjs(),
+  extendShinyjs(text = jscode, functions = c("closeWindow")),
+   actionButton("close", "Close window")
+ ),
+ mainPanel(
+ tabsetPanel(type="tab",
+ tabPanel("Panel 1",textOutput(outputId="text1"),
+ plotOutput(outputId="myhist")),
+ tabPanel("Panel 2",tableOutput(outputId="summary"))
+ ))))
> server<-function(input,output,session){
+ observeEvent(input$close, {
+     js$closeWindow()
+     stopApp()
+ })
+ output$text1<-renderText({
+ colm=as.numeric(input$var)
+ paste("The ...",names(iris[colm]))})
+ output$myhist<-renderPlot({
+ colm=as.numeric(input$var)
+ hist(iris[,colm],col=input$colour,
+ xlim=c(0,max(iris[,colm])),
+ main="Histogram...",breaks=input$bin,
+ xlab=names(iris[colm]))})
+ output$summary<-renderTable({summary(iris)})
+ }
> shinyApp(ui=ui,server=server)

Listening on http://127.0.0.1:4605
```

Ce qui va donner:



et qui fonctionne parfaitement.

Exemple R Shiny 3 (Flux de mise à jour)

L'idée ici est d'avoir une application Shiny qui lit automatiquement les nouveaux flux de données arrivant dans le "c:/tmp/".

D'abord nous avons pour le fichier *server.r* où nous avons mis en rouge les éléments particulièrement intéressants:

```
library("shiny")
library("data.table")
library("ggplot2")
library("gridExtra")
library("readr")
setwd("c:/tmp/")
IsThereNewFile=function(){  #  cheap function whose values over time will
be tested for equality;
                            #  inequality indicates that the underlying
value has changed and needs to be
                            #  invalidated and re-read using valueFunc
        filenames <- list.files(pattern="*.csv", full.names=TRUE)
        length(filenames)
}
ReadAllData=function(){ # A function that calculates the underlying value
        filenames <- list.files(pattern="*.csv", full.names=TRUE)
       read_csv(filenames[length(filenames)])
              }
function(input, output, session) {
 sampled_data <- reactivePoll(10, session,IsThereNewFile, ReadAllData)
         # 10: number of milliseconds to wait between calls to checkFunc
output$plot1<-renderPlot({
        sampled_data=  sampled_data()
        g1= ggplot(sampled_data, aes(depth, fill = cut, colour = cut)) +
                geom_density(alpha = 0.1) +xlim(55,
70)+ggtitle("Distribution of Depth by Cut")+
                theme(plot.title =
element_text(color="darkred",size=18,hjust = 0.5),
                     axis.text.y =
element_text(color="blue",size=12,hjust=1),
                      axis.text.x =
element_text(color="darkred",size=12,hjust=.5,vjust=.5),
                      axis.title.x = element_text(color="red", size=14),
                      axis.title.y = element_text(size=14))
       g2=ggplot(sampled_data, aes(carat, ..count.., fill = cut)) +
                geom_density(position = "stack")+ggtitle("Total Carat by
Count")+
                theme(plot.title =
element_text(color="purple",size=18,hjust = 0.5),
                      axis.text.y =
element_text(color="blue",size=12,hjust=1),
                      axis.text.x =
element_text(color="darkred",size=12,hjust=.5,vjust=.5),
                      axis.title.x = element_text(color="red", size=14),
                      axis.title.y = element_text(size=14))
       g3=ggplot(sampled_data, aes(carat, ..count.., fill = cut)) +
                geom_density(position = "fill")+ggtitle("Conditional
Density Estimate")+
                theme(plot.title = element_text(color="black",size=18,hjust
= 0.5),
                      axis.text.y =
element_text(color="blue",size=12,hjust=1),
```

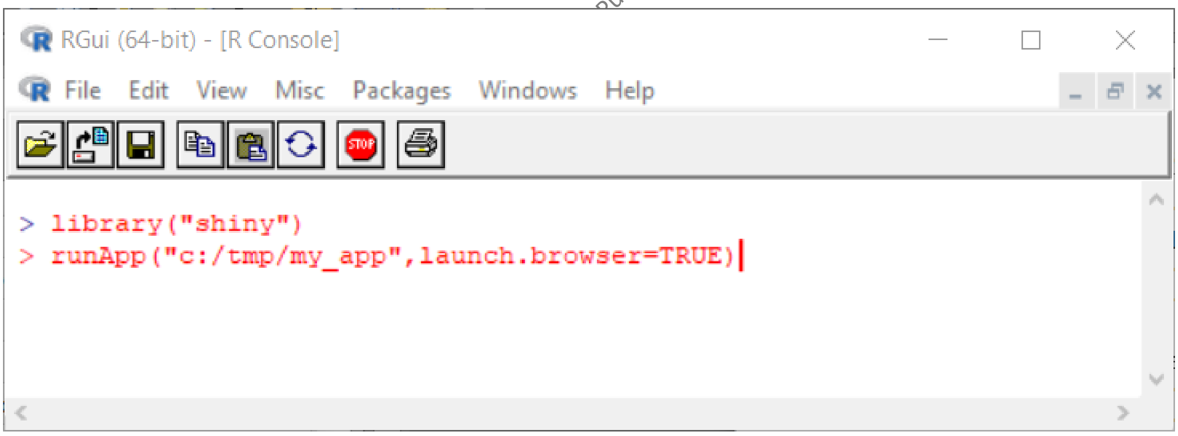
```
                      axis.text.x =
element_text(color="darkred",size=12,hjust=.5,vjust=.5),
                      axis.title.x = element_text(color="red", size=14),
                      axis.title.y = element_text(size=14))

g4=ggplot(sampled_data,aes(carat,price))+geom_boxplot()+facet_grid(.~cut)+
                ggtitle("Price by Carat for each cut")+
                theme(plot.title =
element_text(color="darkblue",size=18,hjust = 0.5),
                      axis.text.y =
element_text(color="blue",size=12,hjust=1),
                      axis.text.x =
element_text(color="darkred",size=12,hjust=.5,vjust=.5),
                      axis.title.x = element_text(color="red", size=14),
                      axis.title.y = element_text(size=14))
        grid.arrange(g1,g2,g3,g4)
})
}
```

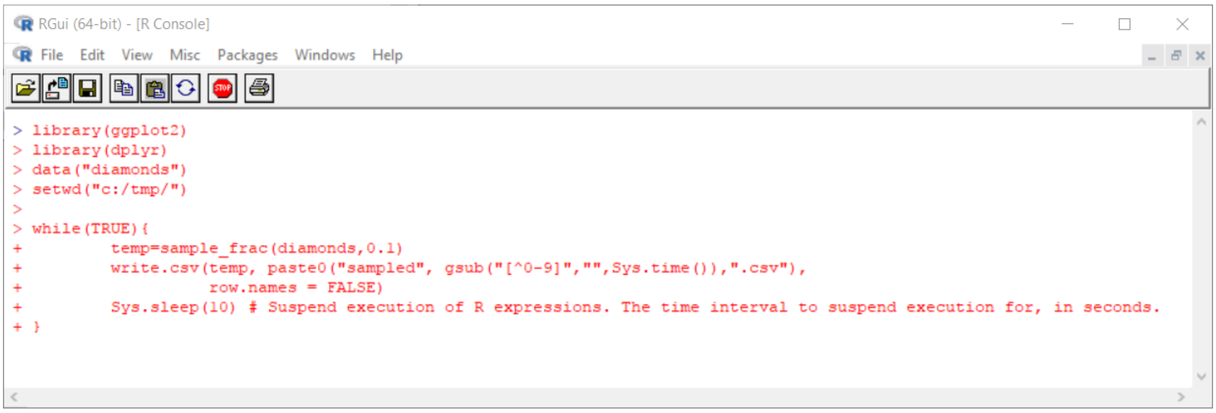
Ensuite, nous avons le fichier *ui.R*:

```
library(shiny)
fluidPage(
        tags$h2("Visualizing Streaming Data with Shiny",
                style="color:blue;text-align:center"),
        plotOutput("plot1",height = "600px")
     )
```

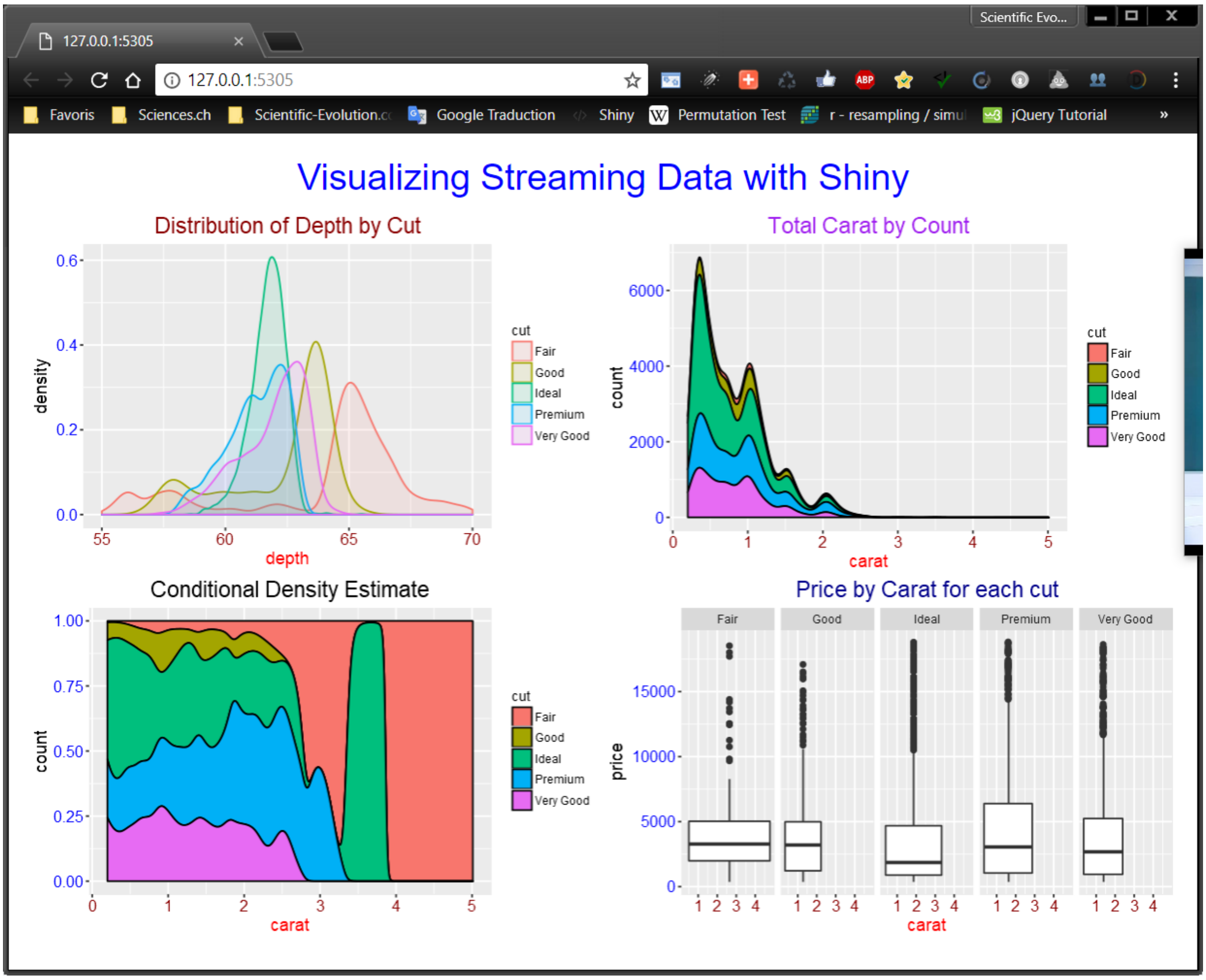
Une fois des deux fichiers enregistrés dans le dossier *c:/tmp/my_app*, nous ouvrons une première console R avec pour lancer l'application Shiny:



Nous ouvrons ensuite une deuxième console R dont le travail sera de mettre à jour le jeu de données:

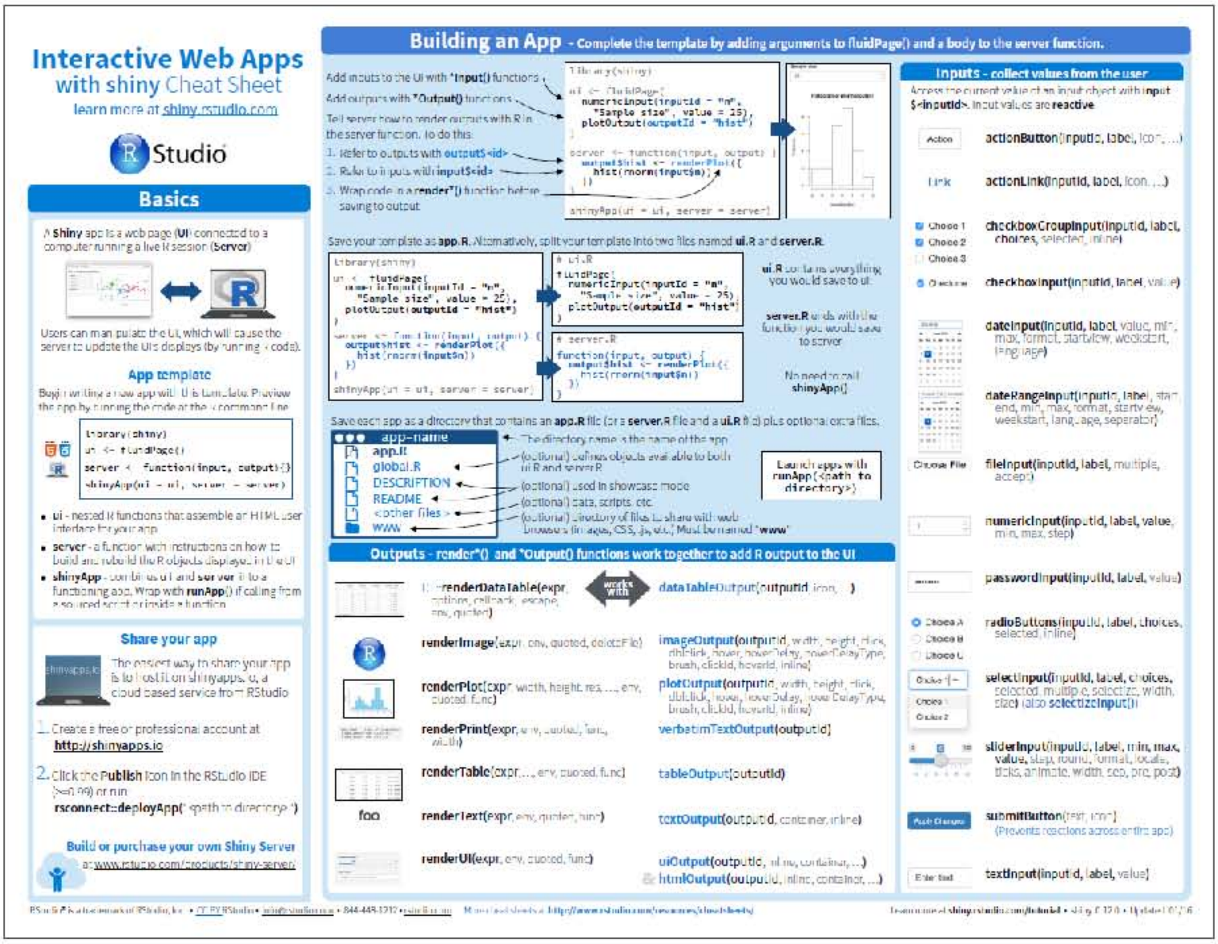


Ce qui donne la page web suivante:



qui se met à jour régulièrement!

Enfin, pour ceux qui aiment R Shiny, voici sa dernière cheatsheet en date:



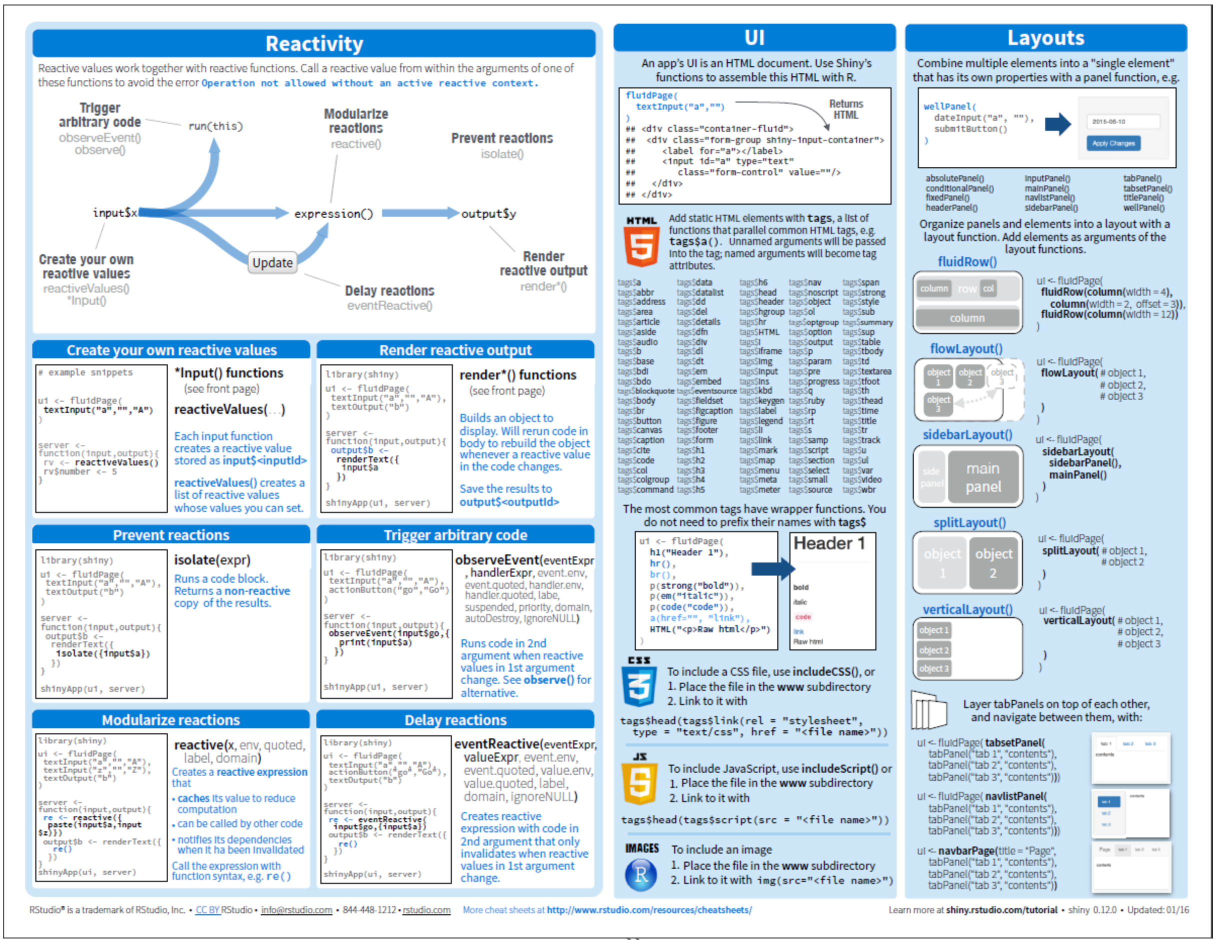


Figure 351 Cheatsheet R Shiny

40. Conclusion

Voilà nous sommes arrivés à la fin de ce livre. Ce dernier n'est bien évidemment pas encore terminé et nécessite aussi de nombreuses mises à jour et compléments d'informations mais constitue toutefois probablement une bonne référence francophone pour tout praticien des statistiques de niveau LMD (Licence, Master, Doctorat) et tout débutant à R qui souhaiterait démarrer du bon pied et ce tout en étant informé des bonnes pratiques d'utilisation.

Nous rappelons, pour ceux qui auraient sauté la lecture de l'introduction, que ce livre est surtout un compagnon au livre théorique *Opera Magistris* et aux formations dispensées par les deux auteurs de ce livre électronique.

41. Quelques abréviations dans R

Commande R	Abréviation étendue	Commentaires
`ls`	[L]i[S]t objects	common command in Unix-like operating systems
`rm`	[R]e[M]ove objects	common command in Unix-like operating systems
`str`	[STR]ucture of an object	
`unz`	[UNZ]ip	
`getwd`	[GET] [W]orking [D]irectory	
`dir`	[DIR]ectory	
`sprintf`	[S]tring [PRINT] [F]ormatted	
`c`	[C]ombine values	
`regexpr`	[REG]ular [EXPR]ession	Why "regular"? See regular sets, regular language
`diag`	[DIAG]onal values of a matrix	
`col`	[COL]umn	
`lapply`	[L]ist [APPLY]	Apply function to each element and return a list
`sapply`	[S]implify [APPLY]	Apply function to each element and attempt to return a vector (i.e., a vector is "simpler" than a list)
`mapply`	[M]ultivariate [APPLY]	Multivariate version of sapply
`tapply`	[T]able [APPLY]	Apply function to sets of values as defined by an index
`apply`	[APPLY] function to sets of values as defined by an index	
`MARGIN = 1 or 2 in apply`	rows [1] come before columns [2]	e.g., a 2 x 3 matrix has 2 rows and 3 columns (note: row count is stated first)
`rmvnorm`	[R]andom number generator for [M]ulti[V]ariate [NORM]al data	
`rle`	[R]un [L]ength [E]ncoding	

Commande R	Abréviation étendue	Commentaires
`ftable`	[F]ormat [TABLE]	
`xtabs`	Cross (i.e., [X]) [TAB]ulation	[X] is the symbol of a cross; [X] is sometimes spoken as "by". Cross-tabulating means to cross one variable with another
`xtable`	[TABLE] of the object [X]	
`formatC`	[FORMAT] using [C] style formats	i.e., [C] the programming language
`Sweave`	[S] [WEAVE]	The R Programming language is a dialect of S. Weaving involves combining code and documentation
`cor`	[COR]relation	
`ancova`	[AN]alysis [O]f [COVA]riance	
`manova`	[M]ultivariate [AN]alysis [O]f [VA]riance	
`aov`	[A]nalysis [O]f [V]ariance	
`TukeyHSD`	[T]ukey's [H]onestly [S]ignificant [D]ifference	
`hclust`	[H]ierarchical [CLUST]er analysis	
`cmdscale`	[C]lassical metric [M]ulti[D]imensional [SCAL]ing	
`factanal`	[FACT]or [ANAL]ysis	
`princomp`	[PRIN]cipal [COMP]onents analysis	
`prcomp`	[PR]incipal [COMP]onents analysis	
`lme`	[L]inear [M]ixed [E]ffects model	
`resid`	[RESID]uals	
`ranef`	[RAN]dom [EF]fects	
`anova`	[AN]alysis [O]f [VA]riance	

Commande R	Abréviation étendue	Commentaires
`fixef`	[FIX]ed [EF]ffects	
`vcov`	[V]ariance-[COV]ariance matrix	
`logLik`	[LOG] [LIK]elihood	
`BIC`	[B]ayesian [I]nformation [C]riteria	
`mcmcsamp`	[M]arkov [Chain] [Monte] [C]arlo [SAMP]ling	
`eval`	[EVAL]uate an R expression	
`cat`	con[CAT]enate	standard Unix command
`apropos`	Search documentation for a purpose or on a topic (i.e., [APROPOS])	Unix command for search documentation;
`read.csv`	[READ] a file in [C]omma [S]eperated [V]alues format	i.e., in each row of the data commas separate values for each variable
`read.fwf`	[READ] a file in [F]ixed [W]idth [F]ormat	
`seq`	Generate [SEQ]uence	
`rep`	[REP]licate values of x	perhaps also [REP]eat
`dim`	[DIM]ension of an object	Typically, number of rows and columns in a matrix
`gl`	[G]enerate factor [L]evels	
`rbind`	[R]ows [BIND]	
`cbind`	[C]olumns [BIND]	
`is.na`	[IS] [N]ot [A]vailable	
`nrow`	[N]umber of [ROW]s	
`ncol`	[N]umber of [COL]umns	
`attr`	[ATTR]ibute	
`rev`	[REV]erse	

Commande R	Abréviation étendue	Commentaires
`diff`	[DIFF]erence between x and a lag of x	
`prod`	[PROD]uct	
`var`	[VAR]iance	
`sd`	[S]tandard [D]eviation	
`cumsum`	[CUM]ulative [SUM]	
`cumprod`	[CUM]ulative [PROD]uct	
`setdiff`	[SET] [DIFF]erence	
`intersect`	[INTERSECT]ion	
`Re`	[RE]al part of a number	
`Im`	[IM]aginary part of a number	
`Mod`	[MOD]ulo opertion	remainder of division of one number by another
`t`	[T]ranspose of a vector or matrix	
`substr`	[SUBSTR]ing	
`strsplit`	[STR]ing [SPLIT]	
`grep`	[G]lobal / [R]egular [E]xpression / [P]rint	Etymology based on text editor instructions in programs such as ed
`sub`	[SUB]stitute identified pattern found in string	
`gsub`	[G]lobal [SUB]stitute identified pattern found in string	
`pmatch`	[P]artial string [MATCH]ing	
`nchar`	[N]umber of [CHAR]acters in a string	
`ps.options`	[P]ost-[S]cript [OPTIONS]	
`win.metafile`	[WIN]dows [METAFILE] graphic	

Commande R	Abréviation étendue	Commentaires
dev.off	[DEV]ice [OFF]	
dev.cur	[CUR]rent [DEV]ice	
dev.set	[SET] the current [DEV]ice	
hist	[HIST]ogram	
pie	[PIE] Chart	
coplot	[CO]nditioning [PLOT]	
matplot	[PLOT] colums of [MAT]rices	
assocplot	[ASSOC]iation [PLOT]	
plot.ts	[PLOT] [T]ime [S]eries	
qqnorm	[Q]uantile-[Q]uantile [P]lot based on normal distribution	
persp	[PERSP]ective [P]lot	
xlim	[LIM]it of the [X] axis	
ylim	[LIM]it of the [Y] axis	
xlab	[LAB]el for the [X] axis	
ylab	[LAB]el for the [Y] axis	
main	[MAIN] title for the plot	
sub	[SUB] title for the plot	
mtext	[M]argin [TEXT]	
abline	[LINE] on plot often of the form y = [A] + [B] x	
h argument in abline	[H]orizontal line	
v argument in abline	[V]ertical line	
par	Graphics [PAR]ameter	
adj as par	[ADJ]ust text [J]ustification	

Commande R	Abréviation étendue	Commentaires
`bg as par`	[B]ack[G]round colour	
`bty as par`	[B]ox [TY]pe	
`cex as par`	[C]haracter [EX]tension or [EX]pansion of plotting objects	
`cex.sub as par`	[C]haracter [EX]tension or [EX]pansion of [SUB]title	
`cex.axis as par`	[C]haracter [EX]tension or [EX]pansion of [AXIS] annotation	
`cex.lab as par`	[C]haracter [EX]tension or [EX]pansion X and Y [LAB]els	
`cex.main as par`	[C]haracter [EX]tension or [EX]pansion of [MAIN] title	
`col as par`	Default plotting [COL]our	
`las as par`	[L]abel of [A]xis [S]tyle	
`lty as par`	[L]ine [TY]pe	
`lwd as par`	[L]ine [W]i[D]th	
`mar as par`	[MAR]gin width in lines	
`mfg as par`	Next [G]raph for [M]atrix of [F]igures	
`mfcol as par`	[M]atrix of [F]igures entered [COL]umn-wise	
`mfrow as par`	[M]atrix of [F]igures entered [ROW]-wise	
`pch as par`	[P]lotting [CH]aracter	
`ps as par`	[P]oint [S]ize of text	Point is a printing measurement
`pty as par`	[P]lot region [TY]pe	
`tck as par`	[T]i[CK] mark length	
`tcl as par`	[T]i[C]k mark [L]ength	

Commande R	Abréviation étendue	Commentaires
`xaxs as par`	[X] [AX]is [S]tyle	
`yaxs as par`	[Y] [AX]is [S]tyle	
`xaxt as par`	[X] [AX]is [T]ype	
`yaxt as par`	[Y] [AX]is [T]ype	
`asp as par`	[ASP]ect ratio	
`xlog as par`	[X] axis as [LOG]arithm scale	
`ylog as par`	[Y] axis as [LOG]arithm scale	
`omi as par`	[O]uter [M]argin width in [I]nches	
`mai as par`	[MA]rgin width in [I]nches	
`pin as par`	[P]lot size in [IN]ches	
`xpd as par`		Perhaps: [X = Cut] [P]lot ? Perhaps D for device
`xyplot`	[X] [Y] [PLOT]	[X] for horizontal axis; [Y] for vertical axis
`bwplot`	[B]ox and [W]hisker plot	
`qq`	[Q]uantile-[Quantile] plot'	
`splom`	[S]catter[PLO]t [M]atrix	
`optim`	[OPTIM]isation	
`lm`	[L]inear [M]odel	
`glm`	[G]eneralised [L]inear [M]odel	
`nls`	[N]onlinear [L]east [S]quare parameter esetimation	
`loess`	[LO]cally [E]stimated [S]catterplot [S]moothing	
`prop.test`	[TEST] null hypothesis that [PROP]ortions in several gropus are the same	

Commande R	Abréviation étendue	Commentaires
rnorm	[R]andom number drawn from [NORM]al distribution	
dnorm	[D]ensity of a given quantile in a [NORM]al distribution	
pnorm	[D]istribution function for [NORM]al distribution returning cumulaive [P]robability	
qnorm	[Q]uantile function based on [NORM]al distribution	
rexp	[R]andom number generation from [EXP]onential distribution	
rgamma	[R]andom number generation from [GAMMA] distribution	
rpois	[R]andom number generation from [POIS]on distribution	
rweibull	[R]andom number generation from [WEIBULL] distribution	
rcauchy	[R]andom number generation from [CAUCHY] distribution	
rbeta	[R]andom number generation from [BETA] distribution	
rt	[R]andom number generation from [t] distribution	
rf	[R]andom number generation from [F] distribution	F for Ronald [F]isher

Commande R	Abréviation étendue	Commentaires
`rchisq`	[R]andom number generation from [CHI][SQ]uare distribution	
`rbinom`	[R]andom number generation from [BINOM]ial distribution	
`rgeom`	[R]andom number generation from [EXP]onential distribution	
`rhyper`	[R]andom number generation from [HYPER]geometric distribution	
`rlogis`	[R]andom number generation from [LOGIS]tic distribution	
`rlnorm`	[R]andom number generation from [L]og [NOR]mal distribution	
`rnbinom`	[R]andom number generation from [N]egative [BINOM]ial distribution	
`runif`	[R]andom number generation from [UNIF]orm distribution	
`rwilcox`	[R]andom number generation from [WILCOX]on distribution	
`ggplot in ggplot2`	[G]rammar of [G]raphics [PLOT]	See Leland Wilkinson (1999)
`aes in ggplot2`	[AES]thetic mapping	
`geom_ in ggplot2`	[GEOM]etric object	
`stat_ in ggplot2`	[STAT]istical summary	
`coord_ in ggplot2`	[COORD]inate system	

Commande R	Abréviation étendue	Commentaires
`qplot` in `ggplot2`	[Q]uick [PLOT]	
`x` as argument	[X] is common letter for unknown variable in math	
`FUN` as argument	[FUN]ction	
`pos` as argument	[POS]ition	
`lib.loc` in `library`	[LIB]rary folder [LOC]ation	
`sep` as argument	[SEP]erator character	
`comment.char` in `read.table`	[COMMENT] [CHAR]acter(s)	
`I`	[I]nhibit [I]nterpretation or [I]nsulate	
`T` value	[T]rue	
`F` value	[F]alse	
`na.rm` as argument	[N]ot [A]vailable [R]e[M]oved	
`fivenum`	[FIVE] [NUM]ber summary	
`IQR`	[I]nter [Q]uartile [R]ange	
`coef`	Model [COEF]ficients	
`dist`	[DIST]ance matrix	
`df` as argument	[D]egrees of [F]reedom	
`mad`	[M]edian [A]bsolute [D]eviation	
`sink`		Divert R output to a connection (i.e., like connecting a pipe to a [SINK])
`eol` in `write.table`	[End] [O]f [L]ine character(s)	

Commande R	Abréviation étendue	Commentaires
`R as software`	[R]oss Ihaka and [R]obert Gentleman or [R] is letter before S	
`CRAN as word`	[C]omprehensive [R] [A]rchive [N]etwork	As I understand it: Inpsired by CTAN (Comprehensive TeX Archive Network); pronunciation of CRAN rhymes with CTAN (i.e., "See" ran as in Iran; "See tan")
`Sexpr`	[S] [EXPR]ession	
`ls.str`	Show [STR]ucture of [L]i[S]ted objects	
`browseEnv`	[BROWSE] [ENV]ironment	
`envir as argument`	[ENVIR]onment	
`q`	[Q]uit	
`cancor`	[CAN]onical [COR]relation	
`ave`	[AVE]rage	
`min`	[MIN]imum	
`max`	[MAX]imum	
`sqrt`	[SQ]uare [R]oo[T]	
`%o%`	[O]uter product	
`&`		& is ampersand meaning [AND]
`|`		| often used to represent OR in computing (http://en.wikipedia.org /wiki /Logical_disjunction)
`:`		sequence generator; aslo used in MATLAB
`nlevels`	[N]umber of [LEVELS] in a factor	
`det`	[DET]erminant of a matrix	
`crossprod`	Matrix [CROSSPROD]uct	
`gls`	[G]eneralised [L]east [S]quares	

Commande R	Abréviation étendue	Commentaires
`dwtest in lmtest`	[D]urbin-[W]atson Test	
`sem in sem`	[S]tructural [E]quation [M]odel	
`betareg in betareg`	[BETA] [REG]ression	
`log`	Natural [LOG]arithm	Default base is e consistent with most mathematics (http://en.wikipedia.org /wiki /Logarithm#Implicit_bases)
`log10`	[LOG]arithm base 10	
`fft`	[F]ast [F]ourier [T]ransform	
`exp`	[EXP]onential function	i.e., e^x
`df.residual`	[D]egrees of [F]reedom of the [R]esidual	
`sin`	[SIN]e function	
`cos`	[COS]ine function	
`tan`	[TAN]gent function	
`asin`	[A]rc[SIN]e function	
`acos`	[A]rc[COS]ine function	
`atan`	[A]rc[TAN]gent function	
`deriv`	[DERIV]ative	
`chol`	[Choleski] decomposition	
`chol2inv`	[CHOL]eski [2=TO] [INV]erse	
`svd`	[S]ingular [V]alue [D]ecomposition	
`eigen`	[EIGEN]value or [EIGEN]vector	
`lower.tri`	[LOWER] [TRI]angle of a matrix	

Commande R	Abréviation étendue	Commentaires
`upper.tri`	[UPPER] [TRI]angle of a matrix	
`acf`	[A]uto [C]orrelation or [C]ovariance [F]unction	
`pacf`	[P]artial A]uto [C]orrelation or [C]ovariance [F]unction	
`ccf`	[C]ross [C]orrelation or [C]ovariance [F]unction	
`Rattle as software`	[R] [A]nalytical [T]ool [T]o [L]earn [E]asily	Perhaps, easy like a baby's rattle
`StatET as software`		Anyone know? Statistics Eclipse?
`JGR as software`	[J]ava [G]UI for [R]	pronounced "Jaguar" like the cat
`ESS as software`	[E]macs [S]peaks [S]tatistics	
`Rcmdr package`	[R] [C]o[m]man[d]e[r] GUI	
`prettyNum`	[PRETTY] [NUM]ber	
`Inf value`	[Inf]inite	
`NaN value`	[N]ot [A] [N]umber	
`is.nan`	[IS] [N]ot [A] [N]umber	
`S3`		R is a dialect of [S]; 3 is the version number
`S4`		R is a dialect of [S]; 4 is the version number
`Rterm as program`	[R] [TERM]inal	
`R CMD as program`		I think: [R] [C]o[m]man[D] prompt
`repos as option`	[REPOS]itory locations	
`bin folder`	[BIN]aries	Common Unix folder for "essential command binaries"

Commande R	Abréviation étendue	Commentaires
`etc folder`	[et cetera]	Common Unix folder for "host-specific system-wide configuration files
`src folder`	[S]ou[RC]e [C]ode	Common Unix folder
`doc folder`	[DOC]umentation	
`RGUI program`	[R] [G]rapical [U]ser [I]nterface	
`.site file extension`	[SITE] specific file	e.g., RProfile.site
`Hmisc package`	Frank [HARRELL]'s package of [MISC]elaneous functions	
`n in debug`	[N]ext step	
`c in debug`	[C]ontinue	
`Q in debug`	[Q]uit	
`MASS package`	[M]odern [A]pplied [S]tatistics with [S]	Based on book of same name by Venables and Ripley
`plyr package`	PL[Y=ie][R]	Double play on words: (1) package manipulates data like pliers manipulate materials; (2) last letter is R as in the program
`aaply`	input [A]rray output [A]rray using [PLY]r package	
`daply`	input [D]ata frame output [A]rray using [PLY]r package	
`laply`	input [L]ist output [A]rray using [PLY]r package	
`adply`	input [A]rray output [D]ata frame using [PLY]r package	
`alply`	input [A]rray output [L]ist using [PLY]r package	
`a_ply`	input [A]rray output Discarded (i.e., _ is blank) using [PLY]r package	

Commande R	Abréviation étendue	Commentaires
RODBC package	[R] [O]bject [D]ata[B]ase [C]onnectivity	
psych package	[PSYCH]ology related functions	
zelig package		"Zelig is named after a Woody Allen movie about a man who had the strange ability to become the physical and psychological reflection of anyone he met and thus to fit perfectly in any situation." - http://gking. harvard.edu/ zelig/
strucchange package	[STRUC]tural [CHANGE]	
relaimpo package	[RELA]tive [IMPO]rtance	
car package	[C]ompanion to [A]pplied [R]egression	Named after book by John Fox
OpenMx packge	[OPEN] Source [M]atri[X] algebra interpreter	Need confirmation that [Mx] means matrix
df in write.foreign	[D]ata [F]rame	
GNU S word	[GNU] is [N]ot [U]nix [S]	
R FAQ word	R [F]requently [A]sked [Q]uestions	
DVI format	[D]e[V]ice [I]ndependent file format	
devel word	[DEVEL]opment	as in code under development
GPL word	[G]eneral [P]ublic [L]icense	
utils package	[UTIL]itie[S]	
mle	[M]aximum [L]ikelihood [E]stimation	
rpart package	[R]ecursive [PART]itioning	
sna package	[S]ocial [N]etwork [A]nalysis	
ergm package	[E]xponential [R]andom [G]raph [M]odels	

Commande R	Abréviation étendue	Commentaires
`rbugs package`	[R] interface to program [B]ayesian inference [Using] [G]ibbs [S]ampling	

Exemple 466.: Identifier le type de données

R 3.0.2

Lorsqu'on décortique un script R dont on n'est pas l'auteur, souvent il nécessaire d'identifier le type d'objet que l'on manipule et parfois de la convertir en un autre type d'object. Nous en verrons quelques cas d'application concrets par la suite mais sinon voici au besoin la liste des commandes utiles pour authentifier le typage et faire une conversion:

Authentification (renvoie un booléen)	Conversion
is.numeric( )	as.numeric( )
is.null( )	as.null( )
is.logical( )	as.logical( )
is.character( )	as.character( )
is.vector( )	as.vector( )
is.factor( )	as.factor( )
is.matrix( )	as.matrix( )
is.list( )	as.list( )
is.data.frame( )	as.data.frame( )
is.na( )	class( )

Tableau 4 Commandes d'identification de type d'objet

42. Index des Tableaux

43. Index des Figures

44. Bibliographie

Voici une bibliographie plus complète et exhaustive que celle présentée au début de ce livre utilisée pour le rédiger. Nous devons de nombreux emprunts à leurs différents auteurs et les remercions pour leurs contributions à l'ingénierie et à la SCIENCE.

45. Index

C

D

E

F

G

M

N

O

P

Q

R

S

T

U